JDF Specification

1.6-DRAFT-IP-1

6 April 2017

CIP4 THANKS ITS PARTNER LEVEL MEMBERS

Legal Notice

Use of this document is subject to the following conditions which are deemed accepted by any person or entity making use hereof.

Copyright Notice

Copyright © 2000-2017, International Cooperation for the Integration of Processes in Prepress, Press and Postpress (CIP4) with registered office in Zurich, Switzerland. All Rights Reserved. CIP4 hereby grants to any person or entity obtaining a copy of the Specification and associated documentation files (the "Specification") a perpetual, worldwide, non-exclusive, fully paid-up, royalty-free copyright license to use, copy, publish, distribute, publicly display, publicly perform, and/or sublicense the Specification in whole or in part verbatim and without modification, unless otherwise expressly permitted by CIP4, subject to the following conditions. This legal notice SHALL be included in all copies containing the whole or substantial portions of the Specification. Copies of excerpts of the Specification which do not exceed five (5) pages SHALL include the following short form Copyright Notice: Copyright © 2000-2013, International Cooperation for the Integration of Processes in Prepress, Press and Postpress (CIP4) with registered office in Zurich, Switzerland.

Trademarks and Tradenames

International Cooperation for the Integration of Processes in Prepress, Press and Postpress, CIP4, Job Definition Format, **JDF**, Job Messaging Format, **JMF** and the CIP4 logo are trademarks of CIP4. Rather than put a trademark symbol in every occurrence of other trademarked names, we state that we are using the names only in an editorial fashion, and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Except as contained in this legal notice or as allowed by membership in CIP4, the name of CIP4 SHALL not be used in advertising or otherwise to promote the use or other dealings in this Specification without prior written authorization from CIP4.

Waiver of Liability

The JDF Specification is provided as is, without warranty of any kind, express, implied, or otherwise, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event will CIP4 be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of, or in connection with the JDF Specification or the use or other dealings in the JDF Specification.

Table of Contents

Legal Notice i

Chapter Table of Contents iii

Preface xxix

Chapter Introduction 1

1.1 Further Information 1
1.2 Background on JDF 1
1.3 Conventions Used in This Specification 1
1.3.1 Document References 1
1.3.2 Text Styles 2
1.3.3 XPath Notation Used in this Specification 2
1.3.4 Modification Notes 2
1.3.5 Specification of Cardinality 3
1.3.6 Template for Narrative Description of Resources 3
1.3.7 Template for Tables that Describe Elements 4
1.4 Glossary 5
1.5 Conformance 10
1.5.1 Conformance Terminology 10
1.5.2 Conformance Requirements for JDF Entities 11
1.5.3 Conformance to Settings Policy 13
1.6 Data Structures 13
1.6.1 Units 15
1.6.2 Counting in JDF 17

Chapter Overview of JDF 19

2.1 System Components 19
2.1.1 Job Components 19
2.1.2 Workflow Component Roles 20
2.1.3 System Interaction 21
2.2 JDF Workflow 22
2.2.1 Job Structure 23
2.3 Hierarchical Tree Structure and Networks in JDF 24
2.4 Role of Messaging in JDF 26
2.5 Coordinate Systems in JDF 26
2.5.1 Introduction 26
2.5.2 Coordinates and Transformations 27
2.5.3 Coordinate Systems of Resources and Processes 28
2.5.4 Coordinate System Transformations 28
2.5.5 Product Example: Simple Brochure 29
2.5.6 General Rules 34
2.5.7 Homogeneous Coordinates 35

Chapter Structure of JDF 37
3.1 Generic Contents of All Elements 37
3.1.1 Structure Diagram 38
3.2 JDF 39
3.2.1 Activation Attribute Values 44
3.2.2 Category Attribute Values 44
3.2.3 Structure Diagram of JDF Node 45
3.3 Common Node Types 45
3.3.1 Product Intent Nodes 46
3.3.2 Process Group Nodes 46
3.3.3 Combined Process Nodes 49
3.3.4 Process Nodes 52
3.4 AncestorPool 52
3.4.1 Ancestor 53
3.5 CustomerInfo 54
3.6 NodeInfo 54
3.7 StatusPool 54
3.8 ResourcePool and its Resource Children 54
3.8.1 ResourcePool 54
3.8.2 Resource 54
3.8.3 Abstract Resource 55
3.8.4 Structure Diagram 58
3.8.5 Resource Classes 59
3.8.6 Position of Resources within JDF Nodes 61
3.8.7 Pipe Resources 61
3.8.8 ResourceUpdate 62
3.9 ResourceLinkPool and ResourceLink 62
3.9.1 ResourceLinkPool 62
3.9.2 ResourceLink 62
3.9.3 Structure Diagram 65
3.9.4 Identification of Physical Resources 74
3.10 ResourcePool and ResourceLinkPool – Deep Structure 75
3.10.1 ResourceElement – Subelement of a Resource 75
3.10.2 ResourceRef – Element for Inter-Resource Linking and refelement 75
3.10.3 Set of Resources and Partitioned Subsets Thereof 77
3.10.4 Resource Amount 78
3.10.5 Description of Partitioned Resources 81
3.10.6 PartIDKeys Attribute and Partition Keys 87
3.10.7 Linking to Resources 97
3.10.8 Splitting and Combining Resources 101
3.11 AuditPool and Audit 103
3.11.1 AuditPool 103
3.11.2 Structure Diagram 104
3.11.3 Abstract Audit 104

3.11.4 Audit . . . 105
3.12 JDF Extensibility . . . 116
3.12.1 Namespaces in XML . . . 116
3.12.2 Creating Extension Intent elements . . . 117
3.12.3 Extending Process Types . . . 117
3.12.4 Extending the NodeInfo and CustomerInfo Nodes . . . 117
3.12.5 Extending Existing Resources . . . 117
3.12.6 Extending NMTOKEN Lists . . . 118
3.12.7 Creating New Resources . . . 118
3.12.8 Future JDF Extensions . . . 118
3.12.9 Maintaining Extensions . . . 118
3.12.10 Processing Unknown Extensions . . . 119
3.12.11 Derivation of Types in XML Schema . . . 119
3.13 JDF Versioning . . . 119
3.13.1 JDF Versioning Requirements . . . 119
3.13.2 JDF Version Definition . . . 119
3.13.3 JDF Version Policies . . . 119

Chapter Life Cycle . . . 123

4.1 Creation and Modification . . . 123
4.1.1 Product Intent Constructs . . . 123
4.1.2 Specification of Delivery of End Products . . . 124
4.1.3 Specification of Process Specifics for Product Intent Nodes . . . 124
4.2 Process Routing . . . 126
4.2.1 Determining Executable Nodes . . . 126
4.2.2 Distributing Processing to Work Centers or Devices . . . 127
4.2.3 Device / Controller Selection . . . 128
4.3 Execution Model . . . 128
4.3.1 Serial Processing . . . 128
4.3.2 Partial Processing of Nodes with Partitioned Resources . . . 129
4.3.3 Overlapping Processing Using Pipes . . . 131
4.3.4 Parallel Processing . . . 135
4.3.5 Iterative Processing . . . 136
4.3.6 Approval, Quality Control and Verification . . . 136
4.4 Spawning and Merging . . . 136
4.4.1 Case 1: Standard Spawning and Merging . . . 138
4.4.2 Case 2: Spawning and Merging with Resource Copying . . . 138
4.4.3 Case 3: Parallel Spawning and Merging of Partitioned Resources . . . 139
4.4.4 Case 4: Nested Spawning and Merging in Reverse Sequence . . . 140
4.4.5 Case 5: Spawning and Merging of Independent Jobs . . . 141
4.4.6 Case 6: Simultaneous Spawning and Merging of Multiple Nodes . . . 141
4.5 Node and Resource IDs . . . 141
4.6 Error Handling . . . 141
4.6.1 Classification of Notifications . . . 141
4.6.2 Event Description . . . 141

4.6.3 Error Logging in the JDF File . . . 142
4.6.4 Error Handling via Messaging (JMF) . . . 142
4.7 Test Running . . . 142
4.7.1 Resource Status During a Test Run . . . 142

Chapter JMF Messaging Format . . . 143

5.1 JMF Root . . . 143
5.1.1 Message . . . 144
5.1.2 Structure Diagram . . . 146
5.2 JMF Message Families . . . 147
5.2.1 Query . . . 147
5.2.2 Command . . . 148
5.2.3 Signal . . . 149
5.2.4 Response . . . 151
5.2.5 Acknowledge . . . 152
5.2.6 Registration . . . 153
5.3 JMF Handshaking . . . 154
5.3.1 Single Query/Command Response Communication . . . 154
5.3.2 Signal and Acknowledge Handshaking . . . 154
5.3.3 Reliable Signalling . . . 154
5.3.4 Persistent Channels . . . 155
5.3.5 Subscription . . . 155
5.3.6 Scope of Subscriptions . . . 157
5.3.7 Deleting Persistent Channels . . . 157
5.4 JMF Messaging Levels . . . 157
5.5 Error and Event Messages . . . 158
5.6 Message Template . . . 158
5.6.1 Object Type Column . . . 159
5.7 List of All JMF Messages . . . 159
5.8 Messages for Events and Capabilities . . . 161
5.9 Messages to Query/Command a Job, Device or Controller . . . 162
5.10 Messages for Pipe Control . . . 163
5.10.1 Common PipeControl Element . . . 163
5.11 Queue Support . . . 164
5.11.1 Queue Entry ID Generation . . . 164
5.12 Messages for Queue Entry Handling . . . 164
5.13 Messages for Global Handling of Queues . . . 167
5.13.1 QueueEntryStatus . . . 169
5.14 Elements for Queues . . . 169
5.14.1 Queue . . . 169
5.14.2 QueueEntry . . . 170
5.14.3 QueueEntryDef . . . 172
5.14.4 QueueFilter . . . 172
5.15 Gang Jobs . . . 173
5.16 Extending Messages . . . 174

5.16.1 IfraTrack Support . . . 174
5.17 AbortQueueEntry . . . 175
5.17.1 AbortQueueEntryParams . . . 176
5.18 CloseQueue . . . 176
5.19 Events . . . 176
5.20 FlushQueue . . . 176
5.20.1 FlushQueue Command . . . 176
5.20.2 FlushQueue Query . . . 177
5.21 FlushResources . . . 178
5.21.1 FlushResources Command . . . 178
5.21.2 FlushResources Query . . . 178
5.22 ForceGang . . . 179
5.22.1 GangCmdFilter . . . 179
5.23 GangStatus . . . 179
5.23.1 GangQuFilter . . . 180
5.23.2 GangInfo . . . 180
5.24 HoldQueue . . . 180
5.25 HoldQueueEntry . . . 180
5.25.1 HoldQueueEntryParams . . . 181
5.26 KnownControllers . . . 181
5.27 KnownDevices . . . 181
5.27.1 DeviceFilter . . . 182
5.27.2 DeviceList . . . 182
5.28 KnownJDFServices . . . 183
5.29 KnownMessages . . . 183
5.29.1 KnownMsgQuParams . . . 183
5.29.2 MessageService . . . 183
5.30 KnownSubscriptions . . . 185
5.30.1 SubscriptionFilter . . . 185
5.30.2 SubscriptionInfo . . . 186
5.31 ModifyNode . . . 186
5.31.1 ModifyNode Command . . . 186
5.31.2 ModifyNode Signal . . . 187
5.32 NewJDF . . . 187
5.32.1 NewJDF Query . . . 188
5.32.2 NewJDF Command . . . 188
5.33 NodeInfo . . . 189
5.34 Notification . . . 189
5.34.1 NotificationFilter . . . 190
5.35 Occupation . . . 191
5.36 OpenQueue . . . 191
5.37 PipeClose . . . 191
5.38 PipePause . . . 191
5.39 PipePull . . . 192

5.40 PipePush . . . 192
5.41 QueueStatus . . . 193
5.42 RemoveQueueEntry . . . 193
5.42.1 RemoveQueueEntryParams . . . 194
5.43 RepeatMessages . . . 194
5.44 RequestForAuthentication . . . 194
5.44.1 RequestForAuthentication Command . . . 194
5.44.2 RequestForAuthentication Query . . . 197
5.45 RequestQueueEntry . . . 199
5.45.1 RequestQueueEntryParams . . . 199
5.46 Resource . . . 200
5.46.1 Resource Query . . . 200
5.46.2 Resource Command . . . 204
5.47 ResourcePull . . . 210
5.47.1 ResourcePullParams 211
5.48 ResubmitQueueEntry . . . 212
5.48.1 ResubmissionParams . . . 212
5.49 ResumeQueue . . . 213
5.50 ResumeQueueEntry . . . 213
5.50.1 ResumeQueueEntryParams . . . 213
5.51 ReturnQueueEntry . . . 213
5.51.1 ReturnQueueEntryParams . . . 214
5.52 SetQueueEntryPosition . . . 214
5.52.1 QueueEntryPosParams . . . 215
5.53 SetQueueEntryPriority . . . 215
5.53.1 QueueEntryPriParams . . . 215
5.54 ShutDown . . . 215
5.54.1 ShutDownCmdParams . . . 216
5.55 Status . . . 216
5.55.1 StatusQuParams . . . 217
5.55.2 DeviceInfo . . . 218
5.55.3 JobPhase . . . 219
5.55.4 ModuleStatus . . . 221
5.56 StopPersistentChannel . . . 223
5.56.1 StopPersChParams . . . 223
5.57 SubmissionMethods . . . 223
5.57.1 SubmissionMethods . . . 224
5.58 SubmitQueueEntry . . . 224
5.58.1 QueueSubmissionParams . . . 225
5.59 SuspendQueueEntry . . . 226
5.59.1 SuspendQueueEntryParams . . . 227
5.60 Track . . . 227
5.61 UpdateJDF . . . 227
5.61.1 UpdateJDF Command . . . 227

5.61.2 UpdateJDF Signal . . . 228
5.62 WakeUp . . . 231
5.62.1 WakeUpCmdParams . . . 231

Chapter Processes . . . 233

6.1 Process Template . . . 233
6.2 General Processes . . . 234
6.2.1 Approval . . . 234
6.2.2 Buffer . . . 235
6.2.3 Combine . . . 235
6.2.4 Delivery . . . 235
6.2.5 ManualLabor . . . 236
6.2.6 Ordering . . . 236
6.2.7 Packing . . . 236
6.2.8 QualityControl . . . 236
6.2.9 ResourceDefinition . . . 237
6.2.10 Split . . . 237
6.2.11 Verification . . . 237
6.3 Prepress Processes . . . 238
6.3.1 AssetListCreation . . . 238
6.3.2 Bending . . . 239
6.3.3 ColorCorrection . . . 239
6.3.4 ColorSpaceConversion . . . 239
6.3.5 ContactCopying . . . 240
6.3.6 ContoneCalibration . . . 241
6.3.7 CylinderLayoutPreparation . . . 241
6.3.8 DBDocTemplateLayout . . . 241
6.3.9 DBTemplateMerging . . . 242
6.3.10 DieDesign . . . 242
6.3.11 DieLayoutProduction . . . 242
6.3.12 DigitalDelivery . . . 245
6.3.13 FilmToPlateCopying . . . 246
6.3.14 FormatConversion . . . 246
6.3.15 ImageEnhancement . . . 246
6.3.16 ImageReplacement . . . 246
6.3.17 ImageSetting . . . 247
6.3.18 Imposition . . . 247
6.3.19 InkZoneCalculation . . . 257
6.3.20 Interpreting . . . 257
6.3.21 LayoutElementProduction . . . 258
6.3.22 LayoutPreparation . . . 258
6.3.23 LayoutShifting . . . 259
6.3.24 PageAssigning . . . 259
6.3.25 PDFToPSConversion . . . 259
6.3.26 PDLCreation . . . 260

6.3.27 Preflight . . . 260
6.3.28 PreviewGeneration . . . 261
6.3.29 Proofing . . . 263
6.3.30 PSToPDFConversion . . . 263
6.3.31 RasterReading . . . 263
6.3.32 Rendering . . . 264
6.3.33 RIPing . . . 265
6.3.34 Scanning . . . 265
6.3.35 Screening . . . 265
6.3.36 Separation . . . 266
6.3.37 ShapeDefProduction . . . 266
6.3.38 SheetOptimizing . . . 267
6.3.39 SoftProofing . . . 267
6.3.40 Stripping . . . 267
6.3.41 Tiling . . . 269
6.3.42 Trapping . . . 269
6.4 Press Processes . . . 270
6.4.1 ConventionalPrinting . . . 270
6.4.2 DigitalPrinting . . . 272
6.4.3 Varnishing . . . 274
6.4.4 IDPrinting . . . 274
6.5 Postpress Processes . . . 274
6.5.1 AdhesiveBinding . . . 274
6.5.2 BlockPreparation . . . 275
6.5.3 BoxFolding . . . 275
6.5.4 BoxPacking . . . 275
6.5.5 Bundling . . . 276
6.5.6 CaseMaking . . . 277
6.5.7 CasingIn . . . 277
6.5.8 ChannelBinding . . . 278
6.5.9 CoilBinding . . . 278
6.5.10 Collecting . . . 278
6.5.11 CoverApplication . . . 279
6.5.12 Creasing . . . 279
6.5.13 Cutting . . . 280
6.5.14 DieMaking . . . 280
6.5.15 Dividing . . . 281
6.5.16 Embossing . . . 281
6.5.17 EndSheetGluing . . . 281
6.5.18 Feeding . . . 282
6.5.19 Folding . . . 283
6.5.20 Gathering . . . 283
6.5.21 Gluing . . . 284
6.5.22 HeadBandApplication . . . 284
6.5.23 HoleMaking . . . 285

6.5.24 Inserting . . . 285
6.5.25 Jacketing . . . 286
6.5.26 Labeling . . . 287
6.5.27 Laminating . . . 287
6.5.28 LongitudinalRibbonOperations . . . 287
6.5.29 Numbering . . . 288
6.5.30 Palletizing . . . 288
6.5.31 Perforating . . . 288
6.5.32 PlasticCombBinding . . . 288
6.5.33 PrintRolling . . . 289
6.5.34 RingBinding . . . 289
6.5.35 SaddleStitching . . . 290
6.5.36 ShapeCutting . . . 290
6.5.37 Shrinking . . . 290
6.5.38 SideSewing . . . 291
6.5.39 SpinePreparation . . . 291
6.5.40 SpineTaping . . . 291
6.5.41 Stacking . . . 291
6.5.42 StaticBlocking . . . 292
6.5.43 Stitching . . . 292
6.5.44 Strapping . . . 293
6.5.45 StripBinding . . . 293
6.5.46 ThreadSealing . . . 294
6.5.47 ThreadSewing . . . 294
6.5.48 Trimming . . . 294
6.5.49 WebInlineFinishing . . . 295
6.5.50 Winding . . . 295
6.5.51 WireCombBinding . . . 296
6.5.52 Wrapping . . . 296
6.6 Postpress Processes Structure . . . 297
6.6.1 Block Production . . . 297
6.6.2 HoleMaking . . . 298
6.6.3 Laminating . . . 298
6.6.4 Numbering . . . 298
6.6.5 Packaging Processes . . . 298
6.6.6 Processes in Hardcover Book Production . . . 298
6.6.7 Sheet Processes . . . 299
6.6.8 Tip-on/in . . . 299
6.6.9 Trimming . . . 299
6.6.10 Web Processes . . . 299

Chapter Product Intent . . . 301

7.0.1 Product Intent Descriptions . . . 301
7.1 Intent Properties Template . . . 302
7.1.1 Abstract Span Element . . . 303
7.1.2 Span Elements . . . 303

7.2 ArtDeliveryIntent . . . 308
7.2.1 ArtDelivery . . . 309
7.3 BindingIntent . . . 311
7.3.1 BindingType Attribute Values . . . 314
7.3.2 AdhesiveNote . . . 314
7.3.3 BindList . . . 314
7.3.4 BindItem . . . 315
7.3.5 AdhesiveBinding . . . 316
7.3.6 BookCase . . . 316
7.3.7 ChannelBinding . . . 316
7.3.8 CoilBinding . . . 316
7.3.9 EdgeGluing . . . 316
7.3.10 HardCoverBinding . . . 317
7.3.11 PlasticCombBinding . . . 319
7.3.12 RingBinding . . . 319
7.3.13 SaddleStitching . . . 320
7.3.14 SideSewing . . . 320
7.3.15 SideStitching . . . 321
7.3.16 SoftCoverBinding . . . 321
7.3.17 StripBinding . . . 322
7.3.18 Tabs . . . 322
7.3.19 Tape . . . 323
7.3.20 ThreadSealing . . . 323
7.3.21 ThreadSewing . . . 324
7.3.22 WireCombBinding . . . 324
7.4 ColorIntent . . . 324
7.4.1 ColorStandard Attribute Values . . . 326
7.4.2 ColorsUsed . . . 326
7.5 DeliveryIntent . . . 326
7.5.1 DropIntent . . . 329
7.5.2 DropItemIntent . . . 330
7.5.3 Pricing . . . 331
7.5.4 Payment . . . 331
7.5.5 CreditCard . . . 331
7.6 EmbossingIntent . . . 331
7.6.1 EmbossingItem . . . 331
7.7 FoldingIntent . . . 332
7.8 HoleMakingIntent . . . 333
7.9 InsertingIntent . . . 333
7.9.1 InsertList . . . 334
7.9.2 Insert . . . 334
7.10 LaminatingIntent . . . 335
7.11 LayoutIntent . . . 335
7.12 MediaIntent . . . 338
7.13 NumberingIntent . . . 342

7.14 PackingIntent . . . 342
7.15 ProductionIntent . . . 343
7.16 ProofingIntent . . . 344
7.16.1 PreflightItem . . . 344
7.16.2 ProofItem . . . 344
7.17 PublishingIntent . . . 345
7.18 ScreeningIntent . . . 346
7.19 ShapeCuttingIntent . . . 346
7.19.1 ShapeCut . . . 347
7.20 SizeIntent . . . 347
7.21 VariableIntent . . . 348

Chapter Resources . . . 349

8.1 AdhesiveBindingParams . . . 349
8.2 ApprovalParams . . . 349
8.2.1 ApprovalPerson . . . 349
8.3 ApprovalSuccess . . . 350
8.3.1 ApprovalDetails . . . 350
8.4 Assembly . . . 350
8.4.1 AssemblySection . . . 352
8.4.2 PageAssignedList . . . 352
8.5 AssetListCreationParams . . . 353
8.6 BendingParams . . . 353
8.7 BinderySignature . . . 354
8.7.1 On the use of Bleed . . . 357
8.7.2 On the use of Trim . . . 357
8.7.3 SignatureCell . . . 360
8.8 BlockPreparationParams . . . 361
8.9 BoxFoldingParams . . . 362
8.9.1 BoxApplication . . . 363
8.9.2 BoxFoldAction . . . 364
8.10 BoxPackingParams . . . 367
8.11 BufferParams . . . 368
8.12 Bundle . . . 368
8.12.1 BundleItem . . . 369
8.13 BundlingParams . . . 371
8.14 ByteMap . . . 372
8.14.1 Band . . . 373
8.14.2 PixelColorant . . . 373
8.15 CaseMakingParams . . . 374
8.16 CasingInParams . . . 375
8.17 ChannelBindingParams . . . 376
8.18 CoilBindingParams . . . 376
8.19 CollectingParams . . . 377

8.20 Color . . . 378
8.20.1 DeviceNColor . . . 381
8.20.2 Diecutting Data (DDES3) . . . 381
8.20.3 PrintConditionColor . . . 382
8.21 ColorantControl . . . 385
8.21.1 ColorantConvertProcess . . . 388
8.21.2 ColorantOrder . . . 388
8.21.3 ColorantParams . . . 388
8.21.4 DeviceColorantOrder . . . 389
8.21.5 ColorSpaceSubstitute . . . 389
8.22 ColorCorrectionParams . . . 390
8.22.1 . . . 391
8.23 ColorPool . . . 391
8.24 ColorSpaceConversionParams . . . 391
8.25 Component . . . 393
8.25.1 ProductType Attribute Values . . . 397
8.26 Contact . . . 398
8.26.1 Company . . . 399
8.27 ContactCopyParams . . . 399
8.28 ContentList . . . 399
8.28.1 ContentData . . . 400
8.28.2 ContentMetadata . . . 401
8.29 ConventionalPrintingParams . . . 403
8.30 CoverApplicationParams . . . 406
8.30.1 Score . . . 406
8.31 CreasingParams . . . 407
8.32 CustomerInfo . . . 407
8.32.1 CustomerMessage . . . 408
8.33 CutBlock . . . 408
8.34 CutMark . . . 409
8.35 CuttingParams . . . 410
8.36 CylinderLayout . . . 410
8.36.1 CylinderPosition . . . 411
8.37 CylinderLayoutPreparationParams . . . 414
8.38 DBMergeParams . . . 414
8.39 DBRules . . . 414
8.40 DBSchema . . . 414
8.41 DBSelection . . . 414
8.42 DeliveryParams . . . 414
8.42.1 Drop . . . 416
8.42.2 DropItem . . . 417
8.43 DevelopingParams . . . 417
8.44 Device . . . 418
8.44.1 IconList . . . 420

8.44.2 Icon . . . 420
8.44.3 Module . . . 420
8.45 DieLayout . . . 421
8.45.1 RuleLength . . . 422
8.45.2 Station . . . 422
8.46 DieLayoutProductionParams . . . 423
8.46.1 RepeatDesc . . . 423
8.47 DigitalDeliveryParams . . . 427
8.48 DigitalMedia . . . 428
8.49 DigitalPrintingParams . . . 429
8.49.1 Coordinate systems in DigitalPrinting . . . 429
8.50 DividingParams . . . 432
8.51 ElementColorParams . . . 432
8.52 EmbossingParams . . . 433
8.52.1 Emboss . . . 433
8.53 Employee . . . 434
8.54 EndSheetGluingParams . . . 435
8.54.1 EndSheet . . . 436
8.55 ExposedMedia . . . 436
8.56 ExternalImpositionTemplate . . . 437
8.57 FeedingParams . . . 438
8.57.1 Feeder . . . 438
8.57.2 FeederQualityParams . . . 439
8.57.3 CollatingItem . . . 440
8.58 FileSpec . . . 440
8.58.1 ResourceUsage Attribute Values . . . 444
8.58.2 Container . . . 445
8.58.3 FileAlias . . . 445
8.59 FoldingParams . . . 446
8.60 FontParams . . . 449
8.61 FontPolicy . . . 450
8.62 FormatConversionParams . . . 451
8.63 GatheringParams . . . 451
8.64 GlueApplication . . . 451
8.65 GluingParams . . . 452
8.65.1 Glue . . . 452
8.66 HeadBandApplicationParams . . . 453
8.67 HoleList . . . 453
8.68 HoleMakingParams . . . 454
8.69 IdentificationField . . . 455
8.69.1 EncodingDetails Attribute Values . . . 457
8.69.2 BarcodeDetails . . . 459
8.69.3 ExtraValues . . . 459
8.69.4 Usage of barcode Attributes . . . 459

8.70 IDPrintingParams . . . 461
8.71 ImageCompressionParams . . . 461
8.71.1 ImageCompression . . . 462
8.71.2 CCITTFaxParams . . . 464
8.71.3 DCTParams . . . 464
8.71.4 FlateParams . . . 465
8.71.5 JBIG2Params . . . 465
8.71.6 JPEG2000Params . . . 465
8.71.7 LZWParams . . . 466
8.72 ImageEnhancementParams . . . 467
8.72.1 ImageEnhancementOp . . . 467
8.73 ImageReplacementParams . . . 467
8.74 ImageSetterParams . . . 468
8.74.1 Sides Attribute Values . . . 470
8.75 Ink . . . 470
8.76 InkZoneCalculationParams . . . 471
8.77 InkZoneProfile . . . 471
8.78 InsertingParams . . . 472
8.79 InterpretedPDLData . . . 473
8.80 InterpretingParams . . . 473
8.80.1 InterpretingDetails . . . 475
8.80.2 PDFInterpretingParams . . . 475
8.80.3 OCGControl . . . 476
8.80.4 ReferenceXObjParams . . . 476
8.80.5 More about PDFInterpretingParams . . . 477
8.81 JacketingParams . . . 477
8.82 LabelingParams . . . 478
8.83 LaminatingParams . . . 479
8.84 Layout . . . 479
8.84.1 LayerList . . . 483
8.84.2 LayerDetails . . . 483
8.84.3 LogicalStackParams . . . 483
8.84.4 Stack . . . 484
8.84.5 PageCondition . . . 484
8.84.6 PlacedObject . . . 485
8.84.7 ContentObject . . . 488
8.84.8 MarkObject . . . 488
8.84.9 CIELABMeasuringField . . . 490
8.84.10 ColorControlStrip . . . 490
8.84.11 DensityMeasuringField . . . 491
8.84.12 DynamicField . . . 492
8.84.13 FillMark . . . 493
8.84.14 MarkActivation . . . 493
8.84.15 SheetCondition . . . 494
8.84.16 More about Layout . . . 495

8.84.17 Signature . . . 499
8.85 LayoutElement . . . 499
8.85.1 ElementType Attribute Values . . . 501
8.85.2 Dependencies . . . 502
8.86 LayoutElementProductionParams . . . 502
8.86.1 LayoutElementPart . . . 504
8.86.2 BarcodeProductionParams . . . 505
8.86.3 PositionObj . . . 505
8.87 LayoutPreparationParams 511
8.88 PageDistributionScheme Attribute Values . . . 519
8.88.1 PageCell . . . 519
8.88.2 ImageShift . . . 520
8.89 LayoutShift . . . 524
8.89.1 ShiftPoint . . . 524
8.90 LongitudinalRibbonOperationParams . . . 525
8.91 ManualLaborParams . . . 525
8.92 Media . . . 526
8.92.1 MediaTypeDetails Attribute Values . . . 533
8.92.2 TabDimensions . . . 535
8.92.3 More about Media . . . 537
8.93 MediaSource . . . 540
8.94 MiscConsumable . . . 540
8.95 NodeInfo . . . 541
8.95.1 NodeStatus Attribute Values . . . 544
8.96 NumberingParams . . . 544
8.97 OrderingParams . . . 544
8.98 PackingParams . . . 544
8.99 PageAssignParams . . . 545
8.100 PageList . . . 545
8.100.1 PageData . . . 546
8.100.2 PageElement . . . 548
8.101 Pallet . . . 548
8.102 PalletizingParams . . . 549
8.103 PDFToPSConversionParams . . . 549
8.104 PDLCreationParams . . . 552
8.105 PDLResourceAlias . . . 552
8.106 PerforatingParams . . . 553
8.107 PlaceHolderResource . . . 553
8.108 PlasticCombBindingParams . . . 553
8.109 PlateCopyParams . . . 554
8.110 PreflightAnalysis . . . 554
8.111 PreflightInventory . . . 554
8.112 PreflightParams . . . 554
8.112.1 PreflightAction . . . 555

8.112.2 BasicPreflightTest . . . 555
8.112.3 PreflightArgument . . . 556
8.112.4 BoxArgument . . . 556
8.112.5 BoxToBoxDifference . . . 557
8.113 PreflightProfile . . . 558
8.114 PreflightReport . . . 558
8.114.1 PRItem . . . 558
8.114.2 PRError . . . 559
8.114.3 PRGroup . . . 559
8.114.4 Abstract PRGroupOccurrenceBase . . . 561
8.114.5 PRGroupOccurrenceBase . . . 562
8.114.6 ArgumentValue . . . 562
8.114.7 PRGroupOccurrence . . . 562
8.114.8 StringListValue . . . 562
8.114.9 PROccurrence . . . 562
8.115 PreflightReportRulePool . . . 563
8.115.1 PRRule . . . 563
8.115.2 PRRuleAttr . . . 563
8.116 Preview . . . 565
8.117 PreviewGenerationParams . . . 566
8.118 PrintCondition . . . 568
8.119 PrintRollingParams . . . 569
8.120 ProductionPath . . . 569
8.120.1 FolderSuperstructureWebPath . . . 570
8.120.2 PostPressComponentPath . . . 570
8.120.3 PrintingUnitWebPath . . . 570
8.121 ProofingParams . . . 571
8.122 PSToPDFConversionParams . . . 571
8.122.1 AdvancedParams . . . 573
8.122.2 PDFXParams . . . 574
8.122.3 ThinPDFParams . . . 576
8.123 QualityControlParams . . . 576
8.123.1 BindingQualityParams . . . 577
8.124 QualityControlResult . . . 577
8.124.1 QualityMeasurement . . . 578
8.124.2 BindingQualityMeasurement . . . 578
8.125 RasterReadingParams . . . 578
8.126 RegisterMark . . . 579
8.127 RenderingParams . . . 580
8.127.1 TIFFFormatParams . . . 581
8.127.2 TIFFtag . . . 582
8.127.3 TIFFEmbeddedFile . . . 582
8.128 ResourceDefinitionParams . . . 582
8.128.1 ResourceParam . . . 583

8.129 RingBindingParams . . . 583
8.130 RollStand . . . 584
8.131 RunList . . . 584
8.131.1 Pages, Documents and Sets for common PDL types . . . 589
8.131.2 DynamicInput . . . 589
8.132 SaddleStitchingParams . . . 591
8.133 ScanParams . . . 591
8.134 ScavengerArea . . . 592
8.135 ScreeningParams . . . 593
8.136 SeparationControlParams . . . 593
8.137 Shape . . . 593
8.138 ShapeCuttingParams . . . 594
8.139 ShapeDef . . . 595
8.139.1 CutLines . . . 596
8.140 ShapeDefProductionParams . . . 596
8.140.1 ObjectModel . . . 597
8.140.2 ShapeTemplate . . . 597
8.141 Sheet . . . 599
8.142 SheetOptimizingParams . . . 599
8.142.1 GangElement . . . 599
8.142.2 SeparationListBack . . . 601
8.142.3 SeparationListFront . . . 601
8.143 ShrinkingParams . . . 601
8.144 SideSewingParams . . . 602
8.145 SpinePreparationParams . . . 602
8.145.1 Operations Attribute Values . . . 603
8.146 SpineTapingParams . . . 603
8.147 StackingParams . . . 604
8.148 StaticBlockingParams . . . 607
8.149 StitchingParams . . . 607
8.150 Strap . . . 610
8.151 StrappingParams . . . 610
8.152 StripBindingParams 611
8.153 StrippingParams . . . 612
8.153.1 Position . . . 615
8.153.2 StripCellParams . . . 616
8.153.3 StripMark . . . 618
8.154 Surface . . . 622
8.155 ThreadSealingParams . . . 622
8.156 ThreadSewingParams . . . 622
8.157 Tile . . . 624
8.158 Tool . . . 625
8.159 TransferCurve . . . 626
8.160 TransferCurvePool . . . 626

8.160.1 TransferCurveSet . . . 627
8.161 TransferFunctionControl . . . 627
8.162 TrappingDetails . . . 628
8.162.1 TrappingOrder . . . 628
8.163 TrappingParams . . . 629
8.163.1 ColorantZoneDetails . . . 631
8.164 TrapRegion . . . 631
8.165 TrimmingParams . . . 632
8.166 UsageCounter . . . 633
8.167 VarnishingParams . . . 633
8.168 VerificationParams . . . 634
8.169 WebInlineFinishingParams . . . 635
8.169.1 FolderProduction . . . 635
8.170 WindingParams . . . 635
8.171 WireCombBindingParams . . . 636
8.172 WrappingParams . . . 637

Chapter Subelements . . . 639

9.1 Address . . . 639
9.2 AutomatedOverPrintParams . . . 639
9.3 BarcodeCompParams . . . 640
9.4 BarcodeReproParams . . . 640
9.5 Certification . . . 641
9.6 ColorantAlias . . . 641
9.7 ColorCorrectionOp . . . 642
9.8 ColorMeasurementConditions . . . 643
9.9 ColorSpaceConversionOp . . . 644
9.9.1 SourceCS Attribute Values . . . 647
9.10 ComChannel . . . 649
9.10.1 ChannelTypeDetails Attribute . . . 650
9.11 Comment . . . 651
9.12 ConvertingConfig . . . 652
9.13 CostCenter . . . 653
9.14 Crease . . . 653
9.15 Cut . . . 654
9.16 DeviceMark . . . 655
9.17 DeviceNSpace . . . 658
9.18 Disjointing . . . 658
9.19 Disposition . . . 659
9.20 FitPolicy . . . 660
9.21 Fold . . . 661
9.22 GangSource . . . 661
9.23 GeneralID . . . 662
9.24 GlueLine . . . 663

9.25 Hole . . . 664
9.26 HoleLine . . . 664
9.27 InsertSheet . . . 665
9.27.1 SheetUsage Attribute Values . . . 668
9.28 JobField . . . 669
9.29 MarkColor . . . 669
9.30 MediaLayers . . . 669
9.31 MetadataMap . . . 670
9.31.1 Expr . . . 671
9.32 MISDetails . . . 674
9.33 ObjectResolution . . . 674
9.34 Perforate . . . 675
9.35 Person . . . 676
9.36 RefAnchor . . . 676
9.37 RegisterRibbon . . . 677
9.38 ScreenSelector . . . 678
9.39 SeparationSpec . . . 679

Chapter Device Capabilities . . . 681

10.1 Capability and Constraint Definitions . . . 681
10.2 Device Capability Definitions . . . 681
10.2.1 DeviceCap . . . 682
10.2.2 ActionPool . . . 685
10.2.3 DevCapPool . . . 686
10.2.4 ModulePool . . . 686
10.2.5 DevCaps . . . 687
10.2.6 DevCap . . . 689
10.2.7 State . . . 691
10.2.8 DisplayGroupPool . . . 705
10.2.9 FeaturePool . . . 705
10.2.10 MacroPool . . . 706
10.2.11 Performance . . . 708
10.2.12 TestPool . . . 709
10.2.13 Term . . . 709
10.2.14 Examples of Device Capabilities . . . 720
10.3 Concept of the Preflight Process . . . 726
10.3.1 Object Classes . . . 727
10.3.2 Properties . . . 730

Chapter Building a System . . . 749

11.1 Implementation Considerations and Guidelines . . . 749
11.2 JDF and JMF Interchange Protocol . . . 749
11.2.1 File-Based Protocol (JDF) . . . 749
11.2.2 HTTP-Based Protocol (JDF + JMF) . . . 749
11.2.3 HTTPS-Based Protocol – SSL with two-way authentication . . . 749

11.2.4 Managing Persistent Channels. . . . 752
11.2.5 Deleting Persistent Channels . . . 752
11.3 JDF Packaging . . . 752
11.3.1 MIME Basics . . . 752
11.3.2 MIME Types and File Extensions . . . 753
11.4 MIS Requirements . . . 756
11.5 Interoperability Conformance Specifications . . . 756

Appendix Data Types and Values . . . 759

A.1 Notes About Encoding . . . 759
A.1.1 List, Range and Range List Data Types . . . 759
A.1.2 Whitespace . . . 759
A.1.3 Infinity Limits . . . 759
A.2 Simple Types — Attribute ValuesXJDF Data Types . . . 759
A.2.1 boolean . . . 759
A.2.2 CMYKColor . . . 759
A.2.3 date . . . 760
A.2.4 dateTime . . . 760
A.2.5 DateTimeRange . . . 760
A.2.6 DateTimeRangeList . . . 760
A.2.7 double . . . 761
A.2.8 DoubleList . . . 761
A.2.9 DoubleRange . . . 761
A.2.10 DoubleRangeList . . . 761
A.2.11 duration . . . 761
A.2.12 DurationRange . . . 762
A.2.13 DurationRangeList . . . 762
A.2.14 gYearMonth . . . 762
A.2.15 hexBinary . . . 762
A.2.16 ID . . . 762
A.2.17 IDREF . . . 763
A.2.18 IDREFS . . . 763
A.2.19 integer . . . 763
A.2.20 IntegerList . . . 763
A.2.21 IntegerRange . . . 763
A.2.22 IntegerRangeList . . . 764
A.2.23 LabColor . . . 764
A.2.24 language . . . 764
A.2.25 languages . . . 764
A.2.26 matrix . . . 765
A.2.27 NameRange . . . 765
A.2.28 NameRangeList . . . 765
A.2.29 NMTOKEN . . . 765
A.2.30 NMTOKENS . . . 766
A.2.31 PDFPath . . . 766

A.2.32 rectangle . . . 766
A.2.33 RectangleRange . . . 766
A.2.34 RectangleRangeList . . . 767
A.2.35 regExp . . . 767
A.2.36 shape . . . 767
A.2.37 ShapeRange . . . 767
A.2.38 ShapeRangeList . . . 767
A.2.39 sRGBColor . . . 768
A.2.40 string . . . 768
A.2.41 TimeRange . . . 768
A.2.42 TransferFunction . . . 768
A.2.43 URI . . . 768
A.2.44 URL . . . 769
A.2.45 XPath . . . 769
A.2.46 XYPair . . . 769
A.2.47 XYPairRange . . . 770
A.2.48 XYPairRangeList . . . 770
A.3 Enumerations . . . 770
A.3.1 Action . . . 770
A.3.2 Anchor . . . 770
A.3.3 Automation . . . 771
A.3.4 Axis . . . 771
A.3.5 BinderMaterial . . . 771
A.3.6 ChannelMode . . . 772
A.3.7 Coating . . . 772
A.3.8 Compensation . . . 772
A.3.9 Drying . . . 772
A.3.10 Edge . . . 773
A.3.11 EmbossDirection . . . 773
A.3.12 EmbossLevel . . . 773
A.3.13 EmbossType . . . 773
A.3.14 FeedQuality . . . 774
A.3.15 FitPolicy . . . 774
A.3.16 GangPolicy . . . 774
A.3.17 Glue . . . 775
A.3.18 IncludeResources . . . 775
A.3.19 ISOPaperSubstrate . . . 775
A.3.20 JDFJMFVersion . . . 775
A.3.21 MappingSelection . . . 776
A.3.22 NamedColor . . . 776
A.3.23 Opacity . . . 776
A.3.24 Orientation . . . 777
A.3.25 Polarity . . . 777
A.3.26 PositionPolicy . . . 777
A.3.27 RenderingIntent . . . 777

A.3.28 Scope . . . 778
A.3.29 Severity . . . 778
A.3.30 SheetLay . . . 778
A.3.31 Side . . . 778
A.3.32 Sides . . . 778
A.3.33 SourceObjects . . . 779
A.3.34 StripMaterial . . . 779
A.3.35 TightBacking . . . 779
A.3.36 Usage . . . 780
A.3.37 WorkingDirection . . . 780
A.3.38 WorkStyle . . . 780
A.3.39 XYRelation . . . 780
A.4 Preferred String and NMTOKEN Values . . . 781
A.4.1 Comb and Coil Shape Names . . . 781
A.4.2 DeviceClass and ModuleType Supported Strings . . . 781
A.4.3 Input Tray and Output Bin Names . . . 786
A.4.4 Media Coating Names . . . 787
A.4.5 Milestone Values . . . 788
A.4.6 NotificationDetails . . . 790
A.4.7 StatusDetails Supported Strings . . . 793
A.5 JDF File Formats . . . 798
A.5.1 PNG Image Format . . . 798

Appendix Schema . . . 799

B.1 Using xsi:type . . . 799
B.1.1 Using xsi:type with JDF Nodes . . . 799
B.1.2 Using xsi:type with JMF Messages . . . 800

Appendix Return Values and Error Codes . . . 801

Appendix Color Adjustment . . . 805

D.1 Adjustment Using Direct Attributes . . . 805
D.2 Adjustment using ICC Profile Attributes . . . 806
D.3 Adjustment using an ICC Abstract Profile Attribute . . . 806
D.4 Adjustment using an ICC DeviceLink Profile Attribute . . . 806

Appendix Paper Weight and Grade . . . 807

E.1 North American Media Weight . . . 807
E.2 Japanese Media Weight . . . 808
E.3 Paper Grade . . . 809

Appendix Media Sizes . . . 811

Appendix MimeTypes . . . 815

Appendix Dynamic String Generation . . . 821

Appendix Pagination Catalog . . . 825

I.1 How to interpret the diagrams . . . 825
I.1.1 Legend . . . 825
I.1.2 Meaning of a Pagination Scheme . . . 826
I.1.3 Settings that Modify the Pagination Schemes . . . 826
I.1.4 Getting a Specific Pagination Scheme . . . 827
I.1.5 Examples . . . 829
I.2 Pagination Diagrams . . . 831

Appendix Resolving Directory URL References . . . 863

J.1 Semantics of the RunList/@Directory Attribute . . . 863

Appendix References . . . 865

Appendix Hole Pattern Catalog . . . 875

L.1 Naming Scheme: . . . 876
L.2 Ring Binding - Two Hole . . . 876
L.3 Ring Binding - Three Hole . . . 878
L.4 Ring Binding - Four Hole . . . 878
L.5 RingBinding - Five Hole . . . 880
L.6 Ring Binding - Six Hole . . . 880
L.7 Ring Binding - Seven Hole . . . 882
L.8 Ring Binding - Eleven Hole . . . 882
L.9 Plastic Comb Binding . . . 883
L.10 Wire Comb Binding . . . 883
L.11 Coil and Spiral Binding . . . 884
L.12 Special Binding . . . 884

Appendix FileSpec Use Cases . . . 885

M.1 Examples of Attribute Values of FileSpec . . . 885
M.2 Corresponding XML examples . . . 886
M.3 Additional examples showing Partitioning of FileSpec . . . 888
M.4 Example of an Intent Job Ticket with a doubly nested ZIP packaging file . . . 895
M.5 AppOS and OSVersion Attributes . . . 896

Appendix Deprecated Items . . . 899

N.1 Deprecated Structures of JDF Nodes and Jobs . . . 899
N.1.1 PlaceHolder Resource . . . 899
N.1.2 ResourceUpdate . . . 899
N.1.3 StatusPool . . . 899
N.2 Lot . . . 900
N.3 Life Cycle of JDF . . . 901
N.3.1 Case 5: Spawning and Merging of Independent Jobs . . . 901
N.4 JMF Messaging Elements . . . 903
N.4.1 Signal . . . 903
N.4.2 Events . . . 904
N.4.3 KnownControllers . . . 905

N.4.4 RepeatMessages . . . 906
N.4.5 NodeInfo . . . 908
N.4.6 KnownJDFServices . . . 910
N.4.7 Occupation . . . 911
N.4.8 Track . . . 912
N.4.9 QueueEntryStatus . . . 913
N.5 Deprecated Processes . . . 914
N.5.1 DBDocTemplateLayout . . . 914
N.5.2 DBTemplateMerging . . . 914
N.5.3 FormatConversion . . . 915
N.5.4 Ordering . . . 915
N.5.5 Packing . . . 916
N.5.6 FilmToPlateCopying . . . 916
N.5.7 PreflightAnalysis . . . 916
N.5.8 PreflightInventory . . . 918
N.5.9 PreflightProfile . . . 919
N.5.10 Proofing . . . 920
N.5.11 SoftProofing . . . 920
N.5.12 IDPrinting . . . 921
N.5.13 AdhesiveBinding . . . 922
N.5.14 Dividing . . . 923
N.5.15 LongitudinalRibbonOperations . . . 923
N.5.16 Numbering . . . 923
N.5.17 SaddleStitching . . . 924
N.5.18 SideSewing . . . 924
N.6 Deprecated Intents . . . 924
N.6.1 BindingIntent Deprecated Subelements . . . 924
N.6.2 DeliveryIntent Deprecated Subelements . . . 925
N.6.3 NumberingIntent . . . 926
N.6.4 SizeIntent . . . 927
N.7 Deprecated Parameters . . . 928
N.7.1 AdhesiveBindingParams . . . 928
N.7.2 BoxFoldingParams Deprecated Subelements . . . 929
N.7.3 CustomerMessage . . . 929
N.7.4 DBMergeParams . . . 930
N.7.5 DBRules . . . 930
N.7.6 DBSchema . . . 930
N.7.7 DBSelection . . . 931
N.7.8 DividingParams . . . 931
N.7.9 FormatConversionParams . . . 931
N.7.10 IDPrintingParams . . . 932
N.7.11 OutputBin Attribute Values . . . 934
N.7.12 Layout Deprecated Subelement . . . 942
N.7.13 LongitudinalRibbonOperationParams . . . 942
N.7.14 MediaSource . . . 943

N.7.15 NumberingParams 944
N.7.16 NumberingParam 944
N.7.17 OrderingParams 944
N.7.18 PackingParams 945
N.7.19 PlaceHolderResource 946
N.7.20 PlateCopyParams 946
N.7.21 ProofingParams 946
N.7.22 RunList Deprecated Subelements 947
N.7.23 SaddleStitchingParams 948
N.7.24 Sheet 948
N.7.25 SideSewingParams 949
N.7.26 Surface 950

Appendix List of Figures 951

Appendix List of Tables 957

Preface

This specification is immense ... there is little doubt about that ... but it is also a keystone standard for the future of graphic communications. The members of CIP4 believe that users and developers alike need to have a clear understanding of what the objectives of the Job Definition Format (**JDF**) are as well as an understanding of its value and purpose. To that end we thought you would find a "non-standard" preface and user overview helpful.

Before we get into the overview, we remind you that **JDF** is a living specification. We would value your comments and input. There are several ways to contact the International Cooperation for the Integration of Processes in Prepress, Press and Postpress (CIP4) and to receive ongoing information about CIP4 activities. To get a list of contacts, join the **JDF** developers form, or sign up for Email updates, visit the contact page at http://www.cip4.org/. *(Of course, we'd love to have you as a CIP4 member too! Be sure to review the membership page when you visit the CIP4 Website.)*

You will also find callouts throughout this document that are identified by three different icons. These callouts, provided for your convenience, are not normative parts of the standard (i.e., they're not technically a part of the *standard*). They provide references to external sources, executive summaries of complex technical concepts, and some thoughts or strategies to consider as you formulate your **JDF** implementation plan. Look for these callout icons

Table 2.1: Callout Icon Usage

ICON	CALLOUT TYPE
	External references to online resources, related standards, tutorials and helpful information.
	Executive-style summaries of technical concepts in easy-to-understand language.
	Thoughts to ponder and strategy ideas for formulating JDF implementation programs.

Value. This revision of **JDF** is significant because it builds upon the fifth version of **JDF** (v.1.4) to deliver a fully functional and mature standard. As such, this revision includes elements from which executives, shop managers and technicians will all benefit equally, though in different ways. In the next few years it is our belief that this specification will positively effect everyone involved in the creation and production of printing; regardless of form (offset, digital, flexographic and so on) or function (direct mail, periodical publication, packaging and so on). Furthermore, **JDF** will be of value to companies both large and small. Some of the benefits that **JDF** provides include:

- A common language for describing a print job across enterprises, departments and software and systems;
- A tool for verifying the accuracy and completeness of job tools;
- A systems interface language that can be used to benchmark the performance of new equipment (hardware and software) and that can reduce the cost of expensive custom integration for printers, prepress services and others;
- A basis for total workflow automation that incorporates all aspects of production: human, machine and computer;
- A standard that can be applied to eliminate wasteful re-keying and redundancy of information; and
- A common computer language for printing and related industries as well as a platform for more effective communication.

Implementation Strategy

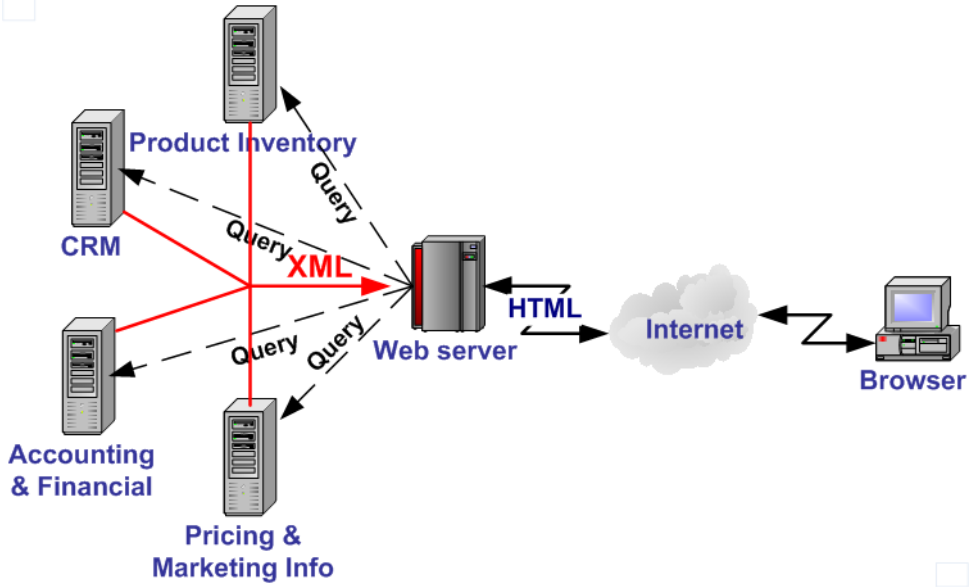
As you read this standard, consider how to make JDF a part of your equipment evaluation and purchasing procedures. Do you add JDF enabled systems slowly with equipment replacement and upgrades, or aggressively as part of a plant re-engineering process? What's your desired competitive position?

Most importantly, **JDF** provides an opportunity for users of graphic arts equipment to get a better return on their technology investment and an opportunity to create a print production and distribution workflow that is more competitive with broadcast media in terms of time-to-market.

XML and Schema: Why? The Extensible Markup Language (XML) is the standard language that is employed by **JDF** is also constructed to the World Wide Web Consortium's (W3C) recommendation for the construction of schema. Why is this important and, in layman's terms, what does it do for you?

First of all, it is helpful to understand how MIS professionals around the world use XML today. Although there are some systems that manage and process XML directly, it is primarily used as an exchange language or "middleware" element to create the "glue" that ties integrated systems together.

For instance, complex systems such as enterprise resource planning (ERP), data warehousing or E-commerce systems often tap into numerous legacy databases and application environments. A manager might wish to have a "view" of corporate information that is actually an aggregate of information that might come from various sources such as billing and invoicing, sales management, inventory and other systems. Rather than merge these systems into a single, monstrous and centralized system, an operator queries the legacy systems and the results are wrapped in XML. This allows programmers to deal with one exchange language or data format instead of a multitude of proprietary data formats.



XML is not a *functional* computer language like JAVA, C++ or FORTRAN—it is incapable of manipulating data in anyway; rather, it is a *descriptive* computer language that can be used to describe your information including its structure, interrelationships, and to some extent, its intended usage. For this reason, modern program languages such as JAVA provide intrinsic support for XML processing. Most modern database applications also provide methods for receiving and delivering XML.

Early XML, based solely upon the XML 1.0 specification, had a few limitations that prevented it from being used widely as a transactional data format *across* enterprises, as opposed to *within* enterprises (where it found its niche as described above). For example, there is probably a database behind each of your major systems and applications. If your database has reserved a fixed space a data particular field and a supplier provides a transaction with a data element larger than that field, you have a problem. The data limitations of XML 1.0 cannot effectively deal with this. The XML Schema specification solved this problem and others.

XML Schema

To learn more about XML Schema, including tools, usage, tutorials and other resources visit http://www.w3.org/XML/Schema

The Plusses of Parsing. Schemas also provide one other feature that is perhaps the greatest benefit. Tagged documents or transactions (called "instances" in XML parlance) are *parsible*. Schemas, such as **JDF**, establish rules for structuring your information. A parser is a software application that reads those rules, checks documents and transactions, and then validates that they conform to the rules as established in your schema ... sort of like preflighting but for XML instances rather than your layout pages.

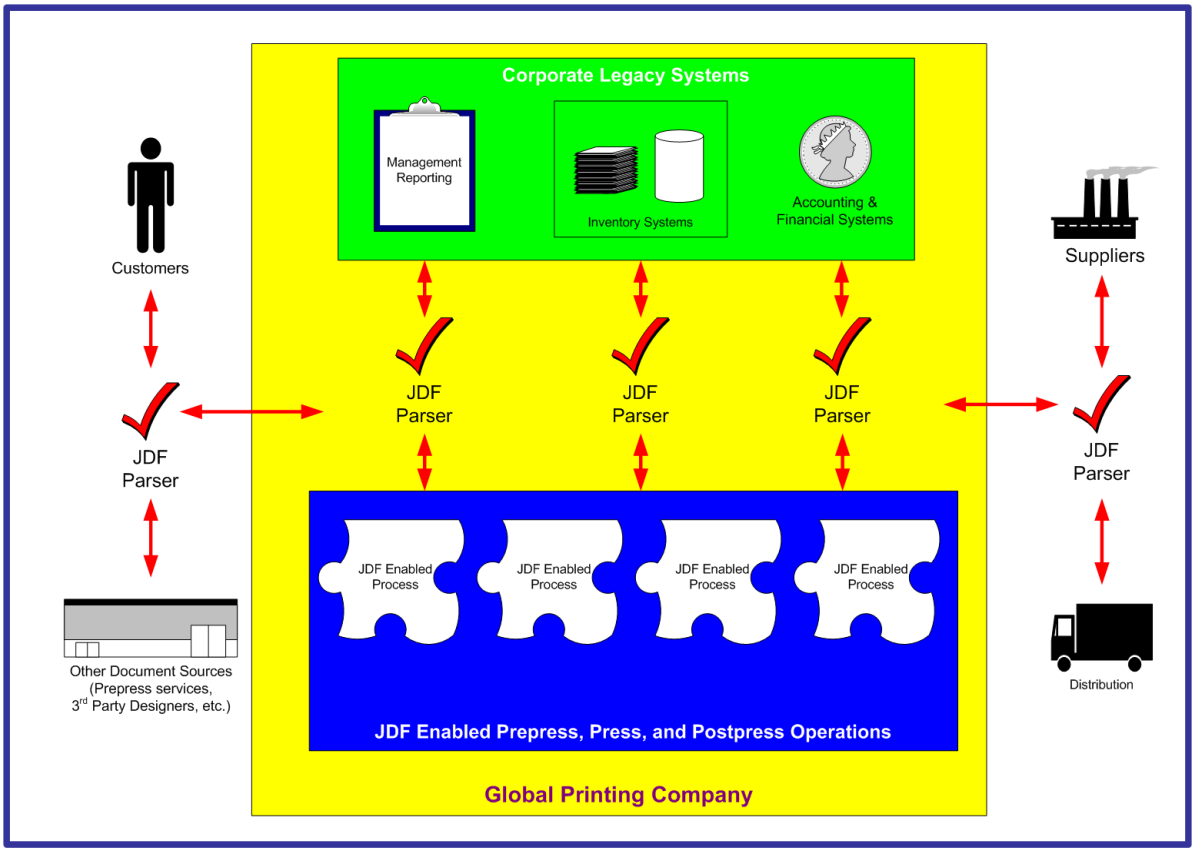
Parsers can play many roles. Like preflighting software, parsers can be run as stand-alone applications, but they can also be found embedded into other applications. Some of the roles parsers can play in your **JDF**-enabled workflow include:

Free Parsers

The JDF schema was validated with the Xerces parser. This parser, as well as other XML tools, is available for free from The Apache Software Foundation open source software community at http://xml.apache.org/

1 Acceptance checking of client job tickets;
2 Validation of **JDF** prior to or following transformation of data into and out of databases;
3 Ensuring that source job information is collected as a document is created (embedded in document layout software);
4 Determining if equipment reads and writes Job Messaging Format (**JMF**) commands, a subset of **JDF**, as part of equipment benchmarking and testing software;
5 Controlling the movement of workflow information and controls within workflow software from process to process and as a specific **JDF** job ticket requires; and
6 Working as a middleware component to communicate between **JDF**-enabled software and systems and your legacy Management Information System (MIS) and corporate applications environments.

It is worth mentioning that parsing can be time consuming and computer intensive. But parsers don't have to be the gatekeepers everywhere in a **JDF**-enabled workflow. Equipment that is **JDF**-enabled and part of a company's internal production operations need not parse every communication. It can be limited to equipment evaluation and problem solving applications. The role of **JDF** parser-enabled software in a printing plant that uses tightly coupled **JDF**-enabled print production equipment might look like this:



The JDF Concept. The **JDF** specification is quite complex and detailed—something best left to programmers and XML experts. But the concepts behind **JDF** are quite simple and straightforward. It provides an explanation of each of the components of **JDF**, its meaning and intended usage. You will want to use the components of **JDF** that fit best with your workflow and the needs of your customers. To start, a basic understanding of the concepts behind **JDF** is in order. There are four primary components that comprise the **JDF** environment:

1 JDF itself,

2 The Job Messaging Format (JMF)

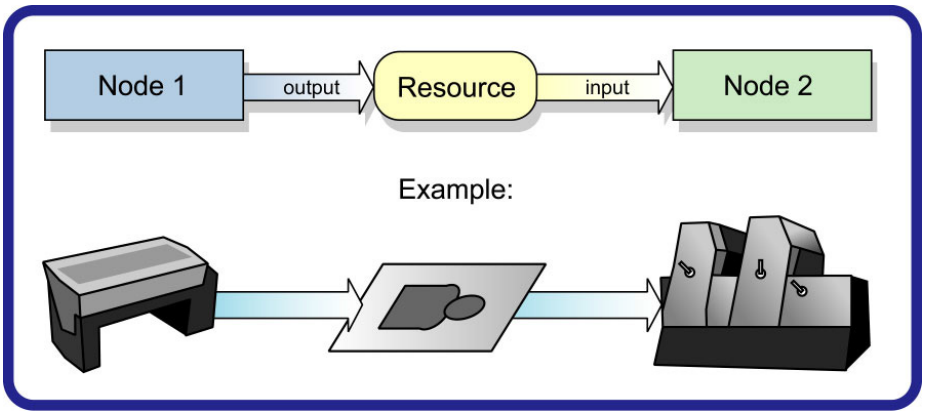
3 JDF Capabilities and

4 ICS Documents

JDF is an exchange format for instructions and job parameters. You can use PDF or its standard variant (PDF/X), to relay content data files from one platform to another. You can do the same with **JDF** to relay job parameters and instructions. **JDF** can be used to describe a printing job logically, as you would in exchanging a job description with a client within an estimate. It can also be used to describe a job in terms of individual production processes and the materials or other process inputs needed to complete a job.

There is no such thing as a standard print workflow. In fact, printing is the ultimate form of *flexible manufacturing*. This makes process automation quite a challenge for our industry. What you'll find in this standard are XML element definitions that describe all the production processes and material types you're likely to encounter, regardless of your workflow. These are the building blocks that you can use to emulate your workflow with **JDF**. Every process in the print production workflow requires input *resources* starting with the client's files or artwork and ending with the final bound, packaged and labeled print product. For example, before you can print, you need paper, ink and plates, and before you can send a document to a bindery line, you need printed and cut signatures.

Process *nodes* and *resources* are the basic elements within **JDF**. They can be strung together to meet the requirements of each job. The output of one process becomes the input of the following process, and a process doesn't begin until its input resources are available:



This specification provides details on how to use these building blocks to describe concurrent processes, spawned processes, dynamic processes and so on. To realize the capabilities of **JDF**, there are two other things you will need: a way of controlling the flow of process and a way of communicating commands to equipment on the shop floor. **JMF** is a subset of **JDF** that handles communication with equipment on the shop floor. This might include major equipment, such as platesetters or subsystems, such as in-line color measurement devices. **JMF** can be used to establish a queue, discover the capabilities of a **JDF**-enabled device, determine the status of a device (e.g., "RIPing," "Idle" and so on).

JMF

The Job Messaging Format (JMF) functions as a standard interface between your equipment and your information systems or other equipment already on the shop floor. By buying only equipment that supports JMF you will reduce the cost and complexity of integrating new equipment into your production operations, and you will improve the flexibility and adaptability of your shop.

Although, theoretically, you can string together equipment that supports **JMF** directly to one another, in almost all cases you will want your production equipment to communicate with your MIS or Production Control system. This way it is the MIS system that controls the scheduling, execution and control of work in progress. The role of the MIS system is described within this standard, but it isn't highly defined. In fact, the **JDF** standard does not dictate how a **JDF** system is to be built. Many printers, prepress services and other graphic arts shops will already have MIS systems in place. **JDF** enabled workflow and MIS systems, custom-tailored to print production requirements, will soon be available on the market. However, many printers already have MIS and workflow systems that have been customized or developed for their own environments. In most cases these legacy systems can be modified to work with the new **JDF** workflows and **JDF** enabled equipment.

Changes to JDF 1.5

JDF 1.5 includes both some wholly new material, as well as many improvements and refinements to **JDF** 1.4. A complete catalog of changes can be found in ▸ Appendix P New, Deprecated & Modified Items. You will also find New in JDF 1.5, Deprecated in JDF 1.5 and Modified in JDF 1.5 flags throughout this document.

The following list gives a high level overview of the areas that have experienced major revisions:

- Job Ganging and **SheetOptimizing**
- ***ResourceLink*** elements are now identical for all resources and contain the sum of all traits of the previously defined ***ResourceLink*** sub-classes. Note that this does not invalidate any previously valid ***ResourceLink*** instances. See ▸ Section 3.9.2 ResourceLink.
- Enhanced control of variable finishing devices using barcodes, (***IdentificationField*** elements) or **JMF** Pipe messages (See ▸ Section 4.3.3 Overlapping Processing Using Pipes).
- The normative language has been adjusted to match the style of ISO specifications. Thus, all normative requirements have been renamed from "MUST" to "SHALL". See ▸ Section 1.5.1 Conformance Terminology.
- The Chapter that describes individual resources (Chapter 7 in previous versions of the specification) has been split into the 5 new Chapters 7-11.
- A new ***Activity*** element, added to ***DeviceInfo*** and ***JobPhase*** in **JMF** and ***PhaseTime Audit***
- A new **Winding** process

ICS Documents and Certification

The concept of Interoperability Conformance Specification of "ICS" documents is introduced. No single device (i.e., printer, press, imagesetter, etc.) is likely to implement all that the **JDF** specification provides for. For instance, if you are in the digital printing business, you might not care to process data used for case binding. A RIP is not a requirement for facilitating **JDF** preflighting. A Stitcher probably doesn't need to handle image rendering data.

To specify exactly what individual classes of devices need to do with **JDF** CIP4 members are developing ICS document that will provide the minimum expectations for individual classes of devices. ICS documents will later be used as the ba-

sis for certification testing. Once the certification program begins, you will start seeing products that are marked as "JDF Certified" and this will be certification to identified levels of one or more specific ICS documents. An initial set of ICS documents is freely available to the public, and we expect that they will become part of your buying practices. ICS documents for additional classes of devices are currently under development.

Introduction

This document defines the technical specification for the Job Definition Format (**JDF**) and its counterpart, the Job Messaging Format (**JMF**).

We will describe the components of **JDF**, both internal and external, and explain how to integrate the format components to create a viable workflow. Ancillary aspects are also introduced, such as how to convert PJTF or PPF to **JDF**, and how **JDF** relates to IfraTrack. It is intended for use by programmers and systems integrators for operations addressed by the International Cooperation for Integration of Processes in Prepress, Press and Postpress (CIP4). In this first chapter, we present the concept of **JDF**, and its relationship to **JDF** and other industry standards. and how to use this document and some basic document navigational aids.

1.1 Further Information

Additional information such as application notes and examples can be found on the CIP4 website at http://www.cip4.org.

-
-
-
-
-
-
-

1.2 Background on JDF

JDF is an extensible, XML-based format built upon the existing technologies of CIP3's Print Production Format ▸ [CIP3 - PPF] and Adobe's Portable Job Ticket Format ▸ [PJTF]. It provides three primary benefits to the printing industry:

1. The ability to unify the prepress, press and postpress aspects of any printing job, unlike any previous format;
2. The means to bridge the communication gap between production services and Management Information Systems (▸ MIS); and
3. The ability to carry out both of these functions no matter what system architecture is already in place and no matter what tools are being used to complete the job. In short, **JDF** is extremely versatile and comprehensive.

JDF is an interchange data format to be used by a system of administrative and implementation-oriented components, which together produce printed products. It provides the means to describe print jobs in terms of the products eventually to be created, as well as in terms of the processes needed to create those products. The format provides a mechanism to explicitly specify the controls needed by each process, which might be specific to the devices that will execute the processes.

JDF works in tandem with a counterpart format known as the Job Messaging Format or **JMF**. **JMF** provides the means for production components of a **JDF** workflow to communicate with system controllers and administrative components. It relays information about the progress of **JDF** jobs and gives ▸ MIS the active ability to query devices about the status of processes being executed or getting ready to be executed. **JMF** will provide the complete job tracking functionality that is defined by IfraTrack messaging standard. Depending on the system architecture, **JMF** might also provide the means to control certain aspects of these processes directly.

JDF and **JMF** are maintained and developed by CIP4 (http://www.cip4.org). They were originally developed by four companies prominent in the graphic arts industry—Adobe, Agfa, Heidelberg and MAN Roland — with significant contributions provided by CIP3, the IfraTrack working group, Fraunhofer IGD and the PrintTalk consortium.

1.3 Conventions Used in This Specification

This section contains conventions and notations used within this document.

1.3.1 Document References

Throughout this specification references to other documents are indicated by short symbolic names inside square brackets (e.g., ▸ [ICC.1]). ▸ Appendix K References lists all such references, with their full title, date, source and availability.

1.3.2 Text Styles

The following text styles are used to identify the components of a **JDF** job.

- Elements and resources are written in ***blue san serif bold italic***.
Examples are ***Comment***, ***BundleItem*** and ***ResourceLink*** elements, and ***Ink***, ***Layout*** and ***ExposedMedia*** resources.
- Attributes are written in dark *blue san serif italic*.
Examples are @*Status*, @*ResourceID* and @*ID*.
- Process types are written in **purple san serif bold**.
Examples are **ColorSpaceConversion** and **Rendering**.
- Enumerative and boolean values of attributes are written in red san serif.
Examples are "true", "Waiting", "Completed" and "Stopped".
- Standard **bold text** is used to highlight; glossary items, defined items inside a table and definitions of local terms.
- Internal cross-reference links are denoted by a red triangle followed by gray text.
An example is 'see ▸ Section 1.3.2 Text Styles'.
- External hyperlinks are denoted by blue underlined text.
An example is http://www.cip4.org.

Extended Backus-Naur Form

The Extended Backus-Naur Form (EBNF) provides a compact notation that is commonly used in the specifications of programming languages. The official EBNF standard, ▸ [ISO14977:1996], is available from ISO.

1.3.3 XPath Notation Used in this Specification

New in JDF 1.2

A simple subset of the XPath Language ▸ [XPath] is used throughout this specification in the description of an element, attribute or value to identify other elements, attributes and/or values. XPath gets its name from its use of a path notation (as in URLs) for navigating through the hierarchical structure of an XML document. The simple subset of XPath used is:

- Element ▸ Subelement hierarchy is indicated by a slash (e.g., "***Element***/***Element***").
- Element attribute hierarchy is indicated by a slash and an at (@) symbol) (ex., "***Element***/@*Attribute*")
- The text styles above in Section 1.3.1 are used to indicate whether an element is a resource, process or other element, or if the subject is an attribute or a value (ex., enumeration, string, etc.).
- Paths beginning with a single slash: "/" indicate root elements: (ex. **/JDF** indicates the root ▸ JDF Node).
- Paths beginning with a double slash "//" indicate elements with any parent (ex. //***GeneralID*** indicates a ***GeneralID*** element in any element).
- Paths containing square brackets that enclose a predicate describe an element that is restricted by the predicate. This document uses three types of predicates as described in the next 3 items:
  - E[@*A*="V"] – the XPath specifies an element ***E*** whose attribute *A* has the value V (e.g., ***Component***[@*ComponentType* = "FinalProduct"] specifies a ***Component*** whose @*ComponentType* has the value of "FinalProduct"). The predicate can be used outside the context of an element (e.g., @*ComponentType* = "FinalProduct" (or without the "@": *ComponentType* = "FinalProduct") means that the @*ComponentType* value equals "FinalProduct").
  - E[contains(@*A*="V")] – the XPath specifies an element ***E*** whose attribute *A* has some value that contains V, *A*'s value is either an enumerations or NMTOKENS and V is an enumeration or NMTOKEN. For example, ***Contact***[contains(@*ContactTypes* = "Delivery"]/***Address*** specifies the ***Address*** of a ***Contact*** whose @*ContactTypes* value contains "Delivery" and possibly other NMTOKEN values. The predicate can be used outside the context of an element (e.g., contains(@*ContactTypes* = "Delivery" means that @*ContactTypes* value contains "Delivery").
  - E[@*A*] – the XPath specifies an element ***E*** in which attribute *A* is present (e.g., ***Layout***[@*Side*] specifies a ***Layout*** in which the @*Side* attribute is present).

Example 1.1: XPath Expression

The XPath expression:

- ***Layout***/***MarkObject***/***DynamicField***/@*Format* = "Replacement Text for %s and %s go in here at %s on %s" ...

Means:

- The value "Replacement Text for %s and %s go in here at %s on %s" of the @*Format* attribute of the ***DynamicField*** ▸ Subelement of the ***MarkObject*** element of the ***Layout*** resource element.

Locally, (and within context), just the basic attribute dependency is noted (for instance — ***DynamicField***/@*Format*) where the discussion occurs within the section describing the element (e.g., ***DynamicField*** in our example) elements or the element's immediate parent (e.g., ***MarkObject*** in our example).

1.3.4 Modification Notes

New in JDF 1.2

To help the reader familiar with earlier versions of **JDF**, this specification indicates additions, deprecations and clarifications using the callouts described in ▸ Table 1.1 Modification Notes. Please note that not all changes are identified with

modified callout flags. When modification occurs in multiple versions, only the most recent version is indicated. A few changes have been made globally and are explained in the body of the document and only significant changes have been flagged with callouts, as determined by CIP4 Working Groups.

Table 1.1: Modification Notes

EXAMPLE	CALLOUT MEANING
New in JDF 1.x	New sections, attributes/elements and attribute values.
Deprecated in JDF 1.x	Deprecated sections, attributes/elements and attribute values. Usually there is a deprecation note describing the mechanism that replaces the deprecated item.
Modified in JDF 1.x	Changed syntax or semantics of sections or attributes/elements. Might include clarification as well. Frequently there is a modification note describing the change.

1.3.4.1 Location of Modification Notes

New in JDF 1.4

A callout occurs after one of the following document elements.

- **Section head:** applies to entire section and the contained table (if any).
- **Attribute/Element name:** applies to entire row for the designated attribute/element.
- **Attribute value:** applies to attribute value.

1.3.5 Specification of Cardinality

The cardinality of **JDF** attributes and elements is expressed using a simple Extended Backus-Naur Form (EBNF) notation.

The symbol T in the table below represents an attribute or element. The symbol T consists of either a single name, such as "***RunList***" or an element name followed by a parenthesized name, such as "***RunList*** (Document)". The name in parentheses "Document" identifies a particular element instance when several of the same type exist in some context. For further details, see ▸ Section 6.1 Process Template and ▸ Section 1.3.7 Template for Tables that Describe Elements.

Table 1.2: Cardinality Symbols

NOTATION	DESCRIPTION
T	T SHALL occur exactly once and represents an attribute or element.
T ?	T is OPTIONAL or is REQUIRED only in the circumstances explained in the description field. T represents an attribute or element. If T is an attribute, a default that is specified in the description will not be inserted into the XML by a schema aware parser if no value is explicitly specified.
T +	T occurs one or more times, and represents an element.
T *	T occurs zero or more times, and represents an element.
T = "V"	T is an OPTIONAL attribute, but has the specified default value *v* when T is not supplied. T MAY be set to other values other than the default. A default that is specified as T = "V" indicates a **JDF** default which SHALL be inserted into the XML by the **JDF** validator if no value is explicitly specified. If no schema is used in validation, it is up to the application to apply these defaults. See ▸ Section 1.5.2.1 Conformance Requirements for Support of Attributes and Attribute Values. This notation is only valid for XML attributes, not XML elements.

1.3.6 Template for Narrative Description of Resources

Each section for a resource begins with a brief narrative description of the resource. Following that is a list containing details about the properties of the resource, as shown below.

Resource Properties

Resource Class:	Defines the resource class or specifies resource element if the element only exists as a resource subelement.
Resource referenced by:	List of parent resources that MAY contain elements or references to elements (IDREF or IDREFS attributes) of this type.
Example Partition	List of RECOMMENDED partition keys: For a complete list of partition keys, see the description of @*PartIDKeys* in ▸ Table 3.24 Partitionable Resource Element . Note that resources MAY be partitioned by keys that are not specified in this list.
Input of Processes:	List of **JDF** node types that use the resource as an input resource.
Output of Processes:	List of **JDF** node types that create the resource as an output resource.

The first item in the above list provides the class of the resource. As was described in ▸ Section 3.8.5 Resource Classes, all resources are derived from one of the following seven superclasses: *Intent*, *Parameter*, *Implementation*, *Consumable*, *Quantity*, *Handling* and *PlaceHolder*. All resources inherit additional contents (i.e., zero or more attributes or zero or more elements) from their respective superclasses, and those attributes and elements are not repeated in this section. Thus those attributes associated with a resource of class *Parameter*, for example, can be found in ▸ Table 3.10 Abstract Resource Element. Note that this inheritance is only valid for atomic resources (i.e., resources that reside directly in a ***ResourcePool***).

Resource elements are listed in separate sections if they can be referenced by more than one resource. For an example, see the resource element ***SeparationSpec***. If the resource is not referenced by multiple resources, it is described inside the resource section of the resource to which it belongs. For example, see the structure of the ***BundleItem*** element of the ***Bundle*** resource. If an element inside a resource section of the resource is needed to be referenced by multiple resources in a revision of **JDF**, then that element is promoted to its own section. For example, ***ColorSpaceConversionOp*** was a subelement of ***ColorSpaceConversionParams*** in **JDF**/1.1. The resource class of an atomic resource also defines the superclasses from which the resource inherits additional contents. The ***Consumable Resource***, ***Quantity Resource*** and ***Handling Resource*** inherit from the ***PhysicalResource*** element, which in turn inherits from the resource element. The ***Parameter Resource*** and ***ImplementationResource*** elements inherit from the resource element directly. Non-atomic resources (i.e., resource subelements) do not inherit contents from resource superclasses.

Examples for resources that can be used as atomic resources or resource elements are: ***Employee***, ***InsertSheet***, ***LayoutElement*** and ***Media***.

After the list describing the resource properties, each section contains tables that outline the structure of each resource and, when applicable, the abstract or subelement information that pertains to the resource structure. The first column contains the name of the attribute or element. In some cases, a resource will contain multiple elements of the same type. If this is the case, the element name is listed as often as it appears, along with a term in parentheses that identifies the occurrence. For an example, see ▸ Section 8.54 EndSheetGluingParams. The following sections provide templates of the tables.

1.3.7 Template for Tables that Describe Elements

Resources and elements are defined by their attributes and sub-elements.

Note: For tables that describe resources or elements:

- the italicized text describes the actual text that would be in its place in an actual resource definition
- *Cardinality* in the Name column refers to a cardinality symbol, which is either empty or consists of a symbol, such as "?". Examples described by the Name column include: "***Media****" and "***Component*** (Proof) ?". For further details, see ▸ Section 1.3.5 Specification of Cardinality.
- The text following a "Note:" in a table field gives further information about the specified table row.

Table 1.3: Template for Element Descriptions (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Attribute-Name Cardinality	Attribute-data-type	Information about the attribute.
Element-Name Cardinality	element	Information about the element. Note: The "element" data type means that the specified element SHALL be an in-line subelement within the ***Resource***.

Table 1.3: Template for Element Descriptions (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Element-Name Cardinality	refelement	*Information about the element* Note: The "refelement" data type means that the specified element is based on other atomic resources or resource elements. The specified element SHALL be either an in-line element or an instance of a ***ResourceRef*** element (see ▸ Section 3.10.2 ResourceRef – Element for Inter-Resource Linking and refelement). In case of a ***ResourceRef*** element, a "Ref" SHALL be appended to the name specified in the table column entitled "Name".
FileSpec (someValue) Cardinality	refelement	Information about the ***FileSpec*** resource. Note: ***FileSpec***/@*ResourceUsage* SHALL match the "someValue" value specified in the parentheses. When a resource potentially contains multiple ***FileSpec*** children, the value of ***FileSpec***/@*ResourceUsage* is used to distinguish them.
Resource-Name (someValue) Cardinality	refelement	*Information about the resource and the attribute whose value is "someValue".* Note: Some specified attribute in the specified resource SHALL match the "someValue" value specified in the parentheses. When a resource potentially contains multiple children of the same resource type, the value of some attribute distinguishes the resources.

1.4 Glossary

The following terms are defined as they are used throughout this specification. For more detail on job and workflow components, see ▸ Section 2.1 System Components.

Table 1.4: Glossary (Sheet 1 of 6)

TERM	DEFINITION
Abstract	**Abstract** is used as a modifier for ▸ Elements and ▸ Resources ▸ Element (e.g., ▸ Abstract Element**,** ▸ Abstract Resource and **Abstract** ▸ PhysicalResource).
Abstract Element	An **Abstract Element** is an abstract data type with ▸ Attributes and ▸ Elements that are inherited by subclass concrete ▸ Elements. For example, ***PlacedObject*** is an **Abstract Element** member of a ***Layout***. ***MarkObject*** and ***ContentObject*** ▸ Elements are concrete ▸ Elements of ***PlacedObject*** that are potential members of ***Layout***
Abstract Resource	An **Abstract Resource** is an abstract data type with ▸ Attributes and ▸ Elements that are inherited by all ▸ Resources. For example, ***Media***, as a ▸ Resource inherits all the ▸ Attributes and ▸ Elements of the **Abstract Resource**. **Abstract Resource** has subclasses, such as ***Abstract PhysicalResource*** and ▸ Abstract ***ImplementationResource***. See ▸ Resource.
Acknowledge Message	An **Acknowledge Message** is a ▸ JMF Message that is delayed response to a ▸ Command Message or ▸ Query Message.
Agent	The component of **a JDF**-based workflow that writes **JDF**.
Allowed values are	The phrase "**Allowed values are:**" precedes the complete (closed) list of values for an ▸ Attribute whose values are enumeration or enumerations.
Allowed values are from	The phrase "**Allowed values are from:**" precedes a reference to the values. The reference may be an XPath to a value or a reference to a table of attribute values. The referenced values are as complete (closed) as a the reference says they are.
Attribute	An XML syntactic construct describing an unstructured characteristic of an ▸ Element. See ▸ [XML] for details.
Attribute Value	The value of an ▸ Attribute.
Binding Side	The binding side of a signature is defined as the last fold.

Table 1.4: Glossary (Sheet 2 of 6)

TERM	DEFINITION
Class	A set of complex data types with common content in an object-oriented sense. A complex data type consist of zero or more ▸ Elements and zero or more ▸ Attributes. Each ▸ Resource belongs to a **Class:** "Consumable", "Handling", "Implementation", "Intent", "Parameter", "PlaceHolder", "Quantity". See ▸ Consumable Resource ***Consumable Resource***, etc. Each ***Notification*** Audit ▸ Element belongs to a **Class:** "Event", "Information", "Warning", "Error", "Fatal".
Combined Process	A ▸ Process which is described by multiple simpler ▸ JDF ▸ Processes. See ▸ Process
Combined Process Node	A ▸ Node that represents a ▸ Combined Process, which is described by multiple simpler ▸ JDF ▸ Processes,.See ▸ Combined Process, ▸ Node and ▸ Section 3.3 Common Node Types.
Command Message	A **Command Message** is a ▸ JMF Message that requests its recipient to change its state.
Consumable Resource	A **Consumable Resource** is consumed during a ▸ Process. See ▸ Resource, ▸ PhysicalResource and ▸ Section 3.8.5.4 Consumable Resource.
Controller	The component of **a JDF** -based workflow that initiates ▸ Devices, routes **JDF** , and communicates status information.
Default	Used to indicate the ▸ Attribute Value that a ▸ JDF Consumer SHALL use if an OPTIONAL ▸ Attribute (as indicated by a "?" or *@Attribute* = "DefaultValue" in this specification) has been omitted from a ▸ JDF instance. See ▸ Section 1.5.2.1 Conformance Requirements for Support of Attributes and Attribute Values.
Default behavior	The phrase "**Default behavior:**" precedes a description of the default behavior for an ▸ Attribute or Span ▸ Element.
Default value is	The phrase "**Default value is:**" precedes the default value for an ▸ Attribute.
Default value is from	The phrase "**Default value is from:**" precedes a reference to a default value – usually an XPath.
Deprecated	Indicates that **a JDF** ▸ Element is being phased out of **JDF** usually in favor of newer **JDF** ▸ Element(s). It is RECOMMENDED that an ▸ Agent ###not include such **a JDF** ▸ Element in **a JDF** instance. Such an indicated **JDF** ▸ Element might be removed from a future version of the **JDF** specification. ▸ JDF Consumers SHOULD only ▸ Support such **JDF** ▸ Elements for backward compatibility with previous versions of **JDF**. **Deprecated** items are flagged with Deprecated in JDF 1.x in this specification.
Device	The component of **a JDF** workflow part that interprets **JDF** and executes the instructions. If a **Device** controls a ▸ Machine, it does so in a proprietary manner. For details, see ▸ Section 2.1.2.2 Device about devices in workflow components.
Document Set	A set of ▸ Instance Documents presumed to be related.
Element	An XML-based syntactic construct describing structured data in **JDF** .
Enumeration	A data type that represents a closed set of values, which are specified in this document.
Finished Page	A page of a final product that normally has no folds inside. The folds of the finished product for packaging (e.g., folding letters into an envelope) or Z-fold of an oversized book, have no effect on the **Finished Page** definition.
Folio	A numbered ▸ Finished Page of a printed book or publication. (Pages are not all necessarily numbered. A 72-page book might have 68 pages that are numbered, which are referred to as either "**Folio pages**" or "**Folios**.")
Form	A collection of imposed (ordered) ▸ Finished Pages set for printing or imaging to plate or film. See also ▸ Signature.
Gear Side	**Gear Side** is the side of a ▸ Machine, where the drives and gear are mounted. **Gear Side** is opposite to ▸ Operating Side.

Table 1.4: Glossary (Sheet 3 of 6)

TERM	DEFINITION
Gray Box	A **Gray Box** specifies a loose combination of several ▸ Processes with a specific goal. A **Gray Box** does not specify all ▸ Processes or all ▸ Resources - except for ▸ Output Resources. In a ▸ JDF instance, a ▸ Process Group with a *@Types* ▸ Attribute and no child ▸ Nodes represents a **Gray Box**.
Handling Resource	A **Handling Resource** is used during a ▸ Process, but are not destroyed by that ▸ Process. See ▸ Resource, ▸ PhysicalResource and ▸ Section 3.8.5.6 Handling Resource.
ImplementationResource	An **ImplementationResource** defines a ▸ Device or operator that executes a given ▸ Node. See ▸ Resource and ▸ Section 3.8.5.3 ImplementationResource.
Input Resource	A resource is an input to a ▸ Process. See ***Resource***.
Instance Document	A document that is part of the output of a job. This generally refers to personalized printing jobs. Each of the individual documents produced from the same input template is referred to as an **Instance Document**. For example, in a credit card statement run, each statement is an **Instance Document**.
Intent Resource	An **Intent Resource** defines the details of products to be produced without defining the process to produce them.See ▸ Resource and ▸ Section 3.8.5.2 Intent Resource.
JDF	Job Definition Format. The overall name of this specification. There is also a **JDF** ▸ Element, which is a top-level ▸ Element within **JDF** that encompasses a ▸ Node (see below).
JDF Consumer	A ▸ Device, ▸ Controller or ▸ Agent that consumes **JDF** instances.
JDF Node	See ▸ Node**.**
JMF	Job Messaging Format. Transfers information between ▸ MIS, ▸ Controllers and ▸ Devices. See ▸ Chapter 5 JMF Messaging Format.
JMF Message	A **JMF Message** is synonymous with ▸ Message. See ▸ Message
Job	A hierarchical tree structure comprised of ▸ Nodes. Describes the output that is desired by a customer.
Job Part	One or more ▸ Nodes which comprise the smallest level of control of interest to ▸ MIS.
Jog Edge	The jog edge of a signature is defined as the last fold that is perpendicular to the ▸ Binding Side. If there is no fold that is perpendicular to the ▸ Binding Side, then the jog edge is the edge on the bottom when the folded signature is not flipped after the final fold.
Leaf	Both the recto and verso ▸ Finished Pages on one piece of paper with “leaves” being the plural usage.
Link	A pointer to information that is located elsewhere in a **JDF** document or that is located in another document.
Machine	The part of a device that does not know **JDF** and is controlled by a **JDF** ▸ Device in a proprietary manner.
Message	The XML element that ▸ Devices and ▸ Controller use to exchange queries, commands, responses, etc. among themselves using HTTP as the underlying protocol and ▸ JMF as the XML format. See ▸ JMF and ▸ Message Family.
Message Family	A **Message Family** is a set of messages. The 6 **Message Families** are ▸ Query Message, ▸ Command Message, ▸ Registration Message, ▸ Response Message, ▸ Acknowledge Message and ▸ Signal Message.
MIS	Management Information System. The functional part of a **JDF** workflow that oversees all ▸ Processes and communication between system components and system control. MIS is assumed to be a role rather than an individual application. A single application may fulfill various roles of an MIS and various roles of an MIS may be implemented by multiple applications.

Table 1.4: Glossary (Sheet 4 of 6)

TERM	DEFINITION
NamedFeature	The term ▸ NamedFeature describes a value that is identified by a name using a ***JDF***/***GeneralID***[@*DataType* = "NamedFeature"]. The value is specified by ***GeneralID***/@*IDValue*, and the name is specified by ***GeneralID***/@*IDUsage* For example, the ***GeneralID*** with @*IDUsage*="a1" and @*IDValue*="a2 b2" and @*DataType*="NamedFeature" is a ▸ NamedFeature with a value "a2 b2" that is identified by the name "a1". ***GeneralID***/@*IDValue* is defined as a string. **Note:** That blanks are allowed in the value.
Node	The **JDF** element type detailing the ▸ Resources and ▸ Process specification needed to produce a final or intermediate product or ▸ Resource. A ▸ Node is also called a ▸ JDF Node.
Operating Side	**Operating Side** is the side of a ▸ Machine, where the operator works. **Operating Side** is opposite to ▸ Gear Side.
Output Resource	A resource that is an output from a ▸ Process. See ***Resource***.
Parameter Resource	A **Parameter Resource** defines the details of ▸ Processes, as well as any non-physical computer data such as files used by a ▸ Process. See ▸ Resource and ▸ Section 3.8.5.1 Parameter Resource.
Page	When **Page** occurs by itself, not in the context of ▸ Finished Page or ▸ Reader Page, it means " ▸ Finished Page."
Partition	A **Partition** is a node of a ▸ Partitioned Resource structure. A leaf node **Partition** represents a single ▸ Resource. A non-leaf node **Partition** represents a set of ▸ Resources. Values of the @*PartIDKeys* ▸ Attribute in the ▸ Partitioned Resource root specify the ▸ Attributes used to identify the individual ▸ Resources in the ▸ Partitioned Resource. Each **Partition** except the ▸ Partitioned Resource root has a ▸ Partition Key ▸ Attribute whose value identifies the **Partition**. See ▸ Table 3.24 Partitionable Resource Element.
Partition Key	A **Partition Key** is an enumeration value of the @*PartIDKeys* ▸ Attribute and a **Partition Key** is an ▸ Attribute that with can identify a ▸ Partition or can reference a ▸ Partition from within a ***Part*** ▸ Element. See ▸ Section 3.10.6 PartIDKeys Attribute and Partition Keys.
Partition, to	The verb **to Partition** means to construct a ▸ Partitioned Resource from a set of ▸ Resources of the same ▸ Class. See ▸ Section 3.10.5.4 Partitioning of Resource Subelements.
Partitionable Resource	A ▸ Resource that can become a ▸ Partitioned Resource.
Partitioned Resource	A **Partitioned Resource** is a structured ▸ Resource that describes a set of ▸ Resources, all of the same ▸ Class and representing multiple physical or logical entities (e.g., a set of ***ExposedMedia*** that represent multiple separated plates).
PDL	Page Description Language. A generic term for any language that describes pages which might be printed. Examples are PDF®, PostScript® or PCL®.
Phase	A Phase is a distinct part of a ▸ Workstep such as setup, production or cleanup.
PhysicalResource	A **PhysicalResource** is a resource whose ▸ Class is "Consumable", "Quantity" or "Handling" is considered a **PhysicalResource**. See ▸ Resource and ▸ Section 3.8.5.7 PhysicalResource.
Process	An individual step in the workflow.
Process Group	A group of ▸ Processes. See ▸ Process and ▸ Process Group Node
Process Group Node	A ▸ Node that contains multiple child ▸ Nodes. See ▸ Process Group, ▸ Node and ▸ Section 3.3 Common Node Types.
Process Node	A ▸ Node that describes an individual ▸ Process. See ▸ Node and ▸ Section 3.3 Common Node Types.

Table 1.4: Glossary (Sheet 5 of 6)

TERM	DEFINITION
Product Intent	Describes the end result that a customer is requesting. See ▸ Product Intent Node.
Product Intent Node	A ▸ Node that describes intent rather than specifying the ▸ Process. See ▸ Node, ▸ Product Intent and see ▸ Section 3.3 Common Node Types.
Quantity Resource	A **Quantity Resource** has been created by a ▸ Process from either a ▸ Consumable Resource or an earlier **Quantity Resource**. See ▸ Resource, ▸ PhysicalResource and ▸ Section 3.8.5.5 Quantity Resource.
Query Message	A **Query Message** is a ▸ JMF Message that requests its recipient to provide information, but not change its state.
Queue	A Queue is a representation of a ▸ Device that receives, manipulates and stores multiple ▸ Queue Entry items for execution.
Queue Entry	A Queue Entry is a the representation of an individual ▸ JMF process in a ▸ Queue. A Queue Entry MAY be comprised of multiple ▸ Workstep processes.
Reader Page	A logical page as perceived by a reader, for example one ***RunList*** entry. One **Reader Page** might span more than one ▸ Finished Page (e.g., a centerfold). One ▸ Finished Page might contain contents defined by multiple **Reader Pages** (e.g., NUp imposition. **Reader Pages** are defined independently of ▸ Finished Pages).
Registration Message	A **Registration Message** is a ▸ JMF Message that requests its recipient to send a ▸ Command Message to some other recipient.
Resource	A physical or conceptual entity that is modified or used by a ▸ Node. Examples include paper, images or ▸ Process parameters.
ResourceLink	An ▸ Element that links to a ▸ Resource. See ▸ Resource and ▸ Section 3.9.2 ResourceLink.
Response Message	A **Response Message** is a ▸ JMF Message that functions as a synchronous response to a ▸ Command Message or ▸ Query Message.
Roll	A **Roll** is media that is mainly used in connection with ▸ Web printing. In British english the name "reel" for "roll" is in widespread use. **Roll** is used as synonym of reel. Also, see the term ▸ Web in this glossary.
Root Node	The top most ▸ JDF Node. See ▸ Node.
Sheet	The printer's ▸ Roll of paper or paper cut for press size, with "recto" and "verso" forms for identification of orientation through the press (facing up versus facing down at the feeder or off the ▸ Roll). ▸ Sheets are press sheets which may be comprised of multiple folding signatures and might also have "recto" and "verso" forms for identification of orientation through the press (facing up versus facing down at the feeder or off the roll). The term "cut sheet" refers to an individual **Sheet**, typically in a phrase, such as "separately cut **Sheets** of an opaque material". The term "Sheet-Fed" is used to describe a press that consumes cut **Sheets**, typically in the phrase "Sheet-Fed Press".
Signal Message	A **Signal Message** is a ▸ JMF Message that is sent asynchronously when some event occurs.
Signature	A **Signature** is a set of printed ▸ Sheets that are folded or yet to be folded. Note that there are multiple usages of the word **Signature** in the industry. A ▸ Sheet MAY contain multiple ***BinderySignature*** ▸ Resources that are the input to folding. This is the standard usage in conventional printing, where multi-page ▸ Sheets are printed and potentially cut into multi-page imposition signatures before folding. The ***Layout*** ▸ Resource , on the other hand, describes a **Signature** as a set of ▸ Sheets. This is appropriate for digital printing, where typically only one or two pages are printed per ▸ Surface and multiple ▸ Sheets are gathered prior to folding.
Slave Controller	The component of a **JDF** workflow that accepts **JDF** as a ▸ Device from other ▸ Controllers and/or **Slave Controllers** and sends **JDF** to other **Slave Controllers** and/or ▸ Devices.

Table 1.4: Glossary (Sheet 6 of 6)

TERM	DEFINITION
Subelement	A child ▸ Element of some other ▸ Element
Subnode	A ▸ Node that is a child of some other ▸ Node. See ▸ Node.
Support	A ▸ JDF Consumer **Supports** a **JDF** syntactic construct (▸ Processes, ▸ Resources, ▸ Elements, ▸ Attributes and ▸ Attribute Values) if the ▸ JDF Consumer performs the action defined in this specification for the **JDF** construct when consuming **a JDF** instance that includes the **JDF** syntactic construct. If the ▸ Machine that a ▸ Device is representing **Supports** a feature which is represented by **a JDF** construct, then the ▸ Device SHOULD **Support** that **JDF** syntactic construct.
Surface	A single side of a ▸ Sheet.
Tag	A syntactic XML construct that marks the start or end of an ▸ Element.
Unique	The word "Unique" without further scope details means "Unique within the job, message or file depending on the context". **Possible scopes are:** **Unique within the machine** (e.g., ***ProductionPath***/@*ProductionPathID*). **Unique within the workflow** – This covers the whole scope at one installation (e.g., ***StatusQuParams***/@*JobID*, ***FileSpec***/@*UID*, ***Device***/@*DeviceID* and @*CreationID* (in document properties)). **Unique within the site** – A site is a physical organization entity within a company (printing plant, finishing subcontractor) (e.g., ***Employee***/@*PersonalID*). It is NOT the ***Contact***/***Company*** used for customers! **Unique within the company** – identification of a company or contact SHALL be unique within the company's database because it can be used on multi tenant systems like web approval) (e.g., ***Contact***[(@*ContactTypes*, "Approver")]/@*ProductID*).
Values include	The phrase "**Values include:**" precedes an open list of values for an ▸ Attribute whose values are NMTOKEN, NMTOKENS, string, ***NameSpan*** or ***StringSpan***. The list includes recommended values but does not include preclude potential vendor or customer extensions.
Values include those from	The phrase "**Values include those from:**" precedes a reference to the open list of values. The reference may be an XPath to a value or a table of ▸ Attribute Values. The referenced values do not include potential vendor or customer extensions.
Web	A **Web** is media that comes from a ▸ Roll and is mainly used in connection with **Web** printing. This specification uses the word "Web-Fed" instead of "roll-fed" It uses the phrase "Web Printing" and "Web Press" to describe printing presses that consumes media that comes from a ▸ Roll.
Work Center	An organizational unit, such as a department or a subcontracting company, that can accomplish a task.
Workstep	A Workstep is an individual ▸ JMF process that can be processed on a single device in one pass. A Workstep is comprised of one or multiple ▸ Phaseparts such as setup, production or cleanup.

1.5 Conformance

1.5.1 Conformance Terminology

The words "SHALL", "SHALL NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", "NEED NOT" and "OPTIONAL" are used in this specification to define a requirement for the indicated **Agent** or the indicated ▸ JDF Consumer as follows.

Table 1.5: Conformance Terminology (Sheet 1 of 2)

TERM	MEANING
SHALL or REQUIRED	Means that the definition is an absolute requirement of the specification.

Table 1.5: Conformance Terminology (Sheet 2 of 2)

TERM	MEANING
SHALL NOT	Means that the definition is an absolute prohibition of the specification.
SHOULD or RECOMMENDED	Means that there might exist valid reasons in particular circumstances for an implementer to ignore a particular item, but the implementer SHALL fully understand the implications and carefully weigh the alternatives before choosing a different course.
SHOULD NOT or NOT RECOMMENDED	Means that there might exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the implementer should fully understand the implications and then carefully weigh the alternatives before implementing any behavior described with this label.
MAY, NEED NOT or OPTIONAL	Means that an item is truly optional. If a ‣ JDF Consumer is using a **JDF** parser, that parser will supply the default values indicated in this specification, if any, for optional ‣ Attributes that the ‣ Agent has omitted (indicated by @*Attribute* = "DefaultValue" in this specification). See ‣ Section 1.3.5 Specification of Cardinality. For features that are optional for a ‣ JDF Consumer to ‣ Support, one vendor might choose to ‣ Support such an item because a particular marketplace requires it or because the vendor feels that it enhances the product, while another vendor might omit ‣ Support of that item. Similarly, one vendor of an ‣ Agent might choose to supply such an item in a **JDF** instance, while another vendor might omit the same item in a **JDF** instance. A ‣ JDF Consumer implementation which does not include ‣ Support of a particular option (element or attribute) SHALL be prepared to interoperate with an ‣ Agent implementation which does supply the option, though with reduced functionality. In the same vein, a ‣ JDF Consumer implementation which does include ‣ Support for a particular option SHALL be prepared to interoperate with an ‣ Agent implementation which does not supply the option in the **JDF** instance.

1.5.2 Conformance Requirements for JDF Entities

The subsections of this section define the general conformance requirements for the **JDF** entities: 1) ‣ Attributes and ‣ Attribute Values, 2) ‣ Resources, 3) ‣ Processes, and 4) ‣ Combined Processes.

1.5.2.1 Conformance Requirements for Support of Attributes and Attribute Values

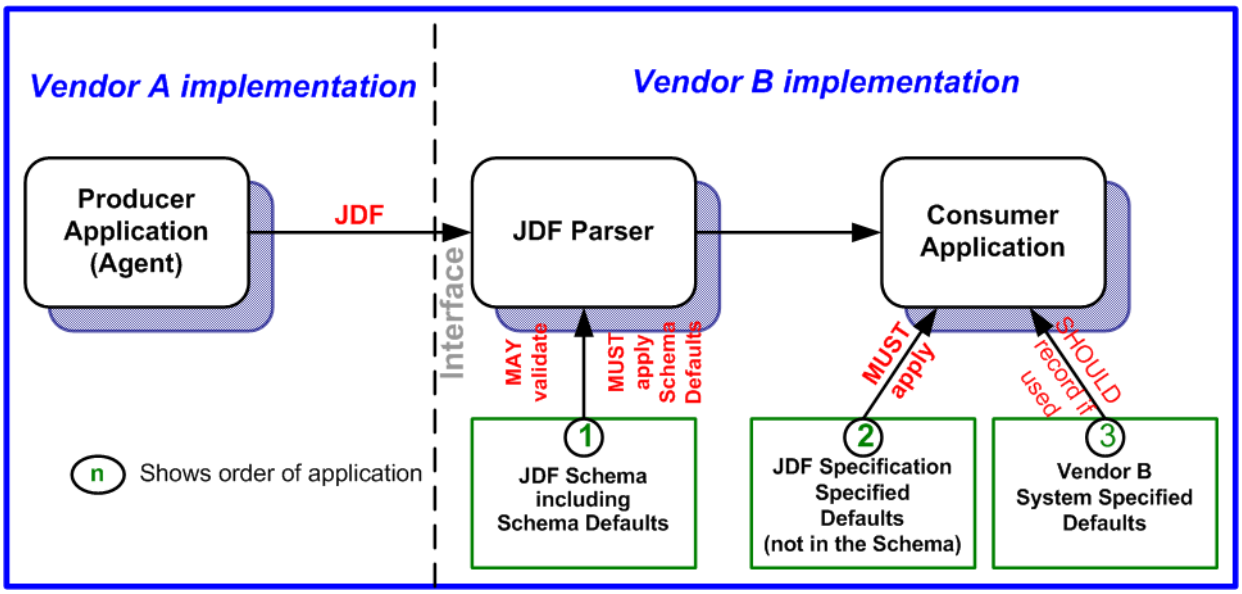
If a ‣ JDF Consumer supports an attribute, it SHALL support all of the values that this specification indicates are REQUIRED for a ‣ JDF Consumer to support (whether or not the attribute is REQUIRED for the ‣ Agent to supply in that context). If this specification is silent on which values are REQUIRED for support of an attribute, then the ‣ JDF Consumer SHALL support at least one value in order to claim support for the attribute.

Attributes that are OPTIONAL for an ‣ Agent to include in a **JDF** instance are indicated by a "?" character following the attribute name or by the notation @*Attribute* = "DefaultValue" as indicated in ‣ Section 1.3.5 Specification of Cardinality.

A Special Note on the Handling of Defaults. Prior to **JDF** 1.2 many OPTIONAL attributes included either explicit default values or the default value was indicated as "system specified" or the "SystemSpecified" enumeration or NMTOKEN value. In **JDF**/1.2, the explicit default values are indicated as default values using the "=" followed by the "value" (See ‣ Section 1.3.5 Specification of Cardinality). The "SystemSpecified" enumeration and NMTOKEN values have been removed and the attribute remains as an OPTIONAL attribute indicated with a "?" with no default value. The **JDF** consuming application SHALL supply the default value when the attribute is omitted from the **JDF** instance. Such an indicated default value SHALL have the same semantic meaning as if an ‣ Agent includes the attribute in the **JDF** instance with the same value. If an OPTIONAL attribute does not have a default value indicated in its description and the **JDF** instance does not include the attribute, then the ‣ JDF Consumer can use a system-specified value.

See ‣ Figure 1-1: Handling of Default Values of JDF Attributes. below. Such a system-specified attribute value can be configurable by a system administrator for the ‣ JDF Consumer or can depend on the values of other supplied attributes and/or the current setting of the ‣ JDF Consumer device or the actual ‣ Machine for which the device is providing a **JDF** interface.

Figure 1-1: Handling of Default Values of JDF Attributes.



1.5.2.2 Conformance Requirements for Support of Elements

If a ▸ JDF Consumer supports an element, it

1 SHALL support all of the attributes (see ▸ Section 1.5.2.1 Conformance Requirements for Support of Attributes and Attribute Values) defined for that element that an ▸ Agent is REQUIRED to include in the element instance, i.e. attributes with either no marks or a "+" as defined in ▸ Section 1.3.5 Specification of Cardinality.

2 SHOULD support the *@SettingsPolicy*, *@BestEffortExceptions*, *@MustHonorExceptions* and *@OperatorInterventionExceptions* (see ▸ Section 3.1 Generic Contents of All Elements) attributes and all of their defined values. These attributes control the policy that a ▸ JDF Consumer SHALL follow when it encounters unsupported settings (i.e., ▸ Subelements, attributes or attribute values in the resource).

1.5.2.3 Conformance Requirements for Support of Processes

All processes are OPTIONAL for a ▸ JDF Consumer to support. However, a device SHALL support at least one process or a ▸ Combined Process. If a ▸ JDF Consumer supports a process, it

1 SHALL support all of the input and output ***ResourceLink*** elements and referenced resources as described in ▸ Section 1.5.2.2 Conformance Requirements for Support of Elements that this specification defines for that process,

2 MAY make its own assumptions regarding attributes and ▸ Subelements of an OPTIONAL ▸ Input Resource (Resources with either a "?" or an "*" – see ▸ Section 1.3.5 Specification of Cardinality) that an ▸ Agent has omitted from the process in the **JDF** instance; therefore, default attribute values defined in this specification are not guaranteed when the ▸ Agent omits the resource from the process in the **JDF** instance (see ▸ Section 6.1 Process Template), and

3 SHOULD find the processes that it supports in a **JDF** instance and SHALL ignore all other processes, independent of the *@SettingsPolicy* attribute for those other processes.

1.5.2.4 Conformance Requirements for Support of Combined Processes

All ▸ Combined Processes are OPTIONAL for a ▸ JDF Consumer to support. The rules for processes specified in ▸ Section 1.5.2.3 Conformance Requirements for Support of Processes apply. If a ▸ JDF Consumer supports a ▸ Combined Process, it

1 SHALL support all of the ▸ Input Resources as defined in ▸ Section 1.5.2.2 Conformance Requirements for Support of Elements that this specification defines for the *first* process in the ▸ Combined Process Node (i.e., the first process listed in the *@Types* attribute),

2 SHALL support all of the ▸ Output Resources as defined in ▸ Section 1.5.2.2 Conformance Requirements for Support of Elements that this specification defines for the *last* process in the ▸ Combined Process,

3 MAY support resources that are used as exchange resources between processes in the process chain of the ▸ Combined Process (i.e., resources that are both produced and consumed within the ▸ Combined Process Node),

4 SHALL support resources in intermediate process steps that are *not* used as exchange resources between processes in the process chain of the ▸ Combined Process.

1.5.3 Conformance to Settings Policy

The *@SettingsPolicy*, *@BestEffortExceptions*, *@MustHonorExceptions* and *@OperatorInterventionExceptions* attributes are defined in ▸ Table 3.1 Any Element (generic content). They define the conformance policy of a device. A ▸ JDF Consumer SHOULD support these attributes and all of the defined values so that an ▸ Agent can depend on the ▸ JDF Consumer following the policy requested by the ▸ Agent in a **JDF** instance.

1.6 Data Structures

The following table describes the data structures as they are used in this specification. For more details on **JDF** Schema and data types, see ▸ Appendix A Data Types and Values.

In **JDF** 1.2, some data types have been enhanced to include unbounded values by defining the explicit tokens "INF" and "-INF". For instance, the IntegerRange "0 ~ INF" specifies all positive integers including 0.

Data Types

An important reason for using a W3C Schema is to make use of user-defined data types. Even data types that are defined in the Schema specification have been more narrowly defined in JDF, including boolean (JDF doesn't permit 1, 0), double (JDF doesn't permit NaN), duration (JDF has INF & -INF) and string (JDF doesn't permit CR LF & FF). Be sure to check ▸ Appendix A Data Types and Values for all data type definitions.

Table 1.6: **JDF** *Data Types (Sheet 1 of 3)*

DATA TYPE	DESCRIPTION
Anchor	Describe the 9 anchor points of a rectangle. See ▸ Table A.2 Anchor Enumeration Values for a list values.
boolean	Binary-valued logic: (true | false).
CMYKColor	Represents a CMYK color specification.
date	Represents a time period that starts at midnight of a specified day and lasts for 24 hours.
dateTime	Represents a specific instant of time. It SHALL be a UTC time or a local time that includes the time zone.
DateTimeRange	Two *dateTime* values separated by a "~" (tilde) character that defines the closed interval of the two. TimeRange corresponds semantically to the time interval (two time instants separated by a slash) defined in ▸ [ISO8601:2004].
DateTimeRangeList	Whitespace-separated list of *DateTimeRange* values.
double	Corresponds to ▸ [IEEE754] double-precision, 64-bit floating point type, including special tokens INF and -INF. This corresponds to the standard XML double with NaN removed. For details, see ▸ [XMLSchema]. **Note**: Prior to **JDF** 1.2 the data type *number* was used. The *double* and *number* data types are syntactically equivalent.
DoubleList New in JDF 1.2	Whitespace-separated list of *double* values. **Note:** This data type was named *NumberList* before JDF 1.2.
DoubleRange New in JDF 1.2	Two *double* values separated by a "~" (tilde) character that define the closed interval of the two. **Note:** This data type was named *NumberRange* before **JDF** 1.2.
DoubleRangeList New in JDF 1.2	Whitespace-separated list of *double* and *DoubleRange* values. **Note:** This data type was named *NumberRangeList* before **JDF** 1.2.
duration	Represents a duration of time.
DurationRange	Two duration values separated by whitespace. Describes a range of time durations. More specifically, it describes a time span that has a relative start and end.
DurationRangeList	Whitespace-separated list of *DurationRange* values.
element	Structured data. The specific data type is defined by the element name.
enumeration	Limited set of *NMTOKEN* values (see below).

Table 1.6: **JDF** *Data Types (Sheet 2 of 3)*

DATA TYPE	DESCRIPTION
enumerations	Whitespace-separated list of *enumeration* values.
gYearMonth	Represents a specific Gregorian month in a specific Gregorian year.
hexBinary	Represents arbitrary hex encoded binary data.
hexBinaryList New in JDF 1.4	Whitespace-separated list of *hexBinary* values.
ID	Unique identifier as defined by ▸ [XML] (see ▸ Section 1.3.1 Document References). SHALL be unique within the scope of the **JDF** document.
IDREF	Reference to an element holding the unique identifier as defined by [XML Specification 1.0].
IDREFS	List of references (IDREF values) separated by white spaces as defined by ▸ [XML].
integer	Represents numerical integer values, including the special tokens INF and -INF. This corresponds to the standard XML integer with INF and -INF added. Values greater than +/-2**31 are not expected to occur for this data type. For details, see ▸ [XMLSchema].
IntegerList	Whitespace-separated list of *integer* values.
IntegerRange	Two *integer* values separated by a "~" (tilde) character that define a closed interval.
IntegerRangeList	Whitespace-separated list of *integer* values and *IntegerRange* values.
JDFJMFVersion	Version label of a **JDF** or **JMF** instance. See ▸ Section 3.13 JDF Versioning for a discussion of versioning in **JDF**. See ▸ Table A.20 JDFJMFVersion Enumeration Values for a list values.
JDFJMFVersions	Whitespace separated list of **JDFJMF***Version*.values
LabColor	Represents a Lab color specification.
language	Represents a language and country code (for example, en-US) for a natural language. Values SHALL conform to ▸ [RFC1766].
languages New in JDF 1.4	Whitespace-separated list of *language* values.
LongInteger	Represents numerical integer values, including the special tokens INF and -INF. This corresponds to the standard XML integer with INF and -INF added. Values greater than +/-2**31 are expected to occur for this data type. For details, see ▸ [XMLSchema].
matrix	Whitespace-separated list of six doubles representing a coordinate transformation matrix.
NamedColor	Represents a color definition by name. See ▸ Table A.22 NamedColor Enumeration Values for a list values.
NameRange	Two *NMTOKEN* values separated by a "~" (tilde) character that define an interval of *NMTOKEN* values.
NameRangeList	Whitespace-separated list of *NMTOKEN* and *NameRange* values.
NMTOKEN	A continuous sequence of special characters as defined by the [XML Specification 1.0].
NMTOKENS	Whitespace-separated list of *NMTOKEN* values.
Orientation New in JDF 1.2	Enumeration that specifies named orthogonal two-dimensional orientations. See ▸ Table A.24 Orientation Enumeration Values for a list values.
Orientations New in JDF 1.2	Whitespace separated list of *Orientation* enumeration values that specify named orthogonal two-dimensional orientations.
PDFPath	Whitespace-separated list of path operators as defined in PDF.

Table 1.6: **JDF** *Data Types (Sheet 3 of 3)*

DATA TYPE	DESCRIPTION
rectangle	Whitespace-separated list of four doubles representing a rectangle.
refelement	**ResourceElement** or a reference to an element. Used to define candidates for inter-Resource linking in resources.
regExp New in JDF 1.2	Regular expression as defined by ▸ [XMLSchema]
shape	Whitespace-separated list of three doubles representing a three-dimensional shape consisting of a width, height and length. Unless specified otherwise in the attribute description, these three numbers are an X-dimension, a Y-dimension and a Z-dimension, respectively.
ShapeRange	Two *shape* values separated by a "~" (tilde) character that defines a 3-dimensional box bounded by x1 y1 z1 ~ x2 y2 z2.
ShapeRangeList	Whitespace-separated list of *shape* values or *ShapeRange* values.
Source	jdf or xml.
string Modified in JDF 1.2	Character strings without tabs or line feeds. Corresponds to the standard XML normalizedString data type ▸ [XMLSchema].
text	Text data contained in an XML element (between start and end tag). A few elements, such as ***Comment***, have text.
text element	Element that contains text between start and end tags. e.g. <Comment>example text</Comment>.
TimeRange	Two *dateTime* values separated by a "~" (tilde) character that defines the closed interval of the two. TimeRange corresponds semantically to the time interval (two time instants separated by a slash) defined in ▸ [ISO8601:2004].
TransferFunction	Whitespace separated list of an even number of doubles representing a set of XY coordinates of a transfer function.
URI Modified in JDF 1.3	URI-reference. Represents a Uniform Resource Identifier (URI) Reference as defined in Section 4 of ▸ [RFC3986]. For the "file:" URL scheme, see ▸ [RFC3987]. URI was modified in **JDF** 1.3 to include Internationalized Resource Identifiers (IRI).
URL Modified in JDF 1.3	URL-reference. Represents a Uniform Resource Locator (URL) Reference as defined in Section 4 of ▸ [RFC3986]. For the "file:" URL scheme, see ▸ [RFC3987]. URL was modified in **JDF** 1.3 to include usage of Internationalized Resource Identifiers (IRI).
WorkStyle New in JDF 1.4	Specifies work styles of a press run. See ▸ Table A.38 WorkStyle Enumeration Values for a list values.
XPath	Represents an XPath expression of an XML node set (attributes or elements), boolean, double or string. ▸ [XPath]
XYPair	Whitespace-separated list of two doubles. Unless specified otherwise in the attribute description, these two doubles are an X-dimension and a Y-dimension, respectively.
XYPairRange	Two *XYPair* values separated by a "~" (tilde) character that defines a rectangle bounded by x1 y1 ~ x2 y2.
XYPairRangeList	Whitespace-separated list of *XYPairRange* values.
XYRelation New in JDF 1.2	Defines the relationship between two ordered doubles. See ▸ Table A.39 XYRelation Enumeration Values for a list of NMTOKEN values.

1.6.1 Units

JDF specifies most values in default units. That means that an implementation SHALL use the defined default units and SHALL NOT use alternate units. All measurable quantities are stated in double precision. Processors SHOULD NOT specify a unit unless no default exists, such as when new resources are defined. Then the units SHALL be based on metric units.

Overriding the default units that are defined in this table is non-standard and MAY lead to undefined behavior. Any exceptions are specified in the appropriate descriptive tables.

The following table lists the units used in **JDF** . The "Representation" column specifies the XML representation to indicate the units used in the following **JDF** attributes:

1 The @*Unit* attribute in resources (see ▸ Table 3.13 Abstract PhysicalResource Element).

2 The @*Unit* attribute in ***ResourceInfo*** (see ▸ Table 5.86 ResourceInfo Element).

3 The @*CounterUnit* attribute in ***DeviceInfo*** (see ▸ Table 5.104 DeviceInfo Element):

Table 1.7: Units Used in JDF

MEASUREMENT	UNIT	REPRESENTATION	REMARKS
Angle	degree°	degree	—
Area	m^2	m2	Used for media, (e.g., in wide format printing).
Countable Objects Modified in JDF 1.4	1	count	Countable objects, such as sheets, MAY be specified as "count".
Length	point (1/72 inch)	pt	Used for all except microscopic lengths (see below)
	micron (μ)	um	Used for microscopic lengths — where used (instead of points) it will be explicitly stated in the definition of the item. See ***Media***/@*Thickness*.
Line Screen	lpi	lpi	The lines per inch (lpi) for conventionally screened halftone, screened grayscale and screened monotone bitmap images.
Paper weight	g/m^2	gsm	Paper weight SHALL be provided in grams per square meter. See ▸ Appendix E Paper Weight and Grade for details of calculating non-gsm paper weights.
Power (electrical) New in JDF 1.6	kilowatt hour	kWh	Used to measure consumption of electricity. **Note:** Current power consumption (kW) MAY be provided in a ***ResourceInfo*** as 'rate of consumption' of electric power, i.e. kWh/h=kW.
Resolution	dpi	dpi	The dots per inch (dpi) for print output and bitmap image (TIFF, BMP, etc.) file resolution.
Screen Resolution	ppi	ppi	The pixels per inch (ppi) for screen display (e.g., soft-proof display and user interface display), scanner capture settings and digital camera settings.
Speed	unit/hour		Speed SHALL be specified in base units per hour. The base units that are used to represent speed SHALL be identical to the base unit.
Spot Resolution	spi	spi	For imaging devices such as filmsetters, platesetters and proofers, the fundamental imaging unit (e.g., one "on" laser or imaging head imaged unit). **Note:** Many imaging devices construct dots from multiple imaging spots, so dpi and spots per inch (spi) NEED NOT be equivalent.
Temperature	C° (Celsius)	C	degree centigrade
Volume (gas) New in JDF 1.6	m^3 (cubic meter)	m3	Used to measure consumption of gas
Volume (liquid)	liter	l	—
Weight	gram	g	—

1.6.2 Counting in JDF

Zero-based indices SHALL be used in **JDF** . Thus the first index is 0, the second index is 1 etc. Note that this restriction applies to the **JDF** representation only. Display of values, for instance in a user interface, is implementation defined.

Overview of JDF

Introduction

This chapter explains the basic aspects of **JDF** . It outlines the terminology that is used and is recognized by the format, and the components of a workflow necessary to execute a printing job using **JDF** . Also provided is a brief discussion of **JDF** process structure and the role of messaging in a **JDF** job.

2.1 System Components

This section defines unique terminology used in this specification for the job and workflow components of **JDF** . Links to additional information is included for some terms.

2.1.1 Job Components

This terminology describes how **JDF** is described conceptually and hierarchically.

2.1.1.1 Jobs and Nodes

A ▸ Job is the entirety of a **JDF** project. Each ▸ Job is organized in a tree structure containing all of the information needed to complete the intended project. The information is collected logically into what is called a **node**. Each node in the tree structure represents an aspect of the ▸ Job to be executed.

The nodes in a ▸ Job are organized in a hierarchical structure that resembles a pyramid. The node at the top of the pyramid describes the overall intention of the ▸ Job. The intermediate nodes describe increasingly process-oriented aspects of the ▸ Job, until the nodes at the bottom of the pyramid each describe a single, simple ▸ Process. Depending on where in the ▸ Job structure a node resides, it can represent a portion of the product to be created, one or many processing steps or other ▸ Job parts. For more information about ▸ Jobs and nodes, see ▸ Section Structure of JDF.

2.1.1.2 Elements

An element is a standard XML syntactic construct ▸ [XML]. (See also: ▸ Section 2.1.1.3 Attributes.) Elements that are subparts of other elements are often referred to as subelements. **JDF** elements are represented by two kinds of data types: element and text element. For more information about elements, see ▸ Section 3.2 JDF.

XML Crash Course

Need a crash course in XML? XML101.com provides online tutorials that non-programmers can easily follow. The site includes examples. See http://xml101.com/

2.1.1.3 Attributes

An attribute is a standard XML syntactic construct ▸ [XML]. (See also: ▸ Section 2.1.1.2 Elements.) Attributes are defined as various different data types, such as **string**, **enumeration**, **dateTime** and so on.

For more information about attributes, see ▸ Section 3.2 JDF. Note that an attribute with an empty (zero length) value string SHALL NOT be specified except when its data type allows an empty string (e.g., when not needed, OPTIONAL attributes are to be omitted rather than included as empty attributes).

2.1.1.4 Relationships

The hierarchical **JDF** structure implies relationships between **nodes** and **elements** within a **JDF** tree structure. The terms used in this document to describe these relationships are defined below, and, in some cases, include a brief representation of the encoding that would express them.

- **Parent**: An element that directly contains a child element.

```
<Parent><Child/></Parent>
```

- **Child**: An element that resides directly in the parent element.
- **Sibling**: An element that resides in the same parent element as another child element.

```
<Any>
    <Sibling/>
</Any>
```

- **Descendent**: An element that is a child or a child of a child, etc.
- **Ancestor**: An element that is a parent or a parent's parent, etc.

```
<Ancestor>
  Any>
    <Descendent/>
  </Any>
  <MoreAnys>
    <Descendent/>
  </MoreAnys>
</Ancestor>
```

- **Root**: The single element that contains all other elements as descendents.
- **Leaf**: Element without further child elements.
- **Branch**: An intermediate node in a hierarchy that contains at least one child node. A branch is never a leaf.

2.1.1.5 Links

There are two kinds of links in **JDF**: internal links and external links. Internal links are pointers to information that is located elsewhere in a **JDF** document. The data that is referenced by the link is located in a target element. External links are used to reference objects that are outside of the **JDF** document itself, such as content files or color profiles. These objects are linked using standard URLs (Uniform Resource Locators).

JDF makes extensive use of links in order to reuse information that is relevant in more than one context of the ‣ Job. The same target can be referenced by multiple links. However, no link references more than one target. See ‣ [URI].

2.1.2 Workflow Component Roles

The components that create, modify, route, interpret and execute a **JDF** job are known as agents, controllers, queues, devices and machines. Overseeing the workflow created by these components is MIS or Management Information Systems. These five aspects of a **JDF** workflow are described in the sections that follow.

By defining these terms, this specification does not intend to dictate to manufacturers how to design, build or implement a **JDF** /**JMF** system.In practice, it is very likely that individual system components will include a mixture of the roles described in the following sections. For example, many controllers are also agents.

2.1.2.1 Machine

A machine is any part of the workflow system designed to execute a process. Most often, this term refers to a piece of physical equipment, such as a press or a binder, but it can also refer to the software components used to run a particular machine or perform a calculation. Computerized workstations, whether run through automated batch files or controlled by a human worker, are also considered machines if they have no **JDF** interface.

Agents, Controllers & Devices

"Agents", "Controllers" and "Devices" are special, logical descriptions. You probably won't ever buy one. An Agent (writes and reads JDF) can be any software tool that can parse JDF. Controllers communicate instructions that Devices act upon. They are functions that can be embedded into your software, production equipment or MIS sys-

2.1.2.2 Device

The most basic function of a ‣ Device is to execute the information specified by an ‣ Agent and routed by a ‣ Controller. devices SHALL be able to execute ‣ JDF Nodes and initiate ‣ Machines that can perform the physical execution. The communication between machines and ‣ Devices is not defined in this specification. ‣ Devices MAY, however, support ‣ JMF messaging in order to interact dynamically with a ‣ Controller.

2.1.2.3 Agent

‣ Agents in a **JDF** workflow are responsible for writing **JDF**. An ‣ Agent has the ability to create or modify a **JDF** node. a ‣ Job, to add ‣ Nodes to an existing ‣ Job, and to modify existing nodes. ‣ Agents can be software processes, automated tools or even text editors. Anything that can be used in composing **JDF** can be considered an ‣ Agent.

Actual implementations of ‣ Devices or ‣ Controllers will most often be able to modify **JDF**. These system components have ‣ Agent properties in the terms of this specification.

2.1.2.4 Queue

Whereas ‣ Device process **JDF** to produce a result, queues provide a method of ordering, prioritizing and scheduling QueueEntrys that represent **JDF** processes. Every ‣ Device that is capable of accepting **JDF** via **JMF** messaging SHALL provide exactly one queue. This specification makes no assumptions on implementation limitations of a queue. Thus a device that can only process a single queue entry and cannot store any waiting queue entries still implements an albeit minimalistic queue.

2.1.2.5 Controller

Agents create and modify **JDF** information; controllers route it to the appropriate Devices. The minimum requirement of a Controller is that it can initiate Processes on at least one Device, or at least one other slave Controller that will then initiate Processes on a Device. In other words, a Controller is not a Controller if it has nothing to control. In some cases, a pyramid-like hierarchy of Controllers can be built, with Controllers at the top of the pyramid controlling a series of lower-level Controllers at the bottom. The lowest-level Controllers in the pyramid, however, SHALL have Device capability. Therefore, Controllers SHALL be able to work in collaboration with other Controllers. In order to communicate with one another, and to communicate with Devices, Controllers SHALL support the **JDF** file-exchange protocol and MAY support JMF. Controllers can also determine Process planning and scheduling data, such as Process times and planned production amounts.

Automating Data Flows

JDF-enabled workflow can require a tremendous amount of information. This could seem daunting to anyone who expects to have to enter information into a system, but it need not be the case. From the style information in a layout file, to automatically generated image file header information, to the color profiles tagged onto images automatically by digital cameras or image editing systems, a great deal of information can be captured and passed along from one JDF-enabled application to another. Furthermore, where, in the specification, there are many options, those options can be set to user-defined default values that represents typical Jobs in your particular workflow. For instance, JDF provides a variety of staple folds. If your plant only supports a crown fold, that becomes the default in your JDF-enabled system and is rarely manually specified or keyed.

2.1.2.6 Management Information System—MIS

The overseer of the relationships between all of the units in a workflow is known as Management Information Systems or MIS. MIS is, in effect, a macrocosmic Controller. It is responsible for dictating and monitoring the execution of all of the diverse aspects of the workflow. To do this, it SHALL remain in contact with the actual production facilities. This can be accomplished either in real time using JMF messaging or post facto using the audit records within a returned **JDF**.

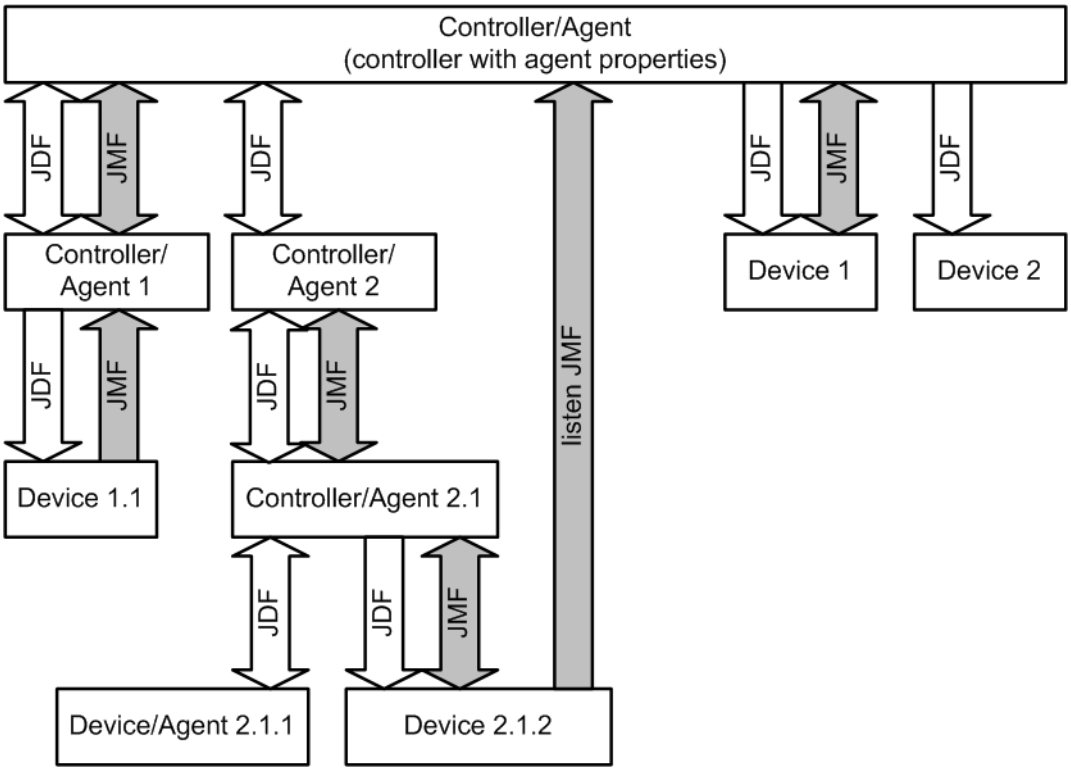
To allow MIS to communicate effectively with the other workflow components, **JDF** supplies what is essentially a messenger service, in the form of JMF, to run between MIS and production. This format is equipped with a variety of Message types, ranging from simple, unidirectional notification to queries and even commands. System designers have a great deal of flexibility in terms of how they choose to use the messaging architecture, so that they can tailor the Processes to the capabilities of the existing workflow mechanism. The Figure 2-1: Example of JDF and JMF workflow interactions depicts how various communication threads can run between MIS and production.

JDF also provides system components the ability to collect performance data for each Node, which can then be passed on to a job-tracking system for use by the MIS system. These data can be derived from the Messages that the Controller receives or from the audit records in the job. For more information on audits, see Section 3.11.4 Audit. Alternatively, the completed job can be passed to the job accounting system, which examines the audit records to determine the costs of all the processes in the Job.

2.1.3 System Interaction

An example of the interaction and hierarchical structure of the components considered in the preceding section is shown in the following figure. Single arrows indicate unidirectional communication channels and double arrows indicate bidirectional communication.

Figure 2-1: Example of JDF and JMF workflow interactions



2.2 JDF Workflow

JDF does not dictate that a workflow be constructed in any pre-specified way. On the contrary, its flexibility has allowed **JDF** to model existing custom solutions for the graphic arts, as well as those yet to be imagined. **JDF** is equally as effective with a simple system using a single controller-Agent and device as it is with a completely automated industrial press workflow with integrated pre- and postpress operations.

Because of workflow system construction in today's industry, the principal subsection procedures of a printing ▸ Job—prepress, press and postpress—remain largely disconnected from one another. **JDF** provides a solution for this lack of unity. With **JDF**, a print ▸ Job becomes an interconnected workflow that runs from job submission through trapping, RIPing, filmmaking, platemaking, inking, printing, cutting, binding, and sometimes even through shipping. **JDF** enables an architecture that defines the process necessary to produce each intended result and identifies the elements necessary to complete the processes. All processes are separated into nodes, and the entire ▸ Job is represented by a tree of these nodes. All of the nodes taken together represent a desired printed product.

Each individual node in **JDF** is defined in terms of inputs and outputs. The inputs for a node consist of the resources it uses and the parameters that control it. For example, the inputs in a node describing the process parameters for imaging the cover of a brochure might include requirements for trapping, raster image processing, and imposing the image. The output of such a node might be a raster image.

Unless they represent the absolutely final product, resources that are produced by one node are in turn modified or consumed by subsequent nodes. For instance, the output of the process described above—the raster image—becomes one of the input resources for a node describing the printing process for the brochure. This input resource would be joined in the node by other input resources such as inks, press sheets, plates and a set of parameters that indicate how many sheets to produce. The output would be a set of printed press sheets that in turn would become the input resource for postpress operations such as folding and cutting. And so on until the brochure is completed.

This system of interlinked nodes effectively unites the prepress, press and postpress processes, and even extends the notion of where a ▸ Job begins. A **JDF** job, like any printing job, is defined by the original intent for the end product. The difference between a **JDF** ▸ Job and a generic printing job, however, is that **JDF** allows the entire ▸ Job, from prepress through postpress, to be defined up front. All of the resources and processes necessary to produce an entire printed product can be identified and organized into nodes before the first prepress process is set in motion. Furthermore, the Product Intent specification can be extremely broad *or* extremely detailed, or anywhere in between. This means that a ▸ Job can be so well defined before production begins that the system administrator only has to set the wheels in motion and let the ▸ Job run its course. It might also mean that the person submitting the ▸ Job has only a general idea of what the final product will look like and that modifications to the intent will be made along the way, depending on the course of the ▸ Job.

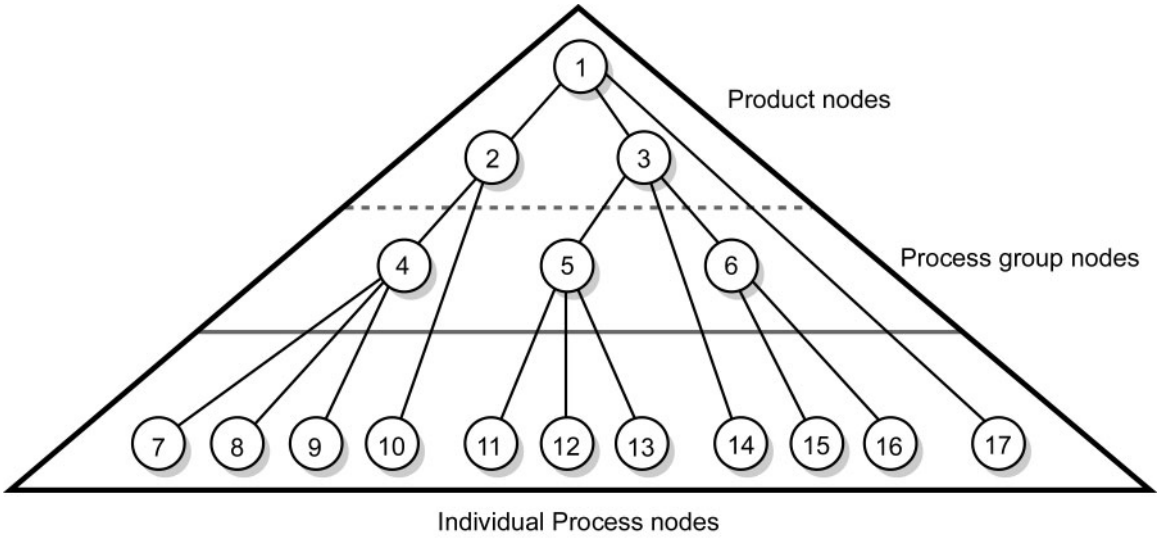
For example, the person submitting the ▸ Job specification for the brochure described above might know that she wants 400 copies, that she wants it done on a four-color press with no spot colors, that the cover will be on a particular paper

stock and the contents on another, that the binding will be stapled, and that she requires the ▸ Job in two weeks. Another person might know only that he wants the pages she's designed to be put into some sort of brochure form, although she doesn't know exactly what. Either person's request can be translated into a **JDF** Product Intent node that will eventually branch into a tree structure describing each process needed to complete the brochure. In the first example, the prepress, press and postpress processes will be well defined from the start. In the second example, information will be included as it is gathered. The following sections describe the way in which nodes can combine to form a ▸ Job.

2.2.1 Job Structure

JDF ▸ Jobs consist of a set of nodes that specify the production steps needed to create the desired end product. The nodes, in addition to being connected through inputs and outputs, are arranged in a hierarchical tree structure. ▸ Figure 2-2: JDF tree structure, below, shows a simple example of a tree of nodes.

Figure 2-2: JDF tree structure



The following table provides a hypothetical breakdown of the nodes in the tree structure shown above.

Table 2.1: Information contained in JDF Nodes, arranged numerically (Sheet 1 of 2)

NODE #	MEANING
1	Entire book
2	Cover
3	Contents
4	Production of cover
5	Production of all color pages
6	Production of all black-and-white pages
7	Cover production Process 1
8	Cover production Process 2
9	Cover production Process 3
10	Cover Finishing Process
11	RIPing for color pages
12	Plate making for color pages
13	Printing for color pages
14	Color page finishing Process
15	RIPing for black-and-white pages

Table 2.1: Information contained in JDF Nodes, arranged numerically (Sheet 2 of 2)

NODE #	MEANING
16	Printing for black-and-white pages on a digital press
17	Binding Process for entire book

The uppermost Nodes (1, 2, & 3) represent the Product Intent in general terms. These nodes describe the desired end product and the components of that product, which, in this case, are the cover and the content pages. As the tree branches, the information contained within the nodes gets more specific. Each subnode defines a component of the product that has a unique set of characteristics, such as different media, different physical size or different color requirements. The nodes that occur in the middle of the tree (4, 5, & 6) represent the groups of processes needed to produce each component of the product. The nodes that occur closest to the bottom of the tree (7–17) each represent individual processes.

In this example, there are two subcomponents of the ▸ Job, the cover and the contents, each with distinct requirements. Therefore, two nodes—Nodes 2 and 3—are needed to describe the elements of the ▸ Job in broad terms. Within the content pages there are some black-and-white pages and some color pages. Since fabricating each requires a different set of processes, further branching is necessary. The following table arranges the nodes in groups according to the processes they will be executing.

Table 2.2: Information contained in JDF Nodes, arranged by group

PROCESS GROUP	NODE #	MEANING
Entire book	1	Entire book
	17	Assemble book
Cover	2	Cover
	4	Cover assembly Processes
	7	Cover production Process 1
	8	Cover production Process 2
	9	Cover production Process 3
	10	Finishing Process for cover
Contents	3	Contents
Color Pages	5	Production of all color pages
	11	RIPing for color pages
	12	Plate making for color pages
	13	Printing for color pages
	14	Color page finishing
Black-and-white pages	6	Production of all black-and-white pages
	15	RIPing for black-and-white pages
	16	Printing for black-and-white pages on a digital press

This hierarchical structure is discussed in more detail in the following section.

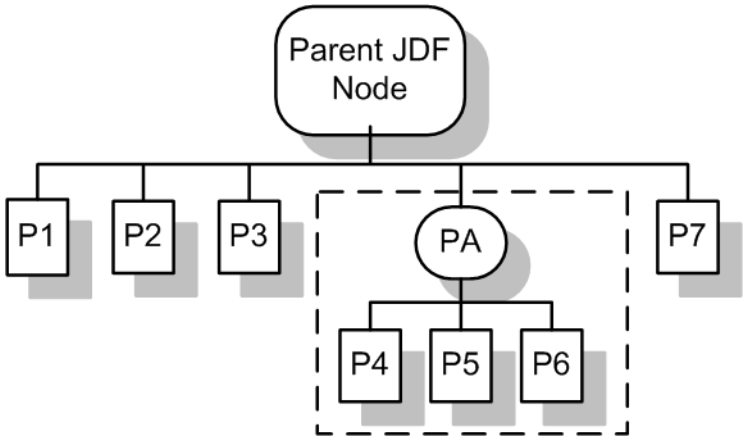
2.3 Hierarchical Tree Structure and Networks in JDF

Output resources of **JDF** nodes are often the input resources for other **JDF** nodes. Nodes SHALL NOT begin executing until all of their input resources are complete and available. This means that the nodes execute in a well defined sequence. One process follows the next. For example, a process for making plates will produce, as output resources, press plates that are needed by a ***ConventionalPrinting*** process.

In the hierarchical organization of a **JDF** ▸ Job, nodes that occur higher in the tree represent high level, more abstract operations, while lower nodes represent more detailed process operations. More specifically, nodes near the top of the tree can represent only intent regarding the components or assemblies that make up the product, while the leaf nodes provide explicit

instructions to a device to perform some operation. ▸ Figure 2-3: Example of a hierarchical tree structure of JDF Nodes shows an example of a hierarchical structure.

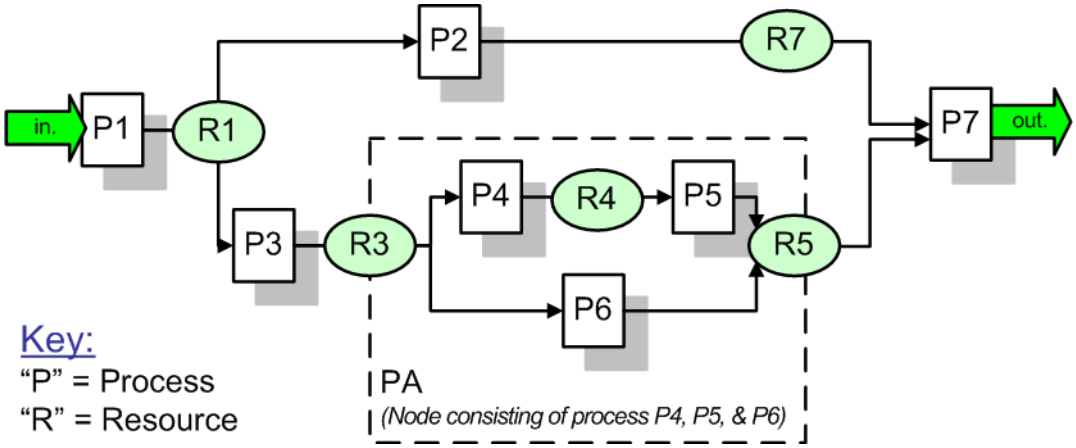
Figure 2-3: Example of a hierarchical tree structure of JDF Nodes



In addition to the hierarchical structure of the node tree, sibling nodes are linked in a process chain by their respective resources. In other words, an output resource of one node ends up representing the input resource of the following node (as represented in ▸ Figure 2-4: Example of a Process chain linked by Input Resources and Output Resources). This interrelationship is known as resource linking.

With resource linking, complex networks of processes can be formed. The ▸ Figure 2-4: Example of a Process chain linked by Input Resources and Output Resources displays an alternate representation of the process described in ▸ Figure 2-3: Example of a hierarchical tree structure of JDF Nodes. Whereas ▸ Figure 2-3: Example of a hierarchical tree structure of JDF Nodes represents a hierarchical structure, ▸ Figure 2-4: Example of a Process chain linked by Input Resources and Output Resources shows an example of the linking mechanism of the same ▸ Job. Note that there are many possible process networks that map to the same node hierarchy.

Figure 2-4: Example of a Process chain linked by Input Resources and Output Resources



In the **JDF** specification, the linking of processes is not explicitly specified. In other words, nodes are not arranged in an abstract chronology, dictating, for example, that the trapping node is to come before the RIPing node. Rather, the links are implicitly defined in the exchange of input and output resources. Resource dependencies form a network of processes, and the sequence of process execution—that is, the routing of processes—can be derived from these dependencies. One resource dependency might have the possibility of multiple process routing scenarios. It is up to MIS to define the proper solution to meet local constraints. Note that the type of exchange resource effectively limits the processes that can be linked.

The agent or set of agents employed by MIS to write the **JDF** ▸ Job SHALL be familiar with these local constraints. They SHALL take into account factors such as the control abilities of the applications that complete the prepress processes, the transport distance between the prepress facility and the press itself, the load capabilities of the press, and the time requirements for the ▸ Job. All of the factors taken together build a process network representing the workflow of production. To aid agents in defining the workflow, **JDF** provides the following four different and fundamental types of process mechanisms, which can be combined in any way.

1 **Serial processing** that is subsequent production and consumption of resources as a whole, represented by a simple process chain

2 **Overlapping processing** that is simultaneous production and consumption of resources by pipes

3 **Parallel processing** that involves the splitting and sharing of resources

4 **Iterative processing** that is a circular or back and forward processing for developing resources by repeated activity

These mechanisms are discussed in greater detail in ▸ Section 4.3 Execution Model.

2.4 Role of Messaging in JDF

Whereas **JDF** provides a container to define a ▸ Job, the Job Messaging Format — **JMF**, defined in Chapter 5, ▸ JMF Messaging Format — provides a method to generate snapshots of ▸ Job status and to interactively manipulate elements of a workflow system.

JMF is specifically designed for communication between the production system controller and the work centers or devices with which it interacts. It provides a series of queries and commands to check the status of processes and, in some cases, to dictate the next course of action. For example, the ***KnownDevices*** query message allows the controller to determine what processes can be executed by a particular device or work center. These processes are likely to be determined at system initialization time. The ***SubmitQueueEntry*** message provides a means for the controller to submit a ▸ Job ticket to individual work centers or devices. And the ***Status*** and ***Resource*** messages allow the device or work center to communicate quasi real-time[1] processing status to a controller. Depending on the system configuration, the message handler can choose to record status changes in the history logs. The status message allows the controller to request status updates from the controller.

JDF also provides mechanisms to define recipients for individual messages on a node-by-node basis. This enables controllers to define the aspects and the parts of ▸ Jobs that they want to track. For more information about messaging, see ▸ Chapter 5 JMF Messaging Format.

2.5 Coordinate Systems in JDF

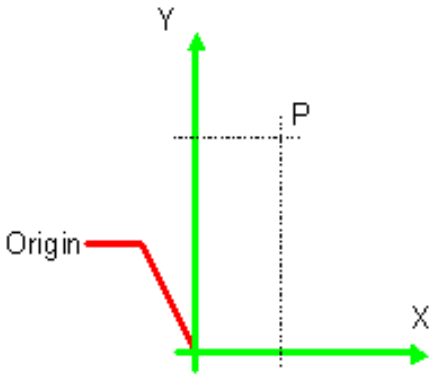
This chapter explains how coordinate systems are defined and used in **JDF** . It also shows how the matrices are used to specify a certain transformation and how these matrices can be used to transform coordinates from one coordinate system to another coordinate system. In addition, it clarifies the meaning of terms like "Top" or "Left".

2.5.1 Introduction

During the production of a printed product it often happens that one object is placed onto another object. During imposition, for example, single pages and marks (like cut, fold or register marks) are placed on a sheet surface. Later, at image setting, a bitmap containing one separation of a sheet surface is imposed on a piece of film. In a following step, the film is copied to a printing plate which then is mounted on a press. In postpress, the printed sheets are gathered on a pile. The objects involved in all these operations have a certain orientation and size when they are put together. In addition, one has to know *where* to place one object on the other.

The position of an object (e.g., a cut mark) on a plane can be specified by a two-dimensional coordinate. Every digital or ***PhysicalResource*** has its own coordinate system. The origin of each coordinate system is located in the lower left corner (i.e., the X coordinate increases from left to the right, and the Y coordinate increases from bottom to top).

Figure 2-5: Standard coordinate system



Each page contained in a PDL file has its own coordinate system. In the same way a piece of film or a sheet of paper has a coordinate system. Within **JDF** each of these coordinate systems is called *Resource coordinate system*.

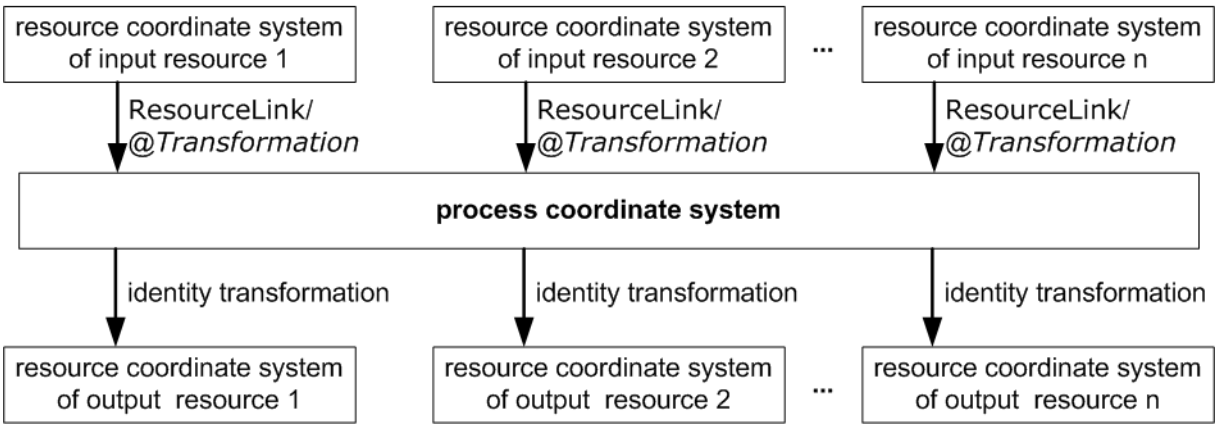
If a process has more than one input resource with a coordinate system, it is necessary to define the relationship between these input coordinate systems. Therefore, a *process coordinate system* is defined for each process. **JDF** tickets are written

1. Quasi real-time is the time-scale typically associated with production control systems. **JMF** is not intended for true real-time, lower level machine control.

assuming an idealized device that is defined in the process coordinate system for each process that the device implements. A real device SHALL map the idealized process coordinate system to its own device coordinate system.

The coordinate systems of the input resources are mapped to the process coordinate system. Each of those mappings is defined by a transformation matrix, which specifies how a coordinate (or position) of the input coordinate system is transformed into a coordinate of the target coordinate system. (See ▸ Section 2.5.7 Homogeneous Coordinates for mathematical background information.) In the same way, the mapping from the process coordinate system to the coordinate systems of the output resources is defined. The process coordinate system is also used to define the meaning of terms like "Top" or "Left", which are used as values for parameters in some processes.

Figure 2-6: Relation between Resource and process coordinate systems



It is important that no implicit transformations (such as rotations) are assumed if the dimensions of the input resources of a process do not match each other. Instead every transformation (e.g., a rotation) SHALL be specified explicitly by using the *@Orientation* or *@Transformation* attribute of the corresponding ***ResourceLink***.The same applies also to other areas in **JDF** (e.g., the **Interpreting** process). A ***FitPolicy*** element MAY define a policy for implied transformations.

2.5.1.1 Source Coordinate Systems

The source coordinate system of a referenced object is defined by the lower left of the object. X values are increasing to the right, Y values are increasing towards the top. In case of PDF the lower left of the MediaBox defines the lower left of the source coordinate system.

Note: Some object coordinate systems have optional tags to indicate internal transformations. These internal transformations SHALL be applied prior to defining the source coordinate system; for instance:

- PDF: the rotation defined by the Rotate key SHALL be applied. The lower left of the MediaBox of the rotated PDF defines the lower left of the PDF source coordinate system.
- TIFF: the orientation defined by the Orientation tag SHALL be applied. The lower left of the rotated TIFF defines the lower left of the TIFF source coordinate system.

2.5.2 Coordinates and Transformations

Table 2.3: Data types for specifying coordinates and transformation

DATA TYPE	EXAMPLE
XYPair	"612 792"
double	"20.7"
rectangle	"0 0 595 843" (Order of elements is "lower-left x, lower-left y, upper-right x, upper-right y" or "left, bottom, right, top".)
Matrix	"1 0 0 1 30.0 235.3" The ordering of elements is defined in ▸ Section 2.5.7 Homogeneous Coordinates.
Orientation	"Rotate180" or "Flip90"

Coordinates and transformations are used throughout **JDF**, to include:

Intent Resources, such as:

- ***LayoutIntent***: specifies size of finished product
- ***MediaIntent***: specifies size of media
- ***InsertingIntent***: specifies rotation and offset of inserts

Process Resources, such as:

- ***Component***: specifies coordinate system
- ***CutBlock***: specifies cut block coordinate system
- ***FoldingParams***: specifies folding operations

2.5.3 Coordinate Systems of Resources and Processes

Each physical input resource (e.g., ***Component***) of a process has, by default, its own coordinate system, which is called the "resource coordinate system." The coordinate system also implies a specific orientation of that ***Resource***. On the other hand there is a coordinate system that is used to define various process-specific parameters. This coordinate system is called a target or process coordinate system.

It is often necessary to change the orientation of an input resource before executing the operation. This can be done by specifying a transformation matrix. It is stored in the @*Orientation* or @*Transformation* attribute of the ***ResourceLink***. This provides the ability to specify different matrices for the individual resources of a process. For details on ***ResourceLink*** elements, see ▸ Section 3.9 ResourceLinkPool and ResourceLink.

2.5.3.1 Use of Preview to Display Resource Orientation

It is often necessary to load printed material into finishing equipment manually. Particularly in the case of imposed sheets, the page orientation will not be unique and even the concept of "Front" or "Back" can be confusing, since front and back pages can be printed on the same surface of the imposed sheet. ***Preview*** resources with ***Preview***/@*PreviewUsage*="ThumbNail" or ***Preview***/@*PreviewUsage*="Viewable" SHOULD be provided to illustrate the desired orientation of the input components with respect to the device.

2.5.3.2 Coordinate Systems of Combined Processes

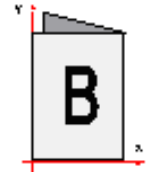
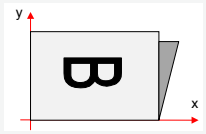
New in JDF 1.2

Combined Processes (see ▸ Section 3.3.3 Combined Process Nodes) combine multiple individual processes and thus also that processes' respective coordinate systems. The process coordinate systems are not modified by the fact that the processes are part of a combined process, they are identical to the process coordinate systems of the processes, were they defined in a linked chain of individual processes. The coordinate systems of an exchange resource can be modified by defining it as a pipe by specifying ***Resource***/@*PipeID* and ***Resource***/@*PipeProtocol* = "Internal" (See ▸ Section 4.3.3 Overlapping Processing Using Pipes) and linking it to the combined process with both an input and output ***ResourceLink***. The input ***ResourceLink*** defines the coordinate transformation using the standard @*Transformation* or @*Orientation* attributes. ***Resource***/@*Status* of the exchange resource SHALL be "Complete".

2.5.4 Coordinate System Transformations

The following table shows some matrices that can be used to change the orientation of a ***PhysicalResource*** . Most of the transformations require the X- (***w***) and the Y-dimension (***h***) of the ***Component*** as specified in the @*Dimensions* attribute. If these are unknown, it is still possible to define a general orientation in the @*Orientation* attribute of the ***ResourceLink***. The naming of the attribute reflects the state of the resource and not necessarily the order of applied transformations. Thus "Rotate90" and "Flip90" specify that the original Y axis as represented by the spine is on top. In the case of Flip90, the ***Component*** is additionally flipped front to back.

Table 2.4: Matrices and Orientation values for describing the orientation of a Component

ORIENTATION VALUE	SOURCE COORDINATE SYSTEM	TRANSFORMATION MATRIX ACCORDING ACTION	TARGET COORDINATE SYSTEM
Rotate0		1 0 0 1 0 0 No Action	
Rotate90		0 1 -1 0 *h* 0 90° Counterclockwise Rotation	
Rotate180		-1 0 0 -1 *w* *h* 180° Rotation	
Rotate270		0 -1 1 0 0 *w* 270° Counterclockwise Rotation	
Flip0		1 0 0 -1 0 *h* Flip around X	
Flip90		0 -1 -1 0 *h* *w* 90° Counterclockwise Rotation + Flip around X	
Flip180		-1 0 0 1 *w* 0 180° Rotation + Flip around X	
Flip270		0 1 1 0 0 0 270° Counterclockwise Rotation + Flip around X	

2.5.5 Product Example: Simple Brochure

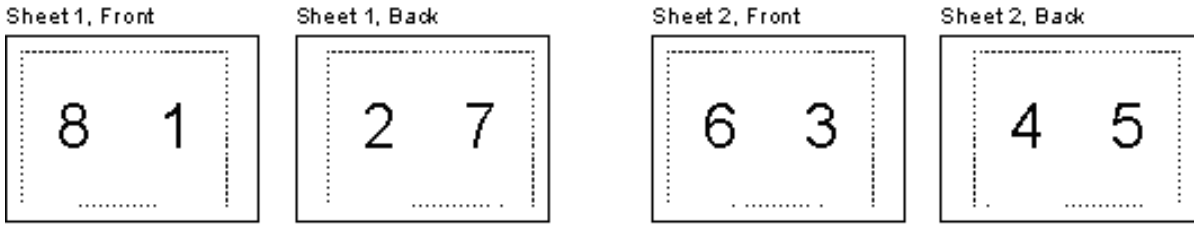
To illustrate the use of coordinate systems in **JDF**, a simple saddle stitched brochure with eight pages is used as an example in ▸ Table 2.5 JDF Processes used for the production of the simple brochure. The brochure is printed on two sheets with front and back. The two sheets are then folded, collected on a saddle, and saddle stitched. Finally the brochure is cut with a three-side trimmer.

Table 2.5: **JDF** *Processes used for the production of the simple brochure*

INPUT RESOURCES	PROCESS	OUTPUT RESOURCES
Layout ***RunList*** (Document) ***RunList*** (Marks)	**Imposition**	***RunList***
RunList	**Interpreting**	***RunList*** (of interpreted PDL data)
RunList (of interpreted PDL data) ***Media*** ***RenderingParams***	**Rendering**	***RunList*** (of rasterized byte maps)
RunList (of rasterized byte maps)	**Screening**	***RunList*** (of bit maps)
ImageSetterParams ***Media*** (of film) ***RunList*** (of bit maps)	**ImageSetting** (to film)	***ExposedMedia*** (of film)
ExposedMedia (of film)	**ContactCopying**	***ExposedMedia*** (of plate)
ExposedMedia (of plate) ***ConventionalPrintingParams***	**Conventional Printing**	***Component***
FoldingParams ***Component***	**Folding**	***Component***
CollectingParams ***Component***	**Collecting**	***Component***
StitchingParams ***Component***	**Stitching**	***Component***
TrimmingParams ***Component***	**Trimming**	***Component***

At imposition, the layout describes a signature with two sheets, each having a front and a back surface. On each surface, two content objects (i.e., pages, are placed).

Figure 2-7: Layout of simple saddle stitched brochure (product example)



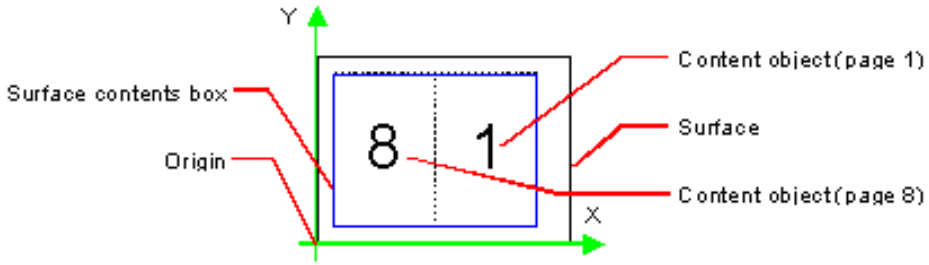
Each surface has its own coordinate system, in which a surface contents box is defined. This coordinate system is also referred to as the ***Layout*** coordinate system because the signature, sheet and surface elements are defined within the hierarchy of the ***Layout*** resource by means of partitioning. The content objects are placed by specifying the CTM attribute relative to the surface contents box. If the position of an object within a page is given in the page coordinate system, this coordinate can be transformed into a position within the surface coordinate system:

Figure 2-8: Equation for Surface Coordinate System Transformations

$$P_{Surface} = P_{Page} \times CTM_{Page} + [SurfaceContentsBox_{Xlowerleft}\ SurfaceContentsBox_{Ylowerleft}\ 0]$$

Please note, that the width and height of the surface NEED NOT be known at this point.

Figure 2-9: Surface coordinate system



The sheet coordinate system is identical with the coordinate system of the front surface. This means that no transformation is needed to convert a coordinate from one system to the other. Instead, the coordinates are valid (and equal) in both coordinate systems. The relation between the coordinate system of the front and the back surfaces depends on the value of the ***Layout***/@*LockOrigins* attribute. The sheet coordinate system is also identical with the signature coordinate system, which in turn is identical with the coordinate system of the **Imposition** process.

The output resource of the **Imposition** process is a run list. Each element of the run list has its own coordinate system, which is identical with the corresponding signature coordinate system. The interpretation, rendering and screening processes do not affect the coordinate systems. This means that the coordinate systems of all these processes are identical.

At the **ImageSetting** process, the digital data is set onto film. The process coordinate system is defined by the ***Media*** input resource. The width and height of the media are defined in the ***Media***/@*Dimension* attribute. The position of the signatures (as defined by the ***RunList*** input resource) on the film is defined by the ***ImageSetterParams***/@*CenterAcross* attribute.

The coordinate system of the conventional and digital printing processes is called *press coordinate system*. It is defined by the press: the X-axis is parallel to the press cylinder, and the Y-axis is going along the paper travel. Y = 0 is at begin of print, X = 0 is at the left edge of the maximum print area. The front side of the press sheet faces up towards the positive Z-axis. The relationship between the layout coordinate system and the press coordinate system is defined by the @*CTM* attributes of the corresponding ***TransferCurveSet*** elements located in the ***TransferCurvePool***.

Figure 2-10: Press coordinate system used for Sheet-Fed Printing

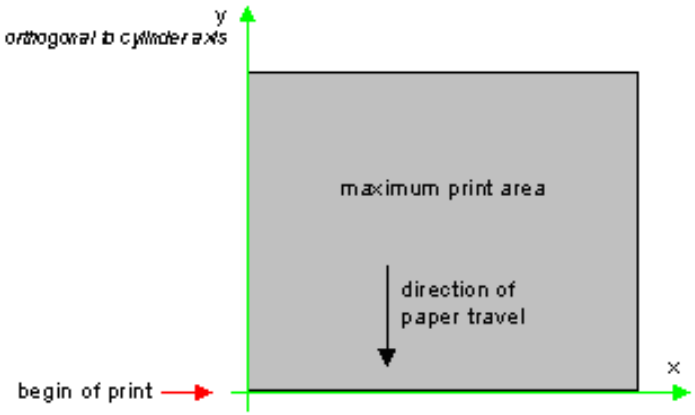
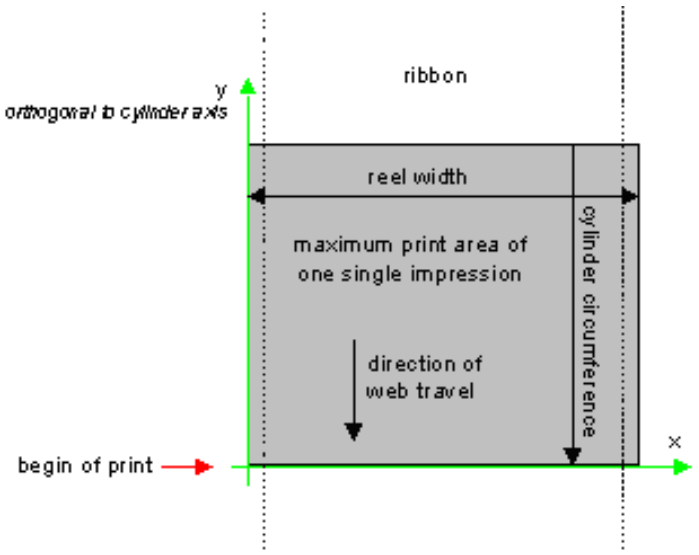


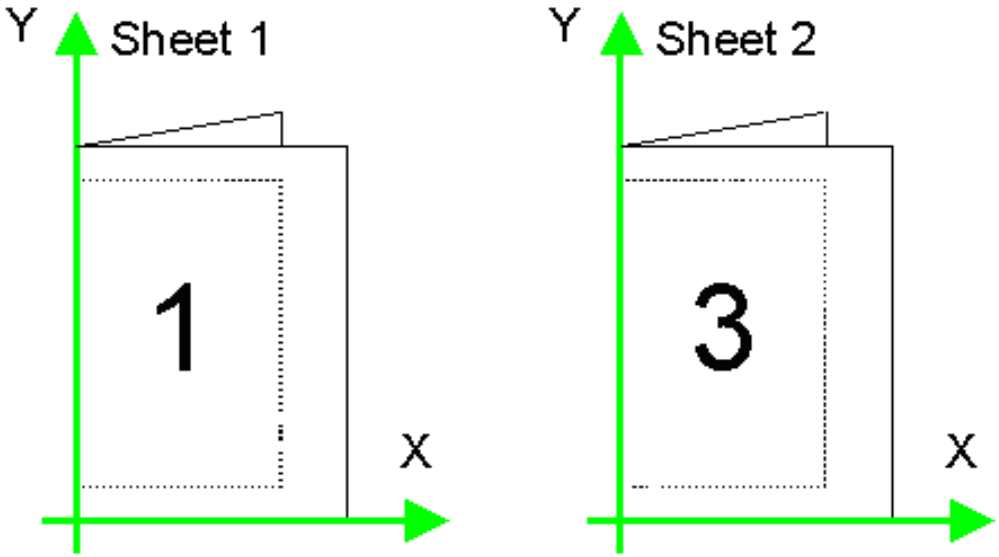
Figure 2-11: Press coordinate system used for Web Printing



The output of the printing process (e.g., a pile of printed sheets) is described as a ***Component*** resource in **JDF**. The coordinate system of the printed sheets is defined by the transformation given in the ***TransferCurveSet***/@*CTM* attribute (where @*Name* = "Paper").

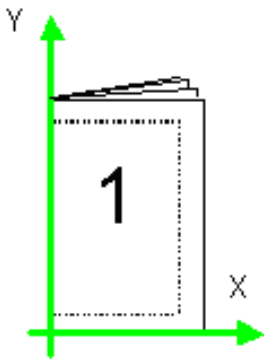
Each of the two sheets is folded in a separate **Folding** process. In this example, the orientation of the sheets is not changed before folding. This can be specified by setting the @*Orientation* attribute of the ***Component*** input resource to "Rotate0" or by setting the @*Transformation* attribute to "1 0 0 1 0 0". The **Folding** process changes the coordinate system. In this example the origin of the coordinate system is moved from the lower left corner of the flat sheet (input) to the lower left corner of the folded sheet (output) (i.e., it is moved to the right by half of the sheet width).

Figure 2-12: Coordinate systems after Folding (product example)



The two folded sheets are now collected. In this example, the orientation of the folded sheets is not changed before collecting. This can be specified by setting the @*Orientation* attribute of the ***Component*** input resource to "Rotate0" or by setting the @*Transformation* attribute to "1 0 0 1 0 0". The ***Collecting*** process does not change the coordinate system.

Figure 2-13: Coordinate systems after Collecting (product example)



The two collected and folded sheets are now trimmed to the final size of the simple brochure. In this example, the orientation of the collected and folded sheets is not changed before trimming. This can be specified by setting the @*Orientation* attribute of the ***Component*** input resource to "Rotate0" or by setting the @*Transformation* attribute to "1 0 0 1 0 0". The **Trimming** process changes the coordinate system: the origin is moved to the lower left corner of the trimmed product.

In looking at the whole production process, a series of coordinate systems is being involved. The relationship between the separate coordinate systems is specified by transformation matrices. This allows transformation of a coordinate from one coordinate system to another coordinate system. As an example, note the position of the title on page 1 of the product example in ▸ Figure 2-13: Coordinate systems after Collecting (product example). By applying the first transformation, this position can be converted into a position of the surface (or layout) coordinate system. This position can then be converted into the paper coordinate system by applying (in this order) the "Film", "Plate", "Press" and "Paper" transformations stored in the ***TransferCurvePool***.

From now on in the workflow, every process is using one or more ***Component*** resources as input and output resources. The ***ResourceLink*** of each input and output ***Component*** contains a @*Transformation* attribute or an @*Orientation* attribute. The @*Transformation* attribute SHALL be used if the width and the height of the ***Component*** are known or a non-orthogonal rotation is needed. Otherwise the @*Orientation* attribute SHOULD be used to specify a change of the orientation (e.g., an orthogonal rotation).

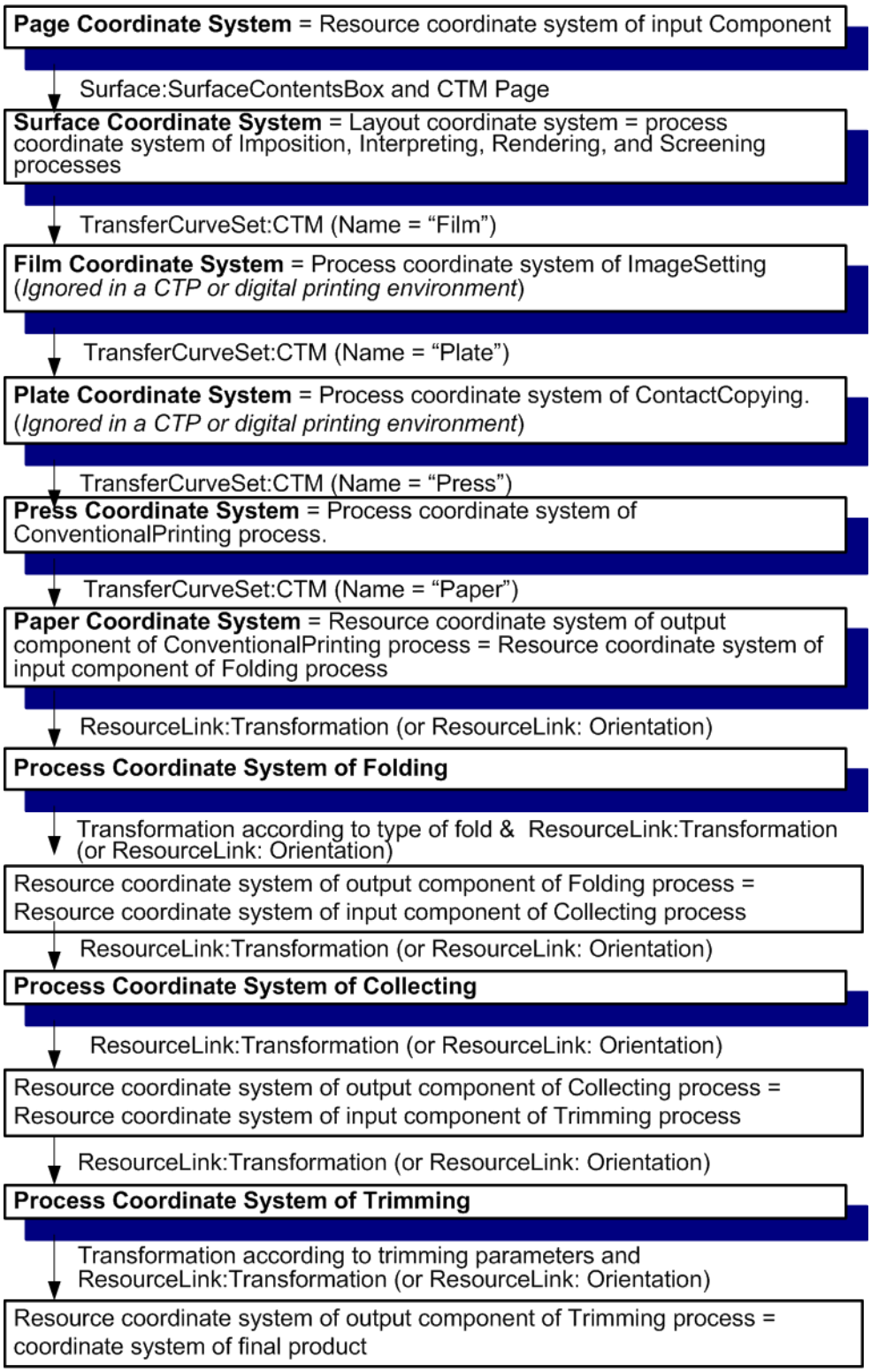
Since the **Folding** process changes the coordinate system depending on the fold type, the transformations specified in the ***ResourceLink*** elements are not sufficient to transform a position given in the paper coordinate system to a position in the coordinate system of the folded sheets (i.e., the resource coordinate system of the output component of the **Folding** process). An additional transformation depending on the fold type and details of the individual folds has to be applied. The corresponding transformation matrix is not explicitly specified in the **JDF** file.

The **Collecting** process does not change the coordinate system. Therefore, only the transformations specified in the ***ResourceLink*** elements of the ***Component*** input and output resources (i.e., components have to be applied).

The ***Trimming*** process again changes the coordinate system depending on the trimming parameters. Therefore, a transformation depending on the trimming parameters has to be applied in addition to the transformations specified in the ***ResourceLink*** elements. The matrix for the additional transformation (depending on the trimming parameters) is not explicitly specified in the **JDF** file.

After having applied all transformations mentioned above, the resulting coordinate specifies the position of the title in the coordinate system of the final product.

Figure 2-14: Examples of Transformations and Coordinate Systems in **JDF**.



2.5.6 General Rules

The following rules summarize the use of coordinate systems in **JDF** .

- Every individual piece of material (film, plate, paper) has a *resource coordinate system.*
- Every process has a *process coordinate system.*
- Terms like *top*, *left*, etc., are used with respect to the *process coordinate system* in which they are used and are independent of orientation (i.e., *landscape* or *portrait*), and the human reading direction.
- The coordinate system of each input component is mapped to the process coordinate system.
- The coordinate system might change during processing (e.g., in **Folding**).

- The description of a product in **JDF** is independent of the particular ▸ Machine used to produce this product. When creating setup information for an individual ▸ Machine, it might be necessary to compensate for certain ▸ Machine characteristics. At printing, for example, it might be necessary to rotate a landscape job because the printing width of the press is not large enough to run the job without rotation.

2.5.7 Homogeneous Coordinates

A convenient way to calculate coordinate transformations in a two-dimensional space is by using so-called homogeneous coordinates. With this concept, a two-dimensional coordinate P=(x,y) is expressed in vector form as [x y 1]. The third element "1" is added to allow the vector being multiplied with a transformation matrix describing scaling, rotation, and translation in one shot. Although this only requires a 2*3 matrix (e.g., as it is used in PostScript) in practice 3*3 matrices are much more common, because they can be concatenated very easily. Thus, the third column is set to "0 0 1".

$$Trf = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$$

Which in JDF is written as
@Trf="a b c d e f"

Commonly used transformation matrices are:

$$Trf = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Identity transformation

$$Trf = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{bmatrix}$$

Translation by dx, dy

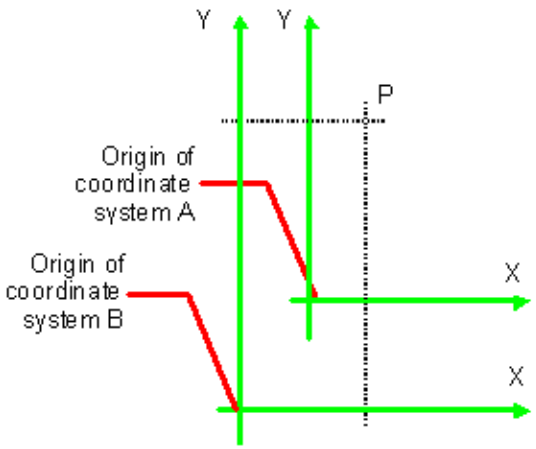
$$Trf = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation by φ degrees counter-clockwise

Transforming a point

In this example, the position P given in the coordinate system A is transformed to a position of coordinate system B. The relationship between the two coordinate systems is given by the transformation matrix *Trf* .

Figure 2-15: Transforming a point (example)



Transformation sequence

$$P_A = \begin{bmatrix} 300 & 100 & 1 \end{bmatrix}$$

Starting position
P_A = (30, 100)

$$P_B = P_A \times Trf$$

Transformation

$$P_B = \begin{bmatrix} 30 & 100 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 40 & 60 & 1 \end{bmatrix}$$

Expanded translation transformation. In JDF, Trf is written as @*Trf*="1 0 0 1 40 60"

$$P_B = \begin{bmatrix} 70 & 160 & 1 \end{bmatrix}$$

Result position
PB = (70, 160)

Structure of JDF

This chapter describes the structure of **JDF** nodes and how they interrelate to form a job. As described in ▸ Section 2.1.1 Job Components, a node is a construct, encoded as an XML element, that describes a particular part of a **JDF** job. Each node represents an aspect of the job in terms of:

1 A process necessary to produce the end result, such as imposing, printing or binding;

2 A product that contributes to the end result, such as a brochure; or

3 Some combination of the previous two.

In short, a node describes a product intent or a process step.

In addition to describing the structure of an individual **JDF** node, this chapter examines in what way those nodes interact to form a coherent job structure. The visual correlative of this structure resembles a family tree with a single node describing the entire job at the top, and a number of nodes at the bottom that each describes only one specific process. **JDF**-supported, leaf-level processes are described in ▸ Chapter 6 Processes.

Resource linking specifies the transformation of input resources into output resources, which in turn might become inputs of other nodes. It also allows nodes to share the same resource. The combination of hierarchical nesting of nodes and resource linking allows complex process networks to be constructed. In a simple case, however, a **JDF** instance MAY contain only one node. The only way that a **JDF** node can identify its input and output resources is by using ***ResourceLink*** elements.

The hierarchical structure of a **JDF** job achieves a functional grouping of processes. For example, a job might be split into a prepress node, a press node and a finishing node that contain the respective process nodes. Each and every node in turn contains attributes that represent various characteristics of that node. Nodes also contain subelements of certain types, such as resources, process information, customer information, audits, logging information and other **JDF** nodes. Some elements, such as those that deal with customer information, typically occur in the root structure, while other elements, such as resources, MAY occur anywhere in the tree. Where the elements can reside depends on their type and their usage scope.

This chapter describes the elements, subelements and attributes commonly found in **JDF** nodes, and provides the characteristics necessary to understand where each belongs and how it is used. Many of these characteristics are presented in tables, and each of these tables includes the following three columns.

- **Name** — Identifies the element being discussed.
- **Data Type** — Refers to the data type, all of which are described in ▸ Section 1.6 Data Structures. Only the data types **element** and **text element** are applied to elements. All other types are attributes.
- **Description** — Provides detail about the element or attribute being discussed.

The **JDF** workflow model is based on a resource/consumer model. **JDF** nodes are the consumers that are linked by input resources and output resources. The ordering of siblings within a node, however, has no effect on the execution of a node. All chronological and logical dependencies are specified using ***ResourceLink Resource*** elements, which are defined in ▸ Section 3.9 ResourceLinkPool and ResourceLink ▸ Section 3.8.2 Resource.

3.1 Generic Contents of All Elements

JDF contains a set of generic structures that MAY occur in any element of a **JDF** or **JMF** document. Some of these are provided as containers for human-readable comments and descriptions and are described below. Others define the usage policy for attributes and subelements.

Table 3.1: Any Element (generic content) (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BestEffortExceptions ? New in JDF 1.1	NMTOKENS	The names of the attributes in this element that are to have the best effort policy applied when @*SettingsPolicy* is not "BestEffort".
CommentURL ?	URL	URL to an external, human-readable description of the element. Note that @*CommentURL* MAY be specified within a ***Comment***.

Table 3.1: Any Element (generic content) (Sheet 2 of 2)

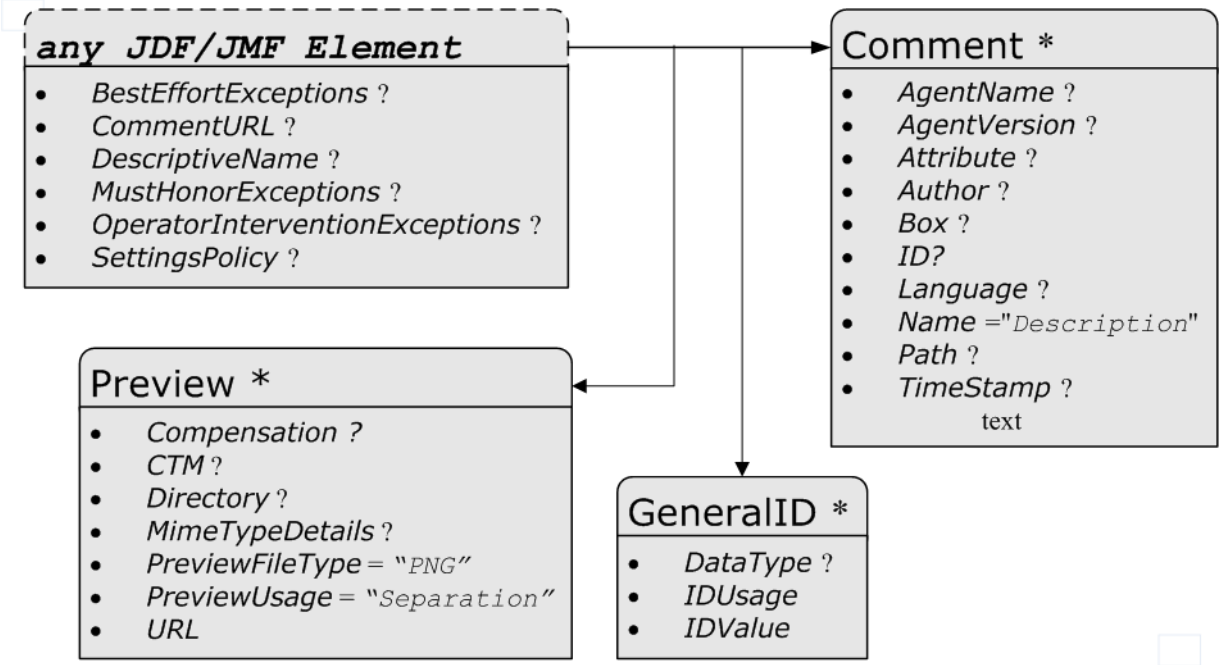
NAME	DATA TYPE	DESCRIPTION
DescriptiveName ?	string	Human-readable descriptive name of the **JDF** element (e.g., a descriptive name of a resource, process or product intent). It is strongly RECOMMENDED to supply @*DescriptiveName* in all **JDF** nodes, ***Quantity Resource***s (for example: ***Component*** resources) and ***Handling Resource***s (for example, ***ExposedMedia***) for communication from applications to humans in order to reference the process or resource.
MustHonorExceptions ? New in JDF 1.1	NMTOKENS	The names of the attributes in this element that are to have the "MustHonor" policy applied when @*SettingsPolicy* is not "MustHonor".
OperatorInterventionExceptions ? New in JDF 1.1	NMTOKENS	The names of the attributes in this element that are to have the operator intervention policy applied when @*SettingsPolicy* is not "OperatorIntervention". If a device has no operator intervention capabilities, @*OperatorIntervention* is treated as "MustHonor".
SettingsPolicy ? New in JDF 1.2	enumeration	The policy for this element indicates what happens when unsupported settings (i.e., subelements, attributes or attribute values) are present in the element. **Default value is from:** parent's @*SettingsPolicy*. If not specified in the parent element or further superior elements, the default value is "BestEffort". **Allowed values are:** BestEffort – Substitute or ignore unsupported attributes, attribute values, default attribute values or elements, and continue processing the job. MustHonor – Reject the job when any unsupported attributes, attribute values or elements are present. OperatorIntervention – Pause job and query the operator when any unsupported attributes, attribute values or elements are present. If a device has no operator intervention capabilities, "OperatorIntervention" is treated as "MustHonor". **Note:** For additional details on @*SettingsPolicy*, see ▸ Section 1.5.3 Conformance to Settings Policy.
Comment *	element	Any human-readable text. The ***Comment*** element is different from an XML comment <!-- XML Comment -->. The **JDF** comment is meant for display in a user interface whereas the XML comment is used to add developers comments to the underlying XML. Comments SHALL NOT be nested within ***Comment*** elements.
GeneralID * New in JDF 1.4	element	Additional identifiers related to the element. **Creation note:** Starting with **JDF** 1.4, ***GeneralID*** has been promoted from being only in a resource to being in any **JDF** element,
Preview * New in JDF 1.4	refelement	Provides a ***Preview*** resource for thumbnails or other images. SHALL not provide multiple ***Preview*** resources with the same ***Preview***/@*PreviewUsage* values **Creation note:** Starting with **JDF** 1.4, ***Preview*** has been moved from ▸ Table 6.1 Template for Input Resources.

3.1.1 Structure Diagram

▸ Figure 3-1: Any-Element (generic content) – a diagram of its structure below shows the structure of the generic content defined above. ▸ Figure 3-1: Any-Element (generic content) – a diagram of its structure and other similar diagrams describe **JDF** structure using the following notation.

- Each box represents an element, with the element's name in the rounded box at the top and its attributes if any, listed below. A rounded box with a dashed line represents an abstract element
- A solid line connects an element to its subelement, where the subelement is at the arrowhead. The cardinality of the subelement is specified after its name. Cardinality in the line overrides that in the box.
- A dashed line connects an element to its abstract element, where the superclass element is at the arrowhead.

Figure 3-1: Any-Element (generic content) – a diagram of its structure



3.2 JDF

The top-level element of a **JDF** instance is a **JDF** element. **JDF** elements MAY also be nested within other **JDF** elements. The individual **JDF** elements are referred to as "nodes" and nodes, in turn, contain various attributes and further subelements, including nested **JDF** nodes.

The following table presents the attributes and elements likely to be found in any given **JDF** node. Three of the attributes in ▸ Table 3.4 JDF, below, SHALL appear in every **JDF** node. Although the rest are designated as OPTIONAL, some OPTIONAL attributes become REQUIRED under circumstances described in the Description column.

The most important of the attributes is the @*Type* attribute, which defines the node type. The value of the @*Type* attribute defines the product intent or process the **JDF** node represents. As is detailed in ▸ Section 3.3 Common Node Types, all nodes fall into one of the following four general categories: Process, Process Group, Combined Processes and Product Intent. Each node is identified as belonging to one of these categories by the value of its @*Type* attribute, as described in the table below. For example, if @*Type* = "Product", the node is a product intent node. Each of these categories is described in greater detail in the sections that follow.

Each attribute/element in ▸ Table 3.4 JDF has a scope. The scope provides further details about the valid range of the attribute/element content, how the content is inherited by descendents (children, grandchildren, etc.), and where the attribute/element can reside in the **JDF** tree.

The scope is specified by the first line of each Description cell in ▸ Table 3.4 JDF. The first line is always: "Scope and Position is *XXX*" where the meaning of *XXX* is defined in ▸ Table 3.2 Definition of "XXX".

Table 3.2: Definition of "XXX"

XXX	DESCRIPTION
Descendent	The content is valid locally within its node and in all descendent nodes, unless a descendent contains an identical attribute that overrides the content.
Local	The content is only valid locally, within the node where the content is defined.
Root	The attribute SHALL be specified only in the root node. An exception from the localization only in the root node occurs if the spawning and merging mechanism for independent job tickets is applied as described in ▸ Section 4.4 Spawning and Merging. All attributes and elements listed in subsequent chapters SHOULD be considered local unless otherwise noted.

Table 3.3: Behavior for Activation Values in ▸ Table 3.4 JDF

ACTIVATION	TEST NODE	EXECUTE NODE
Inactive	false	false
Informative	false	false
Held	false	false
Active	false	true
TestRun	true	false
TestRunAndGo	true	true

Table 3.4: JDF (Sheet 1 of 4)

NAME	DATA TYPE	DESCRIPTION
Activation ? Modified in JDF 1.1	enumeration	Scope and Position is Descendent. Describes the activation status of the **JDF** node. Allows for a range of activity, including deactivation and test running. A child node inherits the value of the @*Activation* attribute from its parent. The value of @*Activation* corresponds to the least active value of @*Activation* of any ancestor, including itself. Therefore, if any ancestor has an @*Activation* of "Inactive", the node itself is "Inactive". If no ancestor is "Inactive" but any ancestor is "Informative", the node is "Informative" unless the node itself is "Inactive". If no ancestor is "Informative" but any ancestor is "TestRun", the node is "TestRun" unless the node itself is "Informative". If no ancestor has a value of "Inactive" or "TestRun" and any ancestor has a value of "TestRunAndGo", the node has a value of "TestRunAndGo" unless that node is "Inactive" or "TestRun" and so on. ▸ Table 3.3 Behavior for Activation Values in ▸ Table 3.4 JDF illustrates the actions to be applied to a node depending on the value of @*Activation*. The values are ordered from least to most active **Allowed values are from:** ▸ Table 3.5 Activation Attribute Values**.**
Category ? New in JDF 1.2 Modified in JDF 1.4	NMTOKEN	Scope and Position is Local. Named category of this node. Used when @*Type* = "Combined" or @*Type* = "ProcessGroup" to identify the general node category. This allows processors to identify the general purpose of a node without parsing the @*Types* field. For instance, a RIP for final output and a RIP for proof process have identical @*Types* attribute values, but have @*Category* = "ProofRIPing" or @*Category* = "RIPing", respectively. **Values include those from:** ▸ Table 3.6 Category Attribute Values**.** **Note:** @*Category* MAY also be the name of a gray box defined by an ICS document.
ICSVersions ? New in JDF 1.2	NMTOKENS	Scope and Position is Descendent. @*ICSVersions* SHALL list all CIP4 Interoperability Conformance Specification (ICS) Versions that this **JDF** node complies with. **Value format is: <ICSName>_L<ICSLevel>-<ICSVersion>.** **Example:** MISPRE_L1-1.3 for the MIS to Prepress ICS. If there is a revision to that ICS: "MISPRE_L1-1.3.1". See ▸ Section 11.5 Interoperability Conformance Specifications for more information on ICS documents.
ID	ID	Scope and Position is Local. Unique identifier of a **JDF** node. This ID is used to refer to the **JDF** node.
JobID ?	string	Scope and Position is Descendent. Job identification used by the application that created the **JDF** job. Typically, a job is identified by the internal order number of the MIS system that created the job.

Table 3.4: JDF (Sheet 2 of 4)

NAME	DATA TYPE	DESCRIPTION
JobPartID ?	string	Scope and Position is Descendent. Identification of a **JDF** node within a job, used by the application that created the job. Typically, @*JobPartID* is internal to the MIS system that created the job and specifies a process or set of processes. Note that a product that is produced by a process or set of processes is identified by ***Resource***/@*ProductID* and not by @*JobPartID*.
MaxVersion ? New in JDF 1.2	JDFJMF-Version	Scope and Position is Descendent. Maximum **JDF** version to be written by an agent that modifies this node. If not specified, an agent that processes the node MAY write any version it is capable of writing. See ▸ Section 3.13 JDF Versioning for a discussion of versioning in **JDF**.
NamedFeatures ? New in JDF 1.2 Deprecated in JDF 1.5	NMTOKENS	Scope and Position is Local. @*NamedFeatures* represents an implementation dependent set of parameters for setting up a device that a device SHALL apply to the **JDF** ticket. It is formatted as an ordered list of name value pairs with an even number of entries. The @*NamedFeatures* names supported by the device MAY be specified in ***DeviceCap*** elements. See ▸ Section 10.2.1 DeviceCap. @*NamedFeatures* SHALL be placed only in combined nodes, process group nodes or product intent nodes . For process group nodes, the @*Types* attribute is typically supplied. See ▸ Section 3.3.2.2 Use of NamedFeature in Product and Process Group Nodes for details. **Deprecation note:** Starting with JDF 1.5, use ***JDF***/***GeneralID***[@*DateType*="NamedFeature"].
ProjectID ? New in JDF 1.1	string	Scope and Position is Descendent. Identification of the project context that this **JDF** belongs to. @*ProjectID* SHOULD be used by a controller to group a set of **JDF** jobs.
RelatedJobID ? New in JDF 1.2	string	Scope and Position is Descendent. Job identification of a related job. Used to identify the @*JobID* of a previous run of this job or job with very similar settings. It MAY be used to retrieve additional job and device specific settings from a data store.
RelatedJobPartID ? New in JDF 1.2	string	Scope and Position is Descendent. Job identification of a related job part. Used to identify the @*JobPartID* of a previous run of this job or job with very similar settings. It MAY be used to retrieve additional job and device specific settings from a data store.
RelatedProjectID ? New in JDF 1.6	string	Scope and Position is Descendent. Identification of a related project context that this **JDF** belongs to. @*RelatedProjectID* SHOULD be used by a controller to group a set of **JDF** jobs.
SpawnID ? New in JDF 1.1	NMTOKEN	Scope and Position is Descendent. Identification of a spawned part of a job. Typically this is used to map ***Audit*** elements and **JMF** messages to a spawned processing step in the workflow. For details on job spawning, see ▸ Section 4.4 Spawning and Merging.
Status Modified in JDF 1.3	enumeration	Scope and Position is Local. Identifies the status of the node. Derivation of the @*Status* of a parent node from the @*Status* of child nodes is non-trivial and implementation-dependent. **Allowed values are from:** ▸ Table 8.179 NodeStatus Attribute Values**.**
StatusDetails ? New in JDF 1.2	string	Scope and Position is Local. Description of the status phase that provides details beyond the enumerative values given by the @*Status* attribute. Values include those from**:** ▸ Section C.11 StatusDetails Supported Strings.

Table 3.4: JDF (Sheet 3 of 4)

NAME	DATA TYPE	DESCRIPTION
Template = "false" New in JDF 1.1	boolean	Scope and Position is Root. Indicates that this **JDF** node (or instance) is a template that is used to generate **JDF** elements but SHALL NOT be exchanged as a job definition. A device SHALL reject a job ticket that contains @*Template* = "true".
TemplateID ? New in JDF 1.2	string	Scope and Position is Descendent. Name or ID that identifies a **JDF** template. Can be used to differentiate between various templates. If @*Template* = "false", @*TemplateID* identifies the template that was used to generate this **JDF**.
TemplateVersion ? New in JDF 1.2	string	Scope and Position is Descendent. Provides the version of the **JDF** template. Can be used to differentiate between various template versions. If @*Template* = "false", @*TemplateVersion* identifies the version of the template that was used to generate this **JDF**.
Type	NMTOKEN	Scope and Position is Local. Identifies the type of the node. Any **JDF** process name is a valid type. The processes that have been predefined are listed in ▸ Chapter 6 Processes, although the flexibility of **JDF** allows anyone to create processes. In addition to these, there are three values which are described in greater detail in the sections that follow. **Values include:** Combined ProcessGroup Product – Identifies a product intent node. **Values include those from:** ▸ Chapter 6 Processes**.**
Types ? Modified in JDF 1.2	NMTOKENS	Scope and Position is Local. List of the @*Type* attributes of the nodes that are combined to create this node. This attribute is REQUIRED if @*Type* = "Combined", OPTIONAL when @*Type* = "ProcessGroup", and is ignored if @*Type* equals any other value. For details on using combined process nodes, see ▸ Section 3.3.3 Combined Process Nodes. If the @*Types* attribute is specified, that **JDF** node SHALL NOT contain child **JDF** nodes. For details on using process group nodes, see ▸ Section 3.3.2 Process Group Nodes. If @*Type* = "ProcessGroup", the tokens MAY also be the name of a gray box that needs expansion. See @*Category* for more details. **Values include those from:** ▸ Chapter 6 Processes**.**
Version ? Modified in JDF 1.2	JDFJMF-Version	Scope and Position is Root and Descendent. Text that identifies the version of the **JDF** node. The @*Version* attribute is REQUIRED in the **JDF** root node but OPTIONAL in child nodes. The version of a **JDF** node is defined by the highest version of the **JDF** node itself or any child **JDF** node or element or any directly or indirectly linked resources. For details on **JDF** versioning see ▸ Section 3.13 JDF Versioning.
xmlns ? New in JDF 1.1	URI	Scope and Position is Root and Descendent. **JDF** supports use of XML namespaces. The namespace SHALL be declared in the root **JDF** element. For details on using namespaces in XML, see ▸ [XMLNS]. For versions 1.1 through 1.5 of **JDF** , @*xmlns* = "http://www.CIP4.org/JDFSchema_1_1".
xsi:type ? New in JDF 1.2	NMTOKEN	Scope and Position is Local. Informs schema aware validators of the **JDF** node type definition that the containing node is to be validated against. The schema for this version includes definitions for all the **JDF** nodes defined in Section 6. If omitted, then a general definition for **JDF** nodes will be used. See ▸ Section 3.2 JDF.
AncestorPool ?	element	Scope and Position is Root. If this element is present, the current **JDF** node has been spawned, and this element contains a list of all ***Ancestor*** elements prior to spawning. See ▸ Section 3.4 AncestorPool.

Table 3.4: JDF (Sheet 4 of 4)

NAME	DATA TYPE	DESCRIPTION
AuditPool ?	element	Scope and Position is Local. List of elements that contains all relevant audit information. ***Audit*** elements are intended to serve the requirements of MIS for evaluation and post calculation. See ▸ Section 3.11 AuditPool and Audit.
CustomerInfo ? Deprecated in JDF 1.3	element	Scope and Position is Descendent. Container element for customer-specific information. See ▸ Section 3.5 CustomerInfo. In **JDF** 1.3 and beyond, ***CustomerInfo*** is a resource that is referenced through a ***CustomerInfoLink*** in the ***ResourceLinkPool***.
JDF *	element	Scope and Position is Local. Child **JDF** nodes. The nesting of **JDF** nodes defines the **JDF** tree.
NodeInfo ? Deprecated in JDF 1.3	element	Scope and Position is Local. Container element for process-specific information such as scheduling and messaging setup. Scheduling affects the planned times when a node is to be executed. Actual times are saved in the ***AuditPool***. See ▸ Section 3.11 AuditPool and Audit. In **JDF** 1.3 and beyond, ***NodeInfo*** is a resource that is referenced through a ***NodeInfoLink*** in the ***ResourceLinkPool***.
ResourceLinkPool ?	element	Scope and Position is Local. Container element for ***ResourceLink*** elements, which describe the input and output resources of the node. See ▸ Section 3.9 ResourceLinkPool and ResourceLink.
ResourcePool ?	element	Scope and Position is Local. Container element for resources. See ▸ Section 3.8 ResourcePool and its Resource Children. **Note:** Resources are local in a ***ResourcePool*** but MAY be referenced from ***ResourceLink*** elements in descendent nodes. For details see ▸ Section 3.9 ResourceLinkPool and ResourceLink.
StatusPool ? Deprecated in JDF 1.3	element	Scope and Position is Local. Container for ***PartStatus*** elements that specify the details of a node's partition dependent @*Status* related attributes if the @*Status* of the node is "Pool". **Deprecation note:** Starting with **JDF** 1.3, ***StatusPool***/***PartStatus***/@*Status* is replaced by ***NodeInfo***/@*NodeStatus* in the respective partition of ***NodeInfo***.

3.2.1 Activation Attribute Values

Table 3.5: Activation Attribute Values

VALUE	DESCRIPTION
Inactive	The node and all its descendents SHALL NOT be executed or tested. This value is set if certain parts of a **JDF** job SHALL NOT be executed or tested.
Informative	The **JDF** ticket is for information only. If a job is "Informative", it SHALL NOT be processed. Jobs with @*Activation* = "Informative" will generally be sent to an operator console for preview but are still completely under the control of an external controller. When a **JDF** ticket is supplied to a customer as proof of execution, its @*Activation* SHOULD also be "Informative". When a new job ticket with an identical @*ID* attribute and a higher @*Activation* is submitted to a device, that **JDF** job ticket SHALL replace the **JDF** job ticket that was submitted to the device with an @*Activation* of "Informative".
Held	Execution has been held. If a job is "Held", it SHALL NOT be processed until its @*Activation* is changed to "Active".
TestRun	The node requests a test run check by a controller or a device. This does not imply that the node is to be automatically executed when the check is completed. Descendents of a node that is being test run are not to be considered "Active".
TestRunAndGo	Similar to "TestRun", but requests a subsequent automatic start if the test run has been completed successfully.
Active	The default value if not specified in a parent node – The node SHALL be executed according to the steps specified in ▸ Section 4.2.1 Determining Executable Nodes.

3.2.2 Category Attribute Values

Table 3.6: Category Attribute Values

VALUE	DESCRIPTION
Binding	Binding of a bound product.
Cutting New in JDF 1.3	Specifies cutting of a ***Component***.
DigitalPrinting	A RIP and print run on a digital printer that produces final output.
FinalImaging	A RIP and image that produces final output that is ready for further processing (e.g., film or plates).
FinalRIPing	A RIP process for generating final output.
Folding	Folding of a product.
Newsprinting New in JDF 1.4	A press run on a newsprinting web press.
PostPress	General postpress. Includes "Folding" and "Binding".
PrePress	General prepress.
Printing	A press run that produces final output.
ProofImaging	A RIP that produces proof output.
ProofRIPing	A RIP process for generating a proof. The processes are identical to those in specified for "FinalRIPing".
PublishingPreparations New in JDF 1.3	Preparing an issue of a newspaper or magazine to be published.
RIPing New in JDF 1.3	General RIP gray box. For details, see ▸ Section 6.3.33 RIPing.

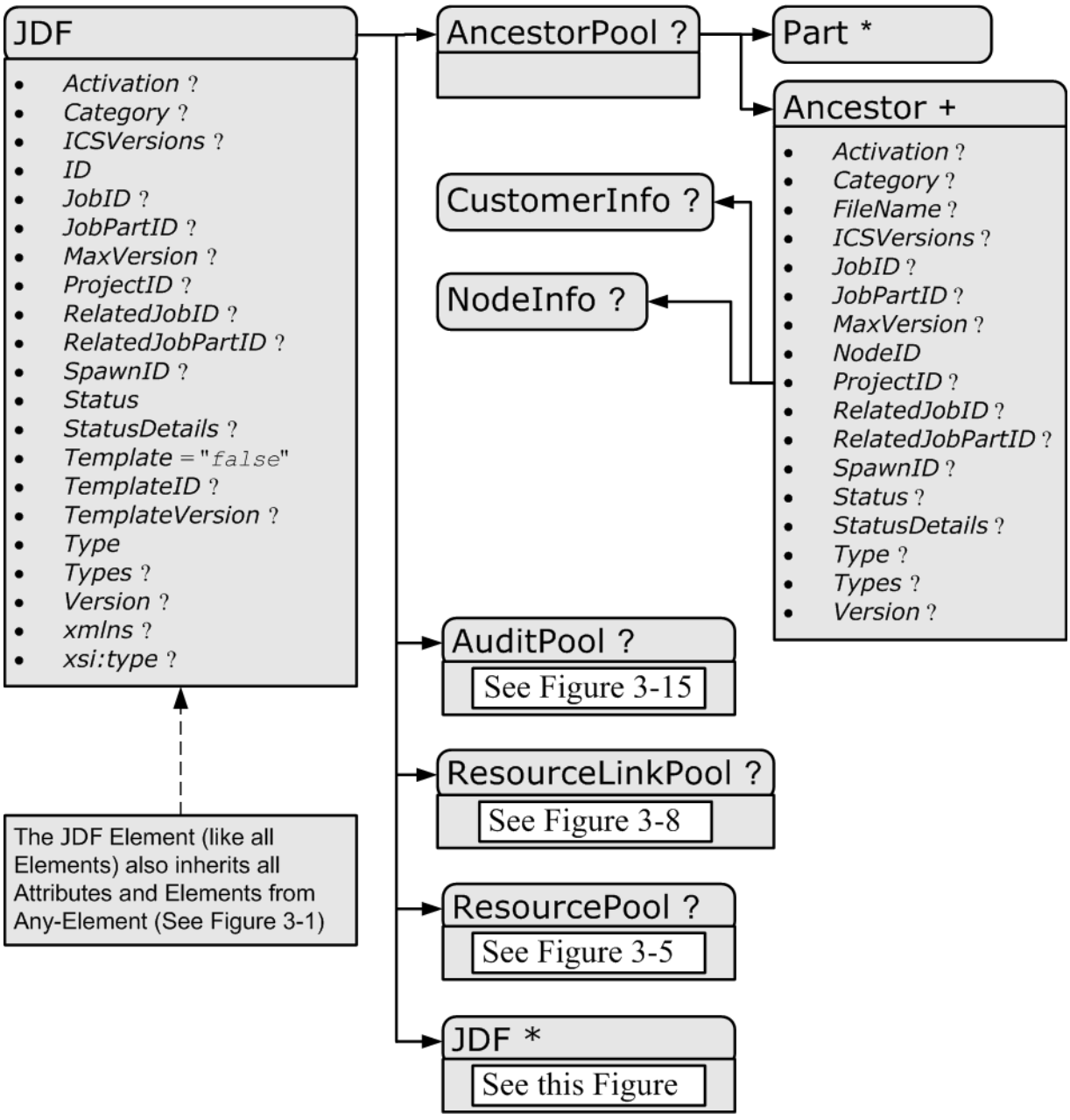
Table 3.6: Category Attribute Values

VALUE	DESCRIPTION
WebPrinting	A press run on a web press can produce one or more components as output at the same time. A web printing press might be equipped with prepress and postpress equipment.
WebToPring	A product description that describes a product order in a web shop.

3.2.3 Structure Diagram of JDF Node

▸ Figure 3-2: JDF Node – a Diagram of its Structure shows the structure of the **JDF** node. Arrows point to child elements.

Figure 3-2: JDF Node – a Diagram of its Structure



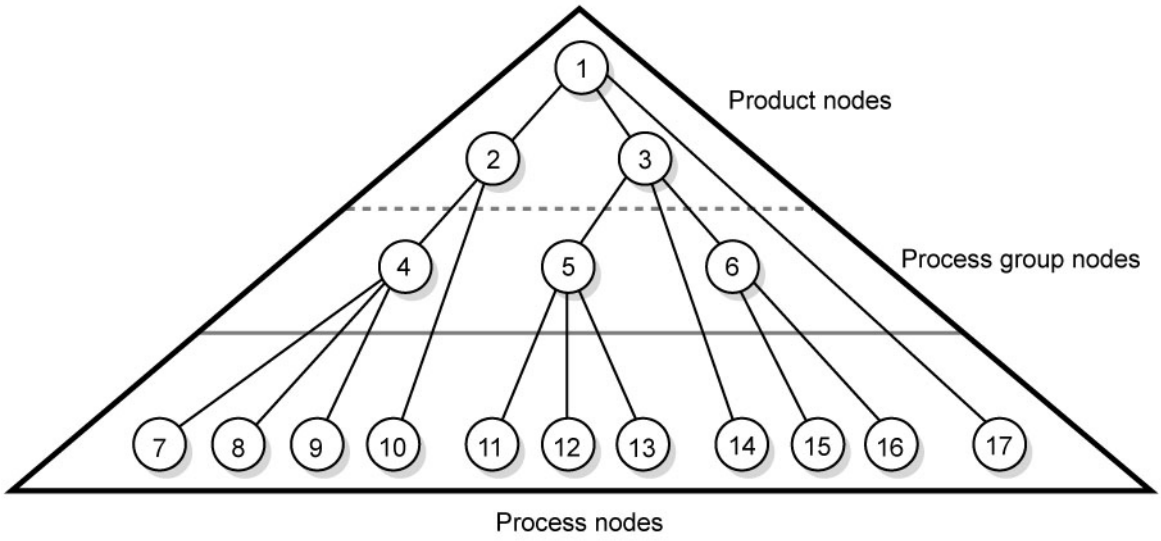
3.3 Common Node Types

As was noted in the preceding section, the @*Type* of a node can fall into four categories. The first is comprised of the specific processes of the kind delineated in ▸ Chapter 6 Processes, known simply as process nodes. The other categories are made up of three enumerative values of the @*Type* attribute: "ProcessGroup", "Combined" and "Product", which is also known as Product Intent. These three node types are described in this section.

The figure below, which was also presented as an illustration in Chapter 2, represents a theoretical job hierarchy comprised of product intent nodes, process group nodes and nodes that represent individual or combined processes. the diagram is divided into three levels to help illustrate the difference between the three kinds of nodes, but these levels do not dictate the hierarchical nesting mechanism of a job. Note, however, that an individual process node MAY be the child

of a product intent node without first being the child of a process group node. Likewise, a process group node MAY have child nodes that are also process groups.

Figure 3-3: Job hierarchy with Process, Process Group and Product Intent Nodes



3.3.1 Product Intent Nodes

Except in certain specific circumstances, the agent assigned to begin writing a **JDF** job will very likely not know every process detail needed to produce the desired results. For example, an agent that is a job-estimating or job-submission tool might not know what devices can execute various steps or even which steps will be needed.

If this is the case, the initiating agent creates a set of top-level nodes to specify the product intent without providing any of the processing details. Subsequent agents then add nodes below these top-level nodes to provide the processing details needed to fulfill the intent specified.

These top-level nodes SHALL have a @*Type* attribute value of "Product" to indicate that they do not specify any processing, (and are referred to as "Product Intent nodes"). All processing needed to produce the products described in these nodes SHALL be specified in process nodes, which exist lower in the job hierarchy.

Product Intent

"Product Intent" is another way of saying "Job Specifications". Rather than describing how a job will be made, Product Intent describes what a finished product (or some aspect of a product) will look like when it is completed. Product Intents can initiate with the customer and in rather vague terms, and they might be later fleshed out or completed by a printer's customer service representative, estimating department or production planners.

Product intent nodes include ***Intent Resources*** that describe the end results the customer is requesting. The ***Intent Resources*** that have already been defined for **JDF** are easily recognizable, as they contain the word "intent" in their titles. Examples include ***ColorIntent*** and ***FoldingIntent***. All ***Intent Resources*** share a set of common subelements, which are described in ▸ Section 7.2 Intent Properties Template. These resources do not attempt to define the processing needed to achieve the desired results; instead they provide a forum to define a range of acceptable possibilities for executing a job.

Each product intent node SHOULD contain at most one ***ResourceLink*** for one type of ***Intent Resource***. If multiple product parts with different intents are needed, each part has its own product intent node. ***DeliveryIntent*** resources are a notable exception. Specifying multiple ***DeliveryIntent*** resources effectively requests multiple options of a quote. A product intent node produces one or more ***Component*** resources as output resources. For more information about product intent, see ▸ Section 4.1.1 Product Intent Constructs.

3.3.2 Process Group Nodes

Intermediate nodes in the **JDF** job hierarchy (i.e., nodes 4, 5 and 6 in ▸ Figure 3-3: Job hierarchy with Process, Process Group and Product Intent Nodes) describe groups of processes. The @*Type* attribute value of these kinds of nodes is "ProcessGroup", (and they are referred to as "Process Group Nodes"). These nodes are used to describe multiple steps in a process chain that have common resources or scheduling data.

Since the agent writing the job has the option of grouping processes in any way that seems logical, custom workflows MAY be modeled flexibly. Process group nodes MAY contain further process group nodes, individual process nodes or a

mixture of both node types. Sequencing of process group nodes SHOULD be defined by linking resources of the appropriate child **JDF** nodes.

The higher the level of the process group nodes within the hierarchy, the larger the number of processes the group contains. A high level process group node (e.g., prepress, finishing or printing processes) might include lower level process group nodes that define a set of individual steps which are executed as a group of steps in the individual workflow hierarchy. For example, all steps performed by one designated individual MAY be grouped in a lower level process group node.

3.3.2.1 Use of the Types Attribute in Process Group Nodes – Gray Boxes

New in JDF 1.2

Process group nodes MAY contain an OPTIONAL @*Types* attribute that allows a controller (e.g., an MIS system) to specify a minimum set of processes to be executed without specifying the complete list of processes or the exact structure or grouping of these processes into individual **JDF** nodes. Process group nodes that contain a @*Types* attribute are commonly referred to as 'Gray Boxes'. Additional processes that are not included in @*Types* MAY be added during expansion of a gray box. A ***ResourceLink***/@*CombinedProcessIndex* is used to map ***ResourceLink*** elements to ***JDF***/@*Types* in the processGroup. Process group nodes with a non-empty @*Types* attribute SHALL NOT be executed. A device that receives the process group node SHALL define the exact structure of the process group node by executing the following steps until the @*Types* list referenced by the process group node is empty:

Step 1 — Select at least one of the process types defined in @*Types* and remove these values from the @*Types* list of values referenced by the process group node.

Step 2 — Create one new **JDF** child node within the processGroup that either:

- Has a @*Type* attribute matching the removed @*Types* entry value, or
- Is a **JDF** node with a @*Type* attribute value of "Combined" or "ProcessGroup" that contains the removed @*Types* value or values.

Step 3 — Link the appropriate resources that were predefined in the original process group node to the newly created subordinate **JDF** node(s). The ***ResourceLink*** SHALL either be retained or deleted from the process group node. If it is retained, the process group node SHALL NOT be executed before the resource that is linked by that ***ResourceLink*** is available. Otherwise, the process group node MAY be executed, even if the resource is not available.

Step 4 — Add missing @*Types* to the subordinate **JDF** node where appropriate. For instance, the original @*Types* attribute list referenced by process group node might have specified "Interpreting Rendering" or simply "RIPing", but the newly created RIP node would specify "Interpreting Rendering Trapping Screening".

Step 5 — Finalize the newly created subordinate **JDF** node by adding any missing resources and resource parameters. Note that newly created resources SHALL NOT be linked to the process group node but only to the subordinate **JDF** node created in this process.

An agent SHALL instantiate all of the processes in the @*Types* attribute of the gray box before releasing the created **JDF** nodes for processing and production. The ordering of the processes in the @*Types* attribute SHALL be maintained when instantiating the child nodes. **JDF** process group nodes that contain both a non-empty @*Types* attribute and child **JDF** nodes are *not* supported, although a process group node MAY contain child process group nodes that have non-empty @*Types* attribute.

3.3.2.2 Use of NamedFeature in Product and Process Group Nodes

New in JDF 1.2

Modified in JDF 1.5

Combined, process group and product intent nodes MAY contain zero or more ***GeneralID***[@*Datatype*="NamedFeature"] elements. These ***GeneralID*** elements that are referred to as "NamedFeatures" in this paragraph allow a controller (e.g., an MIS system) to define a named set of parameters for processes that SHALL be executed without defining the details or even the resources for the individual **JDF** nodes. The agent (e.g., a prepress control system) populates the **JDF** node with the values implied by ▸ NamedFeatures in an implementation-defined manner. This procedure MAY include the addition of additional **JDF** subnodes. The precedence of parameters (attributes or elements) is as follows in order of decreasing precedence:

- Explicitly supplied parameters
- Parameters supplied by the device agent that are associated with the supplied ▸ NamedFeatures closest to the process.
- Parameters supplied by the device agent that are associated with the supplied ▸ NamedFeatures supplied by the device agent at node levels closer to the root.

An individual ▸ NamedFeature is selected by the ***GeneralID***/@*IDUsage* and ***GeneralID***/@*IDValue* that matches entries from ***DeviceCap***/***FeaturePool***/***EnumerationState***/@*Name* and ***DeviceCap***/***FeaturePool***/***EnumerationState***/@*AllowedValueList* (see ▸ Section 10.2.1 DeviceCap), where ***GeneralID***/@*IDUsage* defines the name of the parameter set name (e.g., "Screening"), and ***GeneralID***/@*IDValue* defines the selected parameter set value (e.g., "AM_HighRes"). Multiple ▸ NamedFeatures MAY

be selected. Names and values are implementation dependent. Each ***GeneralID***/@*IDUsage* SHALL occur only once in the ▸ NamedFeatures list.

Use of ▸ NamedFeatures is commonly combined with the use of @*Types* in process group nodes as described in ▸ Section 3.3.2.1 Use of the Types Attribute in Process Group Nodes – Gray Boxes. ***JDF*** /@*Types* abstractly specifies the set of processes to execute, whereas ▸ NamedFeatures abstractly specifies the set of resources for the processes specified in @*Types*.

3.3.2.3 ResourceLink Structure in Process Group Nodes

New in JDF 1.2

The contents of the ***ResourceLinkPool*** of a process group node define the resources that SHALL be available for the process group node itself to be executed.

Example 3.1: ResourceLink Structure for a ProcessGroup

The following example shows the ***ResourceLink*** structure for a "ProcessGroup" digital printing with near-line finishing node. The input ***Media*** is available and the output ***Component*** is of interest to the submitting controller. The ***Parameter Resource***s are assumed to be supplied by the sub-controller that executes the process group node. Note the presence of intermediate component links that link the individual processes. The corresponding ***ResourcePool*** elements and resource elements have been omitted for brevity.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
      Type="ProcessGroup" JobPartID="ID300" Version="1.4">
   <!--the ResourceLink Elements in the ProcessGroup define the Input
         Resources that are to be available for the ProcessGroup to be
         submitted and the Output Resources that are produced by the ProcessGroup
   -->
   <ResourcePool>
      <DigitalPrintingParams ID="L1" Class="Parameter" Status="Available"/>
      <Media ID="L2" Class="Consumable" Status="Available"/>
      <RunList ID="L8" Class="Parameter" Status="Available"/>
      <Component ID="L3" Class="Quantity" Status="Unavailable"
            ComponentType="Sheet"/>
      <GatheringParams ID="L4" Class="Parameter" Status="Available"/>
      <Component ID="L5" Class="Quantity" Status="Unavailable"
            ComponentType="Sheet"/>
      <StitchingParams ID="L6" Class="Parameter" Status="Available"/>
      <Component ID="L7" Class="Quantity" Status="Unavailable"
            ComponentType="Sheet"/>
   </ResourcePool>
   <ResourceLinkPool>
      <!-- print input media -->
      <MediaLink Usage="Input" rRef="L2"/>
      <!-- gathered output components -->
      <ComponentLink Usage="Output" rRef="L7"/>
   </ResourceLinkPool>
   <JDF ID="J2" Status="Waiting" JobPartID="ID301" Type="DigitalPrinting">
      <ResourceLinkPool>
         <!-- digital printing parameters -->
         <DigitalPrintingParamsLink Usage="Input" rRef="L1"/>
         <!-- input sheets -->
         <MediaLink Usage="Input" rRef="L2"/>
         <RunListLink Usage="Input" rRef="L8"/>
         <!-- printed output components -->
         <ComponentLink Usage="Output" rRef="L3"/>
      </ResourceLinkPool>
   </JDF>
```

```
    <JDF ID="J3" Status="Waiting" JobPartID="ID302" Type="Gathering">
        <ResourceLinkPool>
            <!-- gathering parameters -->
            <GatheringParamsLink Usage="Input" rRef="L4"/>
            <!-- printed output components -->
            <ComponentLink Usage="Input" rRef="L3"/>
            <!-- gathered output components -->
            <ComponentLink Usage="Output" rRef="L5"/>
        </ResourceLinkPool>
    </JDF>
    <JDF ID="J4" Status="Waiting" JobPartID="ID303" Type="Stitching">
        <ResourceLinkPool>
            <!-- Stitching parameters -->
            <StitchingParamsLink Usage="Input" rRef="L6"/>
            <!-- gathered output components -->
            <ComponentLink Usage="Input" rRef="L5"/>
            <!-- stitched output components -->
            <ComponentLink Usage="Output" rRef="L7"/>
        </ResourceLinkPool>
    </JDF>
</JDF>
```

3.3.3 Combined Process Nodes

The processes described in ▸ Chapter 6 Processes define individual workflow steps that are assumed to be executed by a single-purpose device. Many devices, however, are able to combine the functionality of multiple single-purpose devices and execute more than one process. For example, a digital printer might be able to execute the **Interpreting**, **Rendering** and **DigitalPrinting** processes. To accommodate such devices, **JDF** allows processes to be grouped within a node whose @*Type* = "Combined", (referred to as "Combined Process Nodes"). Such a node SHALL also contain a @*Types* attribute, which in turn contains an ordered list of the @*Type* values of each of processes that the node specifies. The ordering of the process names in the @*Types* attribute specifies the ordering in which the processes SHOULD be executed. If the final product result would be indistinguishable, the device MAY change the execution order of the processes from that given in the @*Types* attribute.

Furthermore, ***ResourceLink*** elements in combinded process nodes should specify a @*CombinedProcessIndex* attribute in order to define the subprocess to which the resource belongs. Combined process nodes are leaf nodes and SHALL NOT contain further nested **JDF** nodes.

A device with multiple processing capabilities is able to recognize the combinded process node as a single unit of work that it can execute. All input and output resources that are consumed and produced externally by the process SHALL be specified in the ***ResourceLinkPool*** element of the node. This includes all REQUIRED ***Parameter Resource***s as well as the initial input resources and final output resources. Intermediate resources that are internally produced and consumed NEED NOT be specified.

In a combinded process node, the information defined by the various resources linked as input to the various subProcesses are logically available to all processes of the combinded process node. In situations where the ***Parameter Resource*** of more then one subprocess specifies the mapping of sheet surface content to media, the subprocess that specifies such a mapping that is defined earliest in the @*Types* attribute list SHALL be used, and any other mappings specified by any down-stream subprocess resource SHALL be ignored.

3.3.3.1 Combined Process Nodes with Multiple Processes of the Same Type

A combinded process node MAY contain multiple instances of the same process type (e.g., @*Types* = "Cutting Folding Cutting"). In this case, the ordering and mapping of links processes is significant — the parameters of the first **Cutting** process are most likely to be different from those of the second **Cutting** process. Mapping is accomplished using the @*CombinedProcessIndex* attribute in the respective ***ResourceLink***.

Example 3.2: Combined Process Node

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
       Type="Combined" Types="Cutting Folding Cutting" JobPartID="ID345"
       Version="1.4">
    <!--Resources (incomplete...) -->
    <ResourcePool>
       <!-- parameters of the first Cutting Process-->
       <CuttingParams Class="Parameter" ID="L1" Status="Available"/>
       <!-- Folding parameters -->
       <FoldingParams Class="Parameter" ID="L2" Status="Available"/>
       <!-- parameters of the second Cutting Process-->
       <CuttingParams Class="Parameter" ID="L3" Status="Available"/>
       <!-- raw input components -->
       <Component Class="Quantity" ID="L4" Status="Available" ComponentType="Sheet"/>
       <!-- completed output components -->
       <Component Class="Quantity" ID="L5" Status="Unavailable"
              ComponentType="Sheet"/>
    </ResourcePool>
    <!-- Links -->
    <ResourceLinkPool>
       <!-- parameters of the first Cutting Process-->
       <CuttingParamsLink CombinedProcessIndex="0" Usage="Input" rRef="L1"/>
       <!-- Folding parameters -->
       <FoldingParamsLink CombinedProcessIndex="1" Usage="Input" rRef="L2"/>
       <!-- parameters of the second Cutting Process-->
       <CuttingParamsLink CombinedProcessIndex="2" Usage="Input" rRef="L3"/>
       <!-- raw input components -->
       <ComponentLink Usage="Input" rRef="L4"/>
       <!-- completed output components -->
       <ComponentLink Usage="Output" rRef="L5"/>
    </ResourceLinkPool>
</JDF>
```

Example 3.3: ResourceLinkPool for Combined Process Node

The following example of the ***ResourceLinkPool*** of a **JDF** node describes digital printing with in-line finishing and includes the same processes as the previous processGroup example. The node requires the ***Parameter Resource***s and ***Consumable Resource***s of all three processes as inputs, and produces a completed booklet as output. The intermediate

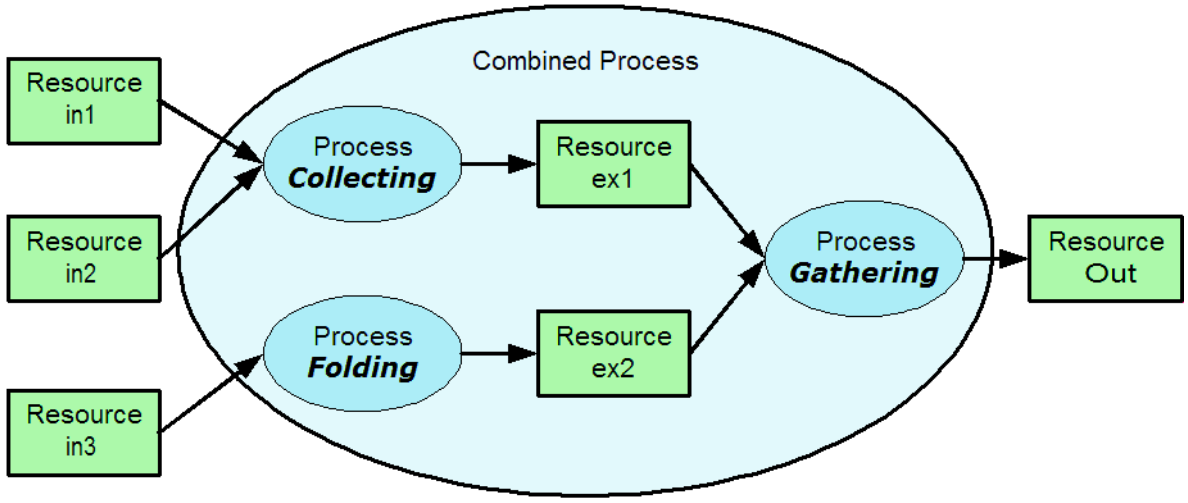
printed sheets and gathered piles are not declared, since they exist only internally within the device and cannot be accessed or manipulated by an external controller.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
       Type="Combined" Types="DigitalPrinting Gathering Stitching" JobPartID="ID200"
       Version="1.4">
    <ResourceLinkPool>
       <!-- digital printing input RunList -->
       <RunListLink CombinedProcessIndex="0" Usage="Input" rRef="L1"/>
       <!-- digital printing parameters -->
       <DigitalPrintingParamsLink CombinedProcessIndex="0" Usage="Input" rRef="L2"/>
       <!-- gathering parameters -->
       <GatheringParamsLink CombinedProcessIndex="1" Usage="Input" rRef="L3"/>
       <!-- Stitching parameters -->
       <StitchingParamsLink CombinedProcessIndex="2" Usage="Input" rRef="L4"/>
       <!-- input sheets -->
       <MediaLink CombinedProcessIndex="0" Usage="Input" rRef="L5"/>
       <!-- stitched output components -->
       <ComponentLink CombinedProcessIndex="2" Usage="Output" rRef="L6"/>
    </ResourceLinkPool>
    <ResourcePool>
       <RunList ID="L1" Class="Parameter" Status="Available"/>
       <DigitalPrintingParams ID="L2" Class="Parameter" Status="Available"/>
       <GatheringParams ID="L3" Class="Parameter" Status="Available"/>
       <StitchingParams ID="L4" Class="Parameter" Status="Available"/>
       <Media ID="L5" Class="Consumable" Status="Available"/>
       <Component ID="L6" Class="Quantity" Status="Unavailable"
              ComponentType="Sheet"/>
    </ResourcePool>
</JDF>
```

3.3.3.2 Specifying non-linear dependencies in a Combined Process Node

A combinded process node typically specified a linear execution chain of the individual process steps defined in ***JDF***/@*Types*. A device that executes a combinded process node MAY execute a more complex network of individual work steps. For instance, a cover might be printed from one tray, the insert from another tray and both be bound to produce a bound component. This behavior is modeled by explicitly declaring the exchange resource and by defining it as a pipe by specifying ***Resource***/@*PipeID* and ***Resource***/@*PipeProtocol* = "Internal". The exchange resource linking it to the combinded process with both an input and output ***ResourceLink*** elements. Multiple input ***ResourceLink*** elements and/or multiple output ***ResourceLink*** elements MAY be declared. ***Resource***/@*Status* of the exchange resource SHALL allow execution of the node.

Figure 3-4: Combined Process Node dependencies



Example 3.4: Complex Combined Process Node

The following example specifies an inline combined folder and collector and gatherer.

```
<JDF ID="ID" xmlns="http://www.CIP4.org/JDFSchema_1_1" Status="Waiting"
      Type="Combined" Types="Collecting Gathering Folding" JobPartID="ID345"
      Version="1.4">
   <ResourcePool>
      <GatheringParams ID="gp1" Class="Parameter" Status="Available"/>
      <FoldingParams ID="fp1" Class="Parameter" Status="Available"/>
      <Component ID="in1" Class="Quantity" Status="Available" ComponentType="Sheet"/>
      <Component ID="in2" Class="Quantity" Status="Available" ComponentType="Sheet"/>
      <Component ID="in3" Class="Quantity" Status="Available" ComponentType="Sheet"/>
      <Component ID="ex1" Class="Quantity" Status="Unavailable" ComponentType="Sheet"
            PipeProtocol="Internal" PipeID="ex1"/>
      <Component ID="ex2" Class="Quantity" Status="Unavailable" ComponentType="Sheet"
            PipeProtocol="Internal" PipeID="ex2"/>
      <Component ID="Out" Class="Quantity" Status="Unavailable"
            ComponentType="Sheet"/>
   </ResourcePool>
   <ResourceLinkPool>
      <GatheringParamsLink Usage="Input" rRef="gp1"/>
      <FoldingParamsLink Usage="Input" rRef="fp1"/>
      <ComponentLink CombinedProcessIndex="0" Usage="Input" rRef="in1"/>
      <ComponentLink CombinedProcessIndex="0" Usage="Input" rRef="in2"/>
      <ComponentLink CombinedProcessIndex="2" Usage="Input" rRef="in3"/>
      <ComponentLink CombinedProcessIndex="0" Usage="Output" rRef="ex1"/>
      <ComponentLink CombinedProcessIndex="2" Usage="Output" rRef="ex2"/>
      <ComponentLink CombinedProcessIndex="1" Usage="Input" rRef="ex1"/>
      <ComponentLink CombinedProcessIndex="1" Usage="Input" rRef="ex2"/>
      <ComponentLink CombinedProcessIndex="1" Usage="Output" rRef="Out"/>
   </ResourceLinkPool>
</JDF>
```

3.3.4 Process Nodes

Process nodes represent the very lowest level in a job hierarchy. They SHALL NOT contain further nested **JDF** nodes, as every process node is a leaf node. These nodes define the smallest work unit that can be scheduled and executed individually within the **JDF** workflow model. In ▸ Figure 3-6: Nodes linked by a Resource below, nodes 7-17 represent process nodes. The various individual process node types are specified in ▸ Section Processes.

3.4 AncestorPool

Ancestor Pool

An ancestor pool contains the job's context when the job is spawned. This includes scheduling information and possibly customer information.

When a job is spawned, an ***AncestorPool*** is created in the spawned **JDF** to identify its parents and grandparents. This allows storing of information about job context in a spawned node as well as allowing the job to be correctly merged with its parent after it is completed. The ***AncestorPool*** element is only REQUIRED in the root of a spawned **JDF**. Spawning and merging are described in ▸ Section 4.4 Spawning and Merging. The ***AncestorPool*** element contains an ordered list of one or more ***Ancestor*** elements, which reflect the family tree of a spawned **JDF**. Each ***Ancestor*** element identifies exactly one ancestor node. The ancestor nodes reside in the original job where the job with the ***AncestorPool*** has been spawned off. The position of the ***Ancestor*** element in the ordered list defines the position in the family tree. The first element in the list is the original root element, the last element in the list is the parent, the last but one, the grandparent and so on. The following table lists the contents of an ***AncestorPool*** element.

Table 3.7: AncestorPool Element

NAME	DATA TYPE	DESCRIPTION
Ancestor +	element	Ordered list of one or more ***Ancestor*** elements, which reflect the family tree of a spawned **JDF**.
Part * New in JDF 1.1	element	List of parts that this node was spawned with. Used in case of parallel spawning of a node. This defines the aggregated ***Part***(s) in the case of nested spawns (i.e., a logical AND of all spawned ***Part***(s)). For instance, the **JDF** that was spawned with a @*SheetName* partition and subsequently spawned with a @*Separation* would contain both @*SheetName* and @*Separation* within ***Part***.

3.4.1 Ancestor

An ***Ancestor*** element SHALL contain read-only copies of all the attributes of the node that it represents with the exception of the @*ID* attribute, which SHALL be copied to the @*NodeID* attribute of that ***Ancestor*** element. ***Ancestor*** elements MAY contain further read-only references to ***CustomerInfo*** and ***NodeInfo***. The attributes and elements of ***Ancestor*** elements are described below.

Table 3.8: Ancestor Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Activation ? New in JDF 1.2	enumeration	Copy of the @*Activation* attribute from the ancestor node. For details, see ▸ Table 3.4 JDF. Allowed values are from: ***JDF***/@*Activation*.
Category ? New in JDF 1.2	NMTOKENS	Copy of the @*Category* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF. Allowed values are from: ***JDF***/@*Category*.
FileName ?	URL	The URL of the **JDF** file where the ancestor node resided prior to spawning. Note that despite of attribute name, @*URL* NEED NOT refer to a file. @*URL* MAY refer to any url scheme.
ICSVersions ? New in JDF 1.2	NMTOKENS	Copy of the @*ICSVersions* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF. Allowed values are from: ***JDF***/@*ICSVersions*.
JobID ?	string	Copy of the @*JobID* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF.
JobPartID ?	string	Copy of the @*JobPartID* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF.
MaxVersion ? New in JDF 1.2	JDFJMFVersion	Copy of the @*MaxVersion* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF.
NamedFeatures ? New in JDF 1.2 Deprecated in JDF 1.5	NMTOKENS	Copy of the @*NamedFeatures* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF.
NodeID	NMTOKEN	Copy of the @*ID* attribute of the ancestor node. **Note:** NMTOKEN is used as the datatype and not ID because the ancestor node and thus @*ID* does not reside in the spawned **JDF**. The corresponding @*ID* attribute resides in the original **JDF**.
ProjectID ?	string	Identification of the project context that this **JDF** belongs to. Used by the application that created the **JDF** job.
RelatedJobID ? New in JDF 1.2	string	Copy of the @*RelatedJobID* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF.
RelatedJobPartID ? New in JDF 1.2	string	Copy of the @*RelatedJobPartID* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF.
SpawnID ? New in JDF 1.1	NMTOKEN	Copy of the @*SpawnID* attribute of the ancestor node.
Status ?	enumeration	Copy of the @*Status* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF. Allowed values are from: ***JDF***/@*Status*.
StatusDetails ? New in JDF 1.2	string	Copy of the @*StatusDetails* attribute from the original ancestor node. For values and details, see ▸ Table 3.4 JDF. **Values include those from:** ***JDF***/@*StatusDetails*
Template = "false" New in JDF 1.1	boolean	Copy of the @*Template* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF.

Table 3.8: Ancestor Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
TemplateID ? New in JDF 1.2	string	Copy of the @*TemplateID* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF.
TemplateVersion ? New in JDF 1.2	boolean	Copy of the @*TemplateVersion* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF.
Type ?	NMTOKEN	Copy of the @*Type* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF. Allowed values are from: ***JDF***/@*Type*.
Types ?	NMTOKENS	Copy of the @*Types* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF. Allowed values are from: ***JDF***/@*Types*.
Version ?	JDFJMFVersion	Copy of the @*Version* attribute from the original ancestor node. For details, see ▸ Table 3.4 JDF.
CustomerInfo ? New in JDF 1.1 Modified in JDF 1.3	refelement	Reference to or copy of the ***CustomerInfo*** element or resource from the original node. In **JDF** 1.3 and beyond, ***CustomerInfo*** SHOULD be a resource reference.For details, see ▸ Table 3.4 JDF.
NodeInfo ? New in JDF 1.1 Modified in JDF 1.3	refelement	Reference to or copy of the ***NodeInfo*** element or resource from the original node. In **JDF** 1.3 and beyond, ***NodeInfo*** SHOULD be a resource reference.For details, see ▸ Table 3.4 JDF.

3.5 CustomerInfo

Deprecated in JDF 1.3

Starting with **JDF** 1.3, ***CustomerInfo*** is deprecated in its use as a direct child of a **JDF** node, and becomes a resource (which is a child of some ***ResourcePool***; see ▸ Section 3.8 ResourcePool and its Resource Children).

3.6 NodeInfo

Deprecated in JDF 1.3

Starting with **JDF** 1.3, ***NodeInfo*** is deprecated in its use as a direct child of a **JDF** node, and becomes a resource (which is a child of some ***ResourcePool***; see ▸ Section 3.8 ResourcePool and its Resource Children).

3.7 StatusPool

Deprecated in JDF 1.3.

Starting with **JDF** 1.3, ***StatusPool*** is deprecated and replaced by a partitioned ***NodeInfo*** resource. For details, see ▸ Section N.1.3 StatusPool.

3.8 ResourcePool and its Resource Children

3.8.1 ResourcePool

All resources are contained in the ***ResourcePool*** element of some node. The ***ResourcePool*** element is described in the following table.

Table 3.9: ResourcePool Element

NAME	DATA TYPE	DESCRIPTION
Resource *	element	List of ***Resource*** elements. The ***Resource*** elements are abstract and serve as placeholders for any resource type.

3.8.2 Resource

Resources represent the "things" that are produced or consumed by processes. They might be physical items such as inks, plates or glue; electronic items such as files or images; or conceptual items such as parameters and device settings. Processes describe what resources they input or output through ***ResourceLink*** elements, discussed in ▸ Section 3.9 Re-

sourceLinkPool and ResourceLink. By examining the input and outputs of a set of processes, it is possible to determine process dependencies, and therefore job routing.

3.8.3 Abstract Resource

Like the @*Type* attribute in abstract **JDF** nodes, the @*Class* attribute in ***Resource*** elements helps to identify how particular resources are to be used. These values are listed in ▸ Table 3.10 Abstract Resource Element, below, and are described in greater detail in the sections that follow.

Modification note: ***GeneralID*** has moved to ▸ Table 3.1 Any Element (generic content).

Table 3.10: Abstract Resource Element (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
AgentName ? New in JDF 1.2	string	The name of the agent application that created the resource. Both the company name and the product name MAY appear, and SHOULD be consistent between versions of the application.
AgentVersion ? New in JDF 1.2	string	The version of the agent application that created the resource. The format of the version string MAY vary from one application to another, but SHOULD be consistent for an individual application.
Author ? New in JDF 1.2	string	Text that identifies the person who generated the resource.
CatalogID ? Deprecated in JDF 1.4	string	Identification of the resource (e.g., in a catalog environment). Defaults to the value of @*ProductID*. **Deprecation note:** Starting with JDF 1.4, use ***GeneralID***.
CatalogDetails ? Deprecated in JDF 1.4	string	Additional details of a resource in a catalog environment. **Deprecation note:** Starting with JDF 1.4, use ***GeneralID***.
Class Modified in JDF 1.5	enumeration	Defines the abstract resource type. For details, see the sections that follow. @*Class* SHALL be specified in the resource root, SHALL NOT be specified in a resource leaf and SHOULD NOT be specified in an inline resource subelement. **Allowed values are:** Consumable Handling Implementation Intent Parameter PlaceHolder Deprecated in JDF 1.5 Quantity
ID	ID	Unique identifier of a resource. @*ID* SHALL be specified in the resource root, SHALL NOT be overwritten in a resource leaf and SHOULD NOT be specified in an inline resource subelement.
Locked ="false"	boolean	If "true", the resource SHALL NOT be modified (e.g., because it resides in a spawned ticket that is spawned in read-only mode or referenced by an ***Audit*** and SHALL NOT be modified without invalidating the ***Audit***).

Table 3.10: Abstract Resource Element (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
PartUsage = "Explicit" New in JDF 1.1 Modified in JDF 1.3	enumeration	Description of the interpretation of partitions. @*PartUsage* SHALL NOT be specified outside of the root of a resource. For details on @*PartUsage* and partitioning, see ▸ Section 3.10.7.4 Implicit, Sparse and Explicit PartUsage in Partitioned Resources. **Allowed values are:** Explicit – Require explicit partition matches. All referenced partitions referenced in ***Part*** SHALL exist, otherwise it is an error. Implicit – The closest matching partition with no non-matching partition keys is returned. If keys with non-matching values exist, the first partition element that is closer to the root than the referenced partition and has no non-matching keys is returned. Sparse – The closest matching partition with no non-matching partition keys is returned. If keys with non-matching values exist the link is in error. @*PartUsage* = "Sparse" is typically used to describe versioned resources, where not all nodes are fully partitioned (e.g., only the black Separations of a 4 color resource are versioned). New in JDF 1.3 **Modification note:** @*PartUsage* was moved to this table from ▸ Table 3.24 Partitionable Resource Element in JDF 1.2.
PipeID ?	string	If this attribute exists, the resource is a pipe. @*PipeID* is used by **JMF** pipe-control messages to identify the pipe. For more information, see ▸ Section 4.3.3 Overlapping Processing Using Pipes.
PipeProtocol ? New in JDF 1.2 Modified in JDF 1.5	NMTOKEN	Defines the protocol use for pipe handling. "JMF" and "Internal" are the only non-proprietary piping protocols that are supported. Proprietary pipe protocols MAY be specified in addition to those defined below but will not necessarily be interoperable. **Values include:** IdentificationField - For barcode push.The pipe data is provided by barcodes that are defined in ***IdentificationField*** elements. Internal – Internal or virtual pipe used within a combinded process. JMF – JMF-based ***PipePush***/***PipePull*** messages. The sequence of pipe initialization is undefined. See next two values: "JMFPush" and "JMFPull". JMFPush – JMF based ***PipePush***/***PipePull*** protocol. The producing device initiates the protocol. New in JDF 1.5 JMFPull – JMF based ***PipePush***/***PipePull*** protocol. The consuming device initiates the protocol. New in JDF 1.5 None – No pipe support.
PipeURL ? New in JDF 1.2 Deprecated in JDF 1.5	URL	Pipe request URL. Dynamic pipe requests to this resource SHOULD be made *to* this URL. Note that this URL is only used for initiating pipe requests. Responses to a pipe request are issued to the URL that is defined in the ***PipePush*** or ***PipePull*** message. For details on using @*PipeURL*, see ▸ Section 4.3.3 Overlapping Processing Using Pipes. **Note:** In most cases @*PipeURL* is the URL of the controller of the *other end* of the pipe. This might seem counterintuitive, but it allows parallel spawning and merging of processes that represent a dynamic pipe without having to include the node that describes the other end in the spawned file. **Deprecation note:** Starting with JDF 1.5, use ***ResourceLink***/@*PipeURL*.
ProductID ?	string	An ID of the resource as defined in the MIS system. For instance item codes or article numbers or identifiers on semi-finished products or ***Handling Resource***s.
rRefs ? Deprecated in JDF 1.2	IDREFS	Array of *ID*s of internally referenced resources. In JDF 1.2 and beyond, it is up to the implementation to maintain references.
SkipIndex ? New in JDF 1.5	integer	Number of indexed partition leaves to omit when evaluating the respective *XXXIndex* partitions. Valid only on partition leaves or branches where the partition key is one of the *XXXIndex* keys (e.g., @*SetIndex*). Used when an index range comprises every Nth index. For example, a range of @*SheetIndex* = "1000 ~ 2000" with @*SkipIndex* = "1" would comprise a list starting at @*SheetIndex* = "1000" including every other index (1000, 1002, 1004, etc.) with a maximum value of @*SheetIndex* = "2000".

Table 3.10: Abstract Resource Element (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
SpawnIDs ? New in JDF 1.1	NMTOKENS	List of @*SpawnID* values. This is used as a reference count for how often the resource has been spawned.
SpawnStatus = "NotSpawned"	enumeration	The spawn status of a resource indicates whether or not a resource has been spawned, and under what circumstances. The @*SpawnStatus* of a resource that has ***ResourceRef*** elements is defined as the maximum @*SpawnStatus* (whose values are ordered) of all recursively linked resources. Value are ordered from lowest to highest **Allowed values are:** NotSpawned — Indicates that the resource has not been copied to another process. SpawnedRO – Indicates that the resource has been copied to another process where it cannot be modified. The "RO" stands for read-only. SpawnedRW – Indicates that the resource has been copied to another process where it can be modified. The "RW" stands for read/write.
Status Modified in JDF 1.2	enumeration	The status of a resource indicates under what circumstances it can be processed or modified. @*Status* SHALL be specified in the resource root, SHALL NOT be specified in an inline resource subelement and MAY be overwritten in a resource leaf. The values listed below are assumed to be ordered so that the @*Status* of a resource that references further resources can be defined as the minimum @*Status* of all recursively linked resources. The values are ordered from lowest to highest **Allowed values are:** Incomplete – Indicates that the resource does not exist, and the metadata is not yet valid. Incomplete resources NEED NOT specify all attributes or elements defined in ▸ Chapter 8 Resources. The structural attributes @*Class* and @*ID* SHALL be specified. Rejected – Indicates that the resource has been rejected by an **Approval** process. The metadata is valid. New in JDF 1.2 Unavailable – Indicates that the resource is not ready to be used or that the resource in the real world represented by the ***PhysicalResource*** in **JDF** is not available for processing. The metadata is valid. InUse – Indicates that the resource exists, but is in use by another process. Also used for active pipes (see ▸ Section 3.8.7 Pipe Resources and ▸ Section 4.3.3 Overlapping Processing Using Pipes). Draft – Indicates that the resource exists in a state that is sufficient for setting up the next process but not for production. Complete – Indicates that the resource is completely specified and the parameters are valid for usage. A ***PhysicalResource*** with @*Status* = "Complete" is not yet available for production, although it is sufficiently specified for a process that references it through a ***ResourceRef*** from a ***Parameter Resource*** to commence execution. Available – Indicates that the whole resource is available for usage.
UpdateID ? New in JDF 1.1 Deprecated in JDF 1.3	NMTOKEN	Unique ID that identifies the ***Resource*** or resource partition. Note that only one ***Resource***, resource partition or ***ResourceUpdate*** with a given value of @*UpdateID* MAY occur per **JDF** document, even though the scope of the ***ResourceUpdate*** is local to the resource that it is defined in.
QualityControlResult * New in JDF 1.2	refelement	Results of quality measurements which were performed during or after the production of this resource.
SourceResource * New in JDF 1.3	element	List of resources that were or SHOULD be taken into account to populate this resource.

3.8.3.1 SourceResource

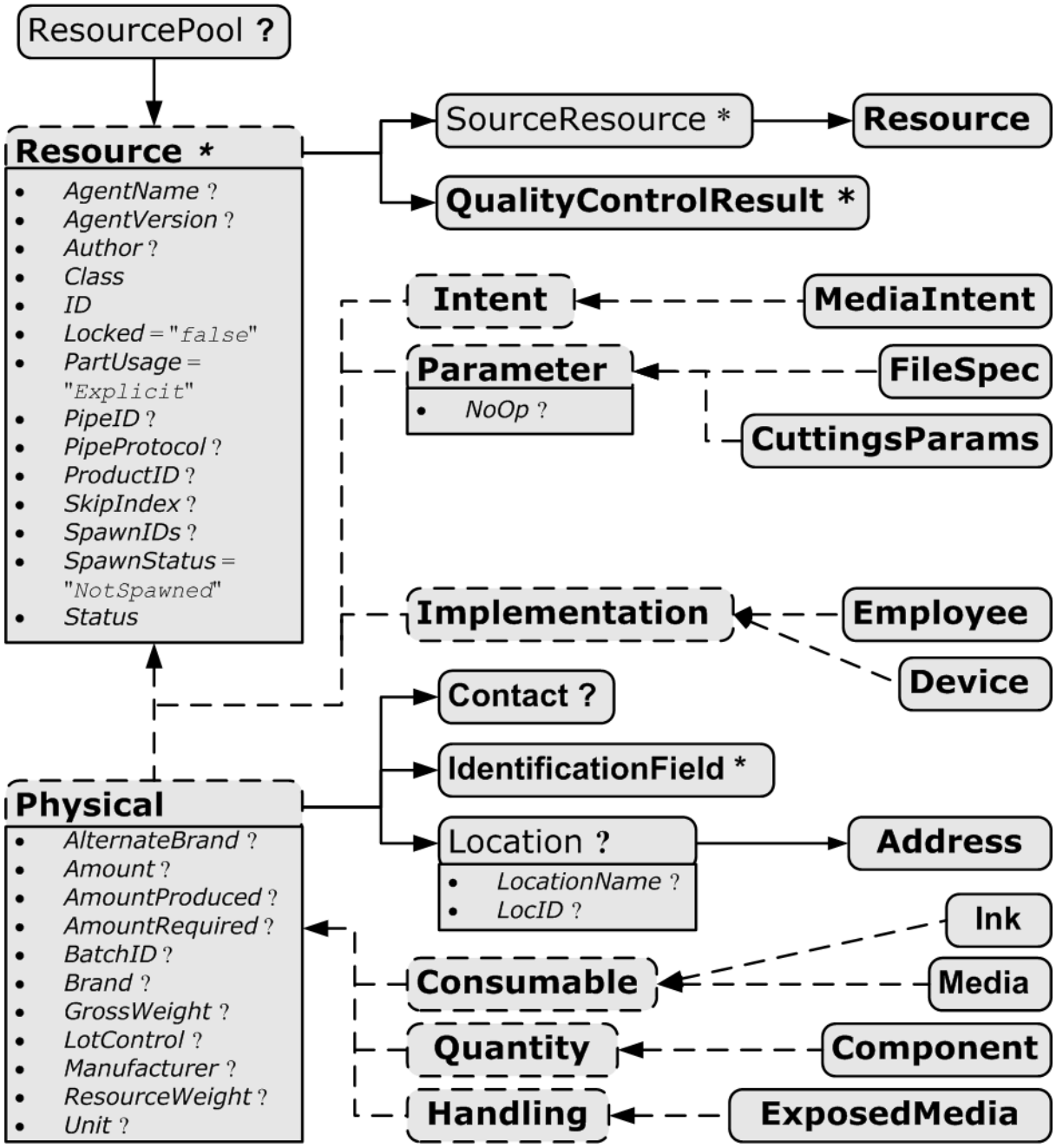
Table 3.11: SourceResource Element

NAME	DATA TYPE	DESCRIPTION
Resource	refelement	Reference to resources that were or SHOULD be taken into account to populate this ***Resource***. ***Resource*** is an abstract element that MAY reference either process resources or ***Intent Resource***s that contain information that is used to populate this resource. Note that resource is an abstract type and designates any valid **JDF** resource (e.g., ***StrippingParams*** or ***ColorIntent***). This element SHALL NOT be an inline ***Resource***.

3.8.4 Structure Diagram

Figure 3-5: ResourcePool and Abstract Resource Element – a diagram of the structure shows the structure of the abstract resource classes defined above. Arrows define inheritance relations and the thin orthogonal lines describe containing relations.

Figure 3-5: ResourcePool and Abstract Resource Element – a diagram of the structure



3.8.5 Resource Classes

The following sections describe the functions of each of the seven values of the @*Class* attribute. All resources fall into one of these classes. In ▸ Chapter 7 Product Intent and ▸ Chapter 8 Resources, the class of each resource is indicated in the resource properties subheading.

Parameter & Intent Resources

Parameter and Intent Resources are *information* about the job. Intent Resources might originate in the customer's RFQ and might include information such as trim size, the number of colors and so on. Later on in the process of estimating and scheduling the job, these intents might be transformed into parameters for production process.

3.8.5.1 Parameter Resource

Parameter Resources define the details of processes, as well as any non-physical computer data such as files used by a process. They are usually associated with a specific process. For example, a REQUIRED input resource of the **DigitalPrinting** process is the ***DigitalPrintingParams*** resource. Most predefined ***Parameter Resource***s contain the suffix "***Params***" in their titles. Examples of ***Parameter Resource***s include ***FoldingParams*** and ***ConventionalPrintingParams***.

3.8.5.1.1 Abstract Parameter Resource

Table 3.12: Abstract Parameter Resource Element

NAME	DATA TYPE	DESCRIPTION
NoOp = "false" New in JDF 1.1	boolean	A value of "true" indicates that the process step that is parameterized by this ***Resource*** or resource partition SHALL NOT be executed. If "false" or not specified, the ***Resource*** is operational and that the process step that is parameterized by this ***Resource*** or resource partition SHALL be executed. The @*NoOp* attribute SHALL only be used for processes that input and output exchange resources of identical resource types (e.g., ***RunList*** or ***Component***).

3.8.5.2 Intent Resource

Intent Resources define the details of products to be produced without defining the process to produce them. In addition, they provide structures to define sets of allowable options and to match these selections with prices. The details of all ***Product Intent*** are described in ▸ Chapter 7 Product Intent. The abstract ***Intent Resource*** element contains no attributes or elements besides those contained in the ***Abstract Resource*** element.

3.8.5.3 ImplementationResource

ImplementationResources define the devices and operators that execute a given node. Only two ***ImplementationResource*** types are defined: ***Employee*** (see ▸ Section 8.53 Employee) and ***Device*** (see ▸ Section 8.44 Device)

ImplementationResources can only be used as input resources and MAY be linked to any process. The abstract ***ImplementationResource*** element contains no attributes or elements besides those contained in the ***Abstract Resource*** element. An example demonstrating how to use ***ImplementationResource***s is provided in ▸ Section Example 3.5: EmployeeLink.

Note that if a node links to a ***Device*** resource in order to specify that the device is intended to execute the node, the ***Device*** resource SHOULD NOT specify the capabilities of the device.

3.8.5.4 Consumable Resource

A ***Consumable Resource*** is consumed during a process. Examples include ***Ink*** and ***Media***. ***Consumable Resource***s are the unmodified inputs in a process chain. A ***Consumable Resource*** is a ***PhysicalResource*** and inherits the contents of the ***Abstract PhysicalResource*** element.

3.8.5.5 Quantity Resource

A ***Quantity Resource*** has been created by a process from either a ***Consumable Resource*** or an earlier ***Quantity Resource***. For example, printed sheets are cut and a pile of cut blocks is created. A ***Component*** resource is an example of a ***Quantity Resource***. A ***Quantity Resource*** is a ***PhysicalResource*** and inherits the contents of the ***Abstract PhysicalResource*** element.

3.8.5.6 Handling Resource

A ***Handling Resource*** is used during a process, but is not destroyed by that process. The ***ExposedMedia*** and ***Tool*** resources are examples of such a resource, although it does describe various kinds of items such as film and plates. A ***Handling Resource*** MAY be created from a ***Consumable Resource***. A ***Handling Resource*** is a ***PhysicalResource*** and inherits the contents of the ***Abstract PhysicalResource*** element.

3.8.5.7 PhysicalResource

A ***PhysicalResource*** is a ***Resource*** that is a ***Consumable Resource***, a ***Quantity Resource*** or a ***Handling Resource*** (whose @*Class* is "Consumable", "Quantity" or "Handling", respectively):

3.8.5.7.1 Abstract PhysicalResource

▸ Table 3.13 Abstract PhysicalResource Element, defines the additional attributes and elements that can be defined for ***PhysicalResource***s. The processes that consume ***PhysicalResource***s—any kind of ***PhysicalResource***—have the option of using these attributes and elements to determine in what way the resources are to be consumed.

Automating Inventory Management

JDF's handling of PhysicalResources provides a bridge between your JDF enabled systems and inventory management, ordering and replenishing systems. This opens the door to just-in-time inventory management driven by real-time scheduling and consumption data.

Table 3.13: Abstract PhysicalResource Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AlternateBrand ?	string	Information, such as the manufacturer or type, about a resource compatible to that specified by the @*Brand* attribute, which is described below.
Amount ?	double	Actual amount of the resource that is available. Note that the amount of consumption and production of a node is specified in the corresponding ***ResourceLink*** element. For details on amount handling, see ▸ Section 3.10.4 Resource Amount. For the unit of measurement, see the @*Unit* attribute below.
AmountProduced ? New in JDF 1.2	double	Total amount of the resource that has been produced by all nodes that reference this resource as output. This corresponds to the sum of all @*ActualAmount* values of output ***ResourceLink*** elements of leaf **JDF** nodes with @*Status* = "Completed" that reference this resource. For the unit of measurement, see the @*Unit* attribute below.
AmountRequired ?	double	Total amount of the resource that is referenced by all nodes that will consume this resource. This corresponds to the sum of all @*Amount* values of input ***ResourceLink*** elements of all processes that consume this resource. In case the resource is the last resource in a process chain, @*AmountRequired* specifies the sum of all @*Amount* values of all output ***ResourceLink*** elements that produce this resource. For the unit of measurement, see the @*Unit* attribute below.
BatchID ?	string	ID of a specific batch of the ***PhysicalResource***
Brand ? Modified in JDF 1.3	string	Information, such as the model, part number and/or type, about the resource being used. Some examples are as follows. • Premium InkProp Glossy 6x642A • Premium Multipurpose 1234, 88 Bright 24 lb. Bond, 8-1/2 x 11, White Copy Paper Reorder 4711 Prior to **JDF** 1.3, @*Brand* included details of the @*Manufacturer*, which SHOULD be specified in @*Manufacturer*.
GrossWeight ? New in JDF 1.3	double	Gross weight of a single resource, as counted in @*Amount*, in grams.
LotControl ? New in JDF 1.3	enumeration	Specifies whether the resource is lot controlled. **Allowed values are:** Controlled – Resource is lot controlled, lot usage SHOULD be reported in resourceAudit elements. NotControlled – Resource is not lot controlled.
Manufacturer ? New in JDF 1.3	string	Specifies the manufacturer of the resource.
ResourceWeight ? New in JDF 1.1	double	Net weight of a single resource, as counted in @*Amount*, in grams.

Table 3.13: Abstract PhysicalResource Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Unit ?	NMTOKEN	Unit of measurement for the values of @*Amount*, @*AmountProduced* and @*AmountRequired*. **Values include those from:** ▸ Table 1.7 Units Used in JDF. **Note:** It is strongly discouraged to specify units other than those that are defined in ▸ Table 1.7 Units Used in JDF.
Contact ?	refelement	If this element is specified, it describes the owner of the resource.
IdentificationField * New in JDF 1.1	refelement	If this element is specified, a bar code or label is associated with this ***PhysicalResource***.
Location ?	element	Description of details of the location of this resource. Note, in order to describe multiple locations, resources MAY be partitioned by the @*Location* partition key as described in ▸ Section 3.10.5 Description of Partitioned Resources.

3.8.5.7.2 Location

Table 3.14: Location Element

NAME	DATA TYPE	DESCRIPTION
LocationName ? New in JDF 1.1	string	Name of the location (e.g., in MIS). This allows the user to describe distributed resources. **Values include those from:** ▸ Table C.10 Input Tray and Output Bin Names. **Note:** The specified values are for printer locations.
LocID ?	string	Location identifier (e.g., within a warehouse system).
Address	element	Address of the storage facility. For more information, see ▸ Section 9.1 Address.

3.8.5.8 PlaceHolder Resource

Deprecated in JDF 1.5

The ***PlaceHolder Resource*** has been deprecated starting with **JDF** 1.5. For details of the deprecated ***PlaceHolder Resource***, see ▸ Section N.1.1 PlaceHolder Resource.

3.8.6 Position of Resources within JDF Nodes

Resources MAY exist in any **JDF** node, but **JDF** nodes SHALL reference only local or global resources. In other words, **JDF** nodes SHALL reference resources only in the two kinds of locations: in the node's own ***ResourcePool*** element, or in **JDF** nodes that are hierarchically closer to the **JDF** root. An exception to this rule, however, occurs if two independent jobs are merged for a process step and are to be separated afterwards, as is the case when two independent jobs are printed on the same web press. For further details on independent job merging, see ▸ Section 4.4.5 Case 5: Spawning and Merging of Independent Jobs.

It is good practice to put resources into the closest node that references the resource. For example, the ***RenderingParams*** resource SHOULD be located in the **Rendering** node, unless it is used by multiple **Rendering** processes, in which case it SHOULD be located in the process group node that contains the **Rendering** process nodes. Resources that link more than one node SHOULD be placed in the parent node of the siblings that are linked by the resource.

A process that needs additional detailed process information specifying the creation of a resource SHALL infer this information by explicitly linking to the appropriate ***Parameter Resource***.

3.8.7 Pipe Resources

A pipe describes the resource dependency in which a process begins to consume a resource while it is being produced by another process (e.g., stacking components while they are being printed) or consuming a data stream while it is being written by an upstream process. Note that defining a pipe resource does not automatically set up communication between processes. The controllers/agents that execute the process SHALL still implement the protocol that defines the pipe.

Using dynamic pipe control, a downstream process can control the total quantity produced by an upstream process, and/or the quantity buffered by an inter-Process transport device (i.e., conveyor belt). Additional description of pipes and process communication via pipes is provided in ▸ Section 4.3.3 Overlapping Processing Using Pipes.

Resources MAY contain a string attribute called @*PipeID* that declares the resource to be a pipe, and identifies it in a dynamic-pipe messaging environment. A pipe that is also controlled by **JMF** pipe messages is called **dynamic pipe**. For more information about dynamic pipes, see ▸ Section 4.3.3.1 Dynamic Pipes.

3.8.8 ResourceUpdate

New in JDF 1.1,

Deprecated in JDF 1.3

For details of the deprecated ***ResourceUpdate*** element, see ▸ Section N.1.2 ResourceUpdate.

3.9 ResourceLinkPool and ResourceLink

3.9.1 ResourceLinkPool

Each **JDF** node contains a ***ResourceLinkPool*** element that in turn contains all of the ***ResourceLink*** elements that link the node to the resources it uses. The following table shows the contents of a ***ResourceLinkPool*** element.

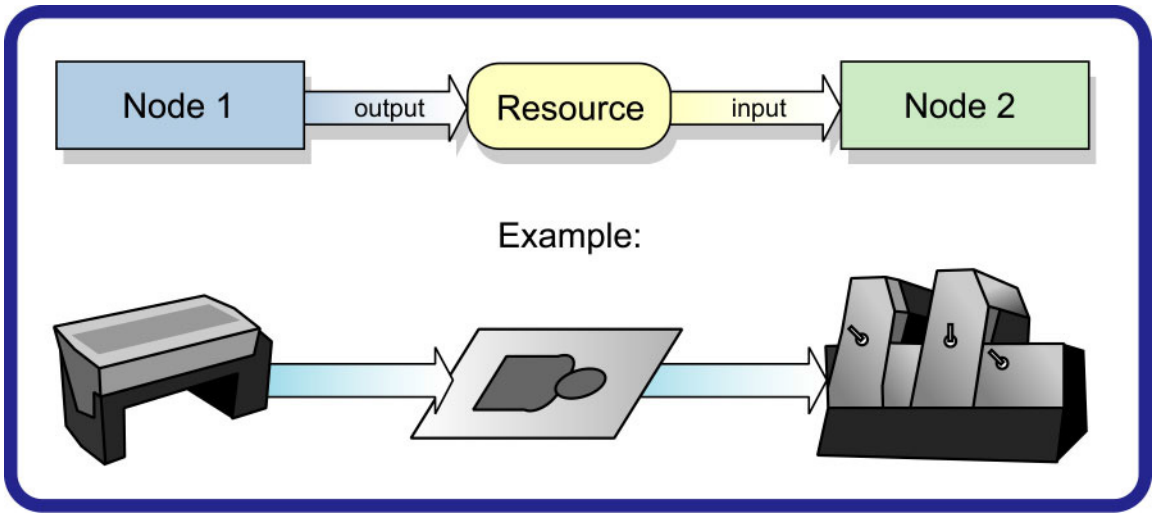
Table 3.15: ResourceLinkPool Element

NAME	DATA TYPE	DESCRIPTION
ResourceLink *	element	List of ***ResourceLink*** elements. A ***ResourceLink*** element is abstract and is a placeholder for a concrete ***ResourceLink*** element, such as ***MediaLink***.

3.9.2 ResourceLink

ResourceLink elements describe what resources a node uses, and how it uses them. They also define whether the resources are inputs or outputs. These inputs and outputs provide conceptual links between the execution elements of **JDF** nodes. Outputs of one node can in turn become inputs in another node, and a given node SHALL NOT be executed before @*Status* all specified input resources is greater than or equal to ***ResourceLink***/@*MinStatus* or ***ResourceLink***/@*MinLateStatus*.[1] Figure 3.6 shows two processes that are linked by a resource. The resource represents the output of node 1, which in turn becomes an input for node 2.

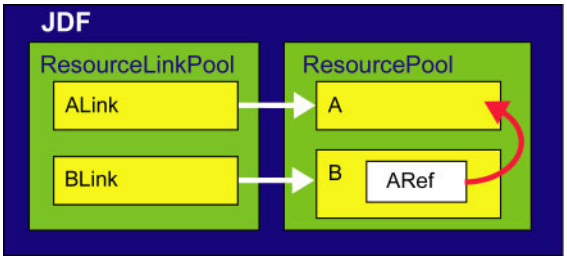
Figure 3-6: Nodes linked by a Resource



ResourceLink elements also allow node dependencies to be calculated. The following diagram summarizes resource linking within a **JDF** node. In this example there are two resources, A and B, which are placed in the node's ***ResourcePool***. To reference the resources, the node has two ***ResourceLink*** elements, ALink and BLink, in the ***ResourceLinkPool***. A ***ResourceLink*** is named by appending "Link" to the type of resource referenced. Resource B also contains a reference to resource A, called ARef. References to resources from within resources are named by appending "Ref" to the type of resource referenced (see ▸ Section 3.10.2 ResourceRef – Element for Inter-Resource Linking and refelement).

1. The availability of a resource that is consumed as a whole is given by the ***Resource*** attribute @*Status* = "Available". In the case of pipe resources, the availability depends on the individual parameter defining the dynamics of a pipe. For details see ▸ Section 4.3.3 Overlapping Processing Using Pipes.

Figure 3-7: ResourceLink Elements and ResourceRef Elements



The previous section describes resources used by the node in which it resides. This section describes how resources can serve as links between nodes. As was described in ▸ Section 2.2 JDF Workflow, any resource that is the output of one process will very likely serve as an input of a subsequent process. Furthermore, some resources are shared between ancestor nodes and their child nodes.

ResourceLink elements MAY also contain attributes to select a part of a resource, such as a single separation. A detailed description of resource partitioning is given in ▸ Section 3.10.5 Description of Partitioned Resources.

Because implementation ***ResourceLink*** elements define the usage of a specific device during the course of a job, situations can arise where that resource is not needed during the whole processing time. For instance, a forklift that only has to transport the completed components need not be available during the entire process run, only during the times when it is needed. This means that, contrary to the general rule that all resources SHALL be "Available" for node execution to commence, a node can commence when ***ImplementationResource***s are still "InUse" by other processes if @*Start* or @*StartOffset* are specified.***ResourceLink*** elements always have a @*Usage* of "Input".

ProcessGroup and product intent nodes can be defined without the knowledge of the individual process nodes that define a specific workflow. In this case, these intermediate nodes will contain ***ResourceLink*** elements that link the appropriate resources. For example, a prepress node might be defined that produces a set of plates. When the processes for creating the plates are defined in detail, the agent that writes the nodes might remove the ***ResourceLink*** elements from the intermediate node. Removing the ***ResourceLink*** specifies that the intermediate node can execute (i.e., it can be sent to the appropriate controller or department), even though the specific resources are not yet available. If the ***ResourceLink*** elements are not removed, the intermediate node cannot execute until the linked input resources become available.

ResourceLink elements MAY be used for process control. For example, if a proof input resource is needed for a print process, a print run can commence only when the proof is signed. The **JDF** format specification also includes a complete specification of how resources are managed when **JDF** tickets are spawned and merged.

In some cases, determining whether to store information in an input or an output resource can be difficult, as the distinction can be ambiguous. For example, is the definition of the color of a separation in the RIP process a property of the output separation or a parameter that describes the RIP process? In order to reduce this ambiguity, the following rules have been defined for input and output resources of processes (see ▸ Chapter 6 Processes and ▸ Chapter 8 Resources).

- Product intent and process parameters are generally input resources, except when one process defines the parameters of a subsequent process.
- ***Consumable Resource***s SHALL always be input resources.
- ***Quantity Resource***s and ***Handling Resource***s are used both as input resources and output resources. Their usage is defined by the "natural" process usage. For example, a printing plate is described as a resource that is the output of a process and the input of a process.
- Processed material is exchanged from node to node using the ***Component*** resource. Product intent nodes also create ***Component*** output resources.
- Every detailed process description SHALL be defined as an input parameter of the first process where it is referenced. This means that a device SHALL NOT infer process parameters from its output resources. For example, paper weight in grams MAY be defined in the ***Component*** output resource of the printing process but SHALL be defined as an input parameter of the ***Media*** of the printing process.
- Any resource parameter that is used SHALL be referenced explicitly. Resource parameters cannot be inferred by following the chain of nodes backwards. This would make spawning of nodes non-local.
- The last process in a chain of processes SHALL define the output resource of its parent process.
- In case of parallel processing, the sum of the outputs of all parallel subnodes SHALL define the output of the parent node.

Like ***Resource*** elements, ***ResourceLink*** elements are an abstract data type. The class tree of abstract ***ResourceLink*** elements is further subdivided into classes defined by the @*Class* attribute of the resource that it references. Individual instances of ***ResourceLink*** elements are named by appending the suffix "Link" to the name of the referenced resource. For example, the link to a ***Component*** resource is entitled ***ComponentLink*** and the link to a ***ScanParams*** resource is entitled ***ScanParamsLink***.

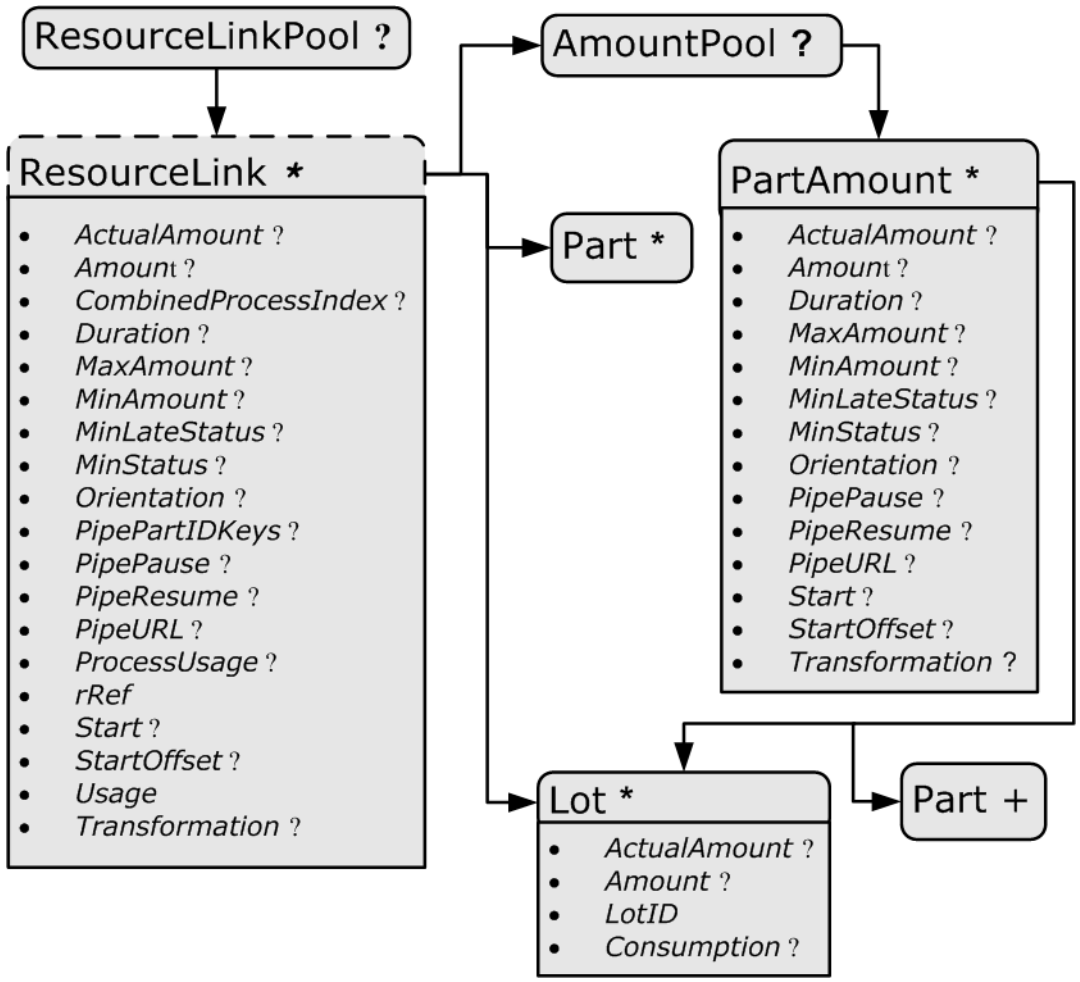
It is important to note that the order of occurrence of links to ***PhysicalResource***s MAY be significant. For example, a **Gathering** process might have among its inputs, links to three ***Component*** resources. The order of these links indicates the order in which the ***Component*** resources are to occur in the new, gathered output ***Component***.

Note: Starting with **JDF** 1.5 ***ConsumableLink***, ***HandlingLink***, ***ImplementationLink***, ***ParameterLink***, ***PlaceHolderLink*** and ***QuantityLink*** have been removed and all their attributes and subelements moved to ***ResourceLink***.

3.9.3 Structure Diagram

Figure 3-8: ResourceLinkPool and ResourceLink Element – a diagram of the structure shows the abstract types derived from the ***ResourceLink*** type.

Figure 3-8: ResourceLinkPool and ResourceLink Element – a diagram of the structure



The following table lists the possible contents of all ***ResourceLink*** elements.

Table 3.16: ResourceLink Element (Sheet 1 of 5)

NAME	DATA TYPE	DESCRIPTION
ActualAmount ? New in JDF 1.2	double	Total amount of the resource that has been produced (in a ***ResourceLink*** with @*Usage* = "Output") or consumed (in a ***ResourceLink*** with @*Usage* = "Input") by this node in every execution. For details see Section 3.10.4 Resource Amount. **Note:** In **JDF** 1.5 @*ActualAmount* was moved from deleted abstract ***PhysicalLink***.
Amount ?	double	For a link with a @*Usage* of "Input", specifies the amount of the resource that is needed by the process, in units as defined in the resource. For a link with a @*Usage* of "Output", specifies the amount of the resource that is to be produced by the process, in units as defined in the resource. Allows resources to be only partially consumed or produced (see Section 3.10.4 Resource Amount). If not specified, ***ResourceLink***/@*Amount* defaults to ***Resource***/@*Amount*. **Note:** In **JDF** 1.5 @*ActualAmount* was moved from the deleted abstract ***PhysicalLink***.

Table 3.16: ResourceLink Element (Sheet 2 of 5)

NAME	DATA TYPE	DESCRIPTION
CombinedProcessIndex ? New in JDF 1.1	IntegerList	Combined process nodes and process group nodes MAY contain resources from multiple process nodes. The @*CombinedProcessIndex* attribute specifies the indices of individual processes in the @*Types* attribute to which a ***ResourceLink*** in a combinded process nodes or process group node belongs. Multiple entries in @*CombinedProcessIndex* specify that the ***ResourceLink*** is used by the respective multiple processes in the combinded process node. It SHALL be specified when multiple resources of the same name and ***ResourceLink***/@*Usage* are specified in one **JDF** node. If @*CombinedProcessIndex* is not specified, even though multiple processes in the combinded process node or process group node MAY link to the resource, the ***ResourceLink*** applies to all of these processes.
CombinedProcessType ? Deprecated in JDF 1.1	NMTOKEN	Combined process nodes contain input resources from multiple process nodes. The @*CombinedProcessType* attribute specifies the name individual process to which a ***ResourceLink*** in a combinded process node belongs. It SHALL match one of the entries in the @*Types* attribute of the node. **Deprecation note:** @*CombinedProcessType* was replaced by @*CombinedProcessIndex* in **JDF** 1.1.
DraftOK ? Deprecated in JDF 1.3	boolean	If "true", the process can commence with a draft resource. Default = "false". Replaced with @*MinLateStatus* and @*MinStatus* in **JDF** 1.3 and beyond.
Duration ? Modified in JDF 1.4	duration	Estimated duration during which the resource will be used. **Modification note:** Starting with **JDF** 1.4, @*Duration* is moved from abstract ImplementationLink table, which was then deleted in **JDF** 1.5.
MaxAmount ? New in JDF 1.3	double	Defines the planned @*Amount* including the maximum overage. If not specified, defaults to a system specified value based on @*Amount*. **Note:** In **JDF** 1.5 @*MaxAmount* was moved from the deleted abstract ***PhysicalLink***.
MinAmount ? New in JDF 1.3	double	Defines the planned @*Amount* including the maximum underage that the customer is willing to accept. If not specified, defaults to a system specified value based on @*Amount*. **Note:** In **JDF** 1.5 @*MinAmount* was moved from the deleted abstract ***PhysicalLink***.
MinLateStatus ? New in JDF 1.3	enumeration	Minimum value of ***Resource***/@*Status* for the execution of this node to commence when deadlines are endangered (i.e., when the time defined by ***NodeInfo***/@*LastStart* or implied by ***NodeInfo***/@*LastEnd* is approaching). **Default value is from: @*MinStatus*.** **Allowed values are from: ***Resource***/@*Status*.**
MinStatus ? New in JDF 1.3	enumeration	Minimum value of ***Resource***/@*Status* for the execution of this node to commence. **Default value is:** "Available" if @*Usage* = "Input" and "Unavailable" if @*Usage* = "Output". **Allowed values are from: ***Resource*/@*Status*.**
Orientation ? New in JDF 1.1	Orientation	Named orientation describing the transformation of the orientation of a ***PhysicalResource*** relative to the ideal process coordinate that uses this resource as input or output. If @*Orientation* is specified for an output resource, the node that processes the ***PhysicalResource*** is to manipulate the resource in such a way as to reflect the transformation. The coordinate system of the resource itself is *not* modified. At most one of @*Orientation* or @*Transformation* SHALL be specified. For details on coordinate systems, see ▸ Section 2.5 Coordinate Systems in JDF. **Note:** In **JDF** 1.5 @*Orientation* was moved from the deleted abstract ***PhysicalLink***. **Allowed values are from:** ▸ Orientation

Table 3.16: ResourceLink Element (Sheet 3 of 5)

NAME	DATA TYPE	DESCRIPTION
PipePartIDKeys ? Deprecated in JDF 1.6	enumerations	Defines the granularity of a dynamic pipe for a partitioned resource. For instance, if a resource were partitioned by sheet, surface and separation (i.e., ***Resource***/@*PartIDKeys* = "SheetName Side Separation") and if the ***ResourceLink***/@*PipePartIDKeys* = "SheetName Side", then pipe requests would be issued only once per surface. The contents of @*PipePartIDKeys* SHALL be a subset of the @*PartIDKeys* attribute of the resource that is linked by this ***ResourceLink***. **Default value is from:** the implied or explicit value of @*PipePartIDKeys* of the referenced resource. Allowed values are from: ***Resource***/@*PartIDKeys*.
PipePause ?	double	Parameter for controlling the pausing of a process if the resource amount in the pipe buffer passes the specified value. For details on using @*PipePause*, see ▸ Section 4.3.3 Overlapping Processing Using Pipes. **Note:** In **JDF** 1.5 @*PipePause* was moved from the deleted abstract ***PhysicalLink***.
PipeProtocol ? New in JDF 1.1 Modified in JDF 1.2 Deprecated in JDF 1.5	NMTOKEN	Defines the protocol use for pipe handling. See ▸ Section 4.3.3 Overlapping Processing Using Pipes. "JMF" and "Internal" are the only non-proprietary piping protocols that are supported. Proprietary pipe protocols MAY be specified in addition to those defined below but will not necessarily be interoperable. **Default value is:** "JMF" (if @*PipeURL* is specified); otherwise referenced ***Resource***/@*PipeProtocol*. **Values include:** Internal – Internal or virtual pipe used within a combinded process. New in JDF 1.2 JMF – JMF based ***PipePush*** / ***PipePull*** messages. None – No pipe support. **Deprecation note:** Starting with **JDF** 1.5, use ***Resource***/@*PipeProtocol*.
PipeResume ?	double	Parameter for controlling the resumption of a process if the resource amount in the pipe buffer passes the specified value. For details on using @*PipeResume*, see ▸ Section 4.3.3 Overlapping Processing Using Pipes. **Note:** In **JDF** 1.5 @*PipeResume* was moved from the deleted abstract ***PhysicalLink***.
PipeURL ? Modified in JDF 1.2	URL	Pipe request URL. Dynamic pipe requests from this end of a pipe SHOULD be made *to* this URL. In most cases this URL is the URL of the controller of the *other end* of the pipe. This might seem counterintuitive, but it allows parallel spawning and merging of processes that represent a dynamic pipe without having to include the node that describes the other end in the spawned file. **Default value is:** referenced ***Resource***/@*PipeURL* **Note:** @*PipeURL* is only used for initiating pipe requests. Responses to a pipe request are issued to the URL that is defined in the ***PipePush*** or ***PipePull*** message. For details on using @*PipeURL*, see ▸ Section 4.3.3 Overlapping Processing Using Pipes.
ProcessUsage ? Modified in JDF 1.4	string	Identifies a process's usage of a resource if multiple resources of the same type can be supplied. For example, this attribute appears when two ***Component*** resources—one Cover and one BookBlock—are used in **CoverApplication**. **Values include those from:** ▸ Table 3.17 ProcessUsage Attribute Values. **Values include those from:** ICS documents. New in JDF 1.4 **Note:** The values of @*ProcessUsage* can be derived from the appropriate process descriptions in ▸ Section Processes. ▸ Section 6.1 Process Template defines the parenthesized notation for denoting the value of @*ProcessUsage* (e.g., ***Component*** (Cover)).
Recommendation ? Deprecated in JDF 1.2	boolean	If "true" and the request cannot be fulfilled, the change MAY be logged as a ***Modified*** audit and the job can continue. If "false", an error occurs if the request is not fulfilled. In **JDF** 1.2 and beyond use @*SettingsPolicy* instead. **Note:** In **JDF** 1.5 @*Recommendation* was moved from the deleted abstract ***ImplementationLink***.

Table 3.16: ResourceLink Element (Sheet 4 of 5)

NAME	DATA TYPE	DESCRIPTION
RemotePipeEndPause ? Deprecated in JDF 1.5	double	Parameter for controlling the pausing of a process at the other end of the pipe if the resource amount in the pipe buffer passes the specified value. For details on using @*RemotePipeEndPause*, see ▸ Section 4.3.3 Overlapping Processing Using Pipes. **Note:** In **JDF** 1.5 @*RemotePipeEndPause* was moved from the deleted abstract ***PhysicalLink***. **Deprecation note:** Starting with **JDF** 1.5, use @*PipePause*.
RemotePipeEndResume ? Deprecated in JDF 1.5	double	Parameter for controlling the resumption of a process at the other end of the pipe if the resource amount in the pipe buffer passes the specified value. For details on using @*RemotePipeEndResume*, see ▸ Section 4.3.3 Overlapping Processing Using Pipes. **Note:** In **JDF** 1.5 @*RemotePipeEndResume* was moved from the deleted abstract ***PhysicalLink***. **Deprecation note:** Starting with **JDF** 1.5, use @*PipeResume*.
rRef	IDREF	Link to the target resource.
rSubRef ? Deprecated in JDF 1.2	IDREF	Link to a subelement within the resource. In **JDF** 1.2 and beyond, a ***ResourceLink*** is able to reference a resource only if it is a direct child of a ***ResourcePool***.
Start ? Modified in JDF 1.4	dateTime	Time and date when the usage of the resource starts. **Note:** In **JDF** 1.4 @*Start* was moved from the deleted abstract ***ImplementationLink***, which was deleted in **JDF** 1.5.
StartOffset ? Modified in JDF 1.4	duration	Offset time when the resource is needed after processing has begun. If both @*Start* and @*StartOffset* are specified, @*Start* has precedence. **Note:** In **JDF** 1.4 @*StartOffset* was moved from the deleted abstract ***ImplementationLink***, which was deleted in **JDF** 1.5.
Usage	enumeration	Resource usage within this **JDF** node. **Allowed values are:** Input – The resource is an input. Output – The resource is an output.
Transformation ? New in JDF 1.1	matrix	Matrix describing the transformation of the orientation of a ***PhysicalResource*** relative to the ideal process coordinate using this resource as input or output. If @*Transformation* is specified for an output resource, the node that processes the ***PhysicalResource*** is to manipulate the resource in such a way as to reflect the transformation. The coordinate system of the resource itself is *not* modified. At most one of @*Orientation* or @*Transformation* SHALL be specified. For details on coordinate systems, see ▸ Section 2.5 Coordinate Systems in JDF. **Note:** In **JDF** 1.5 @*Transformation* was moved from the deleted abstract ***PhysicalLink***.
AmountPool ? New in JDF 1.1 Modified in JDF 1.2	element	Definition of partial amounts and pipe parameters for this ***ResourceLink***. The allowed contents of the ***AmountPool*** are described for the various subclasses of ***ResourceLink*** in the sections below. If ***AmountPool*** is specified, ***ResourceLink*** SHALL NOT contain any of @*Amount*, @*ActualAmount*, @*MaxAmount* or @*MinAmount*
Lot * New in JDF 1.3	element	Group of identifiers that uniquely identifies one lot of a resource. If multiple resource lots are planned to be consumed by a process, this element MAY appear multiple times to identify each resource lot. Examples of resource lots are individual rolls of paper, boxes of paper, cans of ink, etc. See ▸ Section 3.9.4 Identification of Physical Resources for details. For resources that are solely identified by @*ProductID*, ***Lot*** element(s) NEED NOT be specified. **Note:** @*Lot* was moved from abstract ***PhysicalLink*** which was deleted in **JDF** 1.5.

Table 3.16: ResourceLink Element (Sheet 5 of 5)

NAME	DATA TYPE	DESCRIPTION
Part *	element	The ***Part*** elements identify the parts of a partitioned resource that are referenced by the ***ResourceLink***. The structure of the ***Part*** element is defined in ▸ Table 3.26 Part Element. For details on partitioned resources, see ▸ Section 3.10.5 Description of Partitioned Resources.

3.9.3.1 ProcessUsage Attribute Values

Table 3.17: ProcessUsage Attribute Values

VALUE	DESCRIPTION
Accepted	Used for ***Resource*** in an output resource of **Approvall**
Application	Used for ***Component*** in an input resource of **BoxFolding**
BackEndSheet	Used for ***Component*** in an input resource of **EndSheetGluing**
Book	Used for ***Component*** in an input resource of **Jacketing**
BookBlock	Used for ***Component*** in an input resource of **ChannelBinding**, **EndSheetGluing** and **RingBinding**
Box	Used for ***Component*** in an input resource of **BoxPacking**
Case	Used for ***Component*** in an input resource of **CasingIn**
Child	Used for ***Component*** in an input resource of **Inserting**
Cover	Used for ***Component*** in an input resource of **ChannelBinding** and **CoverApplication**.
CoverBoard	Used for ***Media*** in an input resource of **CaseMaking**.
CoverMaterial	Used for ***Component*** and ***Media*** in an input resource of **CaseMaking**.
Cylinder	Used for ***ExposedMedia*** in an input resource of **ConventionalPrinting**.
Document	Used for ***RunList*** in an input resource of **Imposition**, **LayoutPreparation** and **Stripping** and used for ***RunList*** in an output resource of **Stripping**.
FrontEndSheet	Used for ***Component*** in an input resource of **EndSheetGluing**.
Good	Used for ***Component*** in an output resource of **ConventionalPrinting** and **DigitalPrinting**.
Input	Used for ***Component*** in an input resource of **ConventionalPrinting** and **DigitalPrinting**.
Jacket	Used for ***Component*** in an input resource of **Jacketing**.
Label	Used for ***Component*** in an input resource of **Labeling**.
Marks	Used for ***RunList*** in an input resource of **Imposition**, **LayoutPreparation** and **Tiling**, and used for ***RunList*** in an output resource of **LayoutPreparation** and **Stripping**.
Mother	Used for ***Component*** in an input resource of **Inserting**.
Plate	Used for ***ExposedMedia*** in an input resource of **ConventionalPrinting**.
Proof	Used for ***Component*** in an input resource of **ConventionalPrinting** and **DigitalPrinting**, and used for ***ExposedMedia*** in an input resource of **ConventionalPrinting**.
Rejected	Used for ***Resource*** in an output resource of **Approval**.
RingBinder	Used for ***Component*** in an input resource of **RingBinding**.
SpineBoard	Used for ***Media*** in an input resource of **CaseMaking**.
Surface	Used for ***RunList*** in an input resource of **Tiling**.
Tie	Used for ***Media*** in an input resource of **BoxPacking**.
Underlay	Used for ***Media*** in an input resource of **BoxPacking**.
Waste	Used for ***Component*** in an output resource of **ConventionalPrinting** and **DigitalPrinting**.

Example 3.5: EmployeeLink

The following example shows how the operator Smith is linked to a **ConventionalPrinting** process as the only valid oper-

ator.

```
<ResourcePool>
    <Employee Class="Implementation" ID="L1" Status="Available"
          PersonalID="007">
       <Person FamilyName="Smith" JobTitle="Press Operator"/>
    </Employee>
</ResourcePool>
<ResourceLinkPool>
    <EmployeeLink Usage="Input" rRef="L1"/>
</ResourceLinkPool>
```

3.9.3.2 AmountPool and PartAmount

New in JDF 1.1

Whereas ***ResourceLink***/***Part*** identifies the ***Resource*** that the process is consuming or producing, ***AmountPool*** is a container for the amount-related metadata of the resource. Thus process routing is described by ***ResourceLink***/***PartAmount***/***Part*** whereas tracking of amount related attributes are described by ***AmountPool***/***PartAmount***. ***AmountPool***/***PartAmount***/***Part*** SHALL refer to existing partitions or non-existing sub-partitions of existing partitions that are implicitly or explicitly referred to by ***ResourceLink***/***Part***. For instance, if a ***ResourceLink*** refers to a partition with @*SheetName*="Sheet1", ***AmountPool***/***PartAmount*** MAY refer to Sheet1 or any existing or non-existing child of Sheet1, but NOT to Sheet2 or any existing or non-existing child of Sheet2.

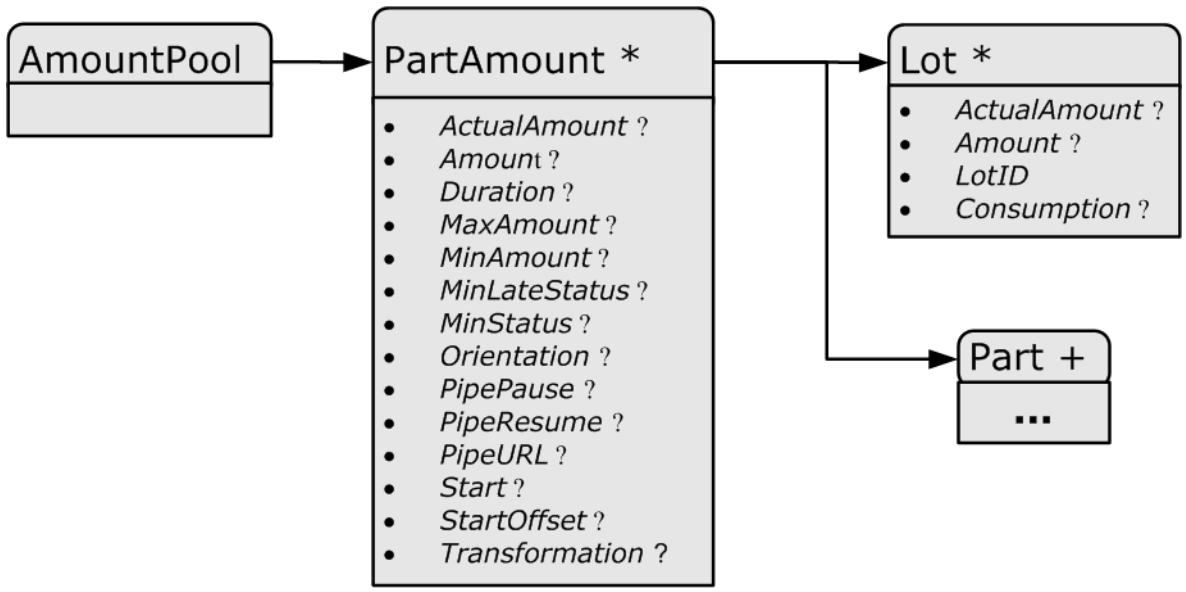
3.9.3.2.1 AmountPool

The following table lists the generic contents of an ***AmountPool*** element. Further parameters of the ***AmountPool*** are described in the sections below.

Table 3.18: AmountPool Element

NAME	DATA TYPE	DESCRIPTION
PartAmount *	element	Element that defines the amounts and pipe parameters for a partitioned resource. The contents of a ***PartAmount*** depends on the type of the ***ResourceLink***

Figure 3-9: AmountPool – a Diagram of its Structure



3.9.3.2.2 PartAmount

The following table lists the generic contents of a ***PartAmount*** element. Note that ***PartAmount*** inherits values from its parent ***ResourceLink***.

Table 3.19: PartAmount Element (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
ActualAmount ? New in JDF 1.2	double	Total amount of the resource that has been produced (in a ***ResourceLink*** with @*Usage* = "Output") or consumed (in a ***ResourceLink*** with @*Usage* = "Input") by this node in every execution. For details see ▸ Section 3.10.4 Resource Amount. **Note:** In **JDF** 1.5 @*AcutalAmount* was moved from the deleted abstract ***PhysicalLink***.
Amount ?	double	For a link with a @*Usage* of "Input", specifies the amount of the resource that is needed by the process, in units as defined in the resource. For a link with a @*Usage* of "Output", specifies the amount of the resource that is to be produced by the process, in units as defined in the resource. Allows resources to be only partially consumed or produced (see ▸ Section 3.10.4 Resource Amount). If not specified, ***ResourceLink***/@*Amount* defaults to ***Resource***/@*Amount*. **Note:** In **JDF** 1.5 @*Amount* was moved from the deleted abstract ***PhysicalLink***.
DraftOK ? Deprecated in JDF 1.3	boolean	If "true", the process can commence with a draft resource partition. Replaced with @*MinLateStatus* and @*MinStatus* in **JDF** 1.3 and beyond.
Duration ? Modified in JDF 1.4	duration	Estimated duration during which the resource will be used. **Note:** In **JDF** 1.4 @*Duration* was moved from ***ImplementationLink***, which was deleted in **JDF** 1.5.
MaxAmount ? New in JDF 1.3	double	Defines the planned @*Amount* including the maximum overage. If not specified, defaults to a system specified value based on @*Amount*. **Note:** In **JDF** 1.5 @*MaxAmount* was moved from the deleted abstract ***PhysicalLink***.
MinAmount ? New in JDF 1.3	double	Defines the planned @*Amount* including the maximum underage that the customer is willing to accept. If not specified, defaults to a system specified value based on @*Amount*. **Note:** In **JDF** 1.4 @*MinAmount* was moved from the deleted abstract ***PhysicalLink***.
MinLateStatus ? New in JDF 1.3	enumeration	Minimum value of ***Resource***/@*Status* for the execution of this node to commence when deadlines are endangered (i.e., when the time defined by ***NodeInfo***/@*LastStart* or implied by ***NodeInfo***/@*LastEnd* is approaching). **Default value is from:** @*MinStatus*. **Allowed values are from:** ***Resource***/@*Status*.
MinStatus ? New in JDF 1.3	enumeration	Minimum value of ***Resource***/@*Status* for the execution of this node to commence. **Default value is:** "Available" if @*Usage*="Input" and "Unavailable" if @*Usage* = "Output". **Allowed values are from:** ***Resource***/@*Status*.
Orientation ? New in JDF 1.1	enumeration	Named orientation describing the transformation of the orientation of a ***PhysicalResource*** relative to the ideal process coordinate that uses this resource as input or output. If @*Orientation* is specified for an output resource, the node that processes the ***PhysicalResource*** is to manipulate the resource in such a way as to reflect the transformation. The coordinate system of the resource itself is *not* modified. At most one of @*Orientation* or @*Transformation* SHALL be specified. For details on coordinate systems, see ▸ Section 2.5 Coordinate Systems in JDF. **Note:** In **JDF** 1.5 @*Orientation* was moved from the deleted abstract ***PhysicalLink***. **Allowed values are from:** ▸ Orientation.
PipePause ?	double	Parameter for controlling the pausing of a process if the resource amount in the pipe buffer passes the specified value. For details on using @*PipePause*, see ▸ Section 4.3.3 Overlapping Processing Using Pipes. **Note:** In **JDF** 1.5 @*PipePause* was moved from the deleted abstract ***PhysicalLink***.

Table 3.19: PartAmount Element (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
PipeResume ?	double	Parameter for controlling the resumption of a process if the resource amount in the pipe buffer passes the specified value. For details on using @*PipeResume*, see ▸ Section 4.3.3 Overlapping Processing Using Pipes. **Note:** In **JDF** 1.5 @*PipeResume* was moved from the deleted abstract ***PhysicalLink***.
PipeURL ?	URL	Pipe request URL for this partition. Dynamic pipe requests from this end of a pipe SHOULD be made to this URL. Note that this URL is only used for initiating pipe requests. Responses to a pipe request are issued to the URL that is defined in the ***PipePush*** or ***PipePull*** Message. For details on using @*PipeURL*, see ▸ Section 4.3.3 Overlapping Processing Using Pipes. **Note:** In most cases @*PipeURL* is the URL of the controller of the *other end* of the pipe. This might seem counterintuitive, but it allows parallel spawning and merging of processes that represent a dynamic pipe without having to include the node that describes the other end in the spawned file.
RemotePipeEndPause ? Deprecated in JDF 1.5	double	Parameter for controlling the pausing of a process at the other end of the pipe if the resource amount in the pipe buffer passes the specified value. For details on using @*RemotePipeEndPause*, see ▸ Section 4.3.3 Overlapping Processing Using Pipes. **Note:** In **JDF** 1.5 @*RemotePipeEndPause* was moved from the deleted abstract ***PhysicalLink***. **Deprecation note:** Starting with **JDF** 1.5, use @*PipePause*.
RemotePipeEndResume ? Deprecated in JDF 1.5	double	Parameter for controlling the resumption of a process at the other end of the pipe if the resource amount in the pipe buffer passes the specified value. For details on using @*RemotePipeEndResume*, see ▸ Section 4.3.3 Overlapping Processing Using Pipes. **Note:** In **JDF** 1.5 @*RemotePipeEndResume* was moved from the deleted abstract ***PhysicalLink***. **Deprecation note:** Starting with **JDF** 1.5, use @*PipeResume*.
Start ? Modified in JDF 1.4	dateTime	Time and date when the usage of the resource starts. **Note:** In **JDF** 1.4 @*Start* was moved from the abstract ***ImplementationLink***, which was deleted in **JDF** 1.5.
StartOffset ? Modified in JDF 1.4	duration	Offset time when the resource is needed after processing has begun. If both @*Start* and @*StartOffset* are specified, @*Start* has precedence. **Note:** In **JDF** 1.4 @*StartOffset* was moved from the abstract ***ImplementationLink***, which was deleted in **JDF** 1.5.
Transformation ? New in JDF 1.1	matrix	Matrix describing the transformation of the orientation of a ***PhysicalResource*** relative to the ideal process coordinate using this resource as input or output. If @*Transformation* is specified for an output resource, the node that processes the ***PhysicalResource*** is to manipulate the resource in such a way as to reflect the transformation. The coordinate system of the resource itself is *not* modified. At most one of @*Orientation* or @*Transformation* SHALL be specified. For details on coordinate systems, see ▸ Section 2.5 Coordinate Systems in JDF. **Note:** In **JDF** 1.5 @*Transformation* was moved from the deleted abstract ***PhysicalLink***.
Lot * New in JDF 1.3	element	Group of identifiers that uniquely identifies one lot of a resource. If multiple resource lots are planned to be consumed by a process, this element MAY appear multiple times to identify each resource lot. Examples of resource lots are individual rolls of paper, boxes of paper, cans of ink, etc. See ▸ Section 3.9.4 Identification of Physical Resources for details. For resources that are solely identified by @*ProductID*, ***Lot*** element(s) NEED NOT be specified. **Note:** In **JDF** 1.5 @*Lot* was moved from the deleted abstract ***PhysicalLink***.

Table 3.19: PartAmount Element (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
Part + Modified in JDF 1.3	***element***	Specifies the selected parts that the ***PartAmount*** is valid for. The granularity of ***Part*** shall be at least that of a leaf partition of the ***Resource***. For instance, a ***Component*** MAY be partitioned by @*SheetName* and ***PartAmount*** could refer to @*SheetName* and @*Condition*. Multiple ***Part*** elements specify that the referenced elements have been processed in one step, for instance two separations on a press run of a two color press.

Example 3.6: PartAmount

The following example shows an ***InkLink*** with an ***AmountPool***.

```
<ResourcePool>
    <Ink Brand="NoName" Class="Consumable" ID="Link0015"
         PartIDKeys="Separation" Status="Available">
        <Ink ColorName="Cyan" Separation="Cyan"/>
        <Ink ColorName="Magenta" Separation="Magenta"/>
        <Ink ColorName="Yellow" Separation="Yellow"/>
        <Ink ColorName="Black" Separation="Black"/>
        <Ink ColorName="Heidelberg Spot Blau"
             Separation="Heidelberg Spot Blau"/>
    </Ink>
</ResourcePool>
<ResourceLinkPool>
    <InkLink Usage="Input" rRef="Link0015">
        <AmountPool>
            <PartAmount Amount="1000">
                <Part Separation="Cyan"/>
            </PartAmount>
            <PartAmount Amount="1200">
                <Part Separation="Magenta"/>
            </PartAmount>
            <PartAmount Amount="700">
                <Part Separation="Yellow"/>
            </PartAmount>
            <PartAmount Amount="3000">
                <Part Separation="Black"/>
            </PartAmount>
            <PartAmount Amount="300">
                <Part Separation="Heidelberg Spot Blau"/>
            </PartAmount>
        </AmountPool>
    </InkLink>
</ResourceLinkPool>
```

3.9.4 Identification of Physical Resources

New in JDF 1.3

MIS systems frequently include functionality for managing inventory. Many ***PhysicalResources*** that are consumed by production processes are things that are tracked for inventory management purposes. This allows estimating the value of the resources, ensuring that sufficient quantities are on hand, and tracking which specific resources are used in production of which jobs. At the most basic level, these ***PhysicalResources*** MAY be identified in **JDF** with ***Resource***/@*ProductID*.

Some MIS systems track these resources at lower levels of detail, tracking individual resource lots. An example of this might include tracking the individual rolls or boxes of paper. While it is theoretically possible to track individual resource lots using a single identifier, many MIS users choose to track them with more than one identifier. Examples of some of these identifiers include roll numbers, lot numbers, purchase order numbers, receipt dates.

Because the required identifiers may be different from site to site, or even from one type resource to another, it is not possible to track these resources with multiple identifiers using **JDF**. Conveying the identification requirements to devices would be too complex. Instead, a single identifier is used in **JDF**. In cases where multiple identifiers are normally used, the MIS SHALL generate a unique identifier for each unique resource lot. This unique identifier SHALL then be mapped back to the correct unique resource lot by the MIS.

3.9.4.0.1 Lot

Deprecated in JDF 1.6

3.10 ResourcePool and ResourceLinkPool – Deep Structure

This section describes the deep structure of a ***ResourcePool*** and ***ResourceLinkPool***. In particular this section describes the ***ResourceRef*** which references a resource from inside another resource. This section also describes resource sets and the partitioning of them.

3.10.1 ResourceElement – Subelement of a Resource

3.10.1.1 ResourceElement

A ***ResourceElement*** is always a subelement of a resource or subelement of a **JMF** message

3.10.1.2 Abstract ResourceElement

An ***Abstract ResourceElement*** is defined in the ▸ Table 3.20 Abstract ResourceElement below. A ***ResourceElement*** does not inherit from the ***Abstract Resource***. Examples of ***ResourceElement*** resources are ***SeparationSpec*** and ***MISDetails***.

Table 3.20: Abstract ResourceElement

NAME	DATA TYPE	DESCRIPTION
ID ? Deprecated in JDF 1.2	ID	Unique identifier of a ***Resource*** element. In **JDF** 1.2 and beyond, an element that is not a direct child of a ***ResourcePool*** SHOULD NOT contain an @*ID*. This @*ID* SHALL NOT be referenced by ***ResourceRef***/@*rRef* or ***ResourceLink***/@*rRef* because a ***ResourceRef*** or ***ResourceLink*** element is able to reference a ***Resource*** only if it is a direct child of a ***ResourcePool***.

3.10.2 ResourceRef – Element for Inter-Resource Linking and refelement

3.10.2.1 ResourceRef

In some cases, it is necessary to reference a ***Resource*** element directly from another element in order to reuse information. Such a reference is a ***ResourceRef*** element. A ***ResourceRef*** element's name is generated by appending the string "Ref" to the element's name. Candidate elements for inter-Resource linking have a data type of refelement in the content description tables of this chapter and ▸ Chapter 8 Resources. A data type of **refelement** allows either a ***ResourceRef*** or an inline ***Resource*** element. In the latter case, the ***Resource*** element inherits attributes and elements from the ***Abstract Resource*** and (where appropriate) from the ***Abstract Parameter Resource*** or the ***Abstract PhysicalResource***. Note that some attributes and elements in these abstract elements have rules for inline resource subelements that differ from the rules for a resource root.

3.10.2.2 Abstract ResourceRef

The following table defines the attributes of the ***Abstract ResourceRef*** element (see also ▸ Figure 3-5: ResourcePool and Abstract Resource Element – a diagram of the structure and ***ResourceElement*** in ▸ Table 3.10 Abstract Resource Element).

The ***Part*** element in a ***ResourceRef*** defines the part of the target that this ***ResourceRef*** references. If both the ***Resource*** that contains ***ResourceRef*** element and the target ***Resource*** are partitioned, the ***ResourceRef*** does *not* implicitly reference the part of the target with the same partitioning attributes, but rather the parts of the target ***Resource*** that are explicitly specified by the ***Part*** element within the ***ResourceRef***.

When a ***ResourceRef*** references a partitioned resource node that is not a resource leaf, the children of the referenced resource are ignored. See ▸ Example 76 MediaRef to Partitioned Media and ▸ Example 76 Equivalent Inline Media for an illustration of this equivalence. Otherwise, the referenced structure would be a partitioned element and thus invalid when inlined. See ▸ Example 76 Invalid Inline Partitioned Media.

Table 3.21: Abstract ResourceRef Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
rRef	IDREF	Reference to the resource. The linked resource SHALL be a direct child of a ***ResourcePool***.

Table 3.21: Abstract ResourceRef Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
rSubRef ? Deprecated in JDF 1.2	IDREF	Reference to a subelement of the ***Resource***. In **JDF** 1.2 and beyond, a ***ResourceRef*** element is able to reference a ***Resource*** only if it is a direct child of a ***ResourcePool***
Part ? New in JDF 1.1	element	Definition of the partition that this ***ResourceRef*** references.

Example 3.7: MediaRef to Partitioned Media

MediaRef references ***Media*** and its children are ignored:

```
<Media Class="Consumable" Dimension="72 72" ID="MediaID" PartIDKeys="Location"
       Status="Available">
    <Comment Name="foo">bar</Comment>
    <Media Location="desk"/>
    <Media Location="drawer"/>
</Media>
<Layout Class="Parameter" ID="Sheet" Status="Available">
    <MediaRef rRef="MediaID"/>
</Layout>
```

Example 3.8: Equivalent Inline Media

Media is inline in ***Layout***. This is equivalent to the preceding ▸ Example 76 MediaRef to Partitioned Media with ***MediaRef***:

```
<Layout Class="Parameter" ID="Sheet" Status="Available">
    <Media Dimension="72 72">
        <Comment Name="foo">bar</Comment>
    </Media>
</Layout>
```

Example 3.9: Invalid Inline Partitioned Media

This example takes the ***Media*** from ▸ Example 76 MediaRef to Partitioned Media and make it be inline in ***Layout***. The result is an invalid partition:

```
<Layout Class="Parameter" ID="Sheet" Status="Available">
    <Media Dimension="72 72" PartIDKeys="Location">
        <Comment Name="foo">bar</Comment>
        <Media Location="desk"/>
        <Media Location="drawer"/>
    </Media>
</Layout>
```

3.10.2.3 ResourceRef Elements in the AncestorPool/Ancestor Element

ResourceRef elements MAY also occur in the ***AmountPool***/***PartAmount*** element of a **JDF** node. Resources that are referenced SHALL reside in a ***ResourcePool***. The restrictions on locations of resource elements described in ▸ Section 3.8.6 Position of Resources within JDF Nodes that apply to ***ResourceLink*** elements similarly apply to ***ResourceRef*** elements.

3.10.2.4 Status of a Resource that Contains an rRef Reference

The @*Status* of a resource that contains an @*rRef* attribute is defined by the lowest @*Status* of all recursively referenced resources. The ordering is defined in ▸ Table 3.10 Abstract Resource Element:

Thus, if any referenced resource has a @*Status* of "Incomplete", the complete resource has a calculated @*Status* of "Incomplete", even though its own @*Status* attribute might be "Unavailable", "Draft", "Available", etc.

3.10.2.5 Alignment of ResourceLink and ResourceRef

New in JDF 1.1A

ResourceRef elements SHALL NOT contain any of the attributes and elements that are specified in the *ResourceLink* as defined in ▸ Section 3.9 ResourceLinkPool and ResourceLink. The value of these properties is implied from the value of the properties for the appropriate part in the *AmountPool* of the *ResourceLink*.

Example 3.10: MediaLink and MediaRef

The following example illustrates the alignment of a *MediaLink* and *MediaRef* in a **DigitalPrinting** node.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n20020626134204"
       Status="Waiting" JobPartID="ID345" Type="DigitalPrinting" Version="1.4">
    <ResourcePool>
       <!-- Media is partitioned so that it can be referenced
          from the AmountPool
       -->
       <Media Class="Consumable" ID="r0006" PartIDKeys="RunIndex"
              Status="Available">
          <Media RunIndex="0 -1"/>
          <Media RunIndex="1 ~ -2"/>
       </Media>
       <DigitalPrintingParams Class="Parameter" ID="r0007" PartIDKeys="RunIndex"
              Status="Available">
          <DigitalPrintingParams RunIndex="0 -1">
             <!-- PartAmount with <Part RunIndex="0 -1"/> contains
                the partition details for this MediaRef -->
             <MediaRef rRef="r0006">
                <Part RunIndex="0 -1"/>
             </MediaRef>
          </DigitalPrintingParams>
          <DigitalPrintingParams RunIndex="1 ~ -2">
             <!-- PartAmount with <Part RunIndex="1 ~ -2"/>
                contains the partition details for this MediaRef
             -->
             <MediaRef rRef="r0006">
                <Part RunIndex="1 ~ -2"/>
             </MediaRef>
          </DigitalPrintingParams>
       </DigitalPrintingParams>
       <RunList Class="Parameter" ID="r0008" Status="Available" />
       <Component Class="Quantity" ID="c0008" Status="Unavailable"
              ComponentType="Sheet" />
    </ResourcePool>
    <ResourceLinkPool>
       <MediaLink Usage="Input" rRef="r0006">
          <!-- the AmountPool contains the ResourceLink partition details -->
          <AmountPool>
             <PartAmount Orientation="Flip180">
                <Part RunIndex="0 -1"/>
             </PartAmount>
             <PartAmount Orientation="Rotate0">
                <Part RunIndex="1 ~ -2"/>
             </PartAmount>
          </AmountPool>
       </MediaLink>
       <DigitalPrintingParamsLink Usage="Input" rRef="r0007"/>
       <RunListLink Usage="Input" rRef="r0008"/>
       <ComponentLink Usage="Output" rRef="c0008"/>
    </ResourceLinkPool>
</JDF>
```

3.10.3 Set of Resources and Partitioned Subsets Thereof

In many cases, a set of similar resources—such as separation films, plates or *RunList* resources—is produced by one process and consumed by another. When this occurs, it is convenient to define one resource element that describes the complete set and allows individual subsets to be referenced. This mechanism also removes process ambiguity if multiple input *ResourceLink* elements and multiple output *ResourceLink* elements exist that are to be unambiguously correlated.

In other cases, there can be a need to change some attribute of a *Parameter Resource* for some subset of the processing to be done by a device. For instance, when printing a document using **DigitalPrinting**, it would be a common application to change the dimensions of the media to be selected based on the actual media box changes in a PDF file.

Resource elements and ***ResourceLink*** elements have OPTIONAL attributes that enable an agent to specify an explicit part of a structured resource. There are two ways to reference a subset of a resource. The first is by quantity (i.e., by specifying an amount in a ***ResourceLink*** that is less than the resource's amount). The second is to select certain parts of a partitioned resource by supplying a filtering ***Part*** element in the ***ResourceLink***.

3.10.4 Resource Amount

Yet another flexible feature of resources is that they can be only partially consumed. For example, in a scenario in which various versions of a product share identical parts—such as versioned books that all have the same cover—each version will only use as many copies of the cover as it needs to fulfill its job requirement, even though all of the covers can be printed in one step for all versions. This feature is specified in the *@Amount* attribute of the ***ResourceLink*** elements and allows multiple **JDF** nodes to share resources. It allows both the sharing of output resources (when a binding process consumes identical sheets from multiple press lines) and the sharing of input resources (when the covers for multiple jobs are identical and are all printed in one press run).

The *@Amount* attribute of a ***PhysicalResource*** element contains the actual amount of a given resource. It is adjusted by the production or consumption amount of every process that is executed and refers to that amount in the corresponding ***ResourceLink*** element. Thus the value of the *@Amount* attribute of a resource that is consumed as an input SHOULD be reduced by the amount that is consumed. It is up to the agent that writes a **JDF** job to ensure that the *@Amount* attributes of resources and the ***ResourceLink*** elements that reference them are consistent. The units used in the *@Amount* attribute of a ***ResourceLink*** element is defined by the unit of the resource element to which the link refers. The definition of *@Amount* for partitioned resources is explained in detail in ‣ Section 3.10.5 Description of Partitioned Resources.

Note that for resources which are the output of processes, the *@Amount* attribute on the ***ResourceLink*** determines the quantity of the resource to be produced. For example, in a **DigitalPrinting** process that included a ***RunList*** as its input with 16 pages to be printed and a ***ComponentLink*** to its output, the *@Amount* and *@AmountProduced* attributes would indicate the number of copies of those 16 pages that the process would produce.

NodeInfoLink/*@Amount* and NodeInfoLink/***AmountPool***/***PartAmount***/*@Amount* describe the amount to be produced in general. This amount describes the number of products to be produced. For instance, on a conventional sheet-fed offset press, this would be press runs, and on a saddle stitcher it would be finished brochures.

3.10.4.1 Evaluating and Updating Amount-Related Attributes in a Device

ResourceLink/*@Amount* specifies the planned amount whereas ***ResourceLink***/*@ActualAmount* specifies the actual production amount. When a device executes a **JDF** node that consumes and produces ***PhysicalResource***s with an amount, it SHALL calculate the needed production amount in the following order: Production Amount(Output)=

1. ***NodeInfoLink***/***AmountPool***/***PartAmount***/*@Amount* -
   NodeInfoLink/***AmountPool***/***PartAmount***/*@ActualAmount*
2. ***NodeInfoLink***/*@Amount* -
   NodeInfoLink/*@ActualAmount*
3. ***ComponentLink***("Output")/***AmountPool***/***PartAmount***/*@Amount* –
   ComponentLink("Output")/***AmountPool***/***PartAmount***/*@ActualAmount*
4. ***ComponentLink***("Output")/*@Amount* –
   ComponentLink("Output")/*@ActualAmount*
5. ***Component***("Output")/*@Amount* –
   ComponentLink("Output")/*@ActualAmount*
6. ***ResourceLink***("Input")/***AmountPool***/***PartAmount***/*@Amount* –
   ResourceLink("Input")/***AmountPool***/***PartAmount***/*@ActualAmount*
7. ***ResourceLink***("Input")/*@Amount* –
   ResourceLink("Input")/*@ActualAmount*
8. ***PhysicalResource***("Input")/*@Amount* –
   ResourceLink("Input")/*@ActualAmount*
9. Implied amount from consuming the complete input resource.

It is strongly RECOMMENDED for MIS systems to explicitly specify the desired production amount of a process by specifying ***ComponentLink***("Output")/*@Amount* or ***ComponentLink***("Output")/***AmountPool***/***PartAmount***/*@Amount* in case of partitioned resources. The device SHOULD increment ***ResourceLink***/*@ActualAmount* or ***ResourceLink***/***AmountPool***/***PartAmount***/*@ActualAmount* by the amount of actual consumption and production. An MIS system that receives a completed process from a device SHALL update ***Resource***/*@Amount* by summing over all ***ResourceLink*** elements that are linked from leaf nodes:

ComponentLink("Output")/***AmountPool***/***PartAmount***/*@Amount*

– ***ComponentLink***("Output")/***AmountPool***/***PartAmount***/*@ActualAmount*

or

ComponentLink("Output")/*@Amount* – ***ComponentLink***(Output)/*@ActualAmount*

and subtracting all links that are linked from leaf nodes:

ComponentLink("Input")/***AmountPool***/***PartAmount***/@*Amount*

– ***ComponentLink***("Input")/***AmountPool***/***PartAmount***/@*ActualAmount*

or

ComponentLink("Output")/@*Amount* – ***ComponentLink***("Input")/@*ActualAmount*

ComponentLink elements from intermediate nodes ("ProcessGroup" or "Product") SHALL be ignored when summing, since they redundantly link to the same resources without specifying and additional production amount.

3.10.4.2 Specifying Amount for a Partially-Completed Process

New in JDF 1.2

A process can be interrupted before the requested amount of output has been produced. When the job is resent from the controller to the device, it SHALL produce only the remaining @*Amount*. The following figure shows the various processes, resources and ***ResourceLink*** elements and their corresponding entries in ▸ Table 3.22 Example of actual amount and amount handling which summarizes the values of the @*Amount*, @*AmountProduced* and @*AmountRequired* attributes in the ***Component***, the @*Amount* and @*ActualAmount* of ***ComponentLink*** in various steps of the process. All planned amounts are multiples of 1000 whereas all actual amounts are randomly adjusted for waste and production overrun or underrun:

Figure 3-10: Amount handling

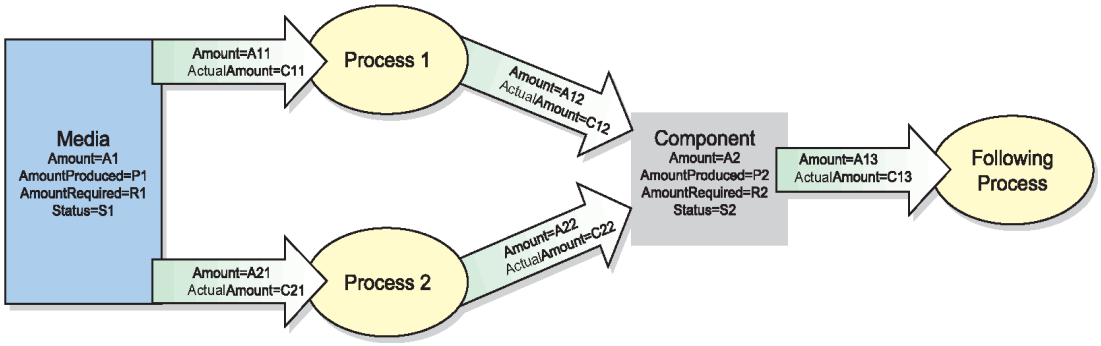


Table 3.22: Example of actual amount and amount handling (Sheet 1 of 3)

PROCESS STEP	A1 P1 R1 S1	A11 C11 A21 C21	A12 C12 A22 C22	A2 P2 R2 S2	A13 C13
Original **JDF**, no processing has commenced. A large Amount of Media (500000) is available. Plan 10% waste. The following Processes are not yet setup.	500000 — 110000 Available	110000 0 — —	100000 0 — —	0 0 — Unavailable	— —
Break after producing exactly 30,000 good copies. Actual waste = 2957	467043 — 110000 Available	110000 32957 — —	100000 30000 — —	30000 30000 — Available	— —
Break after producing exactly an additional 40,000 copies Accumulated actual waste = 6545	423455 — 110000 Available	110000 76545 — —	100000 70000 — —	70000 70000 — Available	— —

Table 3.22: Example of actual amount and amount handling (Sheet 2 of 3)

PROCESS STEP	A1 P1 R1 S1	A11 C11 A21 C21	A12 C12 A22 C22	A2 P2 R2 S2	A13 C13
Completed	390677	110000	100000	101234	—
Overrun = 1234	—	109323	101234	101234	—
Accumulated actual waste = 9323	110000			—	
	Available	—	—	Available	
		—	—		
Consumption of the output by a subsequent process					
A following Process consumes 50,010 copies	390677	110000	100000	51224	50000
	—	109323	101234	101234	50010
	110000			50000	
	Available	—	—	Available	
		—	—		
Additional Copy Request					
A total of 120,000 copies are requested	390677	132000	120000	51224	50000
	—	109323	101234	101234	50010
	110000			50000	
	Available	—	—	Available	
		—	—		
The 20,000 copies are produced(- underrun)	367877	132000	120000	69989	50000
Accumulated actual waste = 12123	—	132123	119999	119999	50010
	132000			50000	
	Available	—	—	Available	
		—	—		
Parallel Production by a second Device					
30,000 additional copies of the same Resource are requested from a different Device.	367877	132000	120000	69989	50000
20% waste is assumed	—	132123	119999	119999	50010
	168000			50000	
	Available	36000	30000	Available	
		0	0		
The 30,000 copies are produced	331856	132000	120000	100089	50000
	—	132123	119999	150099	50010
	168000			50000	
	Available	36000	30000	Available	
		36021	30100		
Consumption by the following process					
The Consuming Node is set up to consume all available Components	331856	132000	120000	100089	150000
	—	132123	119999	150099	50010
	168000			50000	
	Available	36000	30000	Available	
		36021	30100		

Table 3.22: Example of actual amount and amount handling (Sheet 3 of 3)

PROCESS STEP	A1 P1 R1 S1	A11 C11 A21 C21	A12 C12 A22 C22	A2 P2 R2 S2	A13 C13
All intermediate copies are consumed	331856 — 168000 Available	132000 132123 36000 36021	120000 119999 30000 30100	0 150099 150000 Unavailable	150000 150099

3.10.5 Description of Partitioned Resources

Printing workflows contain a number of processes that are repeated over a potentially large number of individual files, sheets, surfaces or separations. In order to define a partitioned resource in a concise manner without having to create a large number of individual nodes and resources, a set of resources might be partitioned by factoring them by one or more attributes. The common elements and defaults are placed in the parent element while partition-specific attributes and overrides are placed in the child elements. This saves space. Also, by providing a single parent ID for the resources, it allows easy access to the entire resource or iteration over each part.

To reference part of a resource, a ***ResourceLink*** references the parent resource and supplies a ***Part*** element that contains an actual value for a partition. The result is all the child elements with matching partition values, including common values and defaults from the parent resource. If @*PartUsage* = "Implicit", the parent attributes are returned if there is no matching partition.

A partitioned resource MAY contain nested elements, each with the same name as the partitioned resource root. The part-independent resource elements and attributes are located in the root of the resource, while the partition-dependent elements are located in the nested elements. Thus one individual part is defined by the convolution of the partition-independent elements and attributes with the elements and attributes contained in the appropriate nested elements. The attributes of nested part elements are overwritten by the equivalent attributes in descendant elements.

Some processes will enumerate a resource in XML order and use its partition key values and actually set the values of those partition keys during its processing. Other processes will treat the resource as a random access resource and look up leaf nodes based on the current settings of @*PartIDKeys* values. For example, the ***RunList*** resource can be used by the **Imposition** process to define key values (such as the @*Run* partition key during consumption of the ***RunList***, and the ***Layout*** resource uses partitioning to define a set of templates chosen based on the current content from the ***RunList*** being processed.

3.10.5.1 Subelements in Partitioned Resources

Subelements of a partitioned resource are inherited by a descendant element if and only if no equivalent subelements exist in the descendant element. Subelements are completely replaced by those in descendant elements even if cardinality of the subelement allows multiple occurrences.

Example 3.11: Inheritance for Subelements of a Partitioned Resource

For example, the following ***SeparationSpec*** is two color duo-tone (only "Black" and SpotGreen) in the part with @*PageNumber* = "1". For additional examples and restrictions, see also ▸ Section 8.84.16.1.2 Position of PlacedObject Elements in Layout which contains ▸ Example 496 Invalid MarkObject and ▸ Example 496 MarkObject.

```
<LayoutElement Class="Parameter" ID="ID1" PartIDKeys="PageNumber"
       Status="Available">
    <SeparationSpec Name="Cyan"/>
    <SeparationSpec Name="Magenta"/>
    <SeparationSpec Name="Yellow"/>
    <SeparationSpec Name="Black"/>
    <FileSpec/>
    <LayoutElement PageNumber="0"/>
    <LayoutElement PageNumber="1">
       <!--These two SeparationSpec Elements completely replace the
             CMYK in the root
       -->
       <SeparationSpec Name="Black"/>
       <SeparationSpec Name="SpotGreen"/>
    </LayoutElement>
</LayoutElement>
```

3.10.5.2 Amount in Partitioned Resources

New in JDF 1.2

The @*Amount* attribute of a partitioned resource is treated formally exactly in the same manner as any other attribute. This implies that the amount specified refers to the amount defined by one leaf and not to the amount defined by the sum of leaves in a branch. The @*Amount* attribute defined in the example below is, therefore, two, even though 24 physical plates are described.

Example 3.12: Partitioned ExposedMedia

The following example defines two sets of 12 plates for two sheets with three surfaces. Each has a common brand attribute called "Gooey". Each individual separation has its own @*ProductID*. Furthermore, the *Status* attribute varies from part to part. For example, if a yellow plate breaks, only it will need to be remade and, therefore, set to "Unavailable"; the others, meanwhile, can remain "Available".

```
<ExposedMedia Amount="2" Brand="Gooey" Class="Handling" ID="L1"
       PartIDKeys="SheetName Side Separation" Status="Available">
    <Media Dimension="500 600" MediaType="Plate"/>
    <ExposedMedia SheetName="S1">
        <ExposedMedia Side="Front">
            <ExposedMedia ProductID="S1FCPlateJ42" Separation="Cyan"/>
            <ExposedMedia ProductID="S1FMPlateJ42" Separation="Magenta"/>
            <ExposedMedia ProductID="S1FYPlateJ42" Separation="Yellow"
                  Status="Unavailable"/>
            <ExposedMedia ProductID="S1FKPlateJ42" Separation="Black"/>
        </ExposedMedia>
        <ExposedMedia Side="Back">
            <ExposedMedia ProductID="S1BCPlateJ42" Separation="Cyan"/>
            <ExposedMedia ProductID="S1BMPlateJ42" Separation="Magenta"/>
            <ExposedMedia ProductID="S1BYPlateJ42" Separation="Yellow"/>
            <ExposedMedia ProductID="S1BKPlateJ42" Separation="Black"/>
        </ExposedMedia>
    </ExposedMedia>
    <ExposedMedia SheetName="S2">
        <ExposedMedia Side="Front">
            <ExposedMedia ProductID="S2FCPlateJ42" Separation="Cyan"/>
            <ExposedMedia ProductID="S2FMPlateJ42" Separation="Magenta"/>
            <ExposedMedia ProductID="S2FYPlateJ42" Separation="Yellow"/>
            <ExposedMedia ProductID="S2FKPlateJ42" Separation="Black"/>
        </ExposedMedia>
    </ExposedMedia>
</ExposedMedia>
```

3.10.5.3 Relating PartIDKeys and Partitions

New in JDF 1.2

The @*PartIDKeys* attribute (see ▸ Section 3.10.6 PartIDKeys Attribute and Partition Keys) describes the partition keys that occur in a partitioned resource. The sequence and number of keys is restricted in order and cardinality to ensure interoperability. The first entry in the @*PartIDKeys* list defines the partition closest to the root, the next entry defines the next intermediate partition node and so forth until the last entry, which defines the partition leaves. Each partition key SHALL occur exactly once in the @*PartIDKeys* list. Note that some of the restrictions specified in this section were assumed to be in place in versions before **JDF** 1.2 but were not explicitly stated in the specification.

The value of each partition key SHALL be unique in the scope of a single resource. Thus two resource siblings SHALL NOT contain a partition key with the same name and value.

3.10.5.3.1 Incomplete Partitions

New in JDF 1.2

Partitioned resources MAY be partitioned by a restricted subset of keys in the @*PartIDKeys* list. Keys from the back of the list MAY be omitted in individual partitions. If a key is omitted, all following keys SHALL also be omitted.

Example 3.13: Legal Incomplete Partition

The following example demonstrates a legal incomplete partition. It is incomplete because the ***Preview*** that is partitioned by @*PreviewType* = "ThumbNail" is not also partitioned by @*Separation*. It is legal because the omitted key @*Separation* is at the end of the @*PartIDKeys* list:

```
<Preview Class="Parameter" ID="P1" PartIDKeys="PreviewType Separation"
       URL="File:///aaa.pdf" Status="Available">
    <Preview PreviewType="Separation">
       <Preview Separation="Cyan"/>
       <Preview Separation="Magenta"/>
    </Preview>
    <Preview PreviewType="ThumbNail"/>
</Preview>
```

Example 3.14: Illegal Incomplete Partition

The following example demonstrates an illegal incomplete partition since the omitted keys are not at the end of the @*PartIDKeys* list:

```
<Preview Class="Parameter" ID="P2" PartIDKeys="PreviewType Separation"
       Status="Available">
    <Preview Separation="Cyan"/>
    <Preview Separation="Magenta"/>
</Preview>
```

3.10.5.3.2 Number of Partition Keys per Partitioned Leaf or Node

New in JDF 1.2

Exactly one partition key SHALL be specified per leaf or node, excluding the root node.
Note: This allows XPath-type searches on partitioned leaves.

Example 3.15: Legal Complete Partition

The following example demonstrates a legal partition:

```
<Preview Class="Parameter" ID="P3" PartIDKeys="PreviewType Separation"
       URL="File:///aaa.pdf" Status="Available">
    <Preview PreviewType="Separation">
       <Preview Separation="Cyan"/>
    </Preview>
</Preview>
```

Example 3.16: Illegal Partition

The following example demonstrates an illegal partition since more than one partition key is specified in the leaf, namely, @*PreviewType* and @*Separation*:

```
<Preview Class="Parameter" ID="P4" PartIDKeys="PreviewType Separation"
       URL="File:///aaa.pdf" Status="Available">
    <Preview PreviewType="Separation" Separation="Cyan"/>
</Preview>
```

3.10.5.3.3 Degenerate Partitions

New in JDF 1.2

A partitioned resource SHALL NOT contain partition keys in the root. Mapping partitioned parameters to non-Partitioned resources is achieved by partitioning the resource with exactly one leaf.

Example 3.17: Degenerate Partition

The following example specifies that only "c1" be folded:

```
<Component Class="Quantity" ID="c1" PartIDKeys="SheetName" Status="Available"
       ComponentType="Sheet">
    <Component SheetName="Sheet 1"/>
</Component>
<Component Class="Quantity" ID="c2" PartIDKeys="SheetName" Status="Available"
       ComponentType="Sheet">
    <Component SheetName="Sheet 2"/>
</Component>
<FoldingParams Class="Parameter" ID="fold" NoOp="true" PartIDKeys="SheetName"
       Status="Available">
    <FoldingParams NoOp="false" SheetName="Sheet 1"/>
</FoldingParams>
```

Example 3.18: Invalid Degenerate Partition

The ***Component*** elements in the following example are NOT valid:

```
<Component Class="Quantity" ID="c12" PartIDKeys="SheetName" SheetName="Sheet 1"
       Status="Available" ComponentType="Sheet"/>
<Component Class="Quantity" ID="c22" PartIDKeys="SheetName" SheetName="Sheet 2"
       Status="Available" ComponentType="Sheet"/>
<FoldingParams Class="Parameter" ID="fold2" NoOp="true" PartIDKeys="SheetName"
       Status="Available">
    <FoldingParams NoOp="false" />
</FoldingParams>
```

3.10.5.4 Partitioning of Resource Subelements

New in JDF 1.2

Only resources can be partitioned. If a resource contains subelements, the subelements SHALL NOT be partitioned. Subelements SHALL always be specified completely in that part where they occur. The content of subelements is not convoluted with the content of subelements in parts closer to the root. Five examples are provided below. The first and the fourth example are valid, the second, third and fifth are invalid.

Example 3.19: Partitioned ExposedMedia with Media Subelements

In the first example, the ***ExposedMedia*** resource is partitioned.

```
<ExposedMedia Class="Handling" ID="L1" PartIDKeys="Separation"
       Status="Available">
    <Media Brand="foo" MediaType="Film"/>
    <ExposedMedia Separation="Cyan"/>
    <ExposedMedia Separation="Magenta">
        <Media Brand="bar" MediaType="Film"/>
    </ExposedMedia>
</ExposedMedia>
```

Example 3.20: Partitioned ExposedMedia with Incomplete Media Subelements

In this incomplete example, the ***Media*** in the leaves is not complete because it does not contain the *@MediaType* attribute. *@MediaType* is *not* inherited from the ***Media*** element in the root resource because in this case ***Media*** is not the partitioned resource.

```
<ExposedMedia Class="Handling" ID="L21" PartIDKeys="Separation"
       Status="Available">
    <Media MediaType="Film"/>
    <ExposedMedia Separation="Cyan">
        <Media Brand="foo"/>
    </ExposedMedia>
    <ExposedMedia Separation="Magenta">
        <Media Brand="bar" Class="Consumable"/>
    </ExposedMedia>
</ExposedMedia>
```

Example 3.21: Partitioned ExposedMedia with Invalid Partitioning of Media Subelements

In this invalid example, ***Media*** is a subelement that SHALL NOT be partitioned.

```
<ExposedMedia Class="Handling" ID="L31" PartIDKeys="Separation" Status="Available">
    <Media MediaType="Film">
        <Media Brand="foo" Separation="Cyan"/>
        <Media Brand="bar" Separation="Magenta"/>
    </Media>
</ExposedMedia>
```

Example 3.22: Partitioned ExposedMedia with MediaRef Subelements

Partitioning MAY be combined with inter-Resource links (i.e., ***ResourceRef*** elements). In the following valid example, each ***MediaRef*** is equivalent to an in-lined leaf with the explicit ***Part*** elements to define the partition (i.e., it is equivalent to the valid ▸ Example 84 Partitioned ExposedMedia with Media Subelements).

```
<Media Class="Consumable" ID="MediaID" MediaType="Film" PartIDKeys="Separation"
       Status="Available">
    <Media Brand="foo" Separation="Cyan"/>
    <Media Brand="bar" Separation="Magenta"/>
</Media>
<ExposedMedia Class="Handling" ID="L41" PartIDKeys="Separation"
       Status="Available">
    <ExposedMedia Separation="Cyan">
        <!--equivalent to <Media MediaType="Film" Brand="foo"/> -->
        <MediaRef rRef="MediaID">
            <Part Separation="Cyan"/>
        </MediaRef>
    </ExposedMedia>
    <ExposedMedia Separation="Magenta">
        <!--equivalent to <Media MediaType="Film" Brand="bar"/> -->
        <MediaRef rRef="MediaID">
            <Part Separation="Magenta"/>
        </MediaRef>
    </ExposedMedia>
</ExposedMedia>
```

Example 3.23: Partitioned ExposedMedia with Invalid MediaRef Subelements

In this invalid example, ***MediaRef*** does not reference the leaves of media but, rather, to the root of ***Media***. It is equivalent to the invalid ▸ Example 85 Partitioned ExposedMedia with Invalid Partitioning of Media Subelements.

```
<Media Class="Consumable" ID="MediaID2" MediaType="Film"
    PartIDKeys="Separation" Status="Available">
    <Media Brand="foo" Separation="Cyan"/>
    <Media Brand="bar" Separation="Magenta"/>
</Media>
<ExposedMedia Class="Handling" ID="L51" PartIDKeys="Separation"
       Status="Available">
    <MediaRef rRef="MediaID2"/>
</ExposedMedia>
```

3.10.5.5 Logical Partitions and the Identical Element

New in JDF 1.3

Partitioning is a mechanism for describing a complete set of similar resources, but always leads to a tree structure of resources. Sometimes it is necessary to describe a set of resources that are not a tree, but where some partitions of the set are 'identical' to another partition. A set of ***ExposedMedia*** resources where the same plate for the separation 'CompanySpot' is reused for all sheets is a practical example.

3.10.5.5.1 Identical

Any partitioned resource MAY contain an ***Identical*** subelement. The resource partition containing the ***Identical*** element is called the logical partition or slave partition. Linking a logical partition using a ***ResourceLink*** or referencing a logical partition using a ***ResourceRef*** is semantically the same as linking/referencing the master partition.

All attributes except for the attributes specified in @*PartIDKeys* and all subelements of the resource (see ▸ Section Table 3.10: Abstract Resource Element) specified or inherited in the logical partition SHALL be ignored and replaced by the attributes and subelements of the master partition.

Table 3.23: Identical Element

NAME	DATA TYPE	DESCRIPTION
Part	element	Identifies the physical partition which will be used instead of the logical partition. The logical partition is defined by the resource partition containing the ***Identical*** element.

Example 3.24: Partitioning with the Identical Element

In the following example the back side of sheet S2 is identical to the back side of sheet S1:

```
<ExposedMedia Class="Handling" ID="L1" PartIDKeys="SheetName Side Separation"
       Status="Available">
    <Media Class="Consumable" MediaType="Film"/>
    <ExposedMedia SheetName="S1">
       <ExposedMedia Side="Front">
          <ExposedMedia ProductID="1" Separation="Cyan"/>
          <ExposedMedia ProductID="2" Separation="Magenta"/>
          <ExposedMedia ProductID="3" Separation="Yellow"/>
          <ExposedMedia ProductID="4" Separation="Black"/>
       </ExposedMedia>
       <!-- Master partition that is referenced by an Identical Element  -->
       <ExposedMedia Side="Back">
          <ExposedMedia ProductID="5" Separation="Cyan"/>
          <ExposedMedia ProductID="6" Separation="Magenta"/>
          <ExposedMedia ProductID="7" Separation="Yellow"/>
          <ExposedMedia ProductID="8" Separation="Black"/>
       </ExposedMedia>
    </ExposedMedia>
    <ExposedMedia SheetName="S2">
       <ExposedMedia Side="Front">
          <ExposedMedia ProductID="9" Separation="Cyan"/>
          <ExposedMedia ProductID="10" Separation="Magenta"/>
          <ExposedMedia ProductID="11" Separation="Yellow"/>
          <ExposedMedia ProductID="12" Separation="Black"/>
       </ExposedMedia>
       <!-- Logical partition with an Identical Element  -->
       <ExposedMedia Side="Back">
          <Identical>
             <Part SheetName="S1" Side="Back"/>
          </Identical>
       </ExposedMedia>
    </ExposedMedia>
</ExposedMedia>
```

3.10.5.5.2 Restrictions when using Identical Elements

The ***Identical*** element SHALL contain exactly one ***Part*** subelement, which identifies the physical or master partition that is identical to the logical partition.

The logical partition SHALL have no other subelements than the ***Identical*** element and no additional attributes other than those specified by @*PartIDKeys*.

The master partition identified by ***Identical***/***Part*** SHALL be either a partition leaf or at the same partition level of the logical partition. Such a master partition SHALL NOT contain an ***Identical*** element. In this way, the logical partition obeys the rules described in ▸ Section 3.10.5.3 Relating PartIDKeys and Partitions.

Example 3.25: ResourceLink with Part Element

The *ExposedMedia* example above is valid, because both the logical and physical partition level equals the @*Side* partition level. The following *ResourceLink* illustrates a valid partition sequence:

```
<ExposedMediaLink Usage="Input" rRef="L1">
    <Part SheetName="S2" Side="Back" Separation="Black"/>
</ExposedMediaLink>
```

Example 3.26: Partitioning with an Invalid Identical Element

This example illustrates an INVALID logical partition, because logical and physical partition level are not equal and the physical partition level is not a leaf.

```
<ExposedMedia Class="Handling" ID="L2" PartIDKeys="SheetName Side Separation"
       Status="Available">
    <ExposedMedia SheetName="S1">
        <ExposedMedia Side="Front">
            <ExposedMedia ProductID="1" Separation="Cyan"/>
            <ExposedMedia ProductID="2" Separation="Magenta"/>
            <ExposedMedia ProductID="3" Separation="Yellow"/>
            <ExposedMedia ProductID="4" Separation="Black"/>
        </ExposedMedia>
        <ExposedMedia Side="Back">
            <ExposedMedia ProductID="5" Separation="Cyan"/>
            <ExposedMedia ProductID="6" Separation="Magenta"/>
            <ExposedMedia ProductID="7" Separation="Yellow"/>
            <ExposedMedia ProductID="8" Separation="Black"/>
        </ExposedMedia>
    </ExposedMedia>
    <ExposedMedia SheetName="S2">
        <ExposedMedia Side="Front">
            <ExposedMedia ProductID="9" Separation="Cyan">
                <!--This Identical is invalid because it references from a Separation
                    partition to a Surface partition -->
                <Identical>
                    <Part SheetName="S1" Side="Back"/>
                </Identical>
            </ExposedMedia>
        </ExposedMedia>
    </ExposedMedia>
</ExposedMedia>
```

3.10.6 PartIDKeys Attribute and Partition Keys

New in JDF 1.2

3.10.6.1 Partitionable Resource

In addition to the usual resource attributes and elements, the partitionable resource element has partition-specific attributes and elements in its root. Specifying @*PartIDKeys* in the root defines a partitioned resource. Throughout this document, the term "Partition Key" (depending on the context) refers to either

- an enumeration value of the @*PartIDKeys* attribute (e.g., "Side").

```
<ExposedMedia PartIDKeys = "Side"...>
```

- an attribute that with two specialized functions:
  - can identify a partition (e.g., @*Side*).

```
<ExposedMedia ID="XM"...>
<ExposedMedia Side="Front"...>
</ExposedMedia>
```

  - can reference a partition from within a *Part* Element (e.g., @*Side*).

```
<ExposedMediaLink rRef="XM" ...>
<Part Side="Front"/>
</ExposedMediaLink>
```

Further attributes that apply to partitioned resources are listed in the following table.

Table 3.24: Partitionable Resource Element

NAME	DATA TYPE	DESCRIPTION
PartIDKeys ? Modified in JDF 1.6	enumerations	List of attribute names that are used to separate the individual parts. @*PartIDKeys* also defines the sequence from root to leaf in which the @*PartIDKeys* SHALL occur in the partitioned resource. Each entry in the @*PartIDKeys* list SHALL occur only once. @*PartIDKeys* SHALL NOT be specified below the root of a partitioned resource. For details, see ▸ Table 3.26 Part Element. **Modification note:** Before **JDF** 1.4, ***Part***/@*Sorting* and ***Part***/@*SortAmount* were not valid values of @*PartIDKeys*. Now they have been deprecated so all values of @*PartIDKeys* are also elements of ***Part***. **Allowed values are from:** ▸ Table 3.25 PartIDKey Attribute Values
PipePartIDKeys ? New in JDF 1.2	enumerations	Defines the granularity of a dynamic pipe for a partitioned resource. For instance, if a resource were partitioned by sheet, surface and separation (i.e., ***Resource***/@*PartIDKeys* ="SheetName Side Separation"), and if the ***ResourceLink***/@*PipePartIDKeys* = "SheetName Side", then pipe requests would be issued only once per surface. The contents of @*PipePartIDKeys* SHALL be a subset of the @*PartIDKeys* attribute of the resource that is linked by this ***ResourceLink***. **Default value is from:** @*PartIDKeys* (i.e., maximum granularity). @*PipePartIDKeys* SHALL NOT be specified below the root of a partitioned resource. For details on partitioned resources, see ▸ Section 3.10.5 Description of Partitioned Resources. **Allowed values are from:** ▸ Table 3.25 PartIDKey Attribute Values**.**
Identical ?	element	Cross reference to a logical partition. For details on logical partitions and the ***Identical*** element, see ▸ Section 3.10.5.5 Logical Partitions and the Identical Element.
Resource*	element	Nested resource elements that contain the appropriate partition keys as specified in @*PartIDKeys*. These elements SHALL be of the same name and type as the root ***Resource*** element. They represent the individual parts or groups of parts.

3.10.6.1.1 PartIDKey Attribute Values

Modified in JDF 1.6

The following table lists the partion key values.

Table 3.25: PartIDKey Attribute Values

PARTIDKEY VALUES			
BinderySignatureName	DocCopies	Metadata5	RunSet
BinderySignaturePaginationIndex	DocIndex	Metadata6	RunTags
BlockName	DocRunIndex	Metadata7	SectionIndex
BundleItemIndex	DocSheetIndex	Metadata8	Separation
CellIndex	DocTags	Metadata9	SetCopies
Condition	Edition	Option	SetDocIndex
DeliveryUnit0	EditionVersion	PageNumber	SetIndex
DeliveryUnit1	FountainNumber	PageTags	SetRunIndex
DeliveryUnit2	ItemNames	PartVersion	SetSheetIndex
DeliveryUnit3	LayerIDs	PlateLayout	SetTags
DeliveryUnit4	Location	PreflightRule	SheetIndex
DeliveryUnit5	LotID	PreviewType	SheetName
DeliveryUnit6	Metadata0	ProductPart	Side
DeliveryUnit7	Metadata1	RibbonName	SignatureName
DeliveryUnit8	Metadata2	Run	StationName SubRun
DeliveryUnit9	Metadata3	RunIndex	TileID
	Metadata4	RunPage	WebName
		RunPageRange	WebProduct
			WebSetup

3.10.6.2 Part

Partitionable resources are uniquely identified by the attribute values listed in @*PartIDKeys* attributes. The choice of which attributes to use depends on how the agent organizes the job.

The following table lists the content of a ***Part*** element, which contains a set of attributes that have a well described meaning. Each of the attributes, except @*Sorting*, MAY be used in the nested resource elements of partitioned resources as the partition key (see example above).

Part elements match a given partition when all of the attributes of a ***Part*** element match the attributes of the referenced resource. This corresponds to Boolean AND operation. Note that a ***Part*** element MAY specify a subset of the partition keys (e.g., only lower level partition keys) and thus implicitly select multiple partitions leaves or nodes from a partitioned resource (see ▸ Section 3.10.7.4 Implicit, Sparse and Explicit PartUsage in Partitioned Resources). If multiple ***Part*** elements are specified, the result is a Boolean OR of the multiple parts. A ***Part*** element with no attributes explicitly references the root resource.

Some attributes of ***Part*** (@*Separation*, @*SheetName*, @*SignatureName*) have a data type of string. Future versions of this specification may restrict the data type to NMTOKEN. Therefore implementations SHOULD write values as NMTOKEN. Compliant implementations SHALL be capable of reading string values.

Table 3.26: Part Element (Sheet 1 of 7)

NAME	DATA TYPE	DESCRIPTION
BinderySignatureName ? New in JDF 1.2	NMTOKEN	Name of the ***BinderySignature*** used in a ***StrippingParams*** description.
BinderySignaturePaginationIndex ? New in JDF 1.4	IntegerRangeList	@*BinderySignaturePaginationIndex* defines indices of the pages of the pagination sequence of ***StrippingParams***/***StripCellParams*** elements or ***BinderySignature***/***SignatureCell*** elements. Elements are counted by their pagination index. The index is zero based and is local in the ***BinderySignature***, not the pagelist of the job.
BlockName ? New in JDF 1.1	NMTOKEN	Identifies a ***CutBlock*** from a **Cutting** process. The value of this attribute SHALL match the value of the @*BlockName* attribute of a ***CutBlock***.
BundleItemIndex ? New in JDF 1.2	IntegerRangeList	The @*BundleItemIndex* attribute selects a set of ***BundleItem*** elements from a ***Component*** resource.

Table 3.26: Part Element (Sheet 2 of 7)

NAME	DATA TYPE	DESCRIPTION
CellIndex ? New in JDF 1.2	IntegerRangeList	Index of ***SignatureCell*** elements in a ***StrippingParams*** or ***BinderySignature***. ***SignatureCell*** elements are counted starting from lower left. Each row is indexed from left to right before moving up to the next row.
Condition ? New in JDF 1.2	NMTOKEN	The @*Condition* attribute was added to **JDF** 1.2 to allow users of **JDF**-enabled systems to define and track different kinds of waste for improved error reporting and production statistics. **Values include those from:** ▸ Table 3.27 Condition Attribute Values.
DeliveryUnit0 ? New in JDF 1.3	NMTOKEN	Specifies a hierarchical manifest of delivery packages where @*DeliveryUnit0* specifies the most granular bundle. @*DeliveryUnit<N+1>* specifies the next most granular bundle in packing after @*DeliveryUnit<N>*. Bundles can be packaged with varying numbers of products. @*DeliveryUnit<N+1>* SHALL occur before @*DeliveryUnit<N>* in @*PartIDKeys*. Note that *N* is a placeholder for the values 0 through 9.
DeliveryUnit1 ? New in JDF 1.3	NMTOKEN	See @*DeliveryUnit0*.
DeliveryUnit2 ? New in JDF 1.3	NMTOKEN	See @*DeliveryUnit0*.
DeliveryUnit3 ? New in JDF 1.3	NMTOKEN	See @*DeliveryUnit0*.
DeliveryUnit4 ? New in JDF 1.3	NMTOKEN	See @*DeliveryUnit0*.
DeliveryUnit5 ? New in JDF 1.3	NMTOKEN	See @*DeliveryUnit0*.
DeliveryUnit6 ? New in JDF 1.3	NMTOKEN	See @*DeliveryUnit0*.
DeliveryUnit7 ? New in JDF 1.3	NMTOKEN	See @*DeliveryUnit0*.
DeliveryUnit8 ? New in JDF 1.3	NMTOKEN	See @*DeliveryUnit0*.
DeliveryUnit9 ? New in JDF 1.3	NMTOKEN	See @*DeliveryUnit0*.
DocCopies ?	IntegerRangeList	Identifies a set of document copies to which the partition applies.
DocIndex ?	IntegerRangeList	The @*DocIndex* attribute selects a set of logical instance documents from a ***RunList*** resource.
DocRunIndex ?	IntegerRangeList	The @*DocRunIndex* attribute selects a set of logical pages from instance documents of a ***RunList*** resource. For example, @*DocRunIndex* = "0 –1" specifies the first and last page of every copy of every selected instance document (assuming that additional partitioning using @*DocCopies* and/or @*DocIndex* is not also specified). The index always refers to entries of the entire ***RunList*** and SHALL NOT be modified if only a part of the ***RunList*** is spawned. Specifying @*DocRunIndex* does not modify the index of a ***RunList*** entry and therefore does not reposition pages on a ***Layout***.

Table 3.26: Part Element (Sheet 3 of 7)

NAME	DATA TYPE	DESCRIPTION
DocSheetIndex ?	IntegerRangeList	The @*DocSheetIndex* attribute selects a set of logical sheets from individual instance documents. For example @*DocSheetIndex* = "0 –1" specifies the first and last sheet of every selected copy of every instance document (assuming that additional partitioning using @*DocCopies* and/or @*DocIndex* is not also specified). The index always refers to entries of the entire ***RunList*** and SHALL NOT be modified if only a part of the ***RunList*** is spawned. Specifying @*DocSheetIndex* does not modify the index of a ***RunList*** entry and therefore does not reposition pages on a ***Layout***.
DocTags ? New in JDF 1.3 Modified in JDF 1.4	NameRangeList	List of tags of documents in a multi-document ***RunList***. Used to partition resources that are linked from processes that also have a ***RunList*** as input. The partition is selected if the implied value (i.e., from the PDL) of the document in the ***RunList*** matches any of the entries in @*DocTags* Note that being a multi-set ***RunList*** implies being a multi-document ***RunList*** as well. **Modification note:** Starting with **JDF** 1.4, the data type was expanded from NMTOKENS to NameRangeList.
Edition ? New in JDF 1.3	NMTOKEN	An @*Edition* addresses a subset of a published product (e.g., newspaper issue). The content of all copies of one edition is the same. Usually, an edition is published for a specific region and/or publishing time (e.g., Asia/Europe edition or Morning/Evening edition).
EditionVersion ? New in JDF 1.3	NMTOKEN	An edition version is an OPTIONAL subset of a single edition. In order to ship inserts, editions might be subdivided into edition versions.
FountainNumber ?	integer	Zero-based position index of the fountain. Used to partition fountains along the axis of a roller; can be used for web printing.
ItemNames ? New in JDF 1.2	NMTOKENS	List of items to select from a ***Bundle***. **Default behavior:** all ***BundleItem*** elements are processed.
LayerIDs ? New in JDF 1.1	IntegerRangeList	The @*LayerIDs* attribute selects a set layers that are defined by @*LayerID*. **Default behavior:** all layers are processed.
Location ? Modified in JDF 1.3	NMTOKEN	Name of the location (e.g., in MIS). This part key allows to describe distributed resources. Note that this name does not define the location by itself. See ▸ Section 3.10.6.4 Locations of PhysicalResources for details on specifying locations. **Values include those from:** ▸ Table C.10 Input Tray and Output Bin Names**.** **Note:** The specified values are for printer locations.
LotID New in JDF 1.6	NMTOKEN	Identifier of the lot of a resource in a lot controlled environment.
Metadata0 ? New in JDF 1.4	NameRangeList	Metadata extracted from a PDL using ***RunList***/***MetadataMap*** elements. See ▸ Section 9.31 MetadataMap.
Metadata1 ? New in JDF 1.4	NameRangeList	See @*Metadata0*.
Metadata2 ? New in JDF 1.4	NameRangeList	See @*Metadata0*.
Metadata3 ? New in JDF 1.4	NameRangeList	See @*Metadata0*.
Metadata4 ? New in JDF 1.4	NameRangeList	See @*Metadata0*.
Metadata5 ? New in JDF 1.4	NameRangeList	See @*Metadata0*.

Table 3.26: Part Element (Sheet 4 of 7)

NAME	DATA TYPE	DESCRIPTION
Metadata6 ? New in JDF 1.4	NameRangeList	See @*Metadata0*.
Metadata7 ? New in JDF 1.4	NameRangeList	See @*Metadata0*.
Metadata8 ? New in JDF 1.4	NameRangeList	See @*Metadata0*.
Metadata9 ? New in JDF 1.4	NameRangeList	See @*Metadata0*.
Option ? Modified in JDF 1.3	NMTOKEN	Generic option that MAY be semantic free. MAY also be used for options from an RFQ in an intent resource.
PageNumber ?	IntegerRangeList	Page number in a ***Component*** or document (e.g., ***FileSpec*** that is not described as a ***RunList***). References an index in a ***PageList***.
PageTags ? New in JDF 1.3 Modified in JDF 1.4	NameRangeList	List of tags of pages in a multi-page ***RunList***. Used to partition resources that are linked from processes that also have a ***RunList*** as input. The partition is selected if the implied value (i.e., from the PDL) of the page in the ***RunList*** matches any of the entries in @*PageTags*. **Modification note:** Starting with **JDF** 1.4, the data type was expanded from NMTOKENS to NameRangeList.
PartVersion ? Modified in JDF 1.3	NMTOKENS	Version identifier (e.g., the language version of a catalog). **Compatibility note:** The data type of @*PartVersion* was changed from string to NMTOKENS in **JDF** 1.3 in order to accommodate resources that contain elements from multiple versions (e.g., Sheets with two language versions).
PlateLayout ? New in JDF 1.3	NMTOKEN	Identifier of a single plate layout (mainly used for newspaper processes, where multiple plates are needed for one cylinder)
PreflightRule ? New in JDF 1.2 Modified in JDF 1.3	NMTOKEN	Definition of the specific parts of a ***PreflightReportRulePool***/***PRRule*** used in preflight applications.
PreviewType ? New in JDF 1.1 Modified in JDF 1.5	enumeration	Preview specifies the type and usage of a ***Preview***. @*PreviewType* SHALL NOT be specified for resources other than ***Preview*** or ***PreviewGenerationParams***. **Allowed values are:** Animation – animated previews for 3D display. New in JDF 1.5. Identification – ***Preview*** is used as a visual help to identify one or more products, e.g. on a gang form. New in JDF 1.5. SeparatedThumbNail – Very low resolution separated preview. Separation – Separated preview in medium resolution. SeparationRaw – Separated preview in medium resolution with no compensation. New in JDF 1.2 Static3D – static 3D model. New in JDF 1.5 ThumbNail – Very low resolution RGB preview. Viewable – RGB preview in medium resolution.
ProductPart ? New in JDF 1.5	NMTOKEN	References the Product/@*ID* that this ***Part*** applies to.
RibbonName ? Modified in JDF 1.3	NMTOKEN	A string that uniquely identifies each ribbon. Multiple ribbons are created out of one web after dividing in case of web printing.
Run ? Modified in JDF 1.3	NMTOKEN	The @*Run* attribute selects an individual ***RunList*** partition from a ***RunList*** resource.

Table 3.26: Part Element (Sheet 5 of 7)

NAME	DATA TYPE	DESCRIPTION
RunIndex ?	IntegerRangeList	The @*RunIndex* attribute selects a set of logical pages from a ***RunList*** resource in a manner that is independent from the internal structure of the ***RunList***. It contains an array of mixed ranges and individual indices separated by whitespace. Each range consists of two indices connected with a tilde (~). For example, @*RunIndex* = "2 ~ 5 8 10 22 ~ -1". Negative numbers reference pages from the back of a file in base-1 counting. In other words, -1 is the last page, -2 the second to last, etc. Thus @*RunIndex* = "0 ~ -1" refers to a complete range of pages, from first to last. The index always refers to entries of the entire ***RunList*** and SHALL NOT be modified if only a part of the ***RunList*** is spawned. Specifying @*RunIndex* does not modify the index of a ***RunList*** entry and therefore does not reposition pages on a ***Layout***.

RunPage ?
New in JDF 1.1
	integer	Zero-based page number. Used when a document/file-based ***RunList*** is broken down into a page based ***RunList***. For instance, a 2-page document ***RunList***:
		`<RunList URL="doc.pdf" (…) />`
		is split into:
		`<RunList PartIDKeys="RunPage" (…)>`
		`<RunList URL="doc_page0.pdf"`
		`RunPage="0" (…) />`
		`<RunList URL="doc_page1.pdf"`
		`RunPage="1" (…) />`
		`</RunList>`

RunPageRange ? New in JDF 1.4	IntegerRangeList	Used when splitting ***RunList*** resources into larger chunks that are not yet based on ***PageList*** indices.
RunSet ? New in JDF 1.3	NMTOKEN	Generic group of elements in a ***RunList***. If partitioning a ***RunList*** by @*RunSet* and @*Run*, then @*RunSet* SHOULD be specified closer to the root.
RunTags ? New in JDF 1.1 Modified in JDF 1.4	NameRangeList	List of names in a named ***RunList***. Used to partition resources that are linked from processes that also have a ***RunList*** as input when the sequence of the ***RunList*** is undefined. The partition is selected if the explicit or implied (e.g., from the PDL) value of @*RunTag* of the ***RunList*** matches any of the entries in @*RunTags*. **Note:** The difference between @*RunTags* and @*PageTags*, @*DocTags* or @*SetTags*. @*PageTags* is used to identify classes of individual pages having differing **JDF** parameterization. Similarly, @*DocTags* is used to identify classes of individual documents and @*SetTags* is used to identify classes of individual sets each having differing **JDF** parameterization. @*RunTags* is used to identify collections of pages, often thought of as a document or a piece of a document, but not limited to that. Also, @*RunTags* MAY be explicitly set for an entire ***RunList*** by use of the @*RunTag* attribute. The @*SetTags*, @*DocTags* and @*PageTags* partition keys are always set implicitly and always refer to the granularity within a @*RunList* implied by their names. **Modification note:** Starting with **JDF** 1.4, the data type was expanded from NMTOKENS to NameRangeList.
SectionIndex ? New in JDF 1.2	IntegerRangeList	List of sections in a ***StrippingParams***.

Table 3.26: Part Element (Sheet 6 of 7)

NAME	DATA TYPE	DESCRIPTION
Separation ?	string	Identifies the separation name. **Values include:** Composite – Non-separated resource. Separated – The resource is separated, but the separation definition is handled internally by the resource, such as a PDF file that contains **SeparationInfo** dictionaries. Cyan – Process color. Magenta – Process color. Yellow – Process color. Black – Process color. Red – Additional process color. Green – Additional process color. Blue – Additional process color. Orange – Additional process color. Spot – Generic spot color. Used when the exact nature of the spot color is unknown. Varnish – Varnish. **Note:** Other values include any separation name defined in the @*Name* attribute of a ***Color*** element in the ***ColorPool***. **Note:** When @*Separation* is applied to a ***ColorantControlLink***, it defines an implicit partition that selects a subset of separations for the process that is described by the ***ColorantControl***. For details, see ▸ Section 8.21 ColorantControl.
SetCopies ? New in JDF 1.5	IntegerRangeList	Identifies a collection of set copies to which the partition applies.
SetDocIndex ? New in JDF 1.2	IntegerRangeList	The @*SetDocIndex* attribute selects a set of logical instance documents from instance document sets of a ***RunList*** resource. For example, @*SetDocIndex* = "0 -1" specifies the first and last document of every copy of every selected instance document set. The index always refers to entries of the entire ***RunList*** and SHALL NOT be modified if only a part of the ***RunList*** is spawned. Specifying @*SetDocIndex* does not modify the index of a ***RunList*** entry and therefore does not reposition pages on a ***Layout***.
SetIndex ? New in JDF 1.1	IntegerRangeList	The @*SetIndex* attribute selects a set of logical instance document sets from a ***RunList*** resource. The index always refers to entries of the entire ***RunList*** and SHALL NOT be modified if only a part of the ***RunList*** is spawned. Specifying @*SetIndex* does not modify the index of a ***RunList*** entry and therefore does not reposition pages on a ***Layout***.
SetRunIndex ? New in JDF 1.2	IntegerRangeList	The @*SetRunIndex* attribute selects a set of logical pages from instance document sets of a ***RunList*** resource. For example, @*SetRunIndex* = "0 -1" specifies the first and last page of every copy of every selected instance document set. The index always refers to entries of the entire ***RunList*** and SHALL NOT be modified if only a part of the ***RunList*** is spawned. Specifying @*SetRunIndex* does not modify the index of a ***RunList*** entry and therefore does not reposition pages on a ***Layout***.
SetSheetIndex ? New in JDF 1.2	IntegerRangeList	The @*SetSheetIndex* attribute selects a set of logical sheets from individual sets of instance documents. For example @*SetSheetIndex* = "0 –1" specifies the first and last sheet of every selected copy of every set. The index always refers to entries of the entire ***RunList*** and SHALL NOT be modified if only a part of the ***RunList*** is spawned. Specifying @*SetSheetIndex* does not modify the index of a ***RunList*** entry and therefore does not reposition pages on a ***Layout***.
SetTags ? New in JDF 1.3 Modified in JDF 1.4	NameRangeList	List of tags of pages in a multi-set ***RunList***. Used to partition resources that are linked from processes that also have a ***RunList*** as input. The partition is selected if the implied value (i.e., from the PDL) of the set in the ***RunList*** matches any of the entries in @*SetTags*. **Modification note:** Starting with **JDF** 1.4, the data type was expanded from NMTOKENS to NameRangeList.

Table 3.26: Part Element (Sheet 7 of 7)

NAME	DATA TYPE	DESCRIPTION
SheetIndex ? Modified in JDF 1.4	IntegerRangeList	The @*SheetIndex* attribute selects a set of logical sheets from a ***RunList*** resource either implicitly or explicitly partitioned by @*SheetIndex*. @*SheetIndex* is only valid when a ***RunList*** is describing sheet/surfaces.
SheetName ?	string	A string that uniquely identifies each sheet.
Side ?	enumeration	Denotes the side of the sheet. If @*Side* is specified, the ***Part*** element refers to one surface of the sheet. If it is not specified, it refers to both sides. In case of web printing, "Front" is a synonym for the upper side and "Back" for the down side of the web. **Allowed values are:** Front Back
SignatureName ?	string	A string that uniquely identifies the signature within the partitioned resource.
Sorting ? Deprecated in JDF 1.4	IntegerRangeList	Mapping from the implied partitioned resource order to a process order. The indices refer to the elements of the complete partitioned resource, not to the index in the selection of parts defined by the ***Part*** element. If not specified the part order is the same as the sorting order. @*Sorting* SHALL NOT be used as a partition key. **Note:** @*Sorting* and @*SortAmount* are semantically different from the other attributes in this table as they define the ordering of parts, whereas the other attributes define the selection of parts. **Deprecation note:** The order of the ***Part*** elements contained in a ***ResourceLink*** is significant: the specified subsets of the resource are selected in the XML order of the ***Part*** elements.
SortAmount ? Deprecated in JDF 1.4	boolean	If a sorted resource has an @*Amount* attribute and @*SortAmount* = "true", each resource SHALL be processed completely. If @*SortAmount* = "false" (the default), each ***Part*** element SHALL be processed the number of times specified in the @*Amount* attribute before starting the next ***Part***. @*SortAmount* SHALL NOT be used as a partition key. **Deprecation note:** See @*Sorting*.
StationName? New in JDF 1.3	string	The name of the 1-up design in a ***DieLayout***.
SubRun? New in JDF 1.3	NMTOKEN	Defines individual sub-runs in a production run. For instance, ***Media*** might vary over the duration of a longer run. The variation might be only stock numbers, but physical characteristics might also vary.
TileID ? Modified in JDF 1.3	XYPair	XYPair of integer values that identifies the tile. Tiles are identified by their X and Y indexes. Values are zero-based and expressed in the PS coordinate system. So "0 0" is the lower left tile and "1 0" is the tile next to it on the right. ***Tile*** resources are described in detail in the ▸ Section 8.157 Tile. In **JDF** 1.3 and beyond, @*TileID* SHOULD NOT be used to specify multiple plates per cylinder. Instead the new resource ***CylinderLayout*** SHOULD be used.
WebName ? Modified in JDF 1.3	NMTOKEN	A string that uniquely identifies each web.
WebProduct ? New in JDF 1.3	NMTOKEN	Name of a product that will be produced on a web press. Multiple web products MAY be produced simultaneously on one web press.
WebSetup ? New in JDF 1.3	NMTOKEN	Defines one setup of a web press that MAY produce multiple web products.

3.10.6.2.1 Condition Attribute Values

Table 3.27: Condition Attribute Values

VALUE	DESCRIPTION
Good	All correct components.
Waste	General waste.
Overrun	Excess ***Component*** resource(s) that were produced by running the device after the specified amount has been produces.
xxxGood	Like "Good" above, but where "xxx" can be the name of any **JDF** process (e.g., "FeedingGood", "TrimmingGood", etc.). In the case of a combinded process or process group, the name of the last **JDF** process in the process chain is used.
xxxWaste	Like "Waste" above, but where "xxx" can be the name of any **JDF** process (e.g., "FeedingWaste", "TrimmingWaste", etc.). In the case of a combinded process or process group, the name of the last **JDF** process in the process chain is used.
AuxiliarySheet New in JDF 1.4	This partition identifies ***Media*** that was consumed as specified by ***InsertSheet***/*@SheetType* = "AccountingSheet", "ErrorSheet", "JobSheet" or "SeparatorSheet".
BindingQualityTestFailed	Failed binding quality test. The ***Component*** resource(s) with this *@Condition* belong to the batch of ***Component*** resource(s) that did not pass the test.
BindingQualityTestPassed	Passed binding quality test. The ***Component*** resource(s) with this *@Condition* belong to the batch of ***Component*** resource(s) that passed the test but were not destroyed in the process.
BindingQualityTestWaste	Passed binding quality test. The ***Component*** element(s) with this *@Condition* belong to the batch of ***Component*** element(s) that passed the test but were destroyed in the process.
CaliperWaste	Waste by caliper on gathering / collecting.
DoubleFeedWaste	Waste by double feeds on feeders.
IncorrectComponentWaste	Waste by the attempted use of an incorrect components (e.g., on a feeder).
BadFeedWaste	Waste caused by a bad feed
ObliqueSheetWaste	Waste by oblique sheets on gathering / collecting chains.
PaperJamWaste	Waste by paper or other media jam.
Reusable New in JDF 1.4	Waste to be used for setup in the next process.
WhitePaperWaste	White paper waste.

3.10.6.3 Options in Intent Resources

JDF defines "Option" as a partition key in order to specify multiple options (e.g., for multiple quotes in a non-redundant manner). A ***ResourceLink*** that links to a resource with an "Option" partition but has no ***Part*** element to choose the "Option" defaults to the root resource.

3.10.6.4 Locations of PhysicalResources

Unlike other kinds of resources, ***PhysicalResource***s can be stored at multiple, distributed locations. This is specified by including a ***Location*** element in the resource element. A *@Location* partition key is provided to define multiple locations of one resource. The partition key carries no semantic meaning and does not by itself define the name of a location.

Example 3.27: ExposedMedia with Location Elements

The following example describes a set of plates that are distributed over two locations.

Note: See ▸ Section C.5 Input Tray and Output Bin Names for additional detail on locating ***PhysicalResource*** items.

```
<ResourcePool>
    <ExposedMedia Class="Handling" ID="L1" PartIDKeys="Location"
          Status="Available">
       <ExposedMedia Amount="42" Location="dd1">
          <Location LocID="PP_01234" LocationName="Desk Drawer 1"/>
       </ExposedMedia>
       <ExposedMedia Amount="100" Location="dd2">
          <Location LocID="PP_01235" LocationName="Desk Drawer 2"/>
       </ExposedMedia>
       <Media/>
    </ExposedMedia>
</ResourcePool>
<ResourceLinkPool>
    <ExposedMediaLink Amount="50" Usage="Input" rRef="L1">
       <Part Location="dd2"/>
       <!-- Note that @Location can but need not match
          Location/@LocationName
       -->
       </ExposedMediaLink>
</ResourceLinkPool>
```

Example 3.28: Media with Location Elements

The following example describes two different media in the top and bottom tray of a **LayoutPreparation** process. The media is selected for the cover and inside pages respectively.

```
<Media Class="Consumable" ID="TopMedia" Status="Available">
    <Location LocationName="Top"/>
</Media>
<Media Class="Consumable" ID="BottomMedia" Status="Available">
    <Location LocationName="Bottom"/>
</Media>
<LayoutPreparationParams Class="Parameter" ID="L1" PartIDKeys="RunIndex"
       Sides="TwoSidedFlipY" Status="Available">
    <!-- Partition that defines the first and last page of the document -->
    <LayoutPreparationParams RunIndex="0 1 -2 -1">
       <MediaRef rRef="TopMedia"/>
    </LayoutPreparationParams>
    <!-- Partition that defines the inside pages of the document -->
    <LayoutPreparationParams RunIndex="2 ~ -3">
       <MediaRef rRef="BottomMedia"/>
    </LayoutPreparationParams>
</LayoutPreparationParams>
```

3.10.7 Linking to Resources

Modification note: Starting with **JDF** 1.4, all text up to ▸ Section 3.10.7.3 Handling Amount in a ResourceLink to a Partitioned Resource is new and replaces now-deleted text that was present in **JDF** 1.3

A **JDF** node can specify a reordering or subset of a resource by including one or more ***Part*** elements in the ***ResourceLink*** element that links to that resource. For details of the ***Part*** element, please refer to ▸ Table 3.26 Part Element.

3.10.7.1 Linking to Subsets of Resources

Each ***ResourceLink***/***Part*** element selects a subset of the resource, where the aggregation of each selected subset (in the case of multiple ***ResourceLink***/***Part*** elements) creates a "virtual" resource that will then be used during node processing. This feature is often useful to reproduce part of the job described by a node, as the default interpretation of the ***Part*** elements maintains the context as if the node had been executed without any ***ResourceLink*** partitioning.

Example 3.29: Linking to Subsets of Resources

For instance, if an **Imposition** process outputs multiple sheets, and each sheet has dynamic marks placed on the sheet based on the value of @*SheetIndex*, selecting a single sheet to be processed by **Imposition** would produce that sheet using the original @*SheetIndex* value. This example would generate the imposed sheet #5 followed by the imposed sheet #1,

where all dynamic marks on both sheets retain the context in which @*SheetIndex* would have been defined when processing the full ***RunList*** resource.

```
<ResourcePool>
    <RunList Class="Parameter" ID="SheetSurfacesGeneratedByImposition"
           PartIDKeys="SheetIndex" Status="Available">
        <RunList SheetIndex="1"/>
        <RunList SheetIndex="3"/>
        <RunList SheetIndex="5"/>
    </RunList>
</ResourcePool>
<ResourceLinkPool>
    <RunListLink rRef="SheetSurfacesGeneratedByImposition" Usage="Output">
        <!-- output of imposition -->
        <Part SheetIndex="5"/>
        <Part SheetIndex="1"/>
    </RunListLink>
</ResourceLinkPool>
```

3.10.7.2 Reordering the Processing of Resources

ResourceLink partitioning may also be used to reorder the processing order of content described by a ***RunList***. This is done by using the ***RunList***/@*IgnoreContext* attribute, which specifies which ***Part*** element partition keys' job context should be ignored during processing. For more information and an example of this, see ***RunList***/@*IgnoreContext* in ▸ Section 8.131 RunList and following the ***RunList*** table, see. ▸ Example 673 RunList/MetadataMap.

3.10.7.3 Handling Amount in a ResourceLink to a Partitioned Resource

The @*Amount* specified in a ***ResourceLink*** to a ***PhysicalResource*** specifies the sum of individual resource partitions. Individual amounts are specified in the ***PartAmount*** elements of the ***AmountPool***.

Example 3.30: Amount in an ExposedMediaLink to a Partitioned ExposedMedia

The following example shows the ***ResourceLink*** that refers to ▸ Example 82 Partitioned ExposedMedia for a total of five plates.

```
<ExposedMediaLink Usage="Input" rRef="E1">
    <Part Separation="Cyan" SheetName="S1"/>
    <Part Separation="Magenta" SheetName="S1"/>
    <AmountPool>
        <PartAmount>
            <Part Separation="Cyan" SheetName="S1" Side="Front"/>
        </PartAmount>
        <PartAmount>
            <Part Separation="Cyan" SheetName="S1" Side="Back"/>
        </PartAmount>
        <PartAmount>
            <Part Separation="Magenta" SheetName="S1" Side="Front"/>
        </PartAmount>
        <PartAmount Amount="2">
            <Part Separation="Magenta" SheetName="S1" Side="Back"/>
        </PartAmount>
    </AmountPool>
</ExposedMediaLink>
```

3.10.7.4 Implicit, Sparse and Explicit PartUsage in Partitioned Resources

New in JDF 1.2

The @*PartUsage* attribute defines how over-specified ***ResourceLink*** elements SHALL be resolved.

If @*PartUsage* = "Explicit", ***ResourceLink*** elements that do not point to an explicitly defined partition of a resource are an error.

If @*PartUsage* = "Implicit", ***ResourceLink*** elements that do not point to an explicitly defined partition of a resource refer to the closest matching resource partition, regardless of the existence of sibling partitions with identical keys but mismatching values.

If @*PartUsage* = "Sparse", ***ResourceLink*** elements that do not point to an explicitly defined partition of a resource refer to the closest matching resource partition, if no sibling partitions with identical keys but mismatching values exist. If sib-

ling partitions with identical keys but mismatching values exist, ***ResourceLink*** elements that do not point to an explicitly defined partition of a resource are in error.

Example 3.31: PartUsage in a Partitioned Resource

Table 3.28 PartUsage Attribute examples below describes the behavior of the **JDF** example that follows. Table 3.28 PartUsage Attribute examples shows the @*ProductID* of the resource partition that is selected for various values of @*SheetName*, @*Side*, @*Separation* and @*PartVersion* for @*PartUsage* = "Implicit", "Explicit" and "Sparse", respectively. Note the effects of the ***Identical*** element in S2B.

Table 3.28: PartUsage Attribute examples

SHEETNAME	SIDE	SEPARATION	PARTVERSION	IMPLICIT	EXPLICIT	SPARSE
—	—	—	—	Root	Root	Root
S1	—	—	—	S1	S1	S1
S2	—	—	—	S2	S2	S2
S3	—	—	—	Root	—	—
S2	Back	Cyan	—	S1BC	S1BC	S1BC
S1	Back	Cyan	—	S1BC	S1BC	S1BC
S1	Back	Orange	—	S1B	—	—
S2	Back	Orange	—	S1B	—	—
S1	—	Cyan	—	S1BC, S1FC	S1BC, S1FC	S1BC, S1FC
S1	Back	Cyan	Deutsch	S1BC	—	S1BC
S2	Back	Cyan	Deutsch	S1BC	—	S1BC
S2	Front	Cyan	Deutsch	S2FC	—	S2FC
S1	Back	Black	Deutsch	S1BKD	S1BKD	S1BKD

Note: The example below has @*PartUsage* = "Implicit" and explicit values for ***ExposedMediaLink*/*Part*** attributes, but ▸ Table 3.28 PartUsage Attribute examples above describes the behavior for all values of @*PartUsage* and all values of ***ExposedMediaLink*/*Part***.

```
<ResourceLinkPool>
    <ExposedMediaLink Usage="Input" rRef="XM_ID">
        <Part SheetName="S1" Side="Front" Separation="Black" PartVersion="Deutsch"/>
    </ExposedMediaLink>
</ResourceLinkPool>
<ResourcePool>
    <ExposedMedia Brand="Gooey" Class="Handling" ID="XM_ID"
          PartIDKeys="SheetName Side Separation PartVersion"
          PartUsage="Implicit" ProductID="Root" Status="Available">
        <Media Dimension="500 600" MediaType="Plate"/>
        <ExposedMedia ProductID="S1" SheetName="S1">
            <ExposedMedia ProductID="S1F" Side="Front">
                <ExposedMedia ProductID="S1FC" Separation="Cyan"/>
                <ExposedMedia ProductID="S1FM" Separation="Magenta"/>
                <ExposedMedia ProductID="S1FY" Separation="Yellow"/>
                <ExposedMedia ProductID="S1FK" Separation="Black">
                    <ExposedMedia ProductID="S1FKD" PartVersion="Deutsch"/>
                    <ExposedMedia ProductID="S1FKE" PartVersion="English"/>
                </ExposedMedia>
            </ExposedMedia>
            <ExposedMedia ProductID="S1B" Side="Back">
                <ExposedMedia ProductID="S1BC" Separation="Cyan"/>
                <ExposedMedia ProductID="S1BM" Separation="Magenta"/>
                <ExposedMedia ProductID="S1BY" Separation="Yellow"/>
                <ExposedMedia ProductID="S1BK" Separation="Black">
                    <ExposedMedia ProductID="S1BKD" PartVersion="Deutsch"/>
                    <ExposedMedia ProductID="S1BKE" PartVersion="English"/>
                </ExposedMedia>
            </ExposedMedia>
        </ExposedMedia>
        <ExposedMedia ProductID="S2" SheetName="S2">
            <ExposedMedia ProductID="S2F" Side="Front">
                <ExposedMedia ProductID="S2FC" Separation="Cyan"/>
                <ExposedMedia ProductID="S2FM" Separation="Magenta"/>
                <ExposedMedia ProductID="S2FY" Separation="Yellow"/>
                <ExposedMedia ProductID="S2FK" Separation="Black"/>
            </ExposedMedia>
            <ExposedMedia Side="Back">
                <Identical>
                    <Part SheetName="S1" Side="Back"/>
                </Identical>
            </ExposedMedia>
        </ExposedMedia>
    </ExposedMedia>
</ResourcePool>
```

3.10.7.5 Referencing Multiple Resources of the Same Type

Some processes (e.g., **Collecting**, **Gathering**) allow multiple input resources of the same type. These multiple input resources MAY be represented by multiple individual resources or by partitioned resources or by a mixture of both. If ordering is significant, the order of the leaves in a partitioned resource defines said ordering. ▸ Example 101 Explicit Reference of Ordered Partitioned Resources and ▸ Example 101 Implicit Reference of Ordered Partitioned Resources illustrate equivalent ways of gathering three input sheets.

For **Gathering**, **Collecting**, **Inserting** and similar processes that have multiple physical input resources, explicit links SHOULD be used to define how the output component is ordered. Implicit references of ordered partitioned resources are strongly discouraged since there is ambiguity if input components have multiple partition levels.

Example 3.32: Explicit Reference of Ordered Partitioned Resources

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Link0037" Status="Waiting"
       Type="Gathering" JobPartID="ID345" Version="1.4" >
    <ResourcePool>
       <GatheringParams Class="Parameter" ID="Gath01" Locked="false"
              Status="Available"/>
       <Component Class="Quantity" ComponentType="Sheet"
              DescriptiveName="printed insert sheets" ID="Sheets01"
              PartIDKeys="SheetName" Status="Available">
          <Component SheetName="Sheet1"/>
          <Component SheetName="Sheet2"/>
          <Component SheetName="Sheet3"/>
       </Component>
       <Component Class="Quantity" ComponentType="Sheet"
              ID="SheetsOut" Status="Available"/>
    </ResourcePool>
    <ResourceLinkPool>
       <GatheringParamsLink Usage="Input" rRef="Gath01"/>
       <!--three ComponentLink explicitly reference individual parts -->
       <ComponentLink Usage="Input" rRef="Sheets01">
          <Part SheetName="Sheet1"/>
       </ComponentLink>
       <ComponentLink Usage="Input" rRef="Sheets01">
          <Part SheetName="Sheet2"/>
       </ComponentLink>
       <ComponentLink Usage="Input" rRef="Sheets01">
          <Part SheetName="Sheet3"/>
       </ComponentLink>
       <ComponentLink Usage="Output" rRef="SheetsOut"/>
    </ResourceLinkPool>
</JDF>
```

Example 3.33: Implicit Reference of Ordered Partitioned Resources

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Link0037" Status="Waiting"
       Type="Gathering" JobPartID="ID345" Version="1.4">
    <ResourcePool>
       <GatheringParams Class="Parameter" ID="Gath01" Locked="false"
              Status="Available"/>
       <Component Class="Quantity" ComponentType="Sheet"
              DescriptiveName="printed insert sheets" ID="Sheets01"
              PartIDKeys="SheetName" Status="Available">
          <Component SheetName="Sheet1"/>
          <Component SheetName="Sheet2"/>
          <Component SheetName="Sheet3"/>
       </Component>
       <Component Class="Quantity" ComponentType="Sheet"
              ID="SheetsOut" Status="Available"/>
    </ResourcePool>
    <ResourceLinkPool>
       <GatheringParamsLink Usage="Input" rRef="Gath01"/>
       <!--the ComponentLink implicitly references all three parts -->
       <ComponentLink Usage="Input" rRef="Sheets01"/>
       <ComponentLink Usage="Output" rRef="SheetsOut"/>
    </ResourceLinkPool>
</JDF>
```

3.10.8 Splitting and Combining Resources

Depending on the circumstances, it MAY be appropriate either to split a resource into multiple new nodes or to specify multiple locations or parts for an individual resource. There are four possible methods for splitting and combining resources. Two methods are shown in ▸ Figure 3-11: Workflow for splitting shared Input Resources and ▸ Figure 3-12: Workflow for combining shared Output Resources and represent workflows that use the *@Amount* attribute of their ***ResourceLink*** elements to share resources. This method is practical when one controller controls all aspects of resource consumption or production. In ▸ Figure 3-11: Workflow for splitting shared Input Resources, the resource amount is split between subsequent processes. In ▸ Figure 3-12: Workflow for combining shared Output Resources, individual processes

produce amounts that are then combined into a unified resource that is, in turn, used by a single process. In both cases, a single, shared resource is employed. To enable independent parallel processing by multiple controllers, however, independent resources are needed. To create independent resources from one resource, the **Split** process is used, as shown in ▸ Figure 3-13: Workflow for splitting independent Input Resources (for further details, see ▸ Section 6.2.10 Split). This process allows multiple processes to be spawned off, after which multiple processes can consume the same resource in parallel and can therefore run in parallel. ▸ Figure 3-14: Workflow for combining independent Output Resources demonstrates the reverse situation, which occurs if resources have been produced by multiple processes and are then consumed, as a unified entity, by a single subsequent process. To accomplish this, the **Combine** process combines multiple resources to create the single resource.

Figure 3-11: Workflow for splitting shared Input Resources

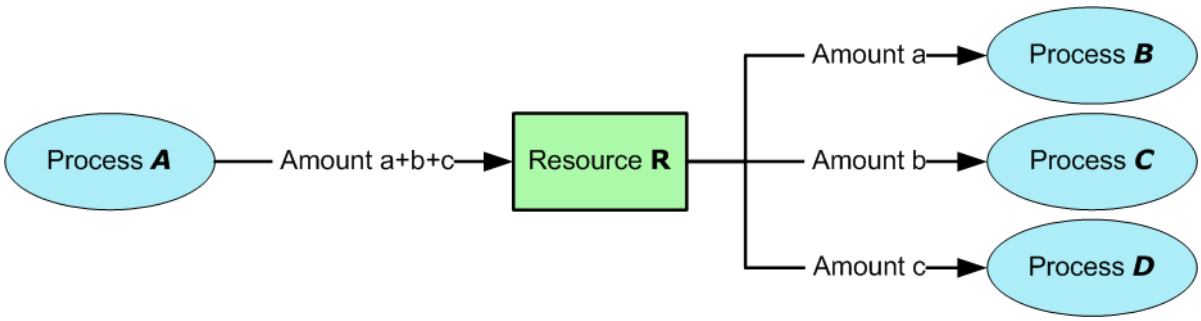


Figure 3-12: Workflow for combining shared Output Resources

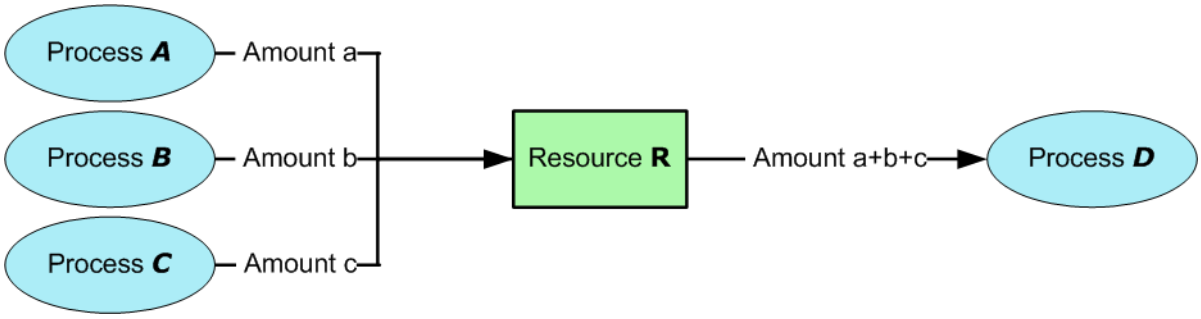


Figure 3-13: Workflow for splitting independent Input Resources

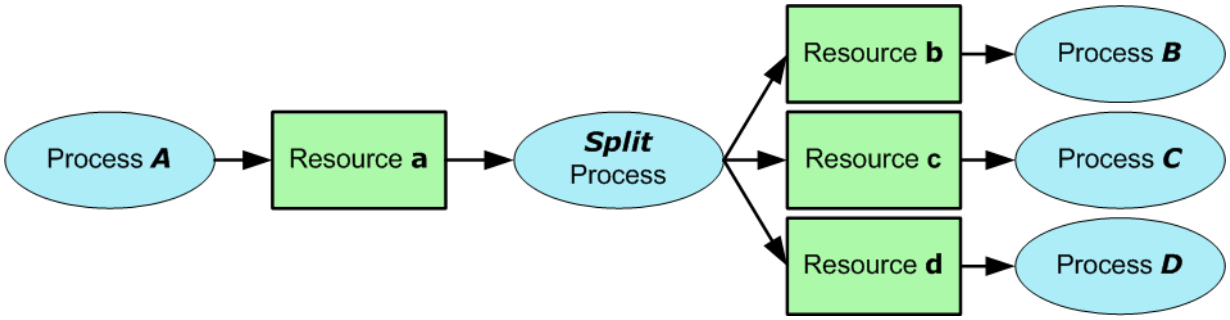
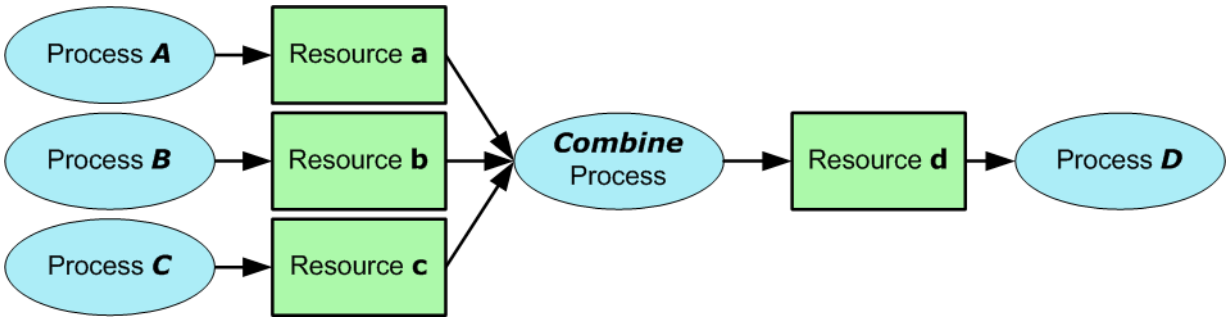


Figure 3-14: Workflow for combining independent Output Resources



3.11 AuditPool and Audit

AuditPool Elements

Audit information is the job's history and can support your daily, quality control and troubleshooting management reporting needs.

Audit elements contain the post-facto recorded results of a process such as the execution of a **JDF** node or modification of the **JDF** itself. ***Audit*** elements become static after a process has been finished. They SHALL NOT be modified after the process has been aborted or completed. Therefore, if ***PhaseTime*** or ***ResourceAudit*** audit elements link to resources, those resources SHOULD be locked in order to inhibit accidental modification of audited information, which is why **JDF** includes a locking mechanism for resources. ***Audit*** elements record any event related to the following situations:

- The creation of a **JDF** node by a ***Created*** audit.
- Spawning and merging, including resource copying by spawned and merged audits.
- Errors such as unnecessary ***ResourceLink*** audits, wrongly linked resources, missing resources or missing links, which might be detected by agents during a test run or by a ***Notification*** audit.
- Actual data about the production and resource consumption by a ***ResourceAudit*** audit.
- Any process phase times. Examples include setting up a device, maintenance and washing, as well as down-times as a result of failure, breaks or pauses. Changes of ***ImplementationResource*** usage, such as a change of operators by a ***PhaseTime*** audit, would also constitute an example of a phase time.
- Actual execution data. For example, the process start and end times, as well as the final process state, as determined by a ***ProcessRun*** audit.
- Any modification of a **JDF** node not covered by the preceding items, as recorded by a ***Modified*** or ***Deleted*** audit.

Audit information might be used by MIS for operations such as evaluation or invoicing. The ▸ Figure 3-15: AuditPool and Abstract Audit Element – a diagram of the structure depicts the structure of the ***AuditPool*** and concrete ***Audit*** elements, such as ***Created***, derived from the ***Abstract Audit*** element. ***Audit*** entries are ordered chronologically, with the last entry in the ***AuditPool*** representing the newest. A ***ProcessRun*** containing the scheduling data finalizes each process run. All subsequent entries belong to the next run.

3.11.1 AuditPool

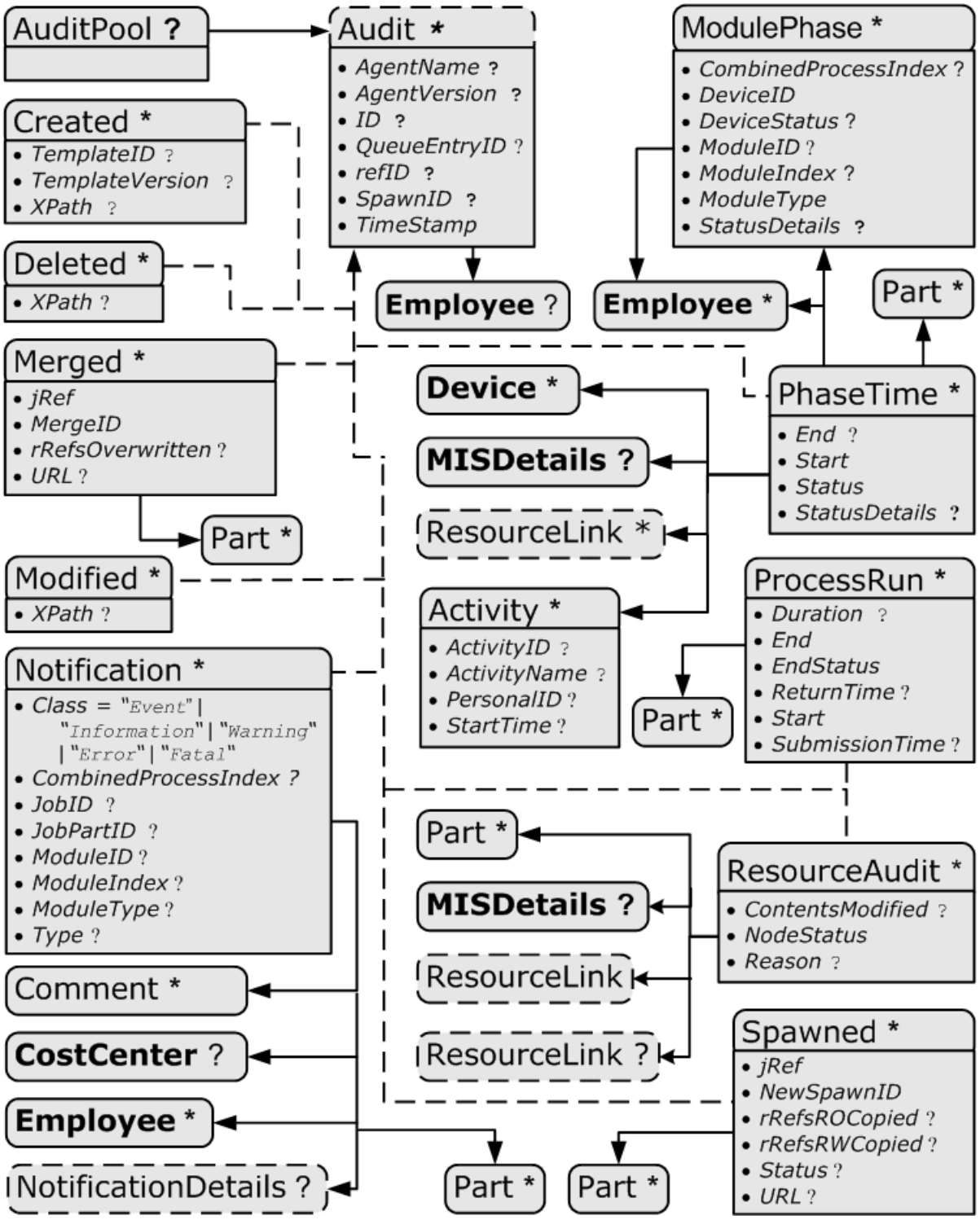
The following table defines the contents of the ***AuditPool*** element.

Table 3.29: AuditPool Element

NAME	DATA TYPE	DESCRIPTION
rRefs ? Deprecated in JDF 1.2	IDREFS	List of all resources that are referenced from within the ***AuditPool***. In **JDF** 1.2 and beyond, it is up to the implementation to maintain references.
Audit *	element	Chronologically ordered list of ***Audit*** elements. The ***Audit*** elements are abstract and serve as placeholders for any concrete element derived from the ***Abstract Audit*** element. ***Audit*** elements are described in the sections that follow.

3.11.2 Structure Diagram

Figure 3-15: AuditPool and Abstract Audit Element – a diagram of the structure



3.11.3 Abstract Audit

All ***Audit*** elements inherit the content from the ***Abstract Audit*** element, described in the following table.

Table 3.30: Abstract Audit Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AgentName ? New in JDF 1.2	string	The name of the agent application that added the ***Audit*** element to the ***AuditPool*** (and was responsible for the creation or modification). Both the company name and the product name MAY appear, and SHOULD be consistent between versions of the application.

Table 3.30: Abstract Audit Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
AgentVersion ? New in JDF 1.2	string	The version of the agent application that added the ***Audit*** element to the ***AuditPool*** (and was responsible for the creation or modification). The format of the version string MAY vary from one application to another, but SHOULD be consistent for an individual application.
Author ? Modified in JDF 1.2 Deprecated in JDF 1.4	string	Text that identifies the person who made the entry. Prior to **JDF** 1.2, @*Author* also contained information that is now encoded in @*AgentName* and @*AgentVersion*. **Deprecation note:** Starting with **JDF** 1.4, use ***Employee***.
ID ? New in JDF 1.2	ID	@*ID* of the ***Audit***. @*ID* SHALL be specified if there is support to subsequently create correction ***Audit*** elements.
QueueEntryID ? New in JDF 1.4	string	@*QueueEntryID* of the ***QueueEntry*** during which this ***Audit*** was generated.
refID ? New in JDF 1.2	IDREF	Reference to a previous ***Audit*** that this ***Audit*** corrects. The referenced ***Audit*** SHALL reside in the same ***AuditPool***.
SpawnID ? New in JDF 1.1	NMTOKEN	Text that identifies the spawned processing step when the entry was generated. This is a copy of the @*SpawnID* attribute of the root **JDF** node of the process that generates the ***Audit*** at the time the ***Audit*** is generated.
TimeStamp	dateTime	For ***Audit*** elements ***Created***, ***Modified***, ***Spawned***, ***Merged*** and ***Notification***, this attribute records the date and time when the related event occurred. For ***Audit*** elements ***PhaseTime***, ***ProcessRun*** and ***ResourceAudit***, the attribute describes the time when the entry was appended to the ***AuditPool***.
Employee ? New in JDF 1.4	element	Employee who created this ***Audit*** element.

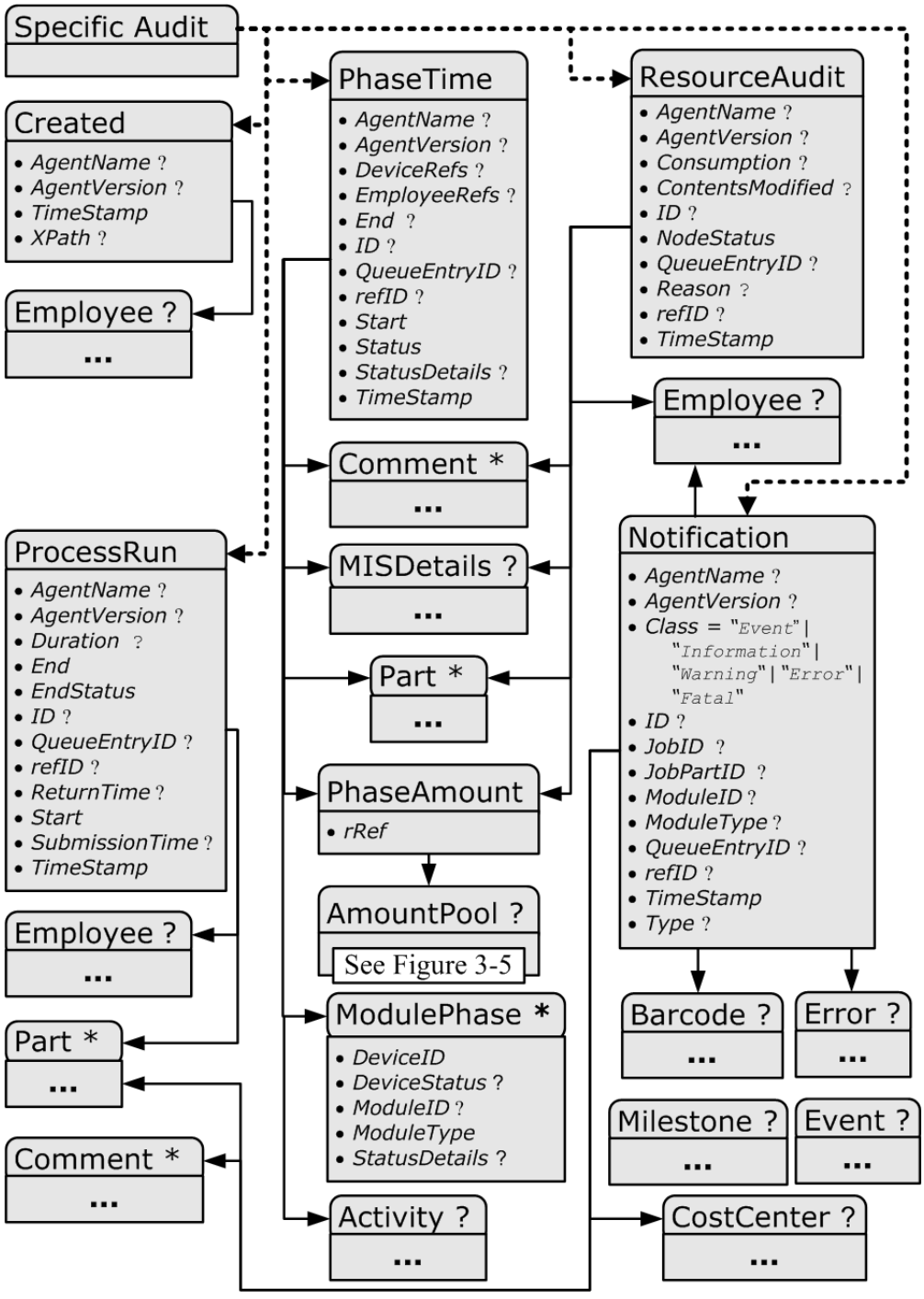
3.11.4 Audit

The following elements are derived from the ***Abstract Audit*** element:

Table 3.31: List of Audit Elements

NAME	PAGE	DESCRIPTION
Created	page 106	Logs creation of **JDF** node or resource
Deleted	page 107	Logs deletion of **JDF** node or resource
Merged	page 107	Logs the merging of a spawned node
Modified	page 107	Logs modifications affecting a **JDF** node or its subelements when the modification is not covered by other ***Audit*** elements
Notification	page 108	Logs individual events that occurred during processing
PhaseTime	page 109	Logs start and end times of any process states and substates, denoted as phases. Phases can reflect any arbitrary subdivisions of a process.
ProcessRun	page 112	Summarizes one complete execution run of a node or delimits a group of ***Audit*** elements for each individual process run.
ResourceAudit	page 113	Describes the usage of resources during execution of a node or the modification of the intended usage of a resource
Spawned	page 115	Logs the spawning of a node.

Figure 3-16: Specific Audit - a Diagram of its Structure



3.11.4.1 Created

This element allows the creation of a **JDF** node or resource to be logged. If the element refers to a **JDF** node, it can be located in the ***AuditPool*** element of the node that has been created or in any ancestor node. If the element refers to a resource, it SHALL be located in the node where the resource resides so that the spawning and merging mechanism can work effectively.

Table 3.32: Created Audit Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ref ? Deprecated in JDF 1.2	IDREF	Represents the ID of the created element. Defaults to the ID of the local **JDF** node. Replaced with @*XPath* in **JDF** 1.2 and beyond.
TemplateID ? New in JDF 1.2	string	Defines the template **JDF** that was used as the template to create the node.

Table 3.32: Created Audit Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
TemplateVersion ? New in JDF 1.2	string	Defines the version of template **JDF** that was used as the template to create the node.
XPath ? New in JDF 1.2	XPath	Location of the created elements or attributes relative to the parent **JDF** node of the ***Created*** element.

3.11.4.2 Deleted

New in JDF 1.2

This element allows any deletions of a **JDF** node or element to be logged. If the corresponding ***Created*** element was not deleted (e.g., in the ***AuditPool*** of a deleted **JDF** node), the ***Deleted*** element SHOULD reside in the same ***AuditPool*** as the corresponding ***Created*** element, otherwise it SHOULD reside in an ancestor of the deleted attribute or element.

Table 3.33: Deleted Audit Element

NAME	DATA TYPE	DESCRIPTION
XPath ?	XPath	Location of the deleted elements or attributes relative to the parent **JDF** node of the ***Deleted*** element.

3.11.4.3 Merged

This element logs a merging event of a spawned node. For more details, see ▸ Section 4.4 Spawning and Merging.

Table 3.34: Merged Audit Element

NAME	DATA TYPE	DESCRIPTION
Independent = "false" Deprecated in JDF 1.5	boolean	Declares that independent jobs are merged into a big job for common production. If it is set to "true", the attributes @*jRefSource* and @*rRefsOverwritten* have no meaning and SHOULD be omitted. **Deprecation note:** Starting with **JDF** 1.5, use **SheetOptimizing**.
jRef	IDREF	ID of the **JDF** node that has been returned or merged.
jRefSource ? Deprecated in JDF 1.5	NMTOKEN	ID of the **JDF** root node of the big job from which the spawned structure has been returned. **Note:** The data type is NMTOKEN and not IDREF because the attribute refers to an external ID. **Deprecation note:** Starting with **JDF** 1.5, use **SheetOptimizing**.
MergeID New in JDF 1.1	NMTOKEN	Copy of the @*SpawnID* of the merged node. Note that a ***Merged*** element MAY also contain a @*SpawnID* attribute, which is the @*SpawnID* of the node that this ***Audit*** is being placed into prior to merging.
rRefsOverwritten ?	IDREFS	Identifies the copied resources that have been overwritten during merging. Resources are usually overwritten during return if they have been copied during spawning with read/write access.
URL ? New in JDF 1.1	URL	Locator that specifies the location of the merged node prior to merging by the merging process.
Part *	element	Specifies the selected parts of the resource that were merged in case of parallel spawning and merging of partitionable resources. See ▸ Section 3.10.5 Description of Partitioned Resources.

3.11.4.4 Modified

This element allows any modifications affecting a **JDF** node or its subelements to be logged. Changes that can be logged by a more specialized ***Audit*** element (e.g., ***ResourceAudit*** for resource changes) SHALL NOT use this common log entry.

The modification can be described textually by adding a generic ***Comment*** element to the ***Modified*** element. The ***Modified*** element SHALL reside in the same ***AuditPool*** as the corresponding ***Created*** element.

Table 3.35: Modified Audit Element

NAME	DATA TYPE	DESCRIPTION
jRef ? Deprecated in JDF 1.2	IDREF	The ID of the modified node. The ***Modified*** element resides in the modified node. Defaults to the ID of the local **JDF** node. Replaced with @*XPath* in **JDF** 1.2 and beyond.
XPath ? New in JDF 1.2	XPath	Location of the modified elements or attributes relative to the parent **JDF** node of the ***Modified*** element.

3.11.4.5 Notification

This element contains information about individual events that occurred during processing. For a detailed discussion of event properties, see ▸ Section 4.6 Error Handling.

Table 3.36: Notification Audit Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Class	enumeration	Class of the notification. **Allowed values are:** (in order of severity from lowest to highest): Event – Indicates that a pure event due to certain operation-related activity has occurred (e.g., machine events, operator activities, etc.). This class is used for the transfer of conventional event messages. In case of @*Class* = "Event", further event information is to be provided by the @*Type* attribute and ***NotificationDetails*** element. See ▸ Section C.8.2 NotificationDetails. Information – Any information about a process which cannot be expressed by the other classes (e.g., the beginning of execution). No user interaction is needed. Warning – Indicates that a minor error has occurred, and an automatic fix was applied. Execution continues. The node's @*Status* is unchanged. This appears in situations such as A4-Letter substitutions when toner is low or when unknown extensions are encountered in a REQUIRED resource. Error – Indicates that an error has occurred that requires user interaction. Execution cannot continue until the problem has been fixed. The node's @*Status* is "Stopped". This value appears in situations such as when resources are missing, when major incompatibilities are detected, or when the toner is empty. Fatal – Indicates that a fatal error led to abortion of the process. The node's @*Status* is "Aborted". This value is seen with most protocol errors or when major device malfunction has occurred.
CombinedProcessIndex ? New in JDF 1.4	IntegerList	@*CombinedProcessIndex* attribute specifies the indices of individual processes in the @*Types* attribute to which a ***Notification***in a combinded process node or process group node belongs. Multiple entries in @*CombinedProcessIndex* specify that the module specified by ***Notification*** is executing the respective multiple processes in the combinded process node.
JobID ? New in JDF 1.3	string	@*JobID* that this ***Notification*** applies to. @*JobID* SHALL NOT be specified when ***Notification*** is used as an ***Audit*** element. ***Notification***/@*JobID* MAY be specified within a **JMF** message.
JobPartID ? New in JDF 1.3	string	@*JobPartID* that this ***Notification*** applies to. @*JobPartID* SHALL NOT be specified when ***Notification*** is used as an ***Audit*** element. ***Notification***/@*JobPartID* MAY be specified within a **JMF** message.
ModuleID ? New in JDF 1.4	string	@*ModuleID* of the ***Module*** that this ***Notification*** relates to.
ModuleIndex ? New in JDF 1.4	IntegerRangeList	0-based indices of the module or modules. The list is based on all modules of the device. If multiple module types are available on one device, each SHALL be unique in the scope of the device. **Constraint:** At least one of @*ModuleID* or @*ModuleIndex* SHALL be specified.

Table 3.36: Notification Audit Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ModuleType ? New in JDF 1.4	NMTOKEN	Module description. **Values include those from:** ▸ Section C.2 DeviceClass and ModuleType Supported Strings**.** **Note:** The allowed values depend on the type of device. Each type of device has a separate table of values.
Type ?	NMTOKEN	Identifies the type of notification. Also defines the name of the ***Abstract NotificationDetails*** element. **Note:** @*Type* allows parsers that do not have access to the schema to find the instance of ***NotificationDetails***. **Values include those from:** ▸ Table C.14 List of NotificationDetails Elements**.**
Comment *	element	A ***Comment*** element contains a verbose, human-readable description of the event. If the value of the @*Class* attribute is one of "Information", "Warning", "Error" or "Fatal", at least one ***Comment*** element SHOULD be specified. Otherwise (including for @*Class* = "Event"), ***Comment*** elements are OPTIONAL.
CostCenter ?	element	The cost center to which this event is related to.
Employee *	refelement	The employees associated with this event.
NotificationDetails ?	element	***NotificationDetails*** is an abstract element that is a placeholder for additional structured information. ▸ Table C.14 List of NotificationDetails Elements. It provides additional information beyond the @*Class* and @*Type* attribute and beyond the ***Comment*** element. For derived elements see ▸ Table C.14 List of NotificationDetails Elements.
Part * New in JDF 1.1	element	Describes which parts of a process this ***Notification*** belongs to. If ***Part*** is not specified for a ***Notification***, it refers to all parts. For example, imagine a print job that is to produce three different sheets. All sheets are described by one partitioned resource. The ***Part*** elements define, unambiguously, the sheet to which the ***Audit*** refers.

3.11.4.6 PhaseTime

This element contains audit information about the start and end times of any process states and substates, denoted as phases. Phases can reflect any arbitrary subdivisions of a process, such as maintenance, washing, plate changing, failures and breaks.***PhaseTime*** elements SHOULD be closed whenever a significant status change that is detected.

PhaseTime elements MAY also be used to log the actual time spans when ***ImplementationResources*** are used by a process. For example, the temporary usage of a fork lift can be logged if a ***PhaseTime*** element is added that contains a link to the fork lift device resource and specifies the actual start and end time of the usage of that fork lift.

PhaseTime elements that apply to identical partitions and contain at least one identical ***ModulePhase*** SHALL NOT overlap in time. ***PhaseTime*** elements that apply to different partitions MAY overlap in time in order to indicate parallel processing. ***PhaseTime*** elements that apply to different modules MAY overlap in time in order to indicate independent processing with individual modules.

Table 3.37: PhaseTime Audit Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
End ? Modified in JDF 1.3	dateTime	Date and time of the end of the phase. If not specified, the ***PhaseTime*** is ongoing and the end of the phase has not yet occurred. This will generally be the case in the last ***PhaseTime*** of a snapshot **JDF** in a status **JMF**. See ▸ Section 5.55 Status for details.**,**
Start	dateTime	Date and time of the beginning of the phase.

Table 3.37: PhaseTime Audit Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Status Modified in JDF 1.3	enumeration	Status of the phase. **Allowed values are:** (a subset of ***JDF***/@*Status*) TestRunInProgress Setup InProgress Cleanup Spawned – Deprecated in JDF 1.3 Suspended – New in JDF 1.3 Stopped **Note:** The values of this @*Status* attribute are a subset of the possible state values ***JDF***/@*Status*. For all possible states of a **JDF** node see ▸ Table 3.4 JDF. The remaining set of states (i.e., "Ready", "FailedTestRun", "Aborted" and "Completed") are end states and are specified in ***ProcessRun***/@*EndStatus*.
StatusDetails ?	string	Description of the status phase that provides details beyond the enumerative values given by the @*Status* attribute. **Values include those from:** ▸ Section C.11 StatusDetails Supported Strings.
Activity * New in JDF 1.5	element	Operator and device activities that are related to a specific job or job phase.
Device *	refelement	Links to ***Device*** resources that are working during this phase. If one or more ***Device*** resource(s) was used during this phase, this refelement SHOULD link to that/those ***Device*** resource(s)
Employee *	refelement	Links to ***Employee*** resources that are working during this phase. If one or more ***Employee*** resources was active during this phase, this refelement SHOULD link to that/those ***Employee*** resource(s).The first employee referenced in this list is the employee who created this ***Audit PhaseTime*** element.
MISDetails ? New in JDF 1.2	element	Definition how the costs for the execution of this ***PhaseTime*** are to be charged.
ModulePhase *	element	Additional phase information of individual device modules, such as print units.
Part *	element	Describes which parts of a job is currently being logged. If a ***Part*** is not specified for a node that modifies partitioned resources, @*PhaseTime* refers to all parts. For example, imagine a print job that is to produce three different sheets. All sheets are described by one partitioned resource. In order to separate the different print phases for each sheet, the ***Part*** elements define, unambiguously, the sheet to which the ***Audit PhaseTime*** refers.
ResourceLink * New in JDF 1.1	element	These ***ResourceLink*** elements specify the actual consumption/usage or production of resources during this production phase. All attributes apply to production and consumption within this ***PhaseTime*** only, thus ***ResourceLink***/@*ActualAmount* specifies the actual amount produced or consumed.

3.11.4.6.1 Activity

New in JDF 1.5

Activity elements allow tracking of device and operator tasks.

Table 3.38: Activity Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ActivityID ?	string	ID of the activity being performed. This ID is unique, site specific and internal to the MIS.
ActivityName ?	string	Name of the activity being performed.
PersonalID ?	string	ID of the employee that performs the activity. This value SHALL match the @*PersonalID* of an ***Employee*** element that is contained in the ***Activity*** element's parent or ancestor element.

Table 3.38: Activity Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
StartTime ?	dateTime	Date and time that the employee started the activity. This value MAY remain the same in multiple messages.

3.11.4.6.2 ModulePhase

It is possible to monitor the states of individual modules of a complex device, such as a press with multiple print units, by defining ***ModulePhase*** elements. One ***PhaseTime*** element MAY contain multiple ***ModulePhase*** elements and can, therefore, record the status of multiple units in a device. ***ModulePhase*** elements describe the set of modules that a given ***PhaseTime*** audit element applies to. ***ModulePhase*** elements are defined in the following table.

Table 3.39: ModulePhase Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
CombinedProcessIndex ? New in JDF 1.3	IntegerList	@*CombinedProcessIndex* attribute specifies the indices of individual processes in the @*Types* attribute to which a ***ModulePhase*** in a combinded process node or process group node belongs. Multiple entries in @*CombinedProcessIndex* specify that the module specified by ***ModulePhase*** is executing the respective multiple processes in the combinded process node.
DeviceID	string	ID of the device that the module described by this ***ModulePhase*** belongs to. This SHALL be the @*DeviceID* attribute of one of the ***Device*** elements specified in the ***PhaseTime***.
DeviceStatus ? Modified in JDF 1.3	enumeration	Status of the device module. **Allowed values are:** Unknown. – The module status is unknown. Idle – The module is not used (e.g., a color print module that is inactive during a black-and-white print). Down – The module cannot be used. It might be broken, switched off etc. Setup – The module is currently being set up. Running – The module is currently executing. Cleanup – The module is currently being cleaned. Stopped – The module has been stopped, but running might be resumed later. This status can indicate any kind of break, including a pause, maintenance or a breakdown, as long as running can be easy resumed. **Note:** These states are analog to the device states of ▸ Table 5.106 ModuleStatus Element.
End ? Modified in JDF 1.3 Deprecated in JDF 1.4	dateTime	Date and time of the end of the module phase. If not specified, the ***ModulePhase*** is ongoing and the end of the phase has not yet occurred. **Deprecation note:** Starting with **JDF** 1.4, all status information is recorded in ***PhaseTime***. ***ModulePhase*** selects only the set of modules that a particular ***PhaseTime*** applies to.
ModuleID ? New in JDF 1.3	string	@*ModuleID* of the module that this ***ModulePhase*** refers to. If not specified, the module is specified in @*ModuleIndex*. **Constraint:** at least one of @*ModuleID* or @*ModuleIndex* SHALL be specified.
ModuleIndex ? Modified in JDF 1.3	IntegerRangeList	0-based indices of the module or modules. The list is based on all modules of the device. If multiple module types are available on one device, each SHALL be unique in the scope of the device. **Constraint:** At least one of @*ModuleID* or @*ModuleIndex* SHALL be specified.
ModuleType ? Modified in JDF 1.5	NMTOKEN	Module description. **Values include those from:** ▸ Section C.2 DeviceClass and ModuleType Supported Strings**.** **Note:** The allowed values depend on the type of device. Each type of device has a separate table of values. **Modification note:** Starting with **JDF** 1.5, @*ModuleType* is optional.
Start Modified in JDF 1.3 Deprecated in JDF 1.4	dateTime	Date and time of the beginning of the module phase. **Deprecation note:** Starting with **JDF** 1.4, all status information is recorded in ***PhaseTime***. ***ModulePhase*** selects only the set of modules that a particular ***PhaseTime*** applies to.

Table 3.39: ModulePhase Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
StatusDetails ?	string	Description of the module status phase that provides details beyond the enumerative values given by the @*DeviceStatus* attribute. **Values include those from:** ▸ Section C.11 StatusDetails Supported Strings**.**
Employee * Deprecated in JDF 1.5	refelement	References to ***Employee*** resources that are working during this module phase on this module. (The module is specified by the attributes @*ModuleIndex* and @*ModuleType*). **Deprecation note:** Starting with **JDF** 1.5, employees SHOULD only be specified in the parent ***PhaseTime***.

3.11.4.7 ProcessRun

This ***Audit*** element serves two related functions.

The first function is to summarize one complete execution run of a node. It contains attributes that record the date and time of the start, the end time, the final process state when the run is finished and, possibly, the process duration of the process run. These attributes are described in ▸ Table 3.40 ProcessRun Audit Element.

The second function is to delimit a group of ***Audit*** elements for each individual process run. Every group of ***Audit*** elements terminates with a ***ProcessRun*** element, which contains the information described in ▸ Table 3.40 ProcessRun Audit Element. If a process is repeated (e.g., as a result of a late change in the order), all ***Audit*** elements belonging to the new run SHALL be appended after the last ***ProcessRun*** element that terminates the ***Audit*** elements of the previous run. The number of ***ProcessRun*** elements is, therefore, always equivalent to the number of process runs. If a node describes partitioned resources, one ***ProcessRun*** MAY be specified for each individual part.

Table 3.40: ProcessRun Audit Element

NAME	DATA TYPE	DESCRIPTION
Duration ?	duration	Time span of the effective process runtime without intentional or unintentional breaks. That time span is the sum of all process phases when the @*Status* is "InProgress", "Setup" or "Cleanup".
End	dateTime	Date and time at which the process ended.
EndStatus Modified in JDF 1.3	enumeration	The @*Status* of the process at the end of the run. For a description of process states, see ▸ Table 3.4 JDF. **Allowed values are:** Aborted Completed FailedTestRun Ready Stopped.– The execution of the node is stopped and might commence at a later time. In **JDF** 1.3 and beyond, "Stopped" is not an end state. Deprecated in JDF 1.3
ReturnTime ? New in JDF 1.4	dateTime	Date and time of the ***ReturnQueueEntry*** submission. If the **JDF** was returned via a Hot Folder, this time corresponds to the time when the **JDF** was placed into the Hot Folder.
Start	dateTime	Date and time at which the process started.
SubmissionTime ? New in JDF 1.4	dateTime	Date and time of the ***SubmitQueueEntry*** submission. This value SHOULD be identical with ***QueueEntry***/@*SubmissionTime*. If the **JDF** was submitted via a Hot Folder, this time corresponds to the time when the **JDF** was extracted from the Hot Folder.
Part * New in JDF 1.1	element	Describes which parts of a process this ***ProcessRun*** belongs to. If ***Part*** is not specified for a ***ProcessRun***, it refers to all parts. For example, imagine a print job that is to produce three different sheets. All sheets are described by one partitioned resource. The ***Part*** elements define, unambiguously, the processing of the sheet to which the ***ProcessRun*** refers.

3.11.4.8 ResourceAudit

The ***ResourceAudit*** element describes the usage of resources during execution of a node or the modification of the intended usage of a resource (i.e., the modification of a ***ResourceLink***). It logs consumption and production amounts of any quantifiable resources, accumulated over one process run or one part of a process run. It contains one or two abstract ***ResourceLink*** elements. The first is REQUIRED and specifies the actual consumption/usage or production of the resource. The second ***ResourceLink*** is OPTIONAL and used to store information about the original ***ResourceLink***, which also refers to the original resource. If the original resource does not need to be saved, a Boolean *@ContentsModified* attribute in the ***ResourceAudit*** SHOULD be specified as "true" to indicate that a change has been made.

Table 3.41: ResourceAudit Audit Element

NAME	DATA TYPE	DESCRIPTION
ContentsModified ?	boolean	Specifies that a modification has occurred but that the original resource has been deleted.
NodeStatus ? New in JDF 1.3	enumeration	Status of the node that was executed during production or consumption of the resource. **Allowed values are:** (a subset of **JDF**/@*Status*): TestRunInProgress Setup InProgress Cleanup Suspended Stopped **Note:** The values of this @*Status* attribute are a subset of the possible state values ***JDF***/@*Status*. For all possible states of a **JDF** node see ▸ Table 3.4 JDF. The remaining set of states (i.e., "Ready", "FailedTestRun", "Aborted" and "Completed") are end states and are specified in ***ProcessRun***/@*EndStatus*.
Reason ? New in JDF 1.1	enumeration	Reason for the modification. **Allowed values are:** OperatorInput – Human update that corrects inconsistencies from automated data collection. PlanChange – The resource was modified due to a change of plan before actual processing. ProcessResult – The actual consumption.
MISDetails ? New in JDF 1.3	element	Specifies how the costs associated with this ***ResourceAudit*** are to be charged.
Part *	element	Describes which parts of a job is currently being logged. If a part is not specified for a node that modifies partitioned resources, ***ResourceAudit*** refers to all parts.
ResourceLink	element	The first ***ResourceLink*** specifies the actual consumption/usage or production of a resource. This current resource after modification NEED NOT be set to @*Locked*="true".
ResourceLink ?	element	The second ***ResourceLink***, which is OPTIONAL, logs the modification of a ***ResourceLink*** and the modification of the resource it refers to. It holds the planned ***ResourceLink*** which also refers to the planned resource. The planned and actual resource MAY be the same.

For details on ***ResourceLink*** elements and ***ResourceLink*** Subclasses, see ▸ Section 3.9 ResourceLinkPool and ResourceLink. The partitioning of resources using ***Part*** elements is defined in ▸ Section 3.10.5 Description of Partitioned Resources.

3.11.4.8.1 Logging Machine Data by Using the ResourceAudit

If a resource is modified during processing, any nodes that also reference the resource MAY also be affected. The following logging procedure is RECOMMENDED in order to track the resource modification and to insure consistency of the job.

1. Create a copy of the original resource with a new ID.
2. Modify the original resource to reflect the changes.
3. Insert a ***ResourceAudit*** element that references the modified original resource with the first ***ResourceLink*** and the copied resource with the second ***ResourceLink*** attribute

Example 3.34: ResourceAudit: Before Logging

The following example describes the logging of a modification of the media weight and amount. The **JDF** document before modification requests 400 copies of 80 gram media.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
       Type="ConventionalPrinting" JobPartID="ID234" Version="1.4">
    <ResourceLinkPool>
       <MediaLink Amount="400" Usage="Input" rRef="RLink"/>
       <ConventionalPrintingParamsLink Usage="Input" rRef="R01"/>
       <ComponentLink Usage="Output" rRef="R02"/>
    </ResourceLinkPool>
    <ResourcePool>
       <Media ID="RLink" Class="Consumable" Status="Available"
              Amount="400" Weight="80"/>
       <ConventionalPrintingParams ID="R01" Class="Parameter" Status="Available"/>
       <Component ID="R02" Class="Quantity" Status="Unavailable"
              ComponentType="Sheet"/>
    </ResourcePool>
</JDF>
```

Example 3.35: ResourceAudit: Logging of Consumption

The **JDF** after modification specifies that 421 copies of 90-gram media have been consumed.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
       Type="ConventionalPrinting" JobPartID="ID234" Version="1.4">
    <ResourceLinkPool>
       <!-- Note that ActualAmount has been added to the ResourceLink -->
       <MediaLink ActualAmount="421" Amount="400" Usage="Input" rRef="RLink"/>
       <ConventionalPrintingParamsLink Usage="Input" rRef="R01"/>
       <ComponentLink Usage="Output" rRef="R02"/>
    </ResourceLinkPool>
    <ResourcePool>
       <Media ID="RPrev" Class="Consumable" Status="Available" Amount="400"
              Weight="80"/>
       <!--Copy of the original resource-->
       <Media ID="RLink" Class="Consumable" Status="Available" Amount="421"
              Weight="90"/>
       <ConventionalPrintingParams ID="R01" Class="Parameter" Status="Available"/>
       <Component ID="R02" Class="Quantity" Status="Unavailable"
              ComponentType="Sheet"/>
       <!--modified resource-->
    </ResourcePool>
    <AuditPool>
       <ResourceAudit TimeStamp="2008-08-28T18:20:00Z">
          <MediaLink ActualAmount="421" Amount="400" Usage="Input" rRef="RLink"/>
          <MediaLink Amount="400" Usage="Input" rRef="RPrev"/>
       </ResourceAudit>
    </AuditPool>
</JDF>
```

3.11.4.8.2 Logging Changes in Product Descriptions by Using the ResourceAudit

ResourceAudit elements MAY also be used to store the original ***Intent Resource***s of a product specification in a change order or request for requote. The mechanism is the same as above.

Example 3.36: ResourceAudit: Logging Changes

The following example shows the structure of a ***MediaIntent*** with @*Option* partitions, where a late change of options from Option1 (80 gram paper) to Option2 (90 gram paper) is requested.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
       Type="Product" JobPartID="ID234" Version="1.4">
    <ResourceLinkPool>
       <MediaIntentLink Usage="Input" rRef="id">
          <Part Option="Option2"/>
       </MediaIntentLink>
       <ComponentLink Usage="Output" rRef="R02"/>
    </ResourceLinkPool>
    <ResourcePool>
       <MediaIntent ID="id" PartIDKeys="Option">
          <!-- the common MediaIntent resource details -->
          <MediaIntent Option="Option1">
             <Weight Preferred="80" DataType="NumberSpan"/>
          </MediaIntent>
          <MediaIntent Option="Option2">
             <Weight Preferred="90" DataType="NumberSpan"/>
          </MediaIntent>
       </MediaIntent>
       <Component ID="R02" Class="Quantity" Status="Unavailable"
             ComponentType="Sheet"/>
    </ResourcePool>
    <AuditPool>
       <ResourceAudit>
          <!-- the actual MediaIntent ResourceLink -->
          <MediaIntentLink Usage="Input" rRef="id">
             <Part Option="Option2"/>
          </MediaIntentLink>
          <!-- the original MediaIntent ResourceLink -->
          <MediaIntentLink Usage="Input" rRef="id">
             <Part Option="Option1"/>
          </MediaIntentLink>
       </ResourceAudit>
    </AuditPool>
</JDF>
```

3.11.4.9 Spawned

This element allows a node that has been spawned to be logged in the ***AuditPool*** of the parent node of the spawned node or in the ***AuditPool*** of the node that has been spawned in case of spawning of individual partitions. For details about spawning and merging, see ▸ Section 4.4 Spawning and Merging.

Table 3.42: Spawned Audit Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Independent = "false" ? Deprecated in JDF 1.5	boolean	Declares that independent jobs that have previously been merged into a big job are spawned. If it is set to "true", the attributes @*jRefDestination*, @*rRefsROCopied* and @*rRefsRWCopied* have no meaning and SHOULD be omitted. **Deprecation note:** Starting with **JDF** 1.5, use **SheetOptimizing**.
jRef	IDREF	ID of the **JDF** node that has been spawned.
jRefDestination ? Deprecated in JDF 1.5	NMTOKEN	ID of the **JDF** node to which the job has been spawned. This attribute SHALL be specified in the parent of the original node if independent jobs are spawned. **Note:** The data type is NMTOKEN and not IDREF because the attribute refers to an external ID. **Deprecation note:** Starting with **JDF** 1.5, use **SheetOptimizing**.
NewSpawnID New in JDF 1.1	NMTOKEN	Copy of the @*SpawnID* of the newly spawned node. Note that a spawned ***Audit*** MAY also contain a @*SpawnID* attribute, which is the @*SpawnID* of the node that this ***Audit*** is being placed into prior to spawning.

Table 3.42: Spawned Audit Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
rRefsROCopied ?	IDREFS	List of IDs separated by whitespace. Identifies the resources copied to the ***ResourcePool*** element of the spawned **JDF** during spawning. These resources SHOULD NOT be modified by the spawned **JDF**.
rRefsRWCopied ?	IDREFS	List of IDs separated by white spaces. Identifies the resources copied to the ***ResourcePool*** element of the spawned **JDF** during spawning. These resources MAY be modified by the spawned **JDF** and SHALL be copied back into their original location by the merging agent. Resource copying is REQUIRED if resources are referenced simultaneously from spawned nodes and from nodes in the original **JDF** document.
Status ? New in JDF 1.1	enumeration	@*Status* of the spawned node at the time of spawning. . Allowed values are from: ***JDF***/@*Status* (▸ Table 3.4 JDF).
URL ? New in JDF 1.1	URL	Locator that specifies the location where the spawned node was stored by the spawning process.
Part *	element	Identifies the parts that were selected for spawning in case of parallel spawning of partitionable resources. See ▸ Section 3.10.5 Description of Partitioned Resources.

3.12 JDF Extensibility

JDF is meant to be flexible and therefore useful to any vendor, as each vendor may have specific data to include in the **JDF** files. This section describes how **JDF** uses the XML extension mechanisms.

3.12.1 Namespaces in XML

JDF extensibility is implemented using XML Namespaces ▸ [XMLNS]. XML namespaces are defined by @*xmlns* attributes. A general example is provided below.

Namespaces are inserted in front of attribute and element names. The associated namespace of element names with no prefix is the default namespace defined by the xmlns attribute. The associated namespace of attributes with no prefix is that one of the element. All namespace prefixes SHALL be declared using the standard @*xmlns*:prefix attribute declarations.

Using Namespaces in JDF

It is REQUIRED to define the JDF namespace in a JDF document, even if no non-JDF extensions are used. JDF can be defined either in the default namespace or in a qualified namespace.

Example 3.37: Namespaces in XML

The example illustrates how private namespaces are declared and used to extend an existing **JDF** resource by adding private attributes and a private element.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1"
       xmlns:foo="fooschema URI" ID="ID1" Status="Ready"
       JobPartID="ID345" Version="1.4" >
    <!-- ... -->
    <SomeJDFDefinedResource name="abc" foo:specialname="cba">
    <!-- ... -->
    <foo:PrivateStuff type=""/>
    <!-- ... -->
    </SomeJDFDefinedResource>
    <!-- ... -->
</JDF>
```

3.12.1.1 JDF Namespace

The official namespace URI for **JDF** Version 1.0 is: http://www.CIP4.org/JDFSchema_1. The official namespace URI for **JDF** Version 1.1 through **JDF** 1.X is: http://www.CIP4.org/JDFSchema_1_1. It is strongly RECOMMENDED to use either the default namespace with no prefix or a prefix of "jdf" as the **JDF** namespace prefix.

3.12.1.2 JDF Extension Namespace

CIP4 defines an extension namespace where new features that are anticipated to be included in a future version of the specification are defined. The official extension namespace URI for **JDF** Version 1.x is: http://www.CIP4.org/JDFSchema_1_1_X. It is strongly RECOMMENDED to use a prefix of "jdfx" as the **JDF** extension namespace prefix.

3.12.2 Creating Extension Intent elements

New intent elements may be defined by creating an intent with @*Name* referring to a proprietary xml namespace. The extension element SHALL reside in the intent element.

Example 3.38: Creating Extension Intent elements

```
<ResourcePool >
  <foo:BarIntentroduct xmlns:foo="www.foo.com" attrib="myAttrib"/>
</ResourcePool>
```

3.12.3 Extending Process Types

JDF defines a basic set of process types. However, because **JDF** allows flexible encoding, this list, by definition, will not be complete. Vendors that have specific processes that do not fit in the general **JDF** processes and that are not combinations of individual **JDF** processes (see ▸ Section 3.3.3 Combined Process Nodes) can create **JDF** process nodes of their own type. Then the content of the @*Type* attribute MAY be specified with a prefix that identifies the organization. The prefix and name SHALL be separated by a single colon (:) as shown in the following example.

Example 3.39: Extending Process Types

```
<JDF Type="myCompaniesNS:MyVeryImportantProcess"
xmlns="http://www.CIP4.org/JDFSchema_1_1"
xmlns:myCompaniesNS="my companies namespace URI"
ID="ID1" JobPartID="ID345" Status="Ready" Version="1.4" >
<!-- ... -->
</JDF>
```

3.12.3.1 Rules about Process Extension

The use of namespace prefixes in the @*Type* attribute is for extensions only. Standard **JDF** process types SHALL be specified without a prefix in the @*Type* attribute or the @*Types* attribute of a combinded process node . If a process is simply an extension of an existing process, it is possible to describe the private data by extending the existing resource types. This is described in greater detail in the sections below.

Extensibility Caution

JDF “Extensibility” simply means that you can add your own XML elements, attributes and enumerations to a JDF application. Although JDF is quite extensive, odds are you’ll find that your current databases and workflow systems use information elements that are unique to your client market or company … *they might have even been defined by your internal MIS staff*. CIP4 acknowledges that it can’t define everything, nor ought it prevent innovation by codifying everything in a static manner, and JDF’s extensibility provides both printers and technology providers with the flexibility they need to make JDF a success.
However, if you or your technology vendors extend JDF, please do so with caution. JDF’s success depends on the ability of MIS systems and JDF-enabled devices to write, read, parse and use JDF. Extensions are *custom* integration applications and great care needs to be made to ensure that extensions made for one systems or device will not *jam* the JDF workflow or other JDF enabled systems and devices. If they use extensions to JDF, your technology providers need to be able to provide you with a fully validated JDF schema and documentation that includes the use of their extensions. Extensions that are not documented, or that are not to be disclosed to third parties for integration purposes, ought to be viewed skeptically.

3.12.4 Extending the NodeInfo and CustomerInfo Nodes

Extending the ***NodeInfo*** and ***CustomerInfo*** nodes is achieved in a manner analogous to the extension of resources, which is described below. On the other hand, extending the direct contents of **JDF** nodes by adding new elements or attributes is discouraged.

3.12.5 Extending Existing Resources

All resources defined by **JDF** MAY be extended by adding attributes and elements using one’s own namespace for these resource extensions. This is useful when the predefined resource types need only a small amount of private data added, or if those resources are the only appropriate place to put the data. The **JDF** namespace of the extended resource SHALL NOT be modified. However, the mechanism for creating new resources in a separate namespace is provided in the next section.

However, duplicate functionality SHALL NOT be added to these resource types. **JDF**-defined attributes and elements SHALL be used where possible and MAY be extended with additional information only when **JDF**-defined constructs don't exist. For example, it is not allowed to extend the RIP resource that controls the bits per colorant with a *@foo:ColorantDepth* or *@foo:ColDepth* attribute that overrides the **JDF** defined parameter for bits per colorant (see ***RenderingParams***/*@ColorantDepth* in ▸ Section 8.127 RenderingParams).

3.12.6 Extending NMTOKEN Lists

Many resources contain attributes of type NMTOKEN and some of these have a set of predefined, suggested enumerative values. These lists MAY be extended with private keywords. In order to identify private keywords, it is strongly RECOMMENDED to prefix these keywords with a namespace-like syntax (i.e., a namespace prefix separated by a single colon ":"). Such a namespace prefix SHOULD be defined in the **JDF** ticket with the standard xmlns:Prefix="someURI" notation, even if no extension elements or attributes from that namespace occur in the **JDF** ticket. Implementations that find an unknown NMTOKEN prefixed by a namespace prefix MAY then attempt to use the default value of that attribute if the value of *@SettingsPolicy* in effect is "BestEffort".

Example 3.40: Extending NMTOKEN Lists

For instance, if an implementation encounters ***TrappingParams***/*@TrapEndStyle* (see below in ▸ Table 3.43 Excerpt from TrappingParams) in the **JDF** snippet shown below, and if the implementation does not support the "HDM" extension, the best assumption is to use *@TrapEndStyle* = "Miter", which is the default for *@TrapEndStyle*.

```
<TrappingParams TrapEndStyle="HDM:FooBar"/>
```

Table 3.43: Excerpt from TrappingParams

NAME	DATA TYPE	DESCRIPTION
TrapEndStyle = "Miter"	NMTOKEN	Instructs the trap engine how to form the end of a trap that touches another object. **Values include:** Miter Overlap **Note:** Other values might be added later as a result of customer requests.

3.12.7 Creating New Resources

There are certain process implementations that have functionality that cannot be specified by the predefined resource types. In these cases, it might be necessary to create a new resource-type element. If so, the resource SHALL be clearly specified and use its own namespace. These resource types SHALL only be linked to custom-type **JDF** process nodes.

3.12.8 Future JDF Extensions

In future versions, certain private extensions will become more widely used, even by different vendors. As private extensions become more of a general rule, those extensions will be candidates for inclusion in the next version of the **JDF** specification. At that time the specific extensions will have to be described and will be included into the **JDF** namespace.

3.12.9 Maintaining Extensions

Given the mix of vendors that will use **JDF**, it is likely that there will be a number of private extensions. Therefore, **JDF** controllers SHALL be prepared to receive **JDF** files that have extensions. These controllers SHOULD ignore all extensions they don't understand, but under no circumstance are they allowed to remove these extensions when making modifications to the **JDF**. If they do, it will break the extensibility mechanism. For example, imagine that **JDF** agent A creates a **JDF** and inserts private information for process P. Furthermore, the information is only understood by agent A and the appropriate device D for executing P. If the **JDF** needs to be processed first by another agent/Device C and that process removes all private data for P, process P will not be able to produce the correct results on device D that were specified by agent A.

Submit Your Extensions to CIP4

Writing JDF extensions? CIP4 encourages you to become part of the standard and submit your private extensions for review and possible inclusion in future versions of the JDF standard. Not only might adoption of extensions into the JDF standard help make it easier for customers to decide to buy your products, but CIP4 is also considering adopting a formal review process for extensions with future editions of the JDF standard. By participating in JDF's development now, you could save time and customer confusion in the future.

3.12.10 Processing Unknown Extensions

If a node is processed by a controller or device and it encounters an unknown extension in one of its input resources, the expected behavior depends on the current value of @*SettingsPolicy*.

If @*SettingsPolicy* = "BestEffort", a ***Notification*** audit element with @*Class* = "Warning" SHOULD be logged.

If @*SettingsPolicy* = "MustHonor", the process SHALL NOT continue and a ***Notification*** audit element with @*Class* = "Error" SHOULD be logged.

If @*SettingsPolicy* = "OperatorIntervention", the process SHALL stop and wait for an operator intervention and a ***Notification*** audit element with @*Class* = "Warning" SHOULD be logged.

3.12.11 Derivation of Types in XML Schema

The XML Schema definition http://www.w3.org/TR/xmlschema-1/ describes a mechanism to create new types by derivation from old types. This is an alternative to extend or create new elements and is described in Section 4 of http://www.w3.org/TR/xmlschema-0/. This mechanism is not allowed to be applied to any elements defined by **JDF** because such new element types can only be understood by agents/Devices that know the extension. The use of the derivation mechanism is allowed only for private extensions.

3.13 JDF Versioning

New in JDF 1.2

The **JDF** Specification is an evolving document that exists in multiple versions. Real workflows will be executed by devices that individually support different versions of the specification. Complete **JDF** workflow descriptions MAY therefore contain sub-**JDF** nodes that SHALL be specified with different versions in one document.

3.13.1 JDF Versioning Requirements

The following list of requirements take the specific needs of a mixed version **JDF** workflow into account:

- JDf documents with mixed versions SHALL be supported.
  - Environments with devices that support different **JDF** versions will exist.
  - It is not feasible to enforce simultaneous software upgrades for devices from multiple vendors in one production facility.
- MIS systems might not support all versions of all devices that are described in the **JDF**.
  - Customers might update a workflow system or device without updating the MIS system.
- Archived **JDF** documents SHALL remain valid when a new version of the **JDF** specification and schema is published.

3.13.2 JDF Version Definition

The version of a **JDF** node is defined as the highest version of all attributes or elements and linked resources. The version of a resource is defined as the highest version of all elements, attributes or resources that are referenced via refelements.

3.13.3 JDF Version Policies

The following specifies the policies for evolving **JDF** 1.x versions. When the term "**JDF**" is used in the remainder of this section the reader also ought to interpret these policies to apply to **JMF** as well. Version policies include three areas of application: **JDF** specification rules, **JDF** schema definition rules and **JDF** application behavior. The policies are applicable to the transition from **JDF** 1.1/1.1A through to **JDF** 1.4, as well as future versions of **JDF**, but are not applicable to **JDF** 1.0.

3.13.3.1 JDF Specification Version Policies

The following list defines the policies that will be followed when extending the **JDF** specification.

- Changes to the **JDF** specification are always backwards compatible.
  - Extension elements or attributes are never required.
    - New attributes in existing elements SHALL be optional.
    - New elements in existing elements SHALL be optional.
    - New elements MAY contain required elements or attributes.
  - Elements and attributes are never removed.
    - Deprecated elements or attributes continue to be valid in all versions of **JDF** 1.x
  - Data type changes SHALL be extensions of existing data types. In other words the data type of an extended attribute SHALL be a complete superset of the existing data type. For instance, only the extensions defined by the arrow directions are valid.
    - enumeration —> NMTOKEN
    - NMTOKEN —> string

- integer —> IntegerList
- integer —> double

- The *JDF*/@*Version* and *JMF*/@*Version* attributes are REQUIRED in the respective root of **JDF** or **JMF** instance documents.
- The semantics of attributes and elements SHALL NOT be altered.
  - New attributes or elements SHALL NOT be introduced that conditionally modify the semantics of existing attributes and elements.
  - Semantics MAY only be altered when the previous definition is clearly wrong and the result is unpredictable with the previous definition (e.g., bug fixes in the specification). These changes SHALL be clearly marked in the specification.
- The default values of attributes and elements SHALL NOT be altered.
  - The default behavior that is specified when an attribute or element is missing SHALL NOT be altered.

3.13.3.2 JDF Schema Version Policies

The following list defines the policies that will be followed when generating new schemas for new versions of the **JDF** specification.

- Changes to the **JDF** schema SHALL always be backwards compatible.
  - **JDF** 1.x documents SHALL validate against **JDF** 1.(x+n) schemas.
- Only one **JDF** schema namespace SHALL be defined for all versions of **JDF** 1.x.
  - The namespace is http://www.CIP4.org/JDFSchema_1_1.
- The xs:version attribute SHALL BE defined in the schema.
  - Applications that read a schema MAY verify that they are compatible with the version of the schema.
  - Applications MAY choose a schema based on the schema's version tag.
    - The schema version selection MAY be based on a best match to both application and **JDF** ticket or even **JDF** node.
- The *JDF*/@*Version* attribute is defined as an enumeration that contains all valid versions for the schema (e.g., "1.1", "1.2" and "1.3" for the **JDF** 1.3 version of the schema). The schema data type of a **JDF** of **JMF** version is "JDFJMFVersion".
  - This allow schema validators to detect incompatible versions when parsing a local legacy schema.
- The version annotations in the schema SHOULD be maintained wherever possible.
- Explicit copies of published legacy schema versions SHALL be available on the CIP4 website.
- The schema default values of deprecated attributes SHALL be removed from the schema. Deprecated attributes SHALL still be valid but SHALL NOT be explicitly defaulted in the schema.

3.13.3.3 JDF Application Version Policies

This section specifies the policies that implementations SHOULD follow in order to support multiple versions of **JDF**. The policies are specified for agents and controllers/devices separately.

3.13.3.3.1 JDF Agent Version Policies

JDF agents SHALL ensure that the **JDF** that they generate is consistently versioned.

- An agent SHALL update the *JDF*/@*Version* attribute when inserting new attributes or elements.
  - If an agent is not aware of versions, it SHALL assume that anything that it writes belongs to the agent's maximum version. In this case, the version of any node that is affected is the maximum of its prior version or the agent's version.
- It is strongly RECOMMENDED that an agent honor the *JDF*/@*MaxVersion* attribute.
  - An agent SHOULD NOT add attributes, elements or attribute values that were introduced in a version that is higher than *JDF*/@*MaxVersion*.
- An agent SHOULD insert the lowest possible *JDF*/@*Version* attribute that is applicable to the nodes version as described in ▸ Section 3.13.2 JDF Version Definition.
- The *JDF*/@*Version* of a spawned **JDF** node is identical to the *JDF*/@*Version* of that node in a complete **JDF**.

3.13.3.3.2 JDF Device/Controller Version Policies

A **JDF** device/controller (i.e., any implementation that reads **JDF**) SHOULD be backwards compatible:

- Implementations SHOULD handle deprecated elements and attributes gracefully.

JDF devices/controllers (i.e., any implementation that reads **JDF**) SHOULD attempt to be forwards compatible.

- Schema validation errors that find an unknown attribute, element or attribute value in a **JDF** with a version that is higher than the schema SHOULD NOT lead to an abort.
  - A device or controller that reads a **JDF** with an element or attribute or attribute value with a version that is higher than the version that it was developed for SHOULD attempt to execute the **JDF** if *@SettingsPolicy* = "BestEffort".

- A device or controller that reads a **JDF** with an element or attribute or attribute value with a version that is higher than the version that it was developed for SHALL NOT execute the **JDF** if @*SettingsPolicy* = "MustHonor".
- Implementations SHOULD handle non-fatal version schema validation errors gracefully.
  - Unknown attributes/elements in the **JDF** namespace SHOULD be treated the same as foreign namespace attributes/elements when handling nodes that are not executed by the device or controller.
  - Unknown versions of the **JDF** namespace SHOULD be treated analog to foreign namespace elements when handling nodes that are not executed by the device or controller.

Life Cycle

Introduction

This chapter describes the life cycle of a **JDF** Job, from creation through modification to processing. Information is provided about the spawning of individual steps of Jobs and in what way they are merged into the Job once the Process step is completed.

4.1 Creation and Modification

The life cycle of a **JDF** Job will likely follow one of two scenarios. In the first scenario, a Job is created all at once by a single Agent and then is consumed by a set of Devices. More often, however, a Job is created by one Agent and is then transformed, or modified, over time by a series of other Agents. This Process might require specification of Product Intent, which is defined in ‣ Section 4.1.1 Product Intent Constructs.

Jobs can be modified in a variety of ways. In essence, any Job is modified as it is executed, since information about the execution is logged. Another instance of modification of a **JDF** Job, however, occurs during processing when more detailed information is learned or understood and then added along the way. This information might be added because an Agent knows more about the processing needed to achieve some result specified in a **JMF** Node than the original, creating Agent knew. For example, one Agent might create a Product Intent Node that specifies the Product Intent of a series of pages. This Product Intent Node might include information about the number of pages and the paper properties. Another Node might then be inserted that includes a Resource describing how the pages are to be RIPed. Later, another Agent might provide more detail about the RIPing Process by appending optional information to the RIP ***Resource***.

Regardless of where in the life cycle they are written, Nodes and their Resources SHALL be valid and include all REQUIRED information in order to have a @*Status* of "Ready" (in case of Nodes) or "Available" (in case of Resources). This restriction allows for the definition of incomplete Output Resources. For example, a URL Resource without a file name might be completed by a Process. On the other hand, it is impossible to define a valid and executable Node with insufficient input parameters.

Once all of the inputs and parameters for the Process requested by a Node are completely specified, a Controller can route the **JDF** Job containing this Node to a Device that can execute the Process. When the Process is completed, the Agent/Controller in charge of the Device will modify the Node to record the results of the Process.

4.1.1 Product Intent Constructs

JDF Jobs, in essence, are requests made by customers for the production of quantities of some product or products. In other words, a Job begins with a particular goal in mind. In **JDF**, product goals are often specified by using a construct called "Product Intent" and represented by ***Product Intent Nodes***. In contrast to Process Resources that define precise values, ***Product Intent Nodes***. allow ranges or sets of preferred values to be specified. Resources of this kind include ***ColorIntent***, ***FoldingIntent***, ***MediaIntent*** and ***ShapeCuttingIntent***, all of which are described in ‣ Chapter 8 Resources.

Product Intent

"Product Intent" is another way of saying "Job Specifications". Rather than describing how a Job will be made, Product Intent describes what a finished product (or some aspect of a product) will look like when it is completed. Product Intents can initiate with the customer and in rather vague terms, and they might be later fleshed out or completed by a printer's customer service representative, estimating department or production planners.

The Product Intent of a Job is like a blue print of a product. The blue print might be extremely vague, detailing only the general goal, or it might be very specific, stipulating the specific requirements inherent in meeting that goal. Product Intent might be defined for an end product about which little is known or about which the processing details for the Job are entirely unknown. Product Intent constructs also allow Agents to describe Jobs that comprise multiple product components and that might share some parts.

The initiating Agent of a Job specifies either Product Intent or a full set or Processes. The various kinds of Process Nodes are described in ‣ Section 3.3.1 Product Intent Nodes, ‣ Section 3.3.2 Process Group Nodes and ‣ Section 3.3.3 Combined Process Nodes. Any Job that specifies Product Intent SHALL include Nodes whose @*Type* = "Product". This representation is described in the following section.

4.1.1.1 Representation of Product Intent

The product description of a Job is a hierarchy of Product Intent Nodes, and the bottom-most level of the product hierarchy represents portions of the product that are each homogeneous in terms of their materials and formats. All Nodes below these Product Intent Nodes begin specifying the Processes needed to produce the products.

Product Intent Nodes are REQUIRED to contain only one thing, and that is a Resource that represents the physical result specified by the Node. This Resource is generally a ***Component***. In addition, somewhere in the hierarchy of Product Intent Nodes, it is a good idea to include an ***Intent Resource*** to describe the characteristics of the intended product. Although these are the only Resources that SHOULD occur, Product Intent Nodes can contain multiple Resources. For example, some Resource types, such as ***LayoutIntent*** and ***MediaIntent***, are defined to provide more general mechanisms to specify Product Intent. The resulting product of a Product Intent Node is specified as an output ***Component*** Resource of the Product Intent Node.

In some cases, more than one high level Product Intent Node will use the output of a Product Intent Node. These high level Nodes represent the combination of homogeneous product parts. In this case, the @*Amount* Attribute of the ***ResourceLink*** Elements that connect the Nodes will identify how the lower level product is shared.

4.1.1.2 Representation of Product Binding

Some ***Intent Resource***s, such as ***BindingIntent*** or ***InsertingIntent***, define how to combine multiple products. To accomplish this, the respective Component Resources SHALL be labeled according to their usage. For example, the Cover and Insert of a product are identified by ***ComponentLink***/@*ProcessUsage* Attribute of the respective input ***ComponentLink*** Elements. For more information about Product Intent, see ▸ Section 3.3.1 Product Intent Nodes.

4.1.2 Specification of Delivery of End Products

A Job can define one or more products and specify a set of deliveries of end products. To accomplish this, a Node **JMF** [@*Type* = "Product"] is created to define each product to be produced. The root Product Intent Node SHOULD contain a ***DeliveryIntent*** Resource that specifies a set of ***DropItemIntent*** Elements. Each ***DropItemIntent*** Element has a common delivery address and time, and a set of ***DropItemIntent*** Elements that specifies the amount of individual ***Component*** Elements to deliver to this address. Quote generation as defined in the previous chapter includes the specification of delivery addresses. For more information, see ▸ Section 6.2.4 Delivery.

4.1.3 Specification of Process Specifics for Product Intent Nodes

Product Intent Nodes are designed to represent a customer's view of the product. In some instances, a knowledgeable customer might want to specify production details that are only available in **JDF** Process Resources for a given product. Examples include scanning or screening parameters. This customer will still have no knowledge or control of the Process workflow. and therefore is expected to specify only the ***Resource*** Elements.

Individual **JDF** Process Resources MAY be referenced from the ***ProductionIntent*** Resource. ***Resource***/@*Status* will most likely be "Incomplete" because generally the customer does not know all parameters of the ***Resource***.

Example 4.1: Product Intent Node

The **highlighted tags** and **highlighted attributes** section shows how specific information about screening is specified in an intent node.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Job1" JobID="J1"
JobPartID="P1"
Status="Waiting" Type="Product" Version="1.4">
       <ResourcePool>
              <Component Amount="10000" Class="Quantity"
DescriptiveName="Complete 16-page Brochure" ID="Link0003"
Status="Unavailable" ComponentType="Sheet" />
              <LayoutIntent Class="Intent" ID="Link0004" Status="Available">
                     <Dimensions DataType="XYPairSpan" Preferred="612 792"
Range="576 720 ~ 648 864"/>
                     <Pages DataType="IntegerSpan" Preferred="16"/>
              </LayoutIntent>
              <MediaIntent Class="Intent" ID="Link0005" PartIDKeys="Option"
Status="Available">
                     <FrontCoatings DataType="EnumerationSpan" Preferred="None"/>
                     <MediaIntent Option="1">
                            <FrontCoatings DataType="EnumerationSpan" Preferred="Glossy"/>
                     </MediaIntent>
                     <BackCoatings DataType="EnumerationSpan" Preferred="None"/>
              </MediaIntent>
              <ProductionIntent Class="Intent" ID="ID_PI" Status="Available">
                     <ScreeningParamsRef rRef="ScreenID"/>
              </ProductionIntent>
              <ScreeningParams Class="Parameter" ID="ScreenID" Status="Incomplete">
                     <ScreenSelector ScreeningFamily="My favorite screen"
SpotFunction="Ellipse"/>
              </ScreeningParams>
       </ResourcePool>
       <ResourceLinkPool>
              <ComponentLink Usage="Output" rRef="Link0003"/>
              <LayoutIntentLink Usage="Input" rRef="Link0004"/>
              <MediaIntentLink Usage="Input" rRef="Link0005"/>
              <ProductionIntentLink Usage="Input" rRef="ID_PI"/>
       </ResourceLinkPool>
</JDF>
```

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Job1" JobID="J1"
JobPartID="P1"
Status="Waiting" Type="Product" Version="1.4">
      <ResourcePool>
            <Component Amount="10000" Class="Quantity"
DescriptiveName="Complete 16-page Brochure" ID="Link0003"
Status="Unavailable" ComponentType="Sheet" />
            <LayoutIntent Class="Intent" ID="Link0004" Status="Available">
                  <Dimensions DataType="XYPairSpan" Preferred="612 792"
Range="576 648 720 864"/>
                  <Pages DataType="IntegerSpan" Preferred="16"/>
            </LayoutIntent>
            <MediaIntent Class="Intent" ID="Link0005" PartIDKeys="Option"
Status="Available">
                  <FrontCoatings DataType="EnumerationSpan" Preferred="None"/>
                  <MediaIntent Option="1">
                        <FrontCoatings DataType="EnumerationSpan" Preferred="Glossy"/>
                  </MediaIntent>
                  <BackCoatings DataType="EnumerationSpan" Preferred="None"/>
            </MediaIntent>
            <ProductionIntent Class="Intent" ID="ID_PI" Status="Available">
                  <ScreeningParamsRef rRef="ScreenID"/>
            </ProductionIntent>
            <ScreeningParams Class="Parameter" ID="ScreenID" Status="Incomplete">
                  <ScreenSelector ScreeningFamily="My favorite screen"
SpotFunction="Ellipse"/>
            </ScreeningParams>
      </ResourcePool>
      <ResourceLinkPool>
            <ComponentLink Usage="Output" rRef="Link0003"/>
            <LayoutIntentLink Usage="Input" rRef="Link0004"/>
            <MediaIntentLink Usage="Input" rRef="Link0005"/>
            <ProductionIntentLink Usage="Input" rRef="ID_PI"/>
      </ResourceLinkPool>
</JDF>
```

4.2 Process Routing

A Controller in a **JDF** workflow system has two tasks. The first is to determine which of the Nodes in a **JDF** document are executable, and the second is to route these Nodes to a Device that is capable of executing them. Both of these procedures are explained in the sections that follow.

In a distributed environment with multiple Controllers and Devices, finding the right Device or Controller to execute a specific Node might be a non-trivial task. Systems with a centralized, smart master Controller might want to route Jobs dynamically by sending them to the appropriate locations. Simple systems, on the other hand, might have a static, well defined routing path. Such a system might, for example, pass the Job from hot folder to hot folder. Both of these extremes are valid examples of **JDF** systems that have no need for additional routing metadata.

In order to accommodate systems between these extremes, the ***NodeInfo*** Resource of a Node contains OPTIONAL *@Route* and *@TargetRoute* Attributes that let an Agent define a static Process route on a Node-by-Node basis. ***JMF***/***QueueSubmissionParams***/*@ReturnURL* takes precedence over ***NodeInfo***/*@TargetRoute* of the **JDF** node that is processed. If no *@Route* or *@TargetRoute* Attribute is specified and if a Controller has multiple options where to route a Job, it is up to the implementation to decide which route to use.

The Controller or Device reading the **JDF** Job is responsible for processing the Nodes. A Device examines the Job and attempts to execute those Nodes that it knows how to execute, whereas a Controller routes the Job to the next Controller or Device that has the appropriate capabilities.

4.2.1 Determining Executable Nodes

In order to determine which Node to execute, the Controller/Device SHALL use the following procedures.

1 It searches the **JDF** document for Node types that it can execute or Gray Boxes that it can expand by comparing the *@Type* and *@Types* Attributes and possibly the *@Category* Attribute of the Node to its own capabilities and by determining the *@Activation* of the Nodes. It SHOULD also verify that the *@Status* of the Node or the respective ***NodeInfo***/*@NodeStatus* is either "Waiting" or "Ready". If a ***Device*** Resource is specified as input to the Node, the Resource SHALL match the Controller/Device. Devices MAY opt to limit the scope of the Node search. The limitations SHOULD be specified in the Device capability description by appropriately setting ***DeviceCap***/*@ExecutionPolicy*.

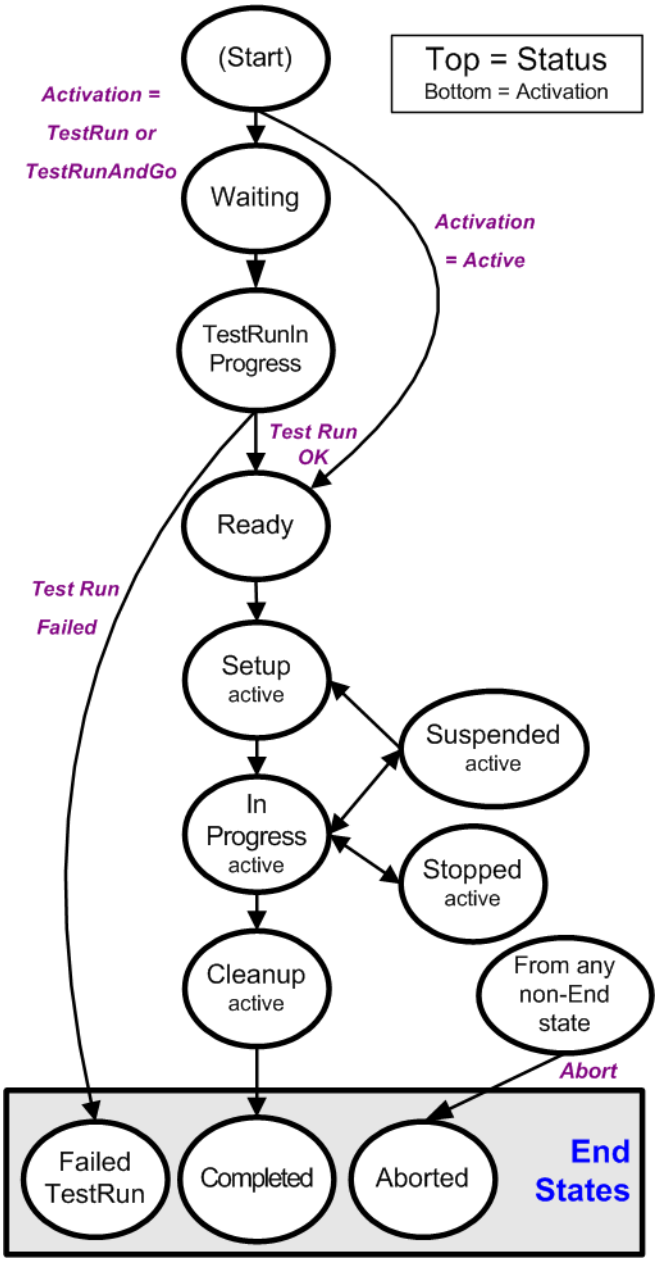
2 The controller/device can then determine if no resources have a @*Status* of "Incomplete" or a @*SpawnStatus* of "SpawnedRW". It SHOULD also determine if all of the input resources of the respective nodes have a @*Status* of "Available" and that all Processes that are attached through pipes are ready to execute. A controller MAY skip these checks and expect the lower level controller or device that it controls to perform this step and return with an error if it fails.

3 If scheduling information is provided in the ***NodeInfo*** resource, the specified start and/or end time SHALL be taken into account by the executing device. If no process times are specified, it is up to the device in charge of queue handling to execute the Process Node.

4 If no executable nodes are found, the device SHALL return the node to the controller. A ***Notification*** Audit Element with ***Notification***/@*Class* = "Error" SHOULD be appended to the ***AuditPool*** of the root **JDF** node. ***Notification***/***Error***/@*ReturnCode* = "102" specifies that no executable Node was found.

The Node will go through various states during its life time as is described in ▸ Figure 4-1: Life Cycle of a JDF Node

Figure 4-1: Life Cycle of a JDF Node



4.2.2 Distributing Processing to Work Centers or Devices

JDF syntax supports two means of distributing Processes to work centers or Devices. Its first option is to use a "smart" Controller that has the ability to parse a **JDF** Job and identify individual Processes or Process Groups that might be distributed to a particular Work Center or Device. This smart Controller MAY use spawning and merging facilities to subdivide the Job ticket and pass specific instructions to a Work Center or Device.

The second option, which is applicable when the Controller being used isn't smart, is to employ a simple Controller implementation that routes the entire Job to each workcenter or Device, thus leaving it up to the recipient to determine

which processing it can accomplish. For this option to work, each **JDF**-capable Device SHALL be able to identify Process Nodes it is capable of executing. Furthermore, each Device SHALL have sufficient **JDF**-handling capabilities to identify Processes that are ready to run.

4.2.3 Device / Controller Selection

The method used to determine which is the appropriate Device or lower level Controller to use to execute a given Node depends greatly on the implemented workflow being used. Although **JDF** provides a method for storing routing information in the *@Route* Attribute of the ***NodeInfo*** Resource of a Node, it does not prescribe any specific routing methods. However, some of the tools available to figure out alternative workflows are described below.

Knowledge of the capabilities of lower level Controllers/Devices either MAY be hard-wired into the system or gained using the ***KnownDevices*** Message. Since **JDF** does not yet provide mechanisms to determine if a given Device is capable of processing a Node without actually performing a test run, a Controller SHALL either have a prior knowledge of the detailed capabilities of its controlled Devices or perform a test run to determine if a Device is capable of executing a Node. Furthermore, in addition to the explicit routing information in the *@Route* Attribute of the ***NodeInfo*** Resource of a Node, **JDF** MAY contain implicit routing information in the form of ***Device ImplementationResource***s.

JMF defines the ***KnownDevices*** Query Message to find Controllers and Devices. The information provided by this query can be used by a Controller to infer the appropriate routing for a Node. In a system that does not support messaging, this information will be provided outside of **JDF**.

4.3 Execution Model

JDF provides a range of options that help Controllers tailor a processing system to the needs of the workflow and of the Job itself. The following sections explain the ways in which Controllers execute processes using these various options.

The processing model of **JDF** is based on a producer/consumer model, which means that the sequencing of events is controlled by the availability of Input Resources. As has been described, Nodes actboth as producers and consumers of Resources. When all necessary inputs are available in a given Node, and not before, the Process can execute. The sequence of processing, therefore, is implied by the chain of Resources in which the Output Resources of one Node become the Input Resources of a subsequent Node.

JDF supports four kinds of Process sequences: serial processing, overlapping processing, parallel processing and iterative processing. All four are described in the following sections.

4.3.1 Serial Processing

The simplest kind of Process routing, known as serial processing, executes Nodes sequentially and with no overlap. In other words, no Nodes are executed simultaneously. Once the Process has acted upon the Resource in some way, the Resource availability is described by the *@Status* Attribute of the Resource, as described above. When the Process state is "Ready" or "Waiting", the Process can begin executing.

In a workflow using serial processing, the Controller is responsible for comparing the actual amount available with the specified amount in the corresponding ***ResourceLink*** Element to determine whether or not the Input Resource can be considered available. If no amount is specified in the ***ResourceLink***, the Process is assumed to consume the entire ***PhysicalResource***.

Figure 4-2: Example of a simple Process chain linked by Resources

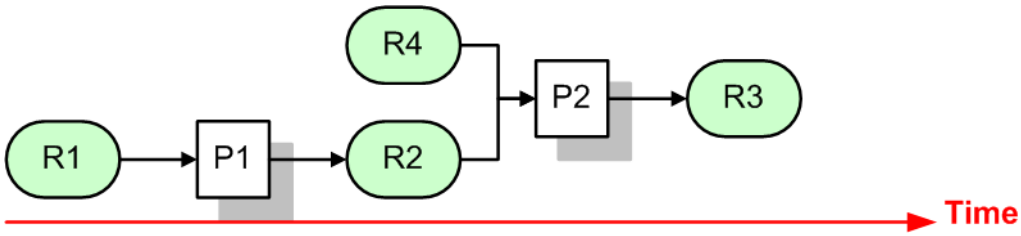


Figure 4-3: Example of a simple Process chain linked by Resources

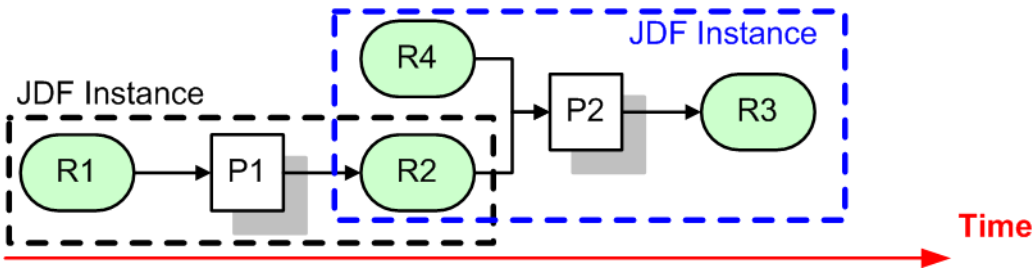


Figure 4-2: Example of a simple Process chain linked by Resources depicts a simple Process chain that produces and consumes ***Quantity Resource***s and uses an ***ImplementationResource***. The Resources R1, R2 and R3 represent ***Quantity Resource***s. Process P1 consumes Resource R1 and produces Resource R2. R2 is then completely consumed by P2, which also requires the ***ImplementationResource*** R4 for processing. Process P2 uses these two Resources and produces Resource R3. All of this is accomplished along a linear time axis.

Table 4.1 Examples of Resource and Process states in the case of simple Process routing shows the value of the @*Status* Attribute of each of the Resources and Processes used in Figure 4-2: Example of a simple Process chain linked by Resources. The time axis runs from left to right both in Figure 4-2: Example of a simple Process chain linked by Resources and in Table 4.1 Examples of Resource and Process states in the case of simple Process routing. Note that no Process can execute until all Resources leading up to that Process are "Available". In other words, the Job executes serially and sequentially. For more information about the values of the @*Status* Attribute of Resources, see Table 3.10 Abstract Resource Element . For more information about the values of the @*Status* Attribute of Processes, see Table 3.4 JDF .

Table 4.1: Examples of Resource and Process states in the case of simple Process routing

OBJECT STATUS	BEFORE RUNNING P1	DURING RUNNING P1	AFTER RUNNING P1, BEFORE P2	DURING P2	AFTER P2
Resource R1	Available	InUse	Unavailable	Unavailable	Unavailable
Resource R2	Unavailable	Unavailable	Available	InUse	Unavailable
Resource R3	Unavailable	Unavailable	Unavailable	Unavailable	Available
Resource R4	Available	Available	Available	InUse	Available
Process P1	Waiting or Ready	InProgress	Completed	Completed	Completed
Process P2	Waiting or Ready	Waiting or Ready	Waiting or Ready	InProgress	Completed

If a Process aborts before completion, its Output Resources are "Unavailable" unless the output has been partially produced in which case the Device MAY update the amount and set the output to "Available".

When the @*Amount* Attribute is used in connection with the quantifiable Resources R1, R2 or R3 and their links, then the Controller SHALL decide whether or not a Resource is available by comparing the individual values. If the amounts are used to define the availability, then the Resource @*Status* MAY be set to "Available" for all ***Quantity Resource***s. Note that when the value of the @*Status* Attribute of the Resource is "Unavailable", the Resource is not available even if a sufficient @*Amount* is specified.

If amounts are specified in the Resource Element, they represent the actual available amount. If they are not specified, the actual amount is unknown, and it is assumed that the Process will consume the entire Resource. Amounts of ***ResourceLink*** Elements SHALL be specified for Output Resources that represent the intended production amount. The specification of the @*Amount* Attribute for Input Resources is OPTIONAL. For details, see Section 3.10.4 Resource Amount . If the Controller cannot determine the amounts, this constitutes a **JDF** content error, which is logged by error handling. This Process is described in Section 4.6 Error Handling.

If a Process in a serial processing run does not finish successfully, the final Process status is designated as "Aborted". In an aborted Job, only a part of the intended production might be available. If this occurs, the actual produced amount is logged into the ***AuditPool*** by a ***ResourceAudit*** Element.

4.3.2 Partial Processing of Nodes with Partitioned Resources

New in JDF 1.2

JDF nodes themselves SHALL NOT be partitioned, although the input and output resources MAY be partitioned. If the input and output ***ResourceLink*** Elements reference one or more individual Partitions, the **JDF** node executes using only the referenced Resources.

If multiple Input Resources are input to a process, the Resource with the highest granularity defines the Partitioning. For instance, a **ConventionalPrinting** Process might consume a non-Partitioned ***ConventionalPrintingParams*** and a set of ***Ink*** and ***ExposedMedia*** (Plate) Resources that are Partitioned by "Separation". The Partition granularity will be defined by the ***Ink*** and ***ExposedMedia*** (Plate) Resources to be "Separation". The "Separation" Partition set is defined by the superset of all defined Partition Key values. If the "Separation" key values of ***Ink*** were "Black" and "Varnish", and the @*Separation* key values of ***ExposedMedia*** (Plate) were "Black", the resulting set is "Black" and "Varnish".

The Partition Keys of both input and output restrict the Process. If the Partition Keys are not identical, both SHALL be applied to restrict the Node. If the Partition Keys are non-overlapping (e.g., in an **Imposition** Node where a ***RunList*** based

input Partition is mapped to a Sheet based output Partition), the application SHALL explicitly calculate the result. The following examples in ▸ Table 4.2 Examples of Partitioning across multiple Resources illustrate the restriction algorithms:

Table 4.2: Examples of Partitioning across multiple Resources

INPUT PARTITION 1	INPUT PARTITION 2	OUTPUT PARTITION	NODE PARTITION	DESCRIPTION
@*SheetName* = "S1"	—	—	@*SheetName* = "S1"	If only the input is Partitioned, the Node Partition is defined by the input.
@*SheetName* = "S1" @*Separation* = "Cyan"	—	—	@*SheetName* = "S1" @*Separation* = "Cyan"	If only the input is Partitioned, the Node Partition is defined by the input.
@*SheetName* = "S1" @*Separation* = "Cyan"	@*Separation* = "Cyan" + @*Separation* = "Black" (@*PartUsage* = "Implicit")	—	@*SheetName* = "S1" @*Separation* = "Cyan" + @*SheetName* = "S1" @*Separation* = "Black"	The first input is Partitioned by @*SheetName* and @*Separation* which defines the Partition Key granularity. The second input is Partitioned by @*Separation* only but has an implied @*SheetName* and has a larger but overlapping set of separation values. The separation value set is therefore defined by the second key.
@*SheetName* = "S1"	—	@*SheetName* = "S1" @*Separation* = "Cyan"	@*SheetName* = "S1" @*Separation* = "Cyan"	The input and output base Partitions are identical. The output further restricts the Partition.
@*SheetName* = "S1"	—	@*SheetName* = "S2" @*Separation* = "Cyan"	Error	Input and output are not overlapping. This specifies the null set.
@*SheetName* = "S1" @*Separation* = "Magenta"	@*Separation* = "Cyan" + @*Separation* = "Black"	—	Error	This is an error and defines the null set. The first input is Partitioned by @*SheetName* and @*Separation* which defines the Partition Key granularity. The second input is Partitioned by @*Separation* only and has a larger but non-overlapping set of separation values. The separation value set is therefore the null set.
@*SheetName* = "S1" @*Separation* = "Cyan"	@*Separation* = "Cyan" + @*Separation* = "Black" (@*PartUsage* = "Explicit")	—	Error	The first input is Partitioned by @*SheetName* and @*Separation* which defines the Partition Key granularity. The second input is Partitioned by @*Separation* only but has no implied @*SheetName* and therefore has a non-overlapping set of Partition Keys. The separation value set is therefore defined by the second key.
@*RunIndex* = "0 ~ 7"	—	@*SheetName* = "s2"	Special	This specifies Sheet s2, with all ***PlacedObject*** Elements with an @*Ord* in the range of 0 to 7. This special case is important when ***RunList*** entries occur multiply on different imposition Sheets.

4.3.3 Overlapping Processing Using Pipes

Whereas pipes themselves are identified in the Resource that represents the pipe by specifying ***Resource***/@*PipeID*, pipe dynamics are declared in the ***ResourceLink*** Elements that reference the pipe. This allows multiple Nodes and Devices to access one pipe, each of them with its own pipe buffering parameters.

In some situations, Resource linking is a continuous, dynamic Process rather than a predefined static Process. In other words, one Process might require the Output Resources of another Process before that Process has completely finished producing them. The ability to accomplish this kind of Resource transfer is known as overlapping processing, and it is accomplished with the use of a mechanism known as pipes. Pipes are considered to be **active** if any Process linking to the pipe simultaneously consumes or produces that pipe Resource.

Any Resource MAY be transformed into a pipe Resource by specifying the @*PipeID* Attribute in the Resource. Pipes resemble reservoir containers that hang between Processes. Processes connected to the pipe via output links fill the container with necessary Resources, while Processes connected via input links deplete it (see ▸ Figure 4-4: Example of a Pipe Resource Linking Two Processes via Pull and ▸ Figure 4-5: Example of a Pipe Resource Linking Two Processes via Push). The level is controlled by the ***ResourceLink*** Attributes @*PipeResume* and @*PipePause* (see ▸ Table 3.16 ResourceLink Element and ▸ Table 3.19 PartAmount Element). The unit of the buffers is defined by the @*Unit* Attribute of the Resource.

Pipe Resources

A pipe Resource is simply an input to a Process that can be exhausted and can be replenished. Examples might include rolls of paper feeding into a press, ink well levels, fountain solution, or even proofing stock loaded into a proofer.
Another type of pipe Resource in everyday use is a “hot-folder” or “watched file.” Hot folders are used to automate functions such as preflighting. When a file is saved to a hot-folder, the system knows to automatically apply a defined Process to the new file. When the folder is empty the processing stops.

The two following diagrams show the ways in which pipes mediate between the Process producing the Resource and the Process consuming the Resource. The following OPTIONAL Attribute Values are defined for pipes:

ResourceLink/@*PipePartIDKeys* – specifies the granularity of a pipe request for partitioned resources.

ResourceLink/@*PipePause* – specifies at which resource level to pause a pipe.

Resource/@*PipeProtocol* – specified the protocol to use to pause, resume and initiate a pipe.

ResourceLink/@*PipeResume* – specifies at which resource level to resume a pipe.

The specified value of each of these Attributes in any given ***ResourceLink*** dictates the levels at which a pipe SHOULD resume or pause execution. ▸ Figure 4-6: Example of status transitions in case of overlapping Processing gives an example of a view on the dynamics of a pipe Resource. The available level of the pipe Resource, represented as R2, and the availability status of two entity Resources, represented as R1 and R3, are changing along a time line. Below the progressions of these Resources is the status of two Processes — P1 and P2. P1 represents the Process producing the pipe Resource and P2 represents the Process consuming that Resource. The Resource status of a active pipe, represented here as R2, is defined to be @*Status*= "InUse" (see also ▸ Table 3.10 Abstract Resource Element).

Figure 4-4: Example of a Pipe Resource Linking Two Processes via Pull

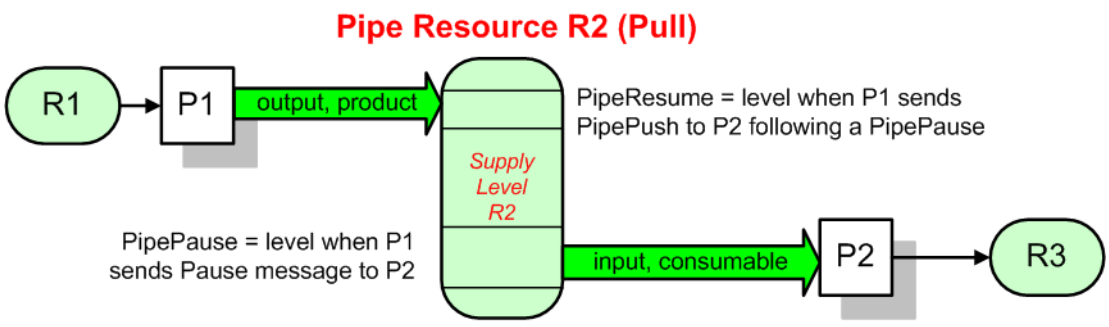


Figure 4-5: Example of a Pipe Resource Linking Two Processes via Push

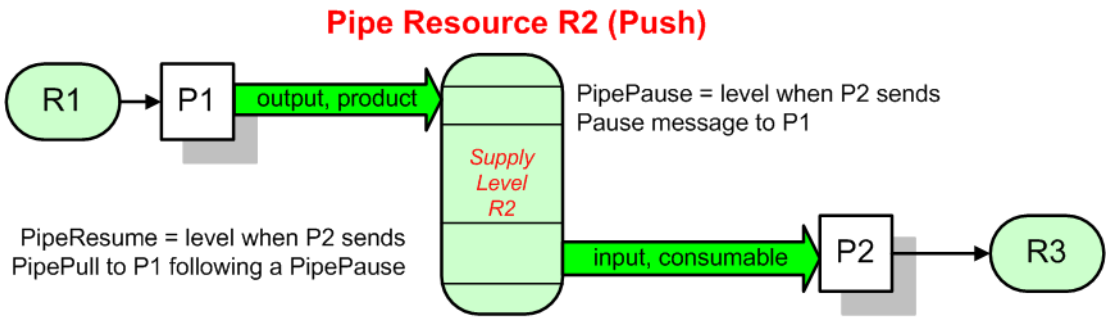
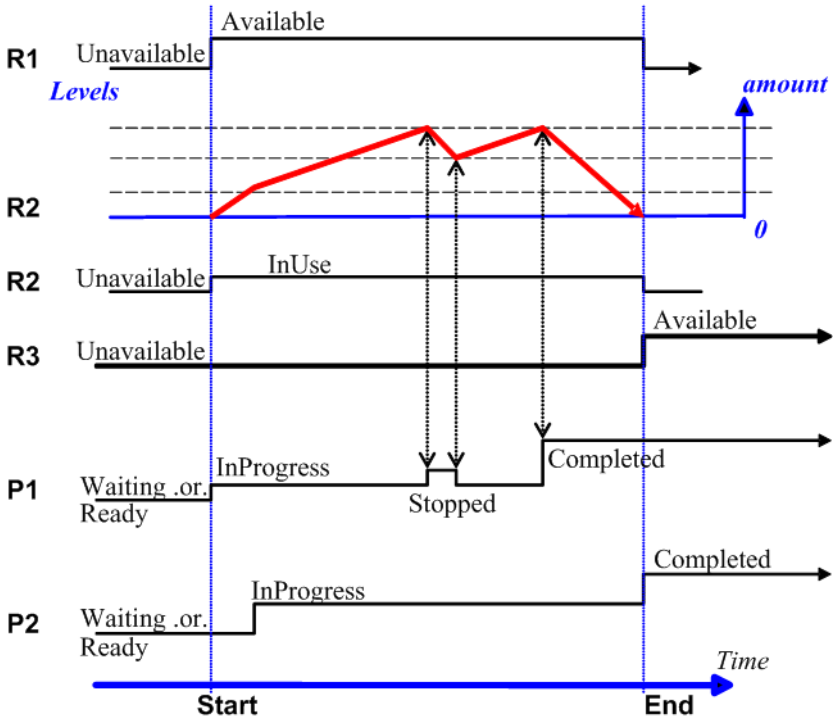


Figure 4-4: Example of a Pipe Resource Linking Two Processes via Pull and Figure 4-5: Example of a Pipe Resource Linking Two Processes via Push are views on the structure and Figure 4-6: Example of status transitions in case of overlapping Processing a view on the dynamics of the pipe example considered here. R1 represents an Input Resource for P1, which feeds into the intermediate pipe Resource R2. Once the container R2 is filled to the predetermined level, it is used as the Input Resource for P2, which in turn produces Output Resource R3.

Figure 4-6: Example of status transitions in case of overlapping Processing



Resource linking through pipes is controlled through the specification of the @*PipePause* and @*PipeResume* Attributes. The intended amount of a Resource MAY be specified in advance in the output ***ResourceLink***. Whenever the level representing the available quantity of the pipe Resource exceeds the @*PipePause* level of the ***ResourceLink***, the Process P1 is halted (@*Status* = "Stopped") so that the Process does not overproduce. Once the level falls below the @*PipeResume* value, the Process P1 resumes execution. P1 is completed when it has produced the intended amount. Once P1 has performed its task, the Resources still in the pipe are consumed by the subsequent Process without level control. In other words, after a Process filling a pipe buffer has completed, pipe buffering becomes disabled.

In the case of output ***ResourceLink*** Elements, the @*PipeResume* value SHALL be smaller than the @*PipePause* value, whereas in the case of input ***ResourceLink*** Elements, the @*PipeResume* value SHALL be greater than the @*PipePause* value. If @*PipePause* is specified for a ***ResourceLink*** and @*PipeResume* is not specified, the related Process might run into a deadlock state. In other words, the Process stops and cannot resume execution automatically. Once a Process is stopped under these circumstances it can only be resumed manually or by sending a pipe control Message for resumption that allows interconnected execution control (halting and resumption of Processes by pipe control Messages is described in Section 5.10 Messages for Pipe Control). If the Attributes @*PipeResume* or @*PipePause* of ***ResourceLink*** Elements to pipe Resources are not specified, the Controller is responsible when the linked Processes start and stop independent of the level.

4.3.3.1 Dynamic Pipes

In addition to abstractly declaring pipe properties, **JMF** provides pipe Messages that allow dynamic control of pipes. Dynamic pipes can be used to model situations where the amount of Resources is not known beforehand but becomes known during processing. An example of this behavior is a long press run where new plates are needed during a press run because of quality deterioration. The exact point in time where quality becomes unacceptable is not predetermined and might even vary from separation to separation. Dynamic pipes provide the flexibility to adjust to changing situations of this nature.

Another usage of dynamic Pipes is linking the output of a variable data print job to various components. Examples include a Pipe describing the ***RunList*** that links the RIP to a print engine or a Pipe describing the ***Component*** that links the printer to finishing equipment or individual finishing devices. In this case, the ***RunList*** and ***Component*** are templates that are logically expanded in increments by the Pipe messages.

Dynamic pipes provide a ***ResourceLink***/@*PipeURL* Attribute that allows dynamic requests for a status change of the pipe while a Process is executing. Dynamic requests use **JMF** pipe control Messages (see ▸ Section 5.10 Messages for Pipe Control) sent to another Controller whose URL address is specified by the @*PipeURL* Attribute of the respective ***ResourceLink***. Depending on the values of the ***Resource***/@*PipeProtocol* Attribute, the following actions are possible.

- "JMFPull": The consumer initiates the pipe by sending a ***PipePull*** Message to its ***ResourceLink***/@*PipeURL*. The consumer MAY request new resources by sending ***PipePull*** Messages. If the producer reaches the pipe-pause (low water) mark, or is incapable of fulfilling ***PipePull*** Messages for other reasons such as a malfunction, it SHOULD send a ***PipePause*** Message to the consumer. Once it has reached the pipe-resume (high water) mark, or the malfunction has been removed, it SHOULD send a ***PipePush*** Message to the consumer to inform the consumer that it can commence sending ***PipePull*** Messages. The consumer SHOULD send a ***PipeClose*** Message to the producer if the consumer does not require any further Resources.
- "JMFPush": The producer initiates the pipe by sending a ***PipePush*** Message to its ***ResourceLink***/@*PipeURL*. The producer MAY dispatch new resources by sending ***PipePush*** messages. If the consumer reaches the pipe-pause (high water) mark, or is incapable of fulfilling ***PipePush*** requests for other reasons such as a malfunction, it SHOULD send a ***PipePause*** message to the producer. Once it has reached the pipe-resume mark (low water), or the malfunction has been removed, it SHOULD send a ***PipePull*** to the producer to inform the producer that it can commence sending ***PipePush*** messages.The producer SHOULD send a ***PipeClose*** Message to the consumer if the producer cannot provide any further Resources.

When dynamic pipes are used, @*PipeResume* and @*PipePause* define the buffering parameters that lead to a Pipe control message to the remote device. The pipe control Messages described later in ▸ Section 5.10 Messages for Pipe Control are designed to establish communication between Processes at both ends of dynamic pipe, even if the corresponding Processes are spawned separately.

The following table summarizes the actions to be taken when the buffer in a dynamic pipe reaches a certain level "L".

Table 4.3: Actions generated when a dynamic-pipe buffer passes various levels (Sheet 1 of 2)

PIPEPROTOCOL (SENDER)	SITUATION	MESSAGE	DESCRIPTION
JMFPull (consumer)	ready to process	***PipePull***	The consumer has processing or buffering available and therefore the producer SHOULD produce resources.
JMFPull (creator)	L < @*PipePause*	***PipePause***	The creator has no resources available and therefore the consumer SHALL refrain from sending ***PipePull*** messages.
JMFPull (creator)	L > @*PipeResume*	***PipePush***	Sufficient Resources have been produced by the creator and are ready for delivery to the consumer.
JMFPush (producer)	resources available	***PipePush***	Sufficient resources have been produced by the creator and are ready for delivery to the consumer. therefore the consumer SHOULD consume resources.
JMFPush (consumer)	L > @*PipePause*	***PipePause***	The consumer has no processing or buffering space available and therefor the creator SHALL refrain from sending ***PipePush*** messages.

Table 4.3: Actions generated when a dynamic-pipe buffer passes various levels (Sheet 2 of 2)

PIPEPROTOCOL (SENDER)	SITUATION	MESSAGE	DESCRIPTION
JMFPush (consumer)	L < @*PipeResume*	***PipePull***	The consumer has processing or buffering available and therefore the producer SHOULD commence sending ***PipePush*** Messages.

Dynamic pipes are initially dormant and SHALL be activated by an explicit request. If ***Resource***/@*PipeProtocol* = "JMF", dynamic pipe requests MAY be initiated by both ends of the pipe. As soon as the Pipe has been initiated, actions that are required by the impled @*PipeProtocol* ("JMFPush" or "JMFPull") SHALL be applied. For example, a print Process might notify an off-line finishing Process when a certain amount is ready by sending a ***PipePush*** Message, or the printing Process might request a new plate by sending a ***PipePull*** Message.

4.3.3.2 Pipes of Partitionable Resources

Pipes of Partitionable Resources MAY also define the granularity of the Resources that are considered to be one part by specifying the @*PipePartIDKeys* Attribute in the appropriate ***ResourceLink*** Element. For instance, a Partitioned **ImageSetting** Process could be defined for multiple Sheet separations, but a complete set containing all separations of both sides of a single Sheet would be sent to the pressroom as one pipe request. In this case, the value of ***ExposedMedia***/@*PartIDKeys* would be "SheetName Side Separation" and the value of the ***ResourceLink***/@*PipePartIDKeys* for the pipe would be "SheetName". The Resources specified in ***PipeParams*** SHOULD be reduced to only define the currently active parts. In the example above, only the selected @*SheetName* partition with all its @*Side* and @*Separation* partition leaves would be included in the message.

4.3.3.3 Example JMFPush Sequence

This section illustrates the concept of dynamic pipes using the example of variable data near line finishing being controlled by a variable data digital press.

The exchange resource is a ***Component*** that is the output of the **DigitalPrinting** Combined Node and the input of a Combined **Folding** and **Stitching** booklet maker. The actual **JMF** messages are provided in ‣ Section O.5.3 JMF XJMF Pipe Messages with example digital finishing.

Table 4.4: Event Sequence in Digital Finishing (Sheet 1 of 2)

DIRECTION	TYPE	PIPEPARAMS DESCRIPTION
P->C	***PipePush*** Sheet 0/1; Cover; Set 0	Initialize Pipe
P->C	***PipePush*** Sheet 0/5; Set 0	Next sheet
P->C	***PipePush***	Lots of next sheets
P->C	***PipePush*** Sheet 4/7; Body; Set 35	Next sheet
P<-C	***PipePause*** Sheet 4/7; Set 35	Paper Jam in finisher - destroys Set 34 and 35
P<-C	***PipePull*** Sheet 0; Set 34	Restart at Page 0 of Doc 34
P->C	***PipePush*** Sheet 0/3; Set 34	Restart Pipe
P->C	***PipePush***	Lots of next sheets
P->C	***PipePause*** Sheet 4/7; Cover; Set 122	Paper Jam in printer - destroys Set 122

Table 4.4: Event Sequence in Digital Finishing (Sheet 2 of 2)

DIRECTION	TYPE	PIPEPARAMS DESCRIPTION
P<-C	***PipePull*** Sheet 0; Set 122	Restart at Page 0 of Doc 34(optional)
P->C	***PipePush*** Sheet 0; Set 122	Restart at Page 0 of Doc 122
P->C	***PipePush***	Lots of next sheets
P->C	***PipePush*** Sheet 2/3; Body; Set 221	Last sheet - note zero based counting for index
P->C	***PipeClose***	Done

4.3.3.4 Comparison of Non-Dynamic and Dynamic Pipes

The ***ResourceLink*** between non-dynamic pipes provides the buffering parameters for the Process to which the ***ResourceLink*** belongs. Therefore, many Processes can link to the same pipe Resource. Furthermore, each Process has its own buffering parameters, whether it is a consumer or a producer. In order to control non-dynamic pipes, one master Controller SHALL control all Processes linked to the pipe Resource.

In contrast, dynamic pipes provide a URL address to control a Process at the other pipe end. Then the buffering parameters of the ***ResourceLink*** control the Process at the other end. In the case of dynamic pipes, no master Controller is needed to control the pipe. Control is accomplished by sending pipe Messages. If pipe Resources are linked to multiple consumers or producers, such as two finishing lines that consume the output of one press one palette at a time, it is up to implementation to ensure consistency of the Processes.

4.3.3.5 Metadata in Pipe Messages

PipeParams/***Resource*** can contain metadata that is required by the recipient of the message. This metadata SHALL be specified as partition keys in ***ResourceLink***/***Part*** and additional details MAY be specified as the actual contents of the ***Resource***. Partition key metadata provides a mechanism to retain context in large variable data jobs without requiring fully fleshed out partitioned ***Resource***s in the **JDF**.

A typical example of partition key metadata is ***Part***/@*DocIndex*, ***Part***/@*RunIndex* and ***Part***/@*Side* to uniquely identify the context of a surface image that is sent from a RIP to a digital press.

4.3.4 Parallel Processing

While serial processing assumes that all Resources will be produced and consumed in a linear fashion, and while overlapping processing uses multiple Processes that work together to use and create Resources, there are times when it makes sense to run more than one Process simultaneously, creating a multi-pronged workflow. This kind of Process routing is known as parallel processing. Subsections of Jobs are spawned off so that Nodes can be executed individually and simultaneously by the appropriate Devices. Once the Processes are complete, the spawned Nodes are merged back into the original Job. The Output Resources of the merged Nodes become inputs for later Processes. For example, an insert could be produced independently of a cover, and both will be bound together later.

In parallel processing, Processes can be run in a coordinated parallel fashion by using independent Resources. An independent Resource is a Resource that is not shared between multiple Processes. ***ImplementationResource***s, for example, cannot be shared and are therefore always independent, and ***Consumable Resource***s and ***Quantity Resource***s can each be split to function as independent Resources. Individual Partitions of Partitionable Resources are independent and can be Processed in parallel. Read-only Resources, such as parameters, can be shared without any restrictions, and can, therefore, be used in read-only mode for parallel processing. Process chains created by the use of independent Resources are known as independent Process chains.

Parallel processing can proceed in one of two ways. Either a Controller can organize the **JDF** Nodes in a way that allows it to initiate parallel processing, or it can use the spawning-and-merging mechanism to field out chunks of the Job to execute simultaneously. If a Controller chooses the latter method, parent Nodes that contain independent Process chains can be spawned off and processed independently. For example, in order to improve production capacity, an Agent could split ***Consumable Resource***s and create independent Process chains in which each chain consumes its own Resource part. Afterwards, the Agent could submit one of the created Job Parts to a subcontractor and Process the other part with its own facilities.

Parallel processing is used only to process multiple aspects of a Job simultaneously; it is not used to process multiple copies of a **JDF** Job. In other words, a Job SHALL NOT be copied and sent to different Controllers for parallel processing. For more information about spawning of Jobs, see ▸ Section 4.4 Spawning and Merging.

4.3.5 Iterative Processing

Some Processes, especially in the prepress area of production, cannot be described as a serial or parallel set of Process steps. Instead, a set of interdependent Processes is iterated in a non-deterministic order. These Processes are known as iterative Processes. For example, an advertisement is laid out that requires a photographic image. During the layout phase, changes are to be made to the color settings of the image, which is then reinserted to the layout. Changes such as these can be described in a high level fashion by defining a Resource *@Status* Attribute of "Draft". As long as an Input Resource to a Process has a *@Status* of "Draft", the *@Status* of the Output Resource SHALL NOT be "Available".

The ***ResourceLink***/*@MinStatus* of a ***ResourceLink*** that links to a draft Input Resource SHALL be set to less than or equal "Draft" to state that a draft Input Resource is acceptable for a Process. Thus a prepress layout Process can be abstractly defined to work on draft Resources until an acceptable output has been achieved, but the output PDL file will not be used for printing until *@Status* is "Available" and no longer designated as a "Draft"

Iterative Processes can be set up in a formal fashion using dynamic pipes to convey parameter change requests or in an informal way that assumes that the operators of the various Processes have an informal communication channel. Both are described in greater detail below.

4.3.5.1 Informal Iterative Processing

Informal iterative processing does not require a complete redefinition of the Resources needed at every iteration. This kind of processing is generally used in a creative workflow where a Job is defined and gets refined in a series of steps until it is completed. The information about the changes is transferred through channels that bypass **JDF**. Nonetheless, the description of these Processes in **JDF** is useful for accounting purposes, as the status of each Process might be monitored individually.

The ***ResourceLink*** Elements for informal processing contain an additional *@MinStatus* Attribute which SHALL be set to "Draft", but in all other ways they are identical to the ***ResourceLink*** Elements used in simple sequential processing. Furthermore, the Nodes run through the same set of phases as they would in sequential processing. Nodes are designated only as "Stopped" and not as "Completed" after being processed for an iterative cycle. They are marked as completed after their Output Resources lose their *@Status* of "Draft".

4.3.5.2 Formal Iterative Processing

In formal iterative processing, all ***ResourceLink*** Elements between interacting Processes are dynamic pipes. Every request for a new Resource is initiated by a ***PipePush*** or ***PipePull*** Message that contains at least one ***Resource***Element with the updated parameters. This Resource is used by the Process, and the resulting new Output Resource can be consumed by the requesting Process. The *@Status* of "Draft" can be removed from a Resource by sending the creator a ***PipeClose*** Message that has the OPTIONAL *@UpdatedStatus* Attribute set to "Available". A Node can only reach a *@Status* of "Completed" if it has no remaining draft Resources. Another method to remove the draft status is to define a Node for an **Approval** Process that accepts draft Resources as inputs and has non-draft Resources representing the same entities as outputs.

4.3.6 Approval, Quality Control and Verification

In many cases, it is desirable to ensure that an executed Process or set of Processes have been executed completely and/or correctly. In the graphic arts industry this is verified by generating approvals and signing them. **JDF** allows modeling of the approval Process and modeling of the verification Processes by allowing an OPTIONAL ***ApprovalSuccess*** Input Resource in any Process.

The **Approval**, **QualityControl** and **Verification** Processes accept any Resource as input and output that Resource along with ***ApprovalSuccess*** Resource if approved. An ***ApprovalSuccess*** Resource SHALL NOT be set as "Available" unless it has been signed by an authorized person. For hard copy proofing, a Combined Process (e.g., ending with the **ImageSetting**, **ConventionalPrinting** or **DigitalPrinting** Process) generates the hard proof which is input to a separate **Approval** Process. For soft proofing, a Combined Process (ending with **Approval** Process) generates the soft proof which is approved by that **Approval** Process.

JDF provides a **QualityControl** Process to verify that the output of a Process fulfills certain quality criteria. This differs from the **Verification** Process, which verifies the completeness of a given set of Resources.

4.4 Spawning and Merging

JDF spawning is the process of extracting a **JDF** Subnode from a Job and creating a new, complete **JDF** document that contains all of the information needed to process the Subnode in the original Job. Merging is the process of recombining the information from a spawned **JDF** part with the original **JDF** Job, even after both documents have evolved independently. By using the mechanism for spawning and merging different parts of a Job, it is possible to submit Job Parts to distributed Controllers, Devices, other work areas or other work centers.

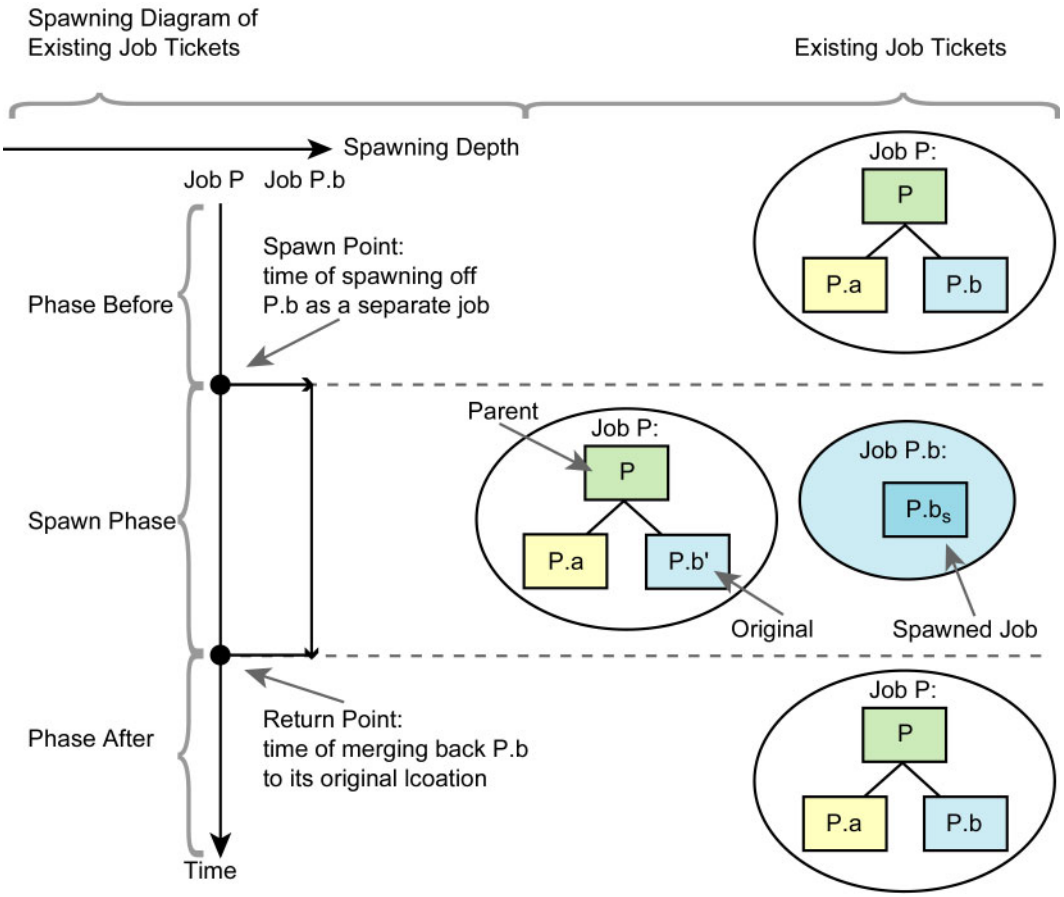
The **JDF** spawning-and-merging mechanism can be applied recursively, which means that subjects that have already been spawned can in turn spawn other sub-Subjobs and so on. However, a Node SHALL NOT be re-spawned. If a Node is to be spawned a second time, the previously submitted version SHALL first be deleted, and the spawning procedure SHALL be applied again to the original Node.

No matter how many Job Parts have been spawned, however, merging is realized by copying Nodes back to their original location and synchronizing the appropriate Resources. Therefore, each spawning SHALL be logged in the Job by the Agent performing the actions that result in a spawned **JDF** Node. Furthermore, in order to avoid inconsistent **JDF** states after merging, each merging SHALL be logged, or the appropriate ***Spawned*** Audit Element SHALL be removed from the ***AuditPool*** Element.

Figure 4-7: The spawning and merging mechanism and its phases shows, schematically, the spawning and merging of a Subjob, designated as P.b. The following three phases are defined on a demonstrational time scale.

1 The first phase occurs before the Subjob is spawned off.
2 The second phase occurs during the spawn phase, when the spawned Subjob is executed separately.
3 The third phase occurs after the spawned **JDF** Node has been merged back into the original **JDF** Job.

Figure 4-7: The spawning and merging mechanism and its phases



The three phases of the Job Part are bordered by the spawning point and the merging point. On a Job scale, denoted as spawning depth in Figure 4-7: The spawning and merging mechanism and its phases, one Job ticket exists during the phases before and after spawning, and the following two Job tickets exist during the spawning phase: the Job with the **parent** (P) of the **original JDF** part (P.b', also denoted as a Subjob) that has been spawned; and the **spawned JDF Node** ($P.b_s$) itself.

This section provides examples that outline the various ways in which spawning and merging can be applied. The following cases are considered in the next six sections.

1 Standard spawning and merging
2 Spawning and merging with Resource copying
3 Parallel spawning and merging of Partitioned Resources
4 Nested spawning and merging in reverse sequence
5 Spawning and merging of independent Job tickets
6 Simultaneous spawning and merging of multiple Nodes

JDF can support any combination of the cases described, but these six represent a cross-section of likely scenarios. Case one is the simplest of all of the cases and is occurs in every instance of spawning and merging, regardless of the circumstances surrounding the Process. Each subsequent case requires additional processing that builds upon the processing described in the cases that precede it.

4.4.1 Case 1: Standard Spawning and Merging

The actions described in this case SHALL be applied in every spawning and merging Process. All cases described in this chapter, as well as any other that might be invented, begin with these procedures.

Spawning

To indicate that a Process has been spawned, the @*Status* Attribute of the original **JDF** Node SHALL be set to the value "Spawned" (see ▸ Table 3.4 JDF). The @*Status* Attribute of the spawned Node remains unchanged.

A unique @*SpawnID* Attribute SHOULD be set in the spawned Node, and a copy of its value SHOULD be set in the @*NewSpawnID* of the newly created ***Spawned*** Audit Element. This simplifies bookkeeping of ***Audit*** Elements and merging in case a Node is multiply spawned, either due to error conditions or in parallel with individual Partitions. The value of @*SpawnID* SHOULD also be appended to the @*SpawnIDs* list of all spawned Resources.

In order to identify all of the ancestors of a Job that has been spawned, an ***AncestorPool*** Element is included in the Root Node of every spawned **JDF** Node. This Element contains an ***Ancestor*** Element that identifies every parent, grandparent, great-grandparent and so on of the spawned Subnode. In this way, the family tree of every spawned Node is tracked in an ordered sequence that allows an unbroken trace back through all predecessors. Consequently, the Elements that comprise the ***AncestorPool*** of a spawned **JDF** Node SHALL be copied into the ***AncestorPool*** Element of the newly spawned **JDF** Node before the ancestor information of the previously spawned **JDF** Node is appended to the ***AncestorPool*** Element of the newly spawned **JDF** Node. The last ***Ancestor*** Element in each ***AncestorPool*** is the parent, the second-to-last the grandparent and so on. ***NodeInfo*** and ***CustomerInfo*** Elements or refelements MAY be copied into the respective ***Ancestor*** Elements.

The complete ancestor information is REQUIRED in order to merge back semi-finished Jobs with nested spawns. If the last spawn is always merged first ("LIFO"—Last In, First Out), then knowing the direct parent is sufficient as each parent will in turn know its own parent back to the original and a complete ancestor line can be inferred.

When a Job is spawned, the action SHALL be logged in the parent Node of the spawned Node in the original Job. This is accomplished by creating a ***Spawned*** Element with the @*jRef* Attribute set to the ID of the spawned **JDF** Node. This ***Spawned*** Element SHALL be appended to the ***AuditPool*** container of the original parent Node. If no ***AuditPool*** container exists in the parent Node, one SHALL be created for the purpose.

Example 4.2: Family Tree of Spawned Nodes

The following code is an example of a family tree:

```
<AncestorPool>
   <Ancestor FileName="file:///grandparent.jdf" NodeID="p_01"/>
   <Ancestor FileName="file:///parent.jdf" NodeID="p_02"/>
</AncestorPool>
```

Merging

After processing, the spawned **JDF** Node SHALL be merged back to its original location. Before this can occur, however, duplicate information contained in any Elements (such as ***Comment***) SHALL be deleted by the Agent executing the spawning and merging. Once this has been accomplished, the spawned Node is copied to the location of the original Node, completely overwriting the original Node. The @*Status* of the original Node is then overwritten with the result.

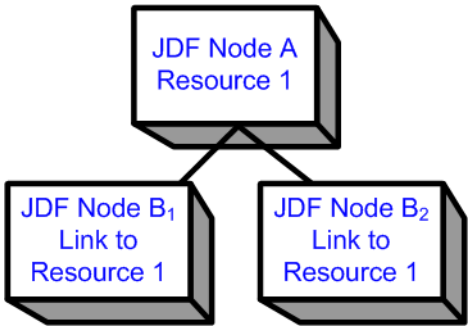
To complete the merging Process, the merging Agent SHALL add a ***Merged*** Audit Element to the ***AuditPool*** (see ▸ Section 3.11 AuditPool and Audit). The @*MergeID* of the ***Merged*** audit Element SHOULD be set to the value of the @*SpawnID* Attribute of the merged Node. Furthermore, the ***AncestorPool*** container with all child Elements SHALL be removed, and the value of @*SpawnID* SHOULD be removed from the @*SpawnIDs* Attribute of the appropriate Resources.

A **JDF** Agent that receives a **JDF** Node that has been spawned individually, and thus has no ***Part*** Element in the ***AncestorPool***, MAY modify any Elements except for Resources that were spawned as read-only data.

4.4.2 Case 2: Spawning and Merging with Resource Copying

The following figure represents an example of a Job that requires that Resources be copied during spawning. In this Job, the Nodes B_1 and B_2 are linked to the same Resource, which is localized in the ***ResourcePool*** of an ancestor Node, denoted as Node A. This Node is the parent Node.

Figure 4-8: JDF Node structure that requires Resource copying during spawning and merging



When Node B_1 is spawned, its Resources SHALL also be duplicated. To accomplish this, the affected Resources SHALL be copied to the spawned **JDF** Node and purged during merging, a Process that is described below.

4.4.2.1 Spawning of Resources with Inter-Resource Links

Resources are linked to a Node by three mechanisms.

- Explicit links defined by a ***ResourceLink*** in the ***ResourceLinkPool*** of the Node.
- Implicit links defined by the ***ResourceRef*** Elements of linked Resources (implicit links are recursive).
- Implicit links defined by the ***ResourceRef*** Elements of the ***AuditPool*** of the Node.

A spawning or merging Agent SHALL resolve all of these links by copying any non-local Resources into the local ***ResourcePool***.

Spawning

Spawning begins as it did in Case 1. The affected Resources SHALL then be copied to the ***ResourcePool*** of the spawned **JDF** Node. The copied Resources retain the same *@ID* values as the original Resources. These Resources can be spawned for read-only access, which allows multiple simultaneous spawning of a Resource, or for read/write access, which allows only one spawning of a Resource. The read/write spawning of a Resource locks the Resource in the original file in order to avoid conflicts that result from simultaneous modification or reading and modification of a Resource. The *@SpawnStatus* Attribute of the original Resource SHALL be set to "SpawnedRW" (which stands for "spawned read/write") or "SpawnedRO" (which stands for "spawned read-only") to indicate that the Resource is spawned. In other words, a copy of the Resource is spawned together with the spawned **JDF** Node. Read/write access effectively locks the original Resources, just as if the Attribute *@Locked* = "true"[1] were present. If a Resource is spawned as read-only, it is not a good idea to modify the original Resource that remains in the parent Job ticket as this might lead to inconsistencies, unless the **JMF** ***Resource Command*** Message is used to inform the Device or Controller that the Resource was spawned to. The *@Locked* Attribute of spawned Resources that are copied read-only SHALL also be set "true". Furthermore, the value of the *@ID* Attribute of each copied Resource SHALL be appended to the appropriate *@rRefsROCopied* or *@rRefsRWCopied* values of the ***Spawned*** Element that resides in the ***AuditPool*** of the parent Node.

Merging

Merging begins as it did in Case 1. Each Read/Write Resource that has been copied for spawning SHALL be copied into its original location, completely overwriting the original Resource. If any Read-only Resource that has been copied for spawning, is not the identical to the original Resource, a **JDF** content error SHOULD be logged by a ***Notification*** Element of *@Class* = "Error" (see ▸ Section 4.6 Error Handling). The *@ID* Attributes of the overwritten Resources SHALL be specified in the *@rRefsOverwritten* Attribute of the ***Merged*** Element. The ***Merged*** Element is then inserted into the ***AuditPool*** container of the parent during the usual merging procedure, which is shown as the return point in the spawning diagram.

4.4.3 Case 3: Parallel Spawning and Merging of Partitioned Resources

In many cases, it is desirable to define a parallel workflow for Partitioned Resources. This is modeled by spawning a Node that defines the Process for each part that is to be processed individually.

Spawning

Spawning begins as it did in Case 1 or Case 2. Then the spawning Agent SHALL loop over all ***ResourceLink*** Elements and ensure that the appropriate ***Part*** Element or Elements exist in any Resources in the spawned ticket, where only the individual parts are REQUIRED. This is accomplished either by adding ***Part*** Elements if none exist in ***ResourceLink*** Elements of the parent Node or by modifying the copies of existing ***Part*** Elements. ***Part*** Elements SHALL be included in all ***ResourceLink*** Elements that point to Resources that are spawned with write access. ***Part*** Elements MAY be included in ***ResourceLink*** Elements that point to Resources that are spawned with read only access (e.g., ***PhysicalResource***s where only

1. Usually Resources become locked (*@Locked* = "true") if they are referenced by ***Audit*** Elements (see also ▸ Section 3.11 AuditPool and Audit).

a part is provided to a Process as input). In addition, copies of the ***Part*** Elements are appended to the ***Spawned*** Audit Element. The @*Status* of any Partitioned Resource is defined individually for each Partition. The @*Status* of the parent Node is set to "Part" and a ***NodeInfo*** Partition for the Partition of this spawn SHALL be created. ***NodeInfo***/@*NodeStatus* of the Partition that describes the newly spawned Node is set to "Spawned".

Exactly one ***Part*** Element that contains the Partition Keys of this spawn and all Partition Keys of previous spawns SHALL be present in the ***AncestorPool*** of the spawned **JDF** Node.

The spawning procedure described in this section can be performed iteratively for multiple parts, effectively generating one ***Spawned*** Audit Element and one ***NodeInfo*** Partition per part. The ***Spawned*** and ***Merged*** audit Elements are not placed in the parent Node of the Node to be spawned, but rather in the Node itself.

An Agent that receives a **JDF** Node that has been spawned in parallel and thus has a ***Part*** Element in the ***AncestorPool*** SHALL NOT modify any Elements except for:

- Resources that were spawned with read-write permission, and
- Adding ***Audit*** Elements.

Synchronizing newly inserted **JDF** Subnode in spawned **JDF** Nodes is OPTIONAL.

Merging

After an individual Partitioned spawned Node has been processed, it is merged back to the parent as described in Case 1. In addition, a copy of the ***Part*** Elements of the corresponding ***Spawned*** Audit Element is appended to the ***Merged*** Element and any read/write Resources are merged into their appropriate parts. The @*Status* of the spawned Node is copied into the appropriate ***PartStatus*** in the ***StatusPool***.

An example of Partitioned Spawning and Merging can be found in ▸ Section O.3 Spawning and Merging.

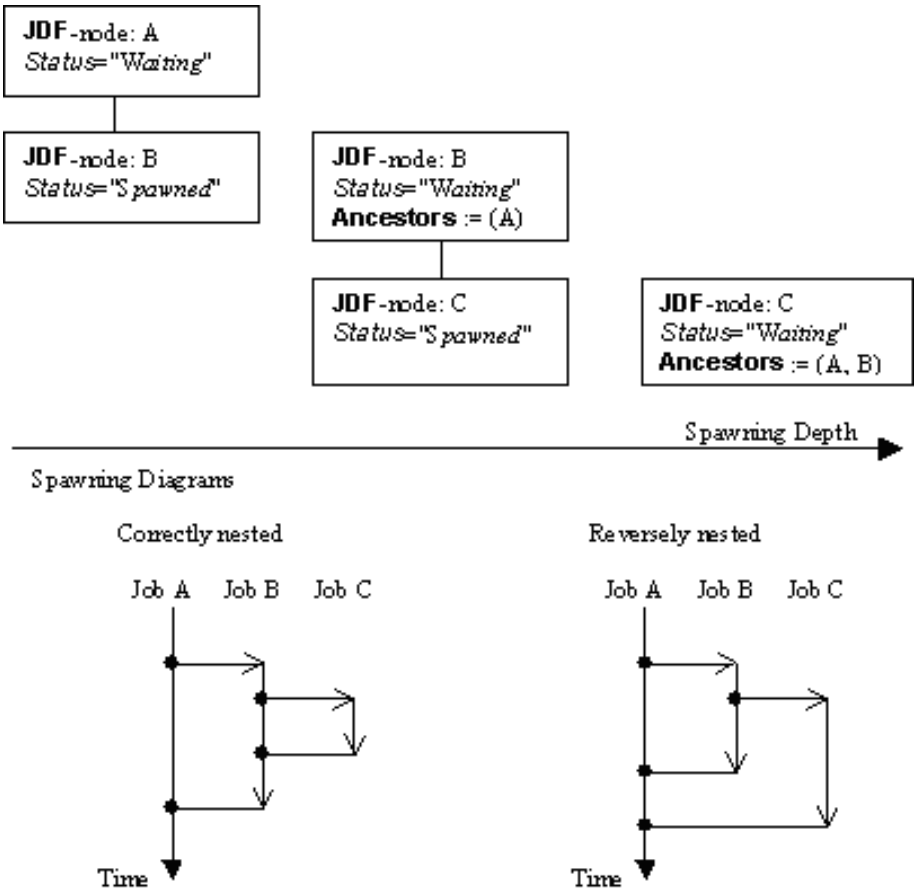
4.4.4 Case 4: Nested Spawning and Merging in Reverse Sequence

Deprecated in JDF 1.2

Note that nested Spawning and Merging in Reverse Sequence has been deprecated because it is highly probable that applications implementing it will not interoperate.

▸ Figure 4-9: Example for a JDF Node structure with nested spawning shows an example of nested spawning and merging in reverse sequence. Process A spawns Node B, and Node B spawns Node C. Even if B is merged back to A for any reason before C is merged back to B, C still contains the information of its grandparent in the ***AncestorPool*** Element. In this way, C can trace back its ancestors and find the location of its parent, Node B, in Node A even though the spawned **JDF** Node, with B as Root Node, has already been deleted.

Figure 4-9: Example for a JDF Node structure with nested spawning



4.4.5 Case 5: Spawning and Merging of Independent Jobs

Deprecated in JDF 1.5

Deprecation note: Starting with **JDF** 1.5, the text from this section has been deprecated and moved to ▸ Section N.3.1 Case 5: Spawning and Merging of Independent Jobs.

Note: Starting with **JDF** 1.5, the objective of Spawning and Merging of Independent Jobs, ganging of Jobs, has been re-defined as a separate Process. See ▸ Section 6.3.38 SheetOptimizing for details on how to describe the collection of multiple Jobs onto a larger Job in order to increase production efficiency.

4.4.6 Case 6: Simultaneous Spawning and Merging of Multiple Nodes

It is not possible to explicitly spawn multiple Nodes simultaneously into one **JDF** job ticket. The Nodes SHALL be grouped into a single Process Group Node. This Node can then be spawned and merged as described in the previous sections.

4.5 Node and Resource IDs

All Nodes and Resources SHALL contain a unique identifier, not only because it is important to be able to identify individual components of a Job, but also because **JDF** uses these IDs for internal linking purposes. Each Agent that creates Resources and Subnodes or that performs spawning and merging is responsible for providing IDs that are unique in the scope of the file, taking into account all of the phases of a Job's life cycle.

IDs come in two flavors: pure and composite. A **pure ID** is an ID that does not contain a period character (“.”). A **composite ID** is made up of pure IDs separated by periods. IDs are used differently under different circumstances. Several different circumstances are described below.

In case of no spawning. If an Agent inserts new Elements requiring IDs into an original Job, then the Agent assigns pure IDs to the new Elements and SHALL guarantee their uniqueness.

In case of single spawning. If an Agent inserts new Elements into a spawned **JDF** Node, then the Agent creates composite IDs by using the ID of the Root Node and appending a unique pure ID delimited by a period. For example:

- ID of spawned Root Node: @*ID* = "Job_01234.Proc1"
- ID used for new Element: @*ID* = "Job_01234.Proc1.newpureID"

In case of independent spawning. The Agent that merges the independent Jobs beneath a Big Job inserts a unique, pure ID (delimited by a period) in front of all IDs of each Small Job it receives. That means that the Agent SHALL replace all IDs of each Job it receives whenever it encounters an ID collision. If an Agent inserts new Elements into a spawned **JDF** Node, then the Agent creates composite IDs by using the ID of the respective Root Node of the Small Job and appends a unique pure ID, delimited by a period. For example:

- ID of the Big Job with Node @*ID* = "A"
- Receives Small Job A_1 with some IDs: @*ID* = "A" @*ID* = "A.A" @*ID* = "A.B" where the first is the ID of the Root Node.
- Receives Small Job A_2 with some IDs: @*ID* = "A" @*ID* = "A.A" @*ID* = "anything" ...
- The Agent creates locally unique pure IDs: @*ID* = "A1" and @*ID* = "A2" each prefixed to all IDs of each received Small Job; the IDs of the Small Job A_1 become: @*ID* = "A1.A" @*ID* = "A1.A.A" @*ID* ="A1.A.B", and the IDs of the Small Job A_2 become: @*ID* = "A2.A" @*ID* = "A2.A.A" @*ID* = "A2.anything". All IDs in the Big Job are unique.
- The Agent creates a new Element added to the Small Job A_1 with ID: @*ID* = "A1.A.C". Here the Agent SHALL resolve the possible conflict if it would append the pure ID = "A" to the root ID = "A1.A". That means the Agent has to check the uniqueness of each created ID.
- Before merging the Jobs back to their original location, the Agent SHALL remove the prefixed pure IDs of all IDs, here "A1", "A2" respectively. Then the newly created Element will be merged back with the @*ID* = "A.C".

4.6 Error Handling

Error handling is an implementation-dependent feature of **JDF**-based systems. The ***AuditPool*** Element provides a container where errors that occur during the execution of a **JDF** Node are to be logged using ***Notification*** Elements. ***Notification*** Elements MAY also be sent in **JMF** messages. The content of the ***Notification*** Element is described in ▸ Table 3.36 Notification Audit Element. For a list of predefined error codes, see ▸ Appendix C Return Values and Error Codes. Further details about error handling are provided in the next four sections.

4.6.1 Classification of Notifications

Notification Audit Elements are classified by the @*Class* Attribute. Every workflow implementation SHALL associate a Class with all events on an event-by-event basis. For values, see ***Notification***/@*Class* in ▸ Table 3.11.4.5 Notification.

4.6.2 Event Description

A description of the event is given by a generic ***Comment*** Element, which is REQUIRED for the ***Notification*** Classes "Information", "Warning", "Error" or "Fatal". For example, after a Process is aborted, error information describing a Device error MAY be logged in the ***Comment*** Element of the ***Notification*** Element. If phase times are logged, the ***PhaseTime*** Element that logged the transition to the "Aborted" state MAY also contain a local ***Comment*** Element that describes the cause of

the Process abortion. ***PhaseTime*** and ***Notification*** Elements are OPTIONAL Subelements of the ***AuditPool***, which is described in ▸ Section 3.11 AuditPool and Audit.

4.6.3 Error Logging in the JDF File

A **JDF**-compliant Controller/Agent SHOULD log an error by inserting a ***Notification*** Element in the ***AuditPool*** of the Node that generated the error. The ***NodeInfo*** Resource MAY contain ***NotificationFilter*** Elements to define the notification events (or, more specifically, errors) that SHOULD be logged.

4.6.4 Error Handling via Messaging (JMF)

A **JMF** with a ***Notification*** Element in the Message body SHOULD be sent through all persistent channels that subscribed events of Class "Error". How to subscribe error events via **JMF**, see ▸ Section 5.3.4 Persistent Channels and ▸ Section 5.19 Events. Note that this is different from the ***NotificationFilter*** Elements of the ***NodeInfo*** Resource, which is defined for logging events by ***Notification*** Elements to the ***AuditPool***.

4.7 Test Running

In **JDF**, the notion of a test run is similar to the press notion of preflight. The goal is to detect **JDF** content errors and inconsistencies in a Job before the Job is executed.

The ability to perform a test run MAY be built into individual Devices or Controllers. Alternatively, a Controller implementation MAY perform test runs on behalf of its Devices. A test run MAY be routed through all of the different Devices and Controllers in a workflow, just as if the test run were a standard execution run. For the routing of Jobs and Nodes through different Devices and Controllers for a test, the spawning and merging mechanism MAY also be applied. The Devices/Controllers receiving a Job read and analyze it WITHOUT initiating execution. Rather, they investigate the content of the Node they would execute. A Device/Controller with Agent capabilities MAY record results into the ***AuditPool*** associated with a given Process.

During test running, the requirements of the Processes specified are compared to the capabilities of the Devices targeted. A Device or Controller explicitly tests if the REQUIRED inputs are actually present, valid and without errors. For example, an input requirement might be a URL that, when a test run is performed, is found to point to an item that no longer exists in that location. Test running is meant to prevent errors as a result of that kind of misinformation. It is particularly useful when running expensive or time-consuming Jobs.

It is also possible to test run specific parts of a workflow, or even individual Nodes. An Agent might request a test of certain Nodes by setting the **JDF** *@Activation* Attribute to "TestRun" (see ▸ Table 3.4 JDF), which is inherited by all descendent Nodes that are not inactive (*@Activation* = "Inactive"). If a Device or Controller[1] detects an error in a Node a ***Notification*** Element containing a textual description SHOULD be appended to the ***AuditPool*** Element of the Node in which the error occurred, and if messaging is supported, the error SHOULD be also communicated to the connected listeners via messaging. For more information, see ▸ Section 5.5 Error and Event Messages. If an error has been detected, the Agent can modify the Job in order to correct the error. Once a test run has been completed successfully, the Device/Controller with Agent capabilities changes the *@Status* Attribute of the tested Node to "Ready". If a test run fails, the Device/Controller SHALL record the Process status as "FailedTestRun". After the test run has finished, the Agent SHOULD log the result by appending a ***ProcessRun*** Element to the ***AuditPool*** Element. For more information about ***Audit*** Elements, see ▸ Section 3.11 AuditPool and Audit.

In principle, execution and test runs might be run simultaneously. For example, one Job Part could be executed while another part requests only a test. **JDF** also defines an *@Activation* value of "TestRunAndGo" that requests a test run and, upon successful completion, automatically initiates processing.

4.7.1 Resource Status During a Test Run

In order to test run a complete set of Nodes, it is sometimes necessary to imply the *@Status* of Resources that are produced by prior Nodes. Successful test running does *not* set the *@Status* Attribute of a Resource to "Available" unless the Resource actually is available. Nodes that require an Output Resource from a Node that has completed test running for purposes of test running itself can assume that these Resources have a *@Status* of "Available" for the purpose of test running as long as the producing Node has a *@Status* of "Ready".

1. Note that only Devices and Controllers with Agent capabilities can write in a **JDF** document.

JMF Messaging Format

Introduction

A workflow system is a dynamic set of interacting Processes, Devices and MIS systems. For the workflow to run efficiently, these Processes and Devices need to communicate and interact in a well defined manner. Messaging is a simple but powerful way to establish this kind of dynamic interaction. The **JDF**-based Job Messaging Format (**JMF**) provides a wide range of capabilities to facilitate interaction between the various aspects of a workflow, from simple unidirectional notification through the issuing of direct commands. This chapter outlines the way in which **JMF**, accomplishes these interactions. The following list of use cases is considered:

- System setup
- Dynamic status and error tracking for Jobs and Devices
- Pipe control
- Device setup and Job changes
- Queue handling and Job submission
- Device Capability description

JMF = ROI

In order to automate aspects of your production without JDF, your technical staff needs to become proficient in each of the command languages that each of your Devices employ. By only buying JDF-enabled Devices that use JMF as their control language, you only have to learn one new Device command language … eventually, the *only one* your MIS staff will need.

Both Controllers and Devices MAY support **JMF**. This support requires hosting by a HTTP(S) server. **JMF** messages are most often encoded in pure XML, without an additional MIME Multipart wrapper. Only Controllers that support **JDF** Job submission via the message channel SHALL support MIME for messages.

JMF messaging uses a Bidirectional protocol — currently HTTP and HTTPS.

JDF messaging supports combining the **JMF** message, the **JDF** Job ticket(s) to which it refers, and, possibly, the digital assets to which the **JDF** Job tickets refer into a single package. See ‣ Section 11.3 JDF Packaging

5.1 JMF Root

JMF and **JDF** have inherently different structures. In order to allow immediate identification of messages, **JMF** uses the unique name **JMF** as its own root-element name.

The root element of the XML fragment that encodes a message, like the root element of a **JDF** fragment, contains a series of predictable Attributes and instances of ***Message*** elements. These contents are defined in the tables that follow and are illustrated in ‣ Figure 5-1: JMF Root Element – a diagram of its structure. ***Message*** elements are Abstract, as is indicated by the dashed line surrounding the ***Message*** element in ‣ Figure 5-1: JMF Root Element – a diagram of its structure.

Table 5.1: JMF Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AgentName ? New in JDF 1.4	string	The name of the Agent application that generated the **JMF**. Both the company name and the product name MAY appear, and SHOULD be consistent between versions of the application.
AgentVersion ? New in JDF 1.4	string	The version of the Agent application that generated the **JMF**. The format of the version string MAY vary from one application to another, but SHOULD be consistent for an individual application.
DeviceID ?	string	Identifies the recipient Device or Controller. If @*DeviceID* is not specified, then the recipient of the message is assumed to be the final recipient. If a Controller receives a message which references a @*DeviceID* that does not match the Controller's @*ControllerID*, the Controller SHOULD attempt to pass the message on to the correct Device. If the Controller is unable to pass the message on, it SHOULD respond to the message with ***Message***/@*ReturnCode* = 121, "Unknown DeviceID". If a Device receives a message with a @*DeviceID* that does not match its own, it SHOULD also respond to the message with ***Response***/@*ReturnCode* = 121.

Table 5.1: JMF Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ICSVersions ? New in JDF 1.3	NMTOKENS	CIP4 Interoperability Conformance Specification (ICS) Versions that this **JMF** message complies with. The semantics are identical to ***JDF***/@*ICSVersions*. **Values include those from:** ***JDF***/@*ICSVersions* (▸ Table 3.4 JDF).
MaxVersion ? New in JDF 1.3	JDFJMFVersion	Maximum **JDF** version to be written by an Agent that modifies this message. If not specified, an Agent that responds to the message MAY write any version it is capable of writing. See ▸ Section 3.13 JDF Versioning for a discussion of versioning in **JDF**.
ResponseURL ? New in JDF 1.2 Deprecated in JDF 1.5 *Unidirectional*	URL	URL of the direct response to this **JMF**. @*ResponseURL* is REQUIRED when using an unidirectional protocol that does not automatically provide a response channel (e.g., the file protocol). If @*ResponseURL* is specified, a ***Response*** SHALL be generated and written to @*ResponseURL*, even if no ***ResponseTypeObj*** is REQUIRED for the ***Message***. The ***Response*** MAY be empty. It SHALL NOT be present when a bidirectional protocol is used (e.g., in HTTP). The URL SHALL be an explicit locator. It is up to the sending Agent to generate a unique locator for the response. Example: "file://master/JMFResponseFolder/Rip1/r12345.jmf" **Deprecation note:** Unidirectional (file based) **JMF** has been deprecated.
SenderID	string	String that identifies the sender Device, Controller or Agent. For a sender Device, the sender's @*DeviceID*. For a sender Controller, the sender's @*ControllerID*. @*SenderID* SHOULD be modified to the proxy Controller's @*ControllerID*when a **JMF** is passed through a proxy. See also ***Message***/@*SenderID* in ▸ Table 5.2 Message Element.
TimeStamp	dateTime	Time stamp that identifies when the message was created.
Version Modified in JDF 1.2	JDFJMFVersion	Text that identifies the version of the **JMF** message. The current version of this specification are "1.1", "1.2", "1.3", "1.4" and "1.5". The version of a **JMF** message is defined by the highest version of the **JMF** message itself or any child Element. For details on **JDF** versioning see ▸ Section 3.13 JDF Versioning. Note that @*Version* was OPTIONAL before **JDF** 1.2, but is REQUIRED in instances that conform to **JDF** 1.2 and beyond. If not specified, the XML schema value for @*Version* SHALL default to "1.1".
xmlns ? New in JDF 1.1	URI	**JDF** supports use of XML namespaces. The **JDF** namespace SHALL be declared. For details on using namespaces in XML, see ▸ [XMLNS].
Employee ? New in JDF 1.4	element	Employee who created this message.
Message + Modified in JDF 1.4	element	Abstract ***Message*** element(s). If a **JMF** instance includes multiple ***Message*** Elements, the messages SHALL be executed in XML order. **Modification note:** Starting with **JDF** 1.4, ***Message*** order is relevant.

5.1.1 Message

The following table describes the contents of the abstract ***Message*** element. All messages contain an @*ID* and a @*Type* attribute.

Table 5.2: Message Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AgentName ? New in JDF 1.4	string	The name of the agent application that generated the **JMF**. Both the company name and the product name MAY appear, and SHOULD be consistent between versions of the application. If not specified, defaults to the value of ***JMF***/@*AgentName*.
AgentVersion ? New in JDF 1.4	string	The version of the agent application that generated the **JMF**. The format of the version string MAY vary from one application to another, but SHOULD be consistent for an individual application. If not specified, defaults to the value of ***JMF***/@*AgentVersion*.

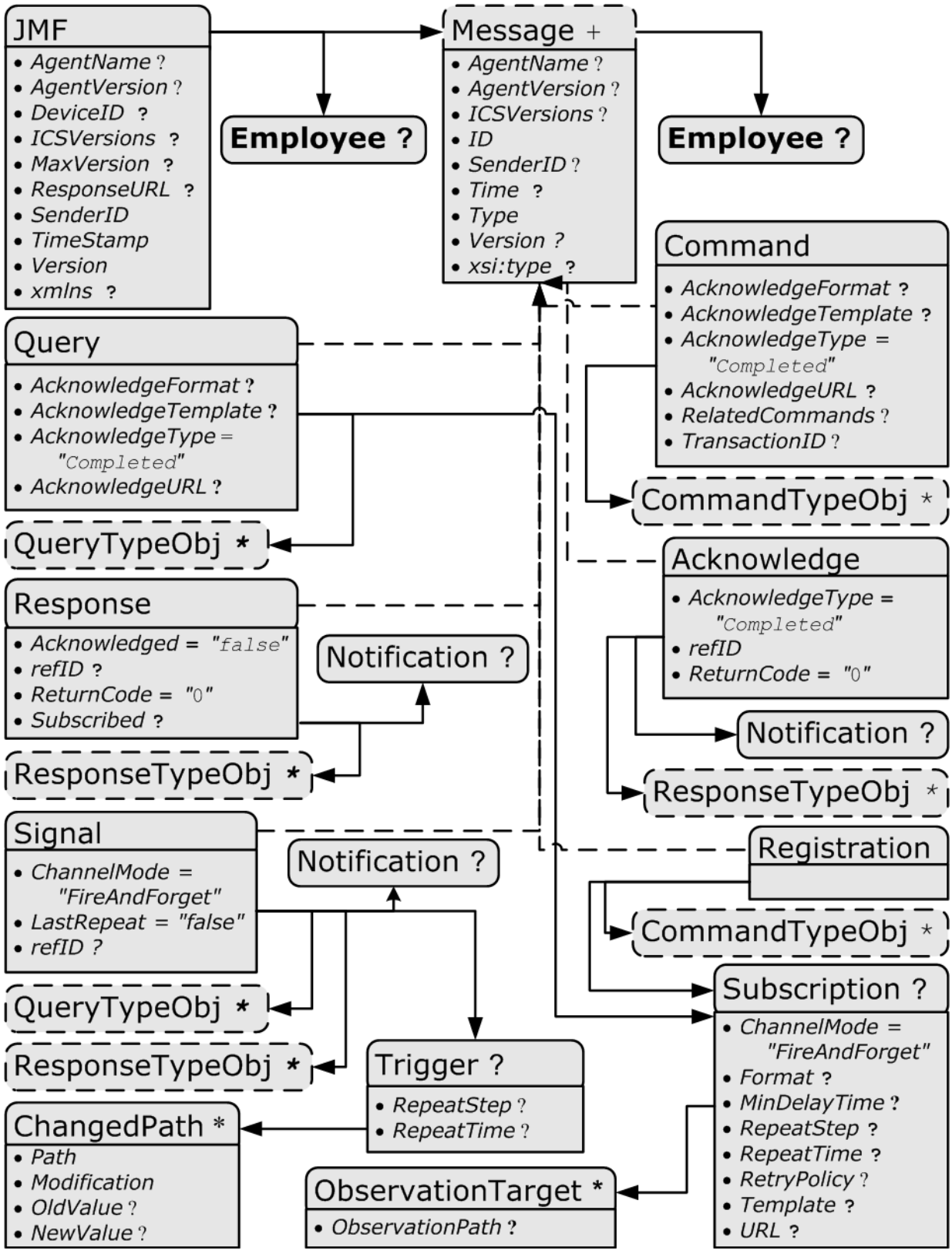
Table 5.2: Message Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ICSVersions ? New in JDF 1.4	NMTOKENS	CIP4 Interoperability Conformance Specification (ICS) Versions that this **JMF** message complies with. The semantics are identical to ***JDF***/@*ICSVersions*. If not specified, defaults to the value of ***JMF***/@*ICSVersions*. **Values include those from:** ***JDF***/@*ICSVersions* (▸ Table 3.4 JDF).
ID	ID	Identifies the message.
SenderID ? New in JDF 1.4	string	@*SenderID* of the original sender of this message Element. If not specified, defaults to the @*SenderID* of the parent **JMF**. @*SenderID* SHALL NOT be modified when a **JMF** is passed through a proxy. See also ***JMF***/@*SenderID* in ▸ Table 5.1 JMF Element.
Time ?	dateTime	Time at which the message was generated. This Attribute NEED NOT be specified unless this time is different from the time specified in the @*TimeStamp* Attribute of the **JMF** Element. **Note:** When a proxy forwards messages and creates a new **JMF** parent for a message, it SHALL update @*Time* to the value of the original ***JMF***/@*TimeStamp* if @*Time* is not provided in the original message.
Type	NMTOKEN	Name that identifies the message type. Message types are described in the remainder of this chapter. **Values include those from:** ▸ Table 5.14 List of JMF Messages.
Version ? New in JDF 1.4	JDFJMFVersion	Text that identifies the version of the **JMF** message. The current version of this specification are "1.1", "1.2" , "1.3" and "1.4". The version of a **JMF** message is defined by the highest version of the **JMF** message itself or any child Element. For details on **JDF** versioning see ▸ Section 3.13 JDF Versioning. If not specified, defaults to the value of ***JMF***/@*Version*.
xsi:type ? New in JDF 1.2	NMTOKEN	Informs schema aware validators of the **JMF** message type definition that the message SHALL be validated against. The schema for this version includes definitions for all the standard **JMF** messages defined in ▸ Section 5.6 Message Template. If omitted then a general definition for the **JMF** message will be used. See ▸ Section 3.2 JDF.
Employee ? New in JDF 1.4	element	Employee who created this message.If not specified, defaults to the value of ***JMF***/***Employee***.

5.1.2 Structure Diagram

The following figure depicts the basic **JMF** messaging structure and the message Families. Dashed boxes show abstract objects.

Figure 5-1: JMF Root Element – a diagram of its structure



5.2 JMF Message Families

A message contains one or more of the following six high level Elements, referred to as message **Families**, in the Root Node. These families are ***Query***, ***Command***, ***Signal***, ***Response***, ***Acknowledge*** and ***Registration***. An explanation of each family is provided in the following sections, along with an encoding example.

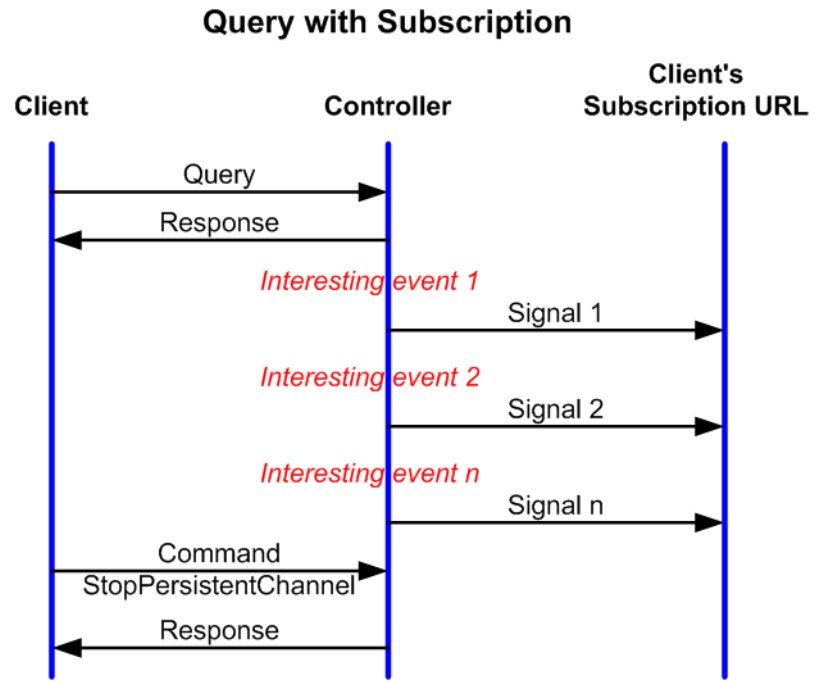
Response & Acknowledgement

The terminology used for Message Families contradicts common usage but will be retained for backwards compatibility. The ***Response*** actually functions as an Acknowledgement that a ***Command*** will be acted upon, while the ***Acknowledge*** could more properly be named Completion or Result. The naming was defined to be consistent with HTTP naming conventions so that a Response is always transported on an HTTP response in case HTTP is used as the JMF transport protocol layer.

5.2.1 Query

A ***Query*** Element is used as a message that retrieves information from a Controller without changing the state of that Controller. A query is sent to a Controller. After a ***Query*** message is sent, a ***Response*** message is returned. If the ***Query*** message included a ***Subscription***, ***Signal*** messages are sent to the designated URL until a ***StopPersistentChannel Command*** message is sent.

Figure 5-2: Interaction of Messages with a Subscription



The ***Query*** contains an @*ID* Attribute and a @*Type* Attribute, which it inherits from the Abstract message type described in ▸ Table 5.2 Message Element. **JMF** supports a number of well defined query types, and each query type can contain additional descriptive Elements, which are described in ▸ Section 5.11 Queue Support and ▸ Section 5.16 Extending Messages.The following table shows the content of a ***Query*** message Element: .

Table 5.3: Query Message Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AcknowledgeFormat ? New in JDF 1.3 Deprecated in JDF 1.5 *Unidirectional*	string	A formatting string used with the @*AcknowledgeTemplate* Attribute to define a sequence of generated URLs. If @*AcknowledgeFormat* is specified, then @*AcknowledgeTemplate* SHALL also be specified and @*AcknowledgeURL* SHALL NOT be specified. **Allowed values are from:** ▸ Appendix H Dynamic String Generation.
AcknowledgeTemplate ? New in JDF 1.3 Deprecated in JDF 1.5 *Unidirectional*	string	A template, used with @*AcknowledgeFormat*, to define a sequence of generated URLs. The resulting set of URLs SHALL be qualified URLs and not a folder. If @*AcknowledgeTemplate* is specified, then @*AcknowledgeFormat* SHALL also be specified and @*AcknowledgeURL* SHALL NOT be specified. **Allowed values are from:** ▸ Appendix H Dynamic String Generation.

Table 5.3: Query Message Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
AcknowledgeType = "Completed" New in JDF 1.3	enumerations	Defines the actions to be acknowledged. This is necessary mainly for Device-Machine pairs where the Machine is not accessible online. **Allowed values are:** Received – The ***Query*** has been received and understood (e.g., by an operator). Applied – The ***Query*** has been applied to the Machine (e.g., by an operator). Completed – The ***Query*** has been completely responded to.
AcknowledgeURL ? New in JDF 1.3	URL	URL of the recipient of any ***Acknowledge***. If specified, the command requests for an Acknowledge message depending on the value of @*AcknowledgeType*. The protocol of the acknowledgement is specified by the scheme of @*AcknowledgeURL*.
QueryTypeObj *	element	Abstract element that is a placeholder for any descriptive elements that provide details for the query. The element type of ***QueryTypeObj*** is defined by the @*Type* attribute of the abstract ***Message*** element.
Subscription ?	element	If specified creates a persistent channel. For the structure of a ***Subscription***Element, see ▸ Section 5.3.4 Persistent Channels.

Example 5.1: Query Message

The following is an example of a Query message:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Controller-1"
       TimeStamp="2005-07-25T11:38:23+02:00"
       MaxVersion="1.4" Version="1.4"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Query ID="M007" Type="KnownDevices" xsi:type="QueryKnownDevices"/>
</JMF>
```

5.2.2 Command

A ***Command*** Element is syntactically equivalent to a ***Query***, but rather than simply retrieving information, it also causes a state change in the target Device. The following table contains the contents of a ***Command*** message. A ***Response*** message is returned immediately after a ***Command***. If the ***Command*** included an @*AcknowledgeURL*, and the ***Command*** was going to take a while, the Device Controller MAY select to return the ***Response*** message with @*Acknowledge* = "true", and send an ***Acknowledge*** message to the @*AcknowledgeURL* when the ***Command*** completes.

Table 5.4: Command Message Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AcknowledgeFormat ? New in JDF 1.2 Deprecated in JDF 1.5 *Unidirectional*	string	A formatting string used with the @*AcknowledgeTemplate* Attribute to define a sequence of generated URLs. If @*AcknowledgeFormat* is specified, then @*AcknowledgeTemplate* SHALL also be specified and @*AcknowledgeURL* SHALL NOT be specified. **Allowed values are from:** ▸ Appendix H Dynamic String Generation.
AcknowledgeTemplate ? New in JDF 1.2 Deprecated in JDF 1.5 *Unidirectional*	string	A template, used with @*AcknowledgeFormat*, to define a sequence of generated URLs. The resulting set of URLs SHALL be qualified URLs and not a folder. If @*AcknowledgeTemplate* is specified, then @*AcknowledgeFormat* SHALL also be specified and @*AcknowledgeURL* SHALL NOT be specified. **Allowed values are from:** ▸ Appendix H Dynamic String Generation.
AcknowledgeType = "Completed" New in JDF 1.1	enumerations	Defines the actions to be acknowledged. This is necessary mainly for Device-Machine pairs where the Machine is not accessible online. **Allowed values are:** Received – The Command has been received and understood (e.g., by an operator). Applied – The Command has been applied to the Machine (e.g., by an operator). Completed – The Command has been executed.

Table 5.4: Command Message Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
AcknowledgeURL ? Modified in JDF 1.2	URL	URL of the recipient of any ***Acknowledge***. If specified, the command requests for a Acknowledge message depending on the value of @*AcknowledgeType*. The protocol of the acknowledgement is specified by the scheme of @*AcknowledgeURL*.
RelatedCommands ? New in JDF 1.4	NMTOKENS	A list of ***Command***/@*ID* values that need to be processed as a single transaction (in other words all Commands needs to succeed or all need to be rejected). The Commands SHALL be processed in the order specified by this attribute. This attribute SHALL only appear in the last Command of a transaction. An application SHOULD wait for a reasonable amount of time to collect all related Commands prior to failing a transaction.
TransactionID ? New in JDF 1.4	string	The ID on the transaction the Command belongs to. All Commands with the same @*TransactionID* SHALL either all succeed or all fail
CommandTypeObj *	element	Abstract Element that is a placeholder for any descriptive Elements that provide details of the command.

Example 5.2: ResumeQueueEntry Command Message

The following example demonstrates how a ***ResumeQueueEntry*** Command message can cause a Job in a queue to begin executing:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="A3 Printer"
       SenderID="MIS master A" TimeStamp="2000-07-25T12:32:48+02:00"
       MaxVersion="1.4" Version="1.4"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <Command ID="M009" Type="ResumeQueueEntry" xsi:type="CommandResumeQueueEntry">
      <QueueEntryDef QueueEntryID="job-0032"/>
   </Command>
</JMF>
```

Example 5.3: ResumeQueueEntry Response Message

The following example shows a possible ***Response*** message to the ***Command*** message example above:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
       TimeStamp="2000-07-25T12:32:48+02:00"
       MaxVersion="1.4" Version="1.4"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <Response ID="M109" Type="ResumeQueueEntry" xsi:type="ResponseResumeQueueEntry"
          refID="M009">
      <Queue DeviceID="A3 Printer" Status="Full">
         <QueueEntry JobID="job-0032" QueueEntryID="job-0032" Status="Running"/>
      </Queue>
   </Response>
</JMF>
```

5.2.3 Signal

A ***Signal*** Element is used as a message, which is equivalent to a combination of a ***Query*** message and a ***Response*** message. It is a unidirectional message sent on any event to other Controllers. This kind of message can be used to automatically broadcast status changes.

Controllers can get ***Signal*** messages in one of three ways. The first way is to subscribe for them with an initiating Query message transmitted via a message channel that includes a ***Subscription*** Element. The second way is to subscribe for them with an initiating ***Query*** message defined in the ***NodeInfo*** Element of a **JDF** Node that also includes a ***Subscription*** Element (see **JMF** Elements in ▸ Section 8.95 NodeInfo). The first Query message is transmitted separately via a mechanism such as HTTP, whereas the second is read together with the corresponding **JDF** Node. Once the subscription has been established, signals are sent to the subscribing Controllers via persistent channels. In both cases, however, the ***Signal*** message contains a @*refID* Attribute that refers to the persistent channel. The value of the @*refID* Attribute identifies the persistent channel that initiated the ***Signal***.

The third way in which a Controller can receive a signal is to have the signal channels hard-wired, for example, by a tool such as a list of Controller-URLs read from an initialization file. For example, signals MAY be generated independently when a service is started, or when sub-Controllers that are newly connected to a network want to inform other Control-

lers about their capabilities. Hard-wired signals, however, SHALL NOT have a @*refID* Attribute. If no @*refID* is specified, the corresponding query parameters SHALL be specified instead.

Table 5.5: Signal Message Element

NAME	DATA TYPE	DESCRIPTION
ChannelMode = "FireAndForget" New in JDF 1.4	enumeration	Specifies reliability of the signal. **Allowed value is from:** ▸ ChannelMode
LastRepeat = "false"	boolean	If "true", the persistent channel is being closed by the Device and no further messages will be generated that fulfill the persistent channel criteria. If "false", further signals will be sent. For further details, see ▸ Section 5.3.4 Persistent Channels.
refID ?	NMTOKEN	Identifies the initiating ***Query*** message that subscribed this ***Signal*** message. Hard-wired signals SHALL NOT contain a @*refID* attribute.
Notification * Modified in JDF 1.5	element	Textual description of the signal. The ***Notification*** Element SHOULD be provided if the severity of the event that caused this signal is greater than "Warning", or if pure events have been subscribed. See ▸ Section 3.11.4.5 Notification. For details about subscribing pure events see ▸ Section 5.19 Events. **Modification note:** Starting with **JDF** 1.5, this element changes from optional to zero or more occurrences.
QueryTypeObj * Modified in JDF 1.4	element	This Element is an Abstract Element and a placeholder for any descriptive Elements that provide details for the virtual ***Query***, which, if sent, would convey the same ***ResponseTypeObj*** Elements. These Element types are the same as in the ***Query*** message Element. If the ***QueryTypeObj*** is required in the corresponding ***Query***, it SHALL also be specified in the ***Signal***, even if the ***QueryTypeObj*** in the Subscription message referred to by @*refID* completely defines the context. The Element type of ***QueryTypeObj*** is defined by the @*Type* Attribute of the Abstract ***Message*** element.
ResponseTypeObj *	element	Abstract Element that is a placeholder for any descriptive Elements that provide details subscribed. These Element types are the same as in the ***Response*** message Element.
Trigger ?	element	Describes the trigger event which caused this signal. The Trigger Element recalls some information provided during the ***Subscription*** of the Signal messages. For details on subscribing signals see ▸ Section 5.3.4 Persistent Channels.

5.2.3.1 Trigger

The following table describes the structure of the ***Trigger*** Element.

Table 5.6: Trigger Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
RepeatStep ?	integer	Recalls the @*RepeatStep* Attribute specified during ***Subscription*** of the signal. For details see ▸ Table 5.11 Subscription Element.
RepeatTime ?	double	Recalls the @*RepeatTime* Attribute specified during ***Subscription*** of the signal. For details see ▸ Table 5.11 Subscription Element.
Added ? Deprecated in JDF 1.2	element	A pool that contains the description of trigger events caused by the adding of Elements like services, Controllers, Devices or messages. Replaced by ***ChangedPath*** in **JDF** 1.2 and above. See ▸ Section N.4.1 Signal for details.
ChangedAttribute * Deprecated in JDF 1.2	element	If a change of an Attribute triggered this signal, this Element describes the Attribute that changed. Replaced by ***ChangedPath*** in **JDF** 1.2 and above. See ▸ Section N.4.1 Signal for details.
ChangedPath * New in JDF 1.2	element	If a change of an Attribute or Element triggered this signal, this Element describes the details of the Element or Attribute that changed.

Table 5.6: Trigger Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Removed ? Deprecated in JDF 1.2	element	A pool that contains the description of trigger events caused by the removal of Elements like services, Controllers, Devices or messages. Replaced by ***ChangedPath*** in **JDF** 1.2 and above. See ▸ Section N.4.1 Signal for details.

5.2.3.2 ChangedPath

New in JDF 1.2

The following describes the structure of the ***ChangedPath*** Element. ***ChangedPath*** replaces the ***Added***, ***ChangedAttribute*** and ***Removed*** Elements.

Table 5.7: ChangedPath Element

NAME	DATA TYPE	DESCRIPTION
Path	XPath	XPath of the Element or Attribute that was modified.
Modification	enumeration	Specifies the modification that occurred with the object specified in @*Path*. **Allowed values are:** Create – The object was created. Delete – The object was deleted. Modify – The object was modified.
OldValue ?	string	Old value of the Attribute if @*Path* specifies an Attribute and @*Modification* != "Create". The string SHALL be cast to the appropriate data type that depends on the Attribute's data type.
NewValue ?	string	New value of the Attribute if @*Path* specifies an Attribute and @*Modification* != "Delete". The string SHALL be cast to the appropriate data type that depends on the Attribute's data type.

Example 5.4: Signal Message

The following is an example of a Signal message:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Press 45"
       TimeStamp="2005-07-25T12:28:01+02:00"
       MaxVersion="1.4" Version="1.4"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Signal ID="s123" Type="Status" xsi:type="SignalStatus">
       <StatusQuParams JobID="42" JobPartID="66"/>
       <DeviceInfo DeviceStatus="Setup"/>
    </Signal>
</JMF>
```

5.2.4 Response

A ***Response*** is a message that a Receiver SHALL synchronously send to a sender as a response to a message. A ***Response*** Element is used to reply to a ***Query*** or a ***Command*** and is always a direct answer of a ***Query*** or a ***Command***. A ***Response*** message is returned from a Controller to the Controller that submitted the ***Query*** or ***Command***; however, ***Response*** message(s) are not acknowledged themselves.

A ***Response*** message indicates that a ***Query*** or ***Command*** has been received and interpreted. The ***Response*** of a ***Query*** or ***Command***s with short latency also includes the information about the execution. An ***Query*** or ***Command*** with long latency MAY additionally generate a separate ***Acknowledge*** message (see ▸ Section 5.2.5 Acknowledge) to broadcast the execution of the ***Query*** or ***Command***. A ***Response*** SHOULD contain a ***Notification*** Element that describes the return status in text if @*ReturnCode* is greater than 0. A ***Response*** contains an Attribute called @*refID*, which identifies the initiating ***Query*** or ***Command***. The following table shows the content of a ***Response*** message.

A ***Signal*** with @*ChannelMode* = "Reliable" SHALL also be replied to with a ***Response***.

Table 5.8: Response Message Element

NAME	DATA TYPE	DESCRIPTION
Acknowledged = "false"	boolean	Indicates whether the ***Command***/***Query*** will be acknowledged separately. If "true", an ***Acknowledge*** message will be supplied after ***Command***/***Query*** execution. If "false", no ***Acknowledge*** message will be supplied. If @*Acknowledged*="true", then no additional information other than protocol information, such as @*AgentName*, @*ID* and @*refID* SHALL be specified. "Real" information SHALL only be specified in the corresponding ***Acknowledge***.
refID ? Modified in JDF 1.2	NMTOKEN	Copy of the @*ID* Attribute of the initiating ***Query*** message or ***Command*** message to which the Response message refers. If not specified, the Response message refers to the entire **JMF** message (e.g., if the **JMF** was not parseable). ***Response***/@*Type* is set to "Notification" if the @*Type* of the incoming ***Message*** is corrupted or unknown.
ReturnCode = "0"	integer	The value "0" indicates success. For all other possible codes see ▸ Appendix C Return Values and Error Codes.
Subscribed ? Modified in JDF 1.2	boolean	If a ***Subscription*** element has been supplied by the corresponding query, this attribute indicates whether the ***Subscription*** has been refused or accepted. If "true", the requested ***Subscription*** is accepted. If "false", the ***Subscription*** is refused because the Controller does not support persistent channels. For details, see ▸ Section 5.3.4 Persistent Channels.
Notification * Modified in JDF 1.5	element	Additional information including textual description of the return code. The ***Notification*** Element SHOULD be provided if the @*ReturnCode* is greater than 0, which indicates that an error has occurred. See ▸ Section 3.11.4.5 Notification. **Modification note:** Starting with **JDF** 1.5, this element changes from optional to zero or more occurrences.
ResponseTypeObj *	element	Abstract Element that is a placeholder for any descriptive Elements that provide details queried for or details about command execution. If ***Response***/@*Acknowledged* = "true", ***ResponseTypeObj*** Element(s) MAY be missing or incomplete in a ***Response***.

Example 5.5: Response Message for Query

An example of a Response message to a Command message is provided in the ▸ Section 5.2.2 Command. The encoding example for the Query message, shown above, might generate the following Response message:

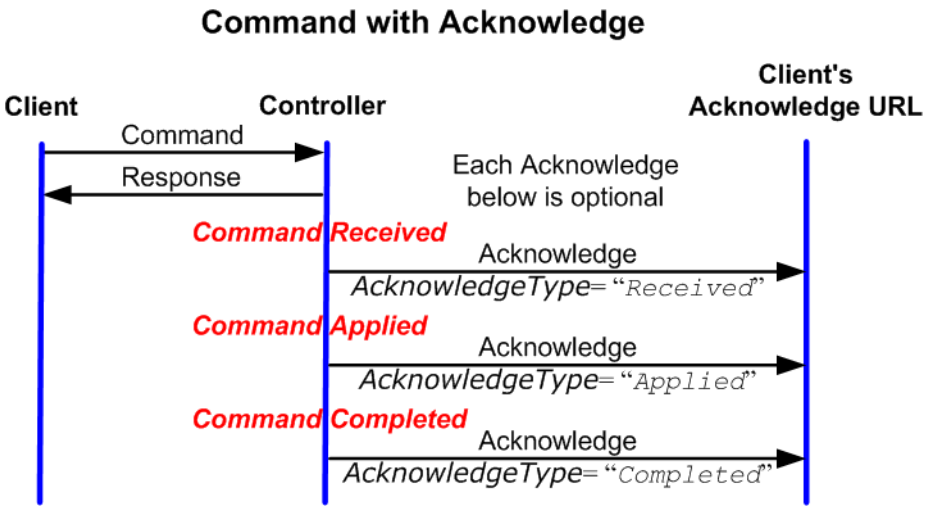
```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="RIP-1"
       TimeStamp="2000-07-25T11:38:25+02:00"
       MaxVersion="1.4" Version="1.4"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <Response ID="M107" Type="KnownDevices" xsi:type="ResponseKnownDevices"
          refID="M007">
       <DeviceList>
          <DeviceInfo DeviceStatus="Unknown">
             <Device DeviceID="Rip1"/>
          </DeviceInfo>
          <DeviceInfo DeviceStatus="Unknown">
             <Device DeviceID="Rip2"/>
          </DeviceInfo>
       </DeviceList>
   </Response>
</JMF>
```

5.2.5 Acknowledge

An ***Acknowledge*** Element is a message that is an asynchronous answer to a ***Command*** message or ***Query*** message issued by a Controller. Each ***Acknowledge*** message is unidirectional and similar to a ***Response*** message, and the @*refID* Attribute of each refers to the initiating command. Acknowledge messages are generated if commands with long latency have been executed in order to inform the ***Command*** message sender about the results. Acknowledge messages are only generated

if the initiating ***Command*** message has specified the *@AcknowledgeURL* Attribute or a pair of *@AcknowledgeFormat* and *@AcknowledgeTemplate* Attributes.

Figure 5-3: Interaction of Command and Acknowledge Messages



They are announced in the ***Response*** message to the Command message by the setting the Attribute *@Acknowledged* = "true".

Table 5.9: Acknowledge Message Element

NAME	DATA TYPE	DESCRIPTION
AcknowledgeType = "Completed" New in JDF 1.1	enumerations	Defines the context of this message. This is necessary mainly for Device-Machine pairs where the Machine is not accessible online. **Allowed values are:** Received – The initiating ***Command*** has been received and understood (e.g., by an operator). Applied – The initiating ***Command*** has been applied to the Machine (e.g., by an operator). Completed – The initiating ***Command*** has been executed. No further acknowledgement will be sent after an acknowledgement with *@AcknowledgeType* = "Completed" has been sent.
refID	NMTOKEN	Identifies the initiating ***Command*** message that the ***Acknowledge*** refers to.
ReturnCode = "0"	integer	Describes the result. "0" indicates success. For all other possible codes see ▸ Appendix C Return Values and Error Codes.
Notification ? Modified in JDF 1.1A	element	Textual description of the command execution. See ▸ Section 3.11.4.5 Notification.
ResponseTypeObj *	element	Abstract Element that is a placeholder for any descriptive Elements that provide details about command execution. Delayed Acknowledge messages contain the same ***ResponseTypeObj*** Elements as direct ***Response*** messages.

Example 5.6: Acknowledge Message

The following is an example of an Acknowledge message:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
       TimeStamp="2000-07-25T12:32:48+02:00" MaxVersion="1.4" Version="1.4"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Acknowledge ID="M109" Type="PipePush" xsi:type="AcknowledgePipePush" refID="M010">
       <JobPhase JobID="J1" JobPartID="1" Status="InProgress"/>
    </Acknowledge>
</JMF>
```

5.2.6 Registration

New in JDF 1.3

A ***Registration*** message is a request to the recipient of the **JMF** to send ***Command*** messages to a Command recipient who is specified in ***Subscription***. See ▸ Section 5.3.4.2 Persistent Channels for Commands for details on persistent channels for Commands.

Table 5.10: Registration Message Element

NAME	DATA TYPE	DESCRIPTION
CommandTypeObj *	element	Abstract Elements that provide details of the ***Command*** that is setup by this ***Registration*** message.
Subscription	element	Creates a persistent channel for a Command. For the structure of a ***Subscription*** Element, see ▸ Section 5.3.4 Persistent Channels.

5.3 JMF Handshaking

JMF can seek to establish communication between system components in several ways. This section describes the actions and appropriate reactions in a communication using **JMF**.

5.3.1 Single Query/Command Response Communication

The handshaking mechanisms for queries and commands are equivalent. The initiating Controller sends a Query message or Command message to the target Controller. The target parses the ***Query*** or ***Command*** and immediately issues an appropriate ***Response*** message. If a ***Command*** with long latency is issued, an additional ***Acknowledge*** message MAY be sent to acknowledge when the command has been executed.

5.3.2 Signal and Acknowledge Handshaking

By default, **JMF** Signal messages and Acknowledge messages are "fire and forget." In case of success, no Response message is sent by the receiver besides the standard protocol HTTP response with an empty body. If an error occurred at the receiver's end, the ***Signal*** or ***Acknowledge*** receiver SHOULD return an error Response message as defined in ▸ Section 5.5 Error and Event Messages.

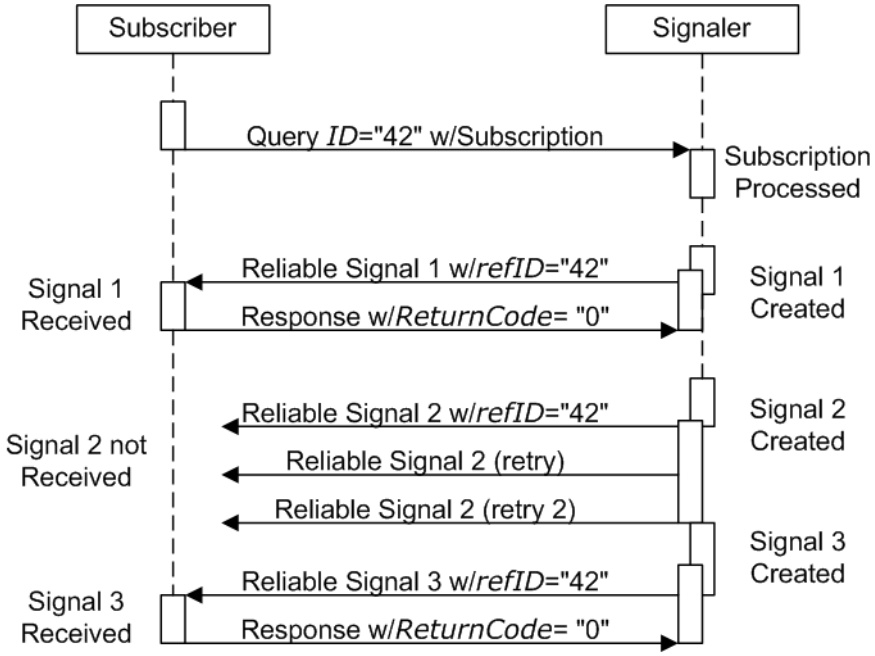
Any Response related to a ***Signal*** or ***Acknowledge*** message SHALL NOT specify that an Acknowledge will be sent (the Acknowledged attribute SHALL be set to false). This is due to the fact that Signal and Acknowledge messages inherently forbid the use of an Acknowledge in Response, since they do not have an *@AcknowledgeURL* to indicate where these Acknowledge messages should be sent.

5.3.3 Reliable Signalling

If reliable signaling has been specified when the persistent channel is set up (see ▸ Table 5.11 Subscription Element), then the receiver of the **JMF** Signal SHALL respond to the message using a **JMF** Response that indicates the appropriate value for the *@ReturnCode* Attribute. If the receiver does not respond to the reliable signal, the sender SHALL retry the reliable signal, based on the *@RetryPolicy* specified in the original ***Subscription*** Element. If a Response is received with a

@*ReturnCode* value other than zero, then the Signal message MAY have to be retried, depending on the ***Error***/@*Resend* attribute in the Response.

Figure 5-4: Example of Reliable Signaling



5.3.4 Persistent Channels

Query and ***Command*** messages are subscribed for using ***Subscription*** Elements.

5.3.4.1 Persistent Channels for Signals

Queries are made persistent by including a ***Subscription*** Element that defines the persistent channel-receiving end (see also ▸ Figure 5-1: JMF Root Element – a diagram of its structure). The responding Controller SHOULD initially send a Response message to the subscribing Controller. Then the responding Controller SHOULD send Signal messages whenever the condition specified by one of the Attributes in the following table is true. This is referred to as a **persistent channel**. The @*refID* Attribute of the ***Signal*** is defined by the @*ID* Attribute of the ***Query***. In other words, the @*refID* of the signal identifies the persistent channel. Any ***Query*** can be set up as a persistent channel, although in some cases this might not make sense.

5.3.4.2 Persistent Channels for Commands

New in JDF 1.3

Commands can also be subscribed for by using a ***Subscription*** Element in an initial ***Registration***. A ***Subscription*** in a ***Registration*** defines a request for the initial Registration message receiver to subsequently send ***Command*** messages to the recipient defined in ***Subscription***/@URL or ***Subscription***/@Format + ***Subscription***/@Template. For instance, an MIS might send a ***Registration*** to a prepress workflow system that directs the prepress workflow system to send ***Command*** messages to a press Controller whenever a plate or preview has been produced.

5.3.5 Subscription

Whether or not a responding Controllers implements a **JDF** Persistent Channel as an HTTP/1.1 ▸ [RFC2616] persistent connection depends on implementation.

Table 5.11: Subscription Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ChannelMode ? New in JDF 1.4	enumerations	Specifies reliability of persistent channel, and whether it is required or just preferred. Ordered list, with most preferred channel mode first. If none of the provided values of @*ChannelMode* are supported by the consumer of the subscription, the Response should indicate @*ReturnCode* 111, which is "Subscription request denied". **Allowed value is from:** ▸ ChannelMode **Note:** See ▸ Table 5.2.3 Signal.

Table 5.11: Subscription Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Format ? New in JDF 1.2 Deprecated in JDF 1.5 *Unidirectional*	string	A formatting string used with the @*Template* Attribute to define a sequence of generated URLs. **Allowed values are from:** ▸ Appendix H Dynamic String Generation. **Constraint:** If @*Format* is specified, then @*Template* SHALL also be specified and @*URL* SHALL NOT be specified
Languages ? New in JDF 1.6	languages	List of languages selected for human readable communication. If not specified, the operating system language SHALL be used. If multiple languages are specified, the second and further languages SHOULD only be used for providing additional localized ***Comment*** elements. Messages SHALL NOT be sent multiple times for the same event.
MinDelayTime ? New in JDF 1.3	duration	Minimum delay between two subsequent ***Signal*** messages that are triggered by this ***Subscription***. If not specified a ***Signal*** SHOULD be fired when any of the conditions described in ***Subscription*** is met. Note that ***Signal*** messages that would be fired before @*MinDelayTime* are lost. @*MinDelayTime* SHOULD NOT be applied to ***Signal*** messages that affect costing. Reliable ***Signal*** messages SHALL NOT be retried more frequently than the interval specified by @*MinDelayTime*.
RepeatStep ?	integer	Requests an update signal whenever the @*ActualAmount* associated with the query is an integer multiple of @*RepeatStep*. If not specified, it is up to the sending Controller to generate signals.
RepeatTime ?	double	Requests an update signal every @*RepeatTime* seconds. If defined, the signal is generated periodically independent of any other trigger conditions. @*RepeatTime* SHALL NOT override any Signals triggered by a change of status. Signals triggered by a status change SHALL be sent regardless of the value of @*RepeatTime*. A sender MAY restart counting for @*RepeatTime* based Signals whenever it sends a Signal to the same subscription.
RetryPolicy ? New in JDF 1.4	enumeration	For reliable subscriptions. Indicates whether or not signals should be retried indefinitely, or only until the next Signal from the same ***Subscription*** (i.e., has the same @*refID*) would be sent. @*RetryPolicy* is ignored for non-reliable subscriptions. **Allowed values are:** DiscardAtNextSignal – if a Signal has not been received, and it is time to send the next Signal related to this ***Subscription*** (the next Signal specifies the same @*refID* value), then discard the current Signal. RetryForever – Continue retrying every Signal indefinitely.
Template ? New in JDF 1.2 Deprecated in JDF 1.5 *Unidirectional*	string	A template, used with @*Format*, to define a sequence of generated URLs. **Allowed values are from:** ▸ Appendix H Dynamic String Generation. **Constraint:** if @*Template* is specified, then @*Format* SHALL also be specified and @*URL* SHALL NOT be specified.
URL ? Modified in JDF 1.2	URL	URL of the persistent channel receiving end. The protocol of the ***Subscription*** is specified by the scheme of @*URL*. **Note:** Starting with **JDF** 1.5, this attribute is no longer specified as "*Bidirectional*" because unidirectional attributes are deprecated.
ObservationTarget *	element	Requests an updating Signal message whenever the value of one of the Attributes specified in ***ObservationTarget*** changes.

5.3.5.1 ObservationTarget

Table 5.12: ObservationTarget Element

NAME	DATA TYPE	DESCRIPTION
Attributes ? Deprecated in JDF 1.2	NMTOKENS	Requests an update signal whenever the value of one of the Attributes specified by @*Attributes* is modified. A value of "*" denotes a message request for any Attribute change which is the default. **Deprecation note:** Replaced with @*ObservationPath* in **JDF** 1.2 and above.
ElementIDs ? Deprecated in JDF 1.2	NMTOKENS	IDs of the Elements that contain Attributes that can change. Used only in conjunction with a query of the state change of a certain Resource or Node which cannot uniquely be addressed by the other Attributes of this Element. **Deprecation note:** Replaced with @*ObservationPath* in **JDF** 1.2 and above.
ElementType ? Deprecated in JDF 1.2	NMTOKEN	Name of the Element that contains Attributes that can change. Defaults to the abstract ***ResponseTypeObj*** of the message. **Deprecation note:** Replaced with @*ObservationPath* in **JDF** 1.2 and above.
ObservationPath ? New in JDF 1.2	XPath	XPath of the Elements or Attributes that are observed. The XPath is in the context of the resulting **JMF**. If not specified, a ***Signal*** is emitted on any change in the abstract ***ResponseTypeObj*** of the message.

If a Controller that does not support persistent channels is queried to set up a persistent channel, it SHALL answer the Query message with a Response message, set @*Subscribed* to "false", and set the @*ReturnCode* to "111".

Multiple Attributes of a ***Subscription*** Element are combined as a Boolean OR operation of these Attributes. For instance, if @*RepeatStep* and @*ObservationTarget* are both specified, messages fulfilling either of the requirements are requested. If the ***Subscription*** Element contains only a URL, it is up to the emitting Controller to define when to emit messages.

5.3.6 Scope of Subscriptions

New in JDF 1.5.

Note: In general, Subscriptions SHOULD be as global in scope as possible. For instance, it is preferable to create one global Status ***Subscription*** for all job related and job unrelated messages rather than creating a new Status ***Subscription*** for each individual queue entry.
Deprecation note: Starting with **JDF** 1.5, support for job and queue entry specific subscriptions is deprecated.

5.3.7 Deleting Persistent Channels

A persistent channel SHALL be deleted by sending a ***StopPersistentChannel*** Command message, as described in ▸ Section 5.56 StopPersistentChannel.

5.4 JMF Messaging Levels

A **JDF**-conforming Controller MAY opt to support one of the following messaging compliance levels offered by **JMF**:

- No messaging — Controllers have the option of supporting no messaging at all. For this level, **JDF** includes ***Audit*** records for each Process that allow the results of the Process to be recorded.
- Notification — Most Controllers will choose to support some level of messaging capability. Notification is the most basic level of support. Devices that support notification provide unidirectional messaging by sending Signal messages. Notification messages inform the Controller when they begin and complete execution of some Process within a Job. They MAY also provide notice of some error conditions. Setup of the notification channel is hard-wired.
- Query support — The next level of communication supports queries. Controllers that support queries respond to requests from other Controllers by communicating their status using such tools as current @*JobID* Attributes, queued @*JobID* Attributes or current Job progress.
- Command support — This level of support provides Controllers with the ability to Process commands. The Controller can receive commands, for instance, to interrupt the current Job, to restart a Job, or to change the status of Jobs in a queue.
- Submission support — Finally, Controllers MAY accept **JDF** Jobs via an HTTP post request to the messaging channel. In this case, the messaging channel SHALL support MIME Multipart/Related documents. For more details on submission, see ▸ Section 5.57 SubmissionMethods.

What's your JMF SOP?

As part of your strategic equipment purchasing procedures and requirements, consider what the JDF Messaging Levels are desired, and what the minimum level of conformance will be for your new equipment purchases.

Each messaging level encompasses all of the lower messaging levels. Note that the message levels are provided for information and are not normative.

5.5 Error and Event Messages

If an Acknowledge message, Command message, Query message, Signal message or a Registration message is not successfully handled, a processor SHALL reply with a standardized error response that may contain a ***Notification*** Element. ***Notification*** Elements, described in detail in ▸ Section 3.11.4.5 Notification, convey a textual description. The information contained in the ***Notification*** element can be used by a user interface to visualize errors.

The Response messages and Acknowledge messages contain a @*ReturnCode* Attribute. @*ReturnCode* defaults to 0, which indicates that the response is successful. In case of success and in responses to commands an informational ***Notification*** Element (@*Class* = "Information") MAY be provided. In case of a warning, error or fatal error, the @*ReturnCode* is greater than 0 and indicates the kind of error committed. In this case, a ***Notification*** Element SHOULD be provided. Error codes are defined in ▸ Appendix C Return Values and Error Codes. The responding application SHOULD fill additional ***Notification***/***Error*** Elements that describe the details of the error.

Example 5.7: Response with Notification Element

The following example uses a ***Notification*** Element to describe an error:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
       TimeStamp="2013-03-25T12:32:48+02:00"
       MaxVersion="1.5" Version="1.5"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Response ID="M109" ReturnCode="5" Type="ResumeQueueEntry"
          xsi:type="ResponseResumeQueueEntry" refID="M009">
       <Notification Class="Error" TimeStamp="2005-03-25T12:32:48+02:00" Type="Error">
          <Comment>StartJob unsuccessful - Device does not handle commands</Comment>
          <Error ErrorID="1234" Resend="Prohibited">
             <ErrorData Path="/JMF/Command" ErrorType="Unsupported"/>
          </Error>
       </Notification>
    </Response>
</JMF>
```

5.6 Message Template

The previous sections in this chapter provide a description of the overall structure of **JMF** messages. This section contains a list of the standard messages that are defined within the **JDF** framework. It is OPTIONAL for a **JDF**-compliant application to support each ***Signal*** message or ***Query*** message described in this list. It is, however, possible to discover which messages are supported in a workflow. A Controller responds to the ***KnownMessages*** Query message by publishing a list of all the messages it supports (see ▸ Section 5.29 KnownMessages, below).

At the beginning of each section there is a table that lists all of the message types in that category. These tables contain three columns. The first is entitled "Message Type," and it lists the names of each message type. The second column is entitled "Family." The values in this (family) column describe the kind of message Element that is applicable in the circumstance being illustrated. The following abbreviations are used to describe the values used in the tables below to describe these major message Element types.
Note: That these are XML elements that are direct children of the **JMF** element.

C: ***Command***

G: ***Registration*** ("G" is the third letter)

Q: ***Query***

R**:** ***Response*** and ***Acknowledge***

S: ***Signal***

More than one of these values can be valid simultaneously. If that is the case, then all applicable letters are included in the column. Additionally, there are a few special circumstances indicated by particular combinations of these letters. The letters "QR" or "CR" indicate that all ***Query*** messages and ***Command*** messages cause a ***Response*** message to be returned. If the message can occur as a ***Signal*** message, either from a ***Subscription*** or independently, the "Family" field in the table also contains the letter "S". Finally, the third column provides a description of each Element.

At the beginning of each section describing the contents and function of the message types listed in the tables described above is a table containing the instantiation (i.e., the type) of all of the abstract Subelements applicable to the message being described. Each table contains an entry that describes the details of the ***Query*** message or ***Command*** message as

well as an additional entry that describes the details of the corresponding response. The tables resemble the following template:

Table 5.13: Template for Message tables

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
Abstract Subelement type of the ***Query*** or ***Command***.	Name and type of the Subelement that defines specifics of the ***Query*** message or ***Command*** message, followed by a cardinality symbol.	Short description of the Subelement(s) if applicable.
Abstract Subelement type of the ***Response*** or ***Acknowledge***.	Name and type of Subelement that contains specific information about the ***Response*** message or ***Acknowledge*** message to a ***Query*** message or ***Command*** message followed by cardinality symbol.	Short description of the Subelement(s) if applicable.

5.6.1 Object Type Column

Each message in the remainder of this chapter has two cells in the Object Type column. The first is either ***QueryTypeObj*** or ***CommandTypeObj***. The second is always a ***ResponseTypeObj***.

5.6.1.1 QueryTypeObj

A ***QueryTypeObj*** is an abstract Element that is a placeholder for Subelements of a ***Query*** or ***Signal*** message. See ***Query***/***QueryTypeObj*** (▸ Table 5.3 Query Message Element) and ***Signal***/***QueryTypeObj*** (▸ Table 5.5 Signal Message Element). ***QueryTypeObj*** also appears in the first row of the Object Type column for each Query message below. For each such Query message, the corresponding Elements in the Element Name column are intended to replace the ***QueryTypeObj*** in ***Query***/***QueryTypeObj*** or ***Signal***/***QueryTypeObj***.

5.6.1.2 CommandTypeObj

A ***CommandTypeObj*** is an abstract Element that is a placeholder for Subelements of a ***Command*** or ***Registration*** message. See ***Command***/***CommandTypeObj*** (▸ Table 5.4 Command Message Element) and ***Registration***/***CommandTypeObj*** (▸ Table 5.10 Registration Message Element). ***CommandTypeObj*** also appears in the first row of the Object Type column for each Command message below. For each such Command message, the corresponding Elements in the Element Name column are intended to replace the ***CommandTypeObj*** in ***Command***/***CommandTypeObj*** or ***Registration***/***CommandTypeObj***.

5.6.1.3 ResponseTypeObj

A ***ResponseTypeObj*** is an abstract Element that is a placeholder for Subelements of a ***Response***, ***Signal*** or ***Acknowledge*** message. See ***Response***/***ResponseTypeObj*** (▸ Table 5.8 Response Message Element), ***Signal***/***ResponseTypeObj*** (▸ Table 5.8 Response Message Element) and ***Acknowledge***/***ResponseTypeObj*** (▸ Table 5.9 Acknowledge Message Element). ***CommandTypeObj*** also appears in the second row of the Object Type column for each message below. For each such message, the corresponding Elements in the Element Name column are intended to replace the ***ResponseTypeObj*** in the ***Response***/***ResponseTypeObj***, ***Signal***/***ResponseTypeObj*** or ***Acknowledge***/***ResponseTypeObj***.

5.7 List of All JMF Messages

Table 5.14: List of JMF Messages (Sheet 1 of 3)

MESSAGE TYPE	FAMILY	DESCRIPTION
AbortQueueEntry	CR	The ***QueueEntry*** is aborted and remains in the ***Queue*** with ***QueueEntry***/@*Status* = "Aborted".
CloseQueue	CR	The queue is closed. No Jobs are to be accepted by the queue.
Events Deprecated in JDF 1.5	QRS	Used to subscribe pure events occurring randomly like scanning of a bar code, activation of function keys at a console, error messages, etc.
FlushQueue	CQRS	All entries in the queue are removed.
FlushResources New in JDF 1.2	CQRS	Remove temporary Resource from a Device.
ForceGang New in JDF 1.3	CR	A gang is forced to execute.

Table 5.14: List of JMF Messages (Sheet 2 of 3)

MESSAGE TYPE	FAMILY	DESCRIPTION
GangStatus New in JDF 1.3	CR	The status of a gang is queried.
HoldQueue	CR	The queue is held. No Jobs within the queue are to be executed.
HoldQueueEntry	CR	The entry remains in queue but is not executed until a ***ResumeQueueEntry*** Command message is received.
KnownControllers Deprecated in JDF 1.5	QRS	Returns a list of **JMF**-capable Controllers.
KnownDevices	QRS	Returns information about the Devices that are controlled by a Controller.
KnownJDFServices Deprecated in JDF 1.2	QRS	Returns a list of services (**JDF** Node Types) that are defined in the **JDF** specification.
KnownMessages	QRS	Returns a list of all messages that are supported by the Controller.
KnownSubscriptions New in JDF 1.4	QRS	Returns a list of active persistent channels.
ModifyNode New in JDF 1.3	CRS	modifies details of **JDF** Nodes.
NewJDF New in JDF 1.2	CQRS	Initiates or reports modifications of new **JDF** Nodes.
NodeInfo New in JDF 1.2 Deprecated in JDF 1.3	CQRS	Initiates or reports modifications of **JDF** Node information (e.g., scheduling).
Notification	S	Used to signal usual events due to any activities of a Device, operator, etc. (e.g., scanning a bar code).
Occupation Deprecated in JDF 1.5	QRS	Queries the occupation of an employee.
OpenQueue	CR	The queue is opened. Jobs are to be accepted.
PipeClose	CR	Closes a pipe because no further Resources are needed. This is typically used to terminate the producing Process.
PipePause	CR	Pauses a Process if no further Resources can be consumed or produced.
PipePull	CR	Requests a new Resource from a pipe.
PipePush	CR	Notifies that a new Resource is available in a pipe.
QueueEntryStatus Deprecated in JDF 1.2	QRS	Returns a ***QueueEntry*** Element.
QueueStatus	QRS	Returns the ***Queue*** Elements that describe a queue or set of queues.
RemoveQueueEntry	CR	A Job is removed from the queue.
RepeatMessages Deprecated in JDF 1.5	QR	Returns a set of previously sent messages that have been stored by the Controller.
RequestForAuthentication New in JDF 1.4	CQRS	Used as a Command to exchange certificates or as a Query to obtain the authentication status of previously exchanged certificates.

Table 5.14: List of JMF Messages (Sheet 3 of 3)

MESSAGE TYPE	FAMILY	DESCRIPTION
RequestQueueEntry New in JDF 1.2	CR	A new Job is requested by the Device. This message is used to signal that a Device has processing Resources available.
Resource	CGQRS	Queries and/or modifies **JDF** Resources that are used by a Device, such as Device settings, or by a Job. This message can also be used to query the level of ***Consumable Resource*** Elements in a Device.
ResourcePull New in JDF 1.2	CGR	Creates a new ***QueueEntry*** from an already existing ***QueueEntry*** and submits it to the queue in order to be executed.
ResubmitQueueEntry	CR	Replaces a queue entry without affecting the entry's parameters. The command is used, for example, for late changes to a submitted **JDF**.
ResumeQueue	CR	The queue is activated and queue entries are to be executed.
ResumeQueueEntry	CR	A held Job is resumed. The Job is re-queued at the position defined by its current priority. Submission time is set to the current time stamp.
ReturnQueueEntry New in JDF 1.2	CR	Returns a Job that had been submitted with a ***SubmitQueueEntry*** to the queue that represents the Controller that originally submitted the Job.
SetQueueEntryPosition	CR	Queues a Job behind a given position n, where n represents a numerical value. "0" = pole position. Priority is set to the priority of the Job at position n.
SetQueueEntryPriority	CR	Sets the priority of a queued Job to a new value. This does not apply to Jobs that are already running.
ShutDown New in JDF 1.2	CR	Shuts down a Device.
Status	QRS	Queries the general status of a Device, Controller or Job.
StopPersistentChannel	CR	Closes a persistent channel.
SubmissionMethods	QR	Queries a list of supported submission methods to the queue.
SubmitQueueEntry	CR	A Job is submitted to a queue in order to be executed.
SuspendQueueEntry New in JDF 1.2	CR	The entry is suspended if it is already running. It remains suspended until a ***ResumeQueueEntry*** Command message is received.
Track Deprecated in JDF 1.5	QRS	Queries the location of a given Job or Job Part.
UpdateJDF New in JDF 1.3	CRS	Synchronizes and relinks modified **JDF** Nodes.
WakeUp New in JDF 1.2	CR	Wakes up a Device that is in standby mode.

5.8 Messages for Events and Capabilities

The message types of the following table are defined in order to exchange metadata about Controller or Device abilities and for general communication.

Table 5.15: Messages for events and capabilities (Sheet 1 of 2)

MESSAGE TYPE	FAMILY	DESCRIPTION
Events Deprecated in JDF 1.5	QRS	Used to subscribe pure events occurring randomly like scanning of a bar code, activation of function keys at a console, error messages, etc.
KnownControllers Deprecated in JDF 1.5	QRS	Returns a list of **JMF**-capable Controllers.

Table 5.15: Messages for events and capabilities (Sheet 2 of 2)

MESSAGE TYPE	FAMILY	DESCRIPTION
KnownDevices	QRS	Returns information about the Devices that are controlled by a Controller.
KnownJDFServices Deprecated in JDF 1.2	QRS	Returns a list of services (**JDF** Node Types) that are defined in the **JDF** specification.
KnownMessages	QRS	Returns a list of all messages that are supported by the Controller.
KnownSubscriptions New in JDF 1.4	QRS	Returns a list of active persistent channels.
Notification	QRS	Generally sent as Signals. A Query allows Subscriptions for ***Notification*** messages.
RepeatMessages Deprecated in JDF 1.5	QR	Returns a set of previously sent messages that have been stored by the Controller.
RequestForAuthentication New in JDF 1.4	CQR	Used as a Command to exchange certificates or as a Query to obtain the authentication status of previously exchanged certificates.
StopPersistentChannel	CR	Closes a persistent channel.

5.9 Messages to Query/Command a Job, Device or Controller

JDF Messaging provides methods to trace the status of individual Devices and Resources and additional Job-dependent Job-tracking data. The status of a Job is described by the ***Status*** Elements of that Job.

Devices are uniquely identified by a *name* — that is, by the Attribute @*DeviceID* of the ***Device*** Resource (see ▸ Section 8.44 Device) — while Controllers are uniquely identified by their URL. In other words, Controllers are implicitly identified as a result of the fact that they are responding to a message. One Controller MAY control multiple Devices. The following queries and commands are defined for status and progress tracking.

Table 5.16: Messages to query/affect a Job, Device or Controller (Sheet 1 of 2)

MESSAGE TYPE	FAMILY	DESCRIPTION
FlushResources New in JDF 1.2	CQRS	Remove temporary Resources from a Device.
ModifyNode New in JDF 1.3	CRS	Modifies details of **JDF** Nodes that have previously been submitted to a Device.
NewJDF New in JDF 1.2	CQRS	Initiates or reports modifications of new **JDF** Nodes.
NodeInfo New in JDF 1.2 Deprecated in JDF 1.3	CQRS	Initiates or reports modifications of **JDF** Node information (e.g., scheduling). Use either ***Resource Command*** messages with ***ResourceCmdParams***/@*ResourceName* = "NodeInfo" or ***Resource Query*** messages with ***ResourceQuParams***/@*ResourceName* = "NodeInfo" instead.
Occupation Deprecated in JDF 1.5	QRS	Queries the occupation of an employee. **Deprecation note:** Use ***Status*** signals with ***JobPhase***/***Activity*** or ***DeviceInfo***/***Activity*** instead.
Resource	CGQRS	Queries and/or modifies **JDF** Resources that are used by a Device, such as Device settings, or by a Job. This message can also be used to query the level of consumables in a Device.
ResourcePull New in JDF 1.2	CGR	Creates a new ***QueueEntry*** from an already existing ***QueueEntry*** and submits it to the queue in order to be executed.
ShutDown New in JDF 1.2	CR	Shuts down a Device.

Table 5.16: Messages to query/affect a Job, Device or Controller (Sheet 2 of 2)

MESSAGE TYPE	FAMILY	DESCRIPTION
Status	QRS	Queries the general status of a Device, Controller or Job.
Track Deprecated in JDF 1.5	QRS	Queries the location of a given Job or Job Part.
UpdateJDF New in JDF 1.3	CRS	Synchronizes and relinks modified **JDF** Nodes.
WakeUp New in JDF 1.2	CR	Wakes up a Device that is in standby mode.

5.10 Messages for Pipe Control

JDF Messaging provides methods to control dynamic pipes. Dynamic pipes are described in detail in ▸ Section 4.3.3 Overlapping Processing Using Pipes.

Table 5.17: Messages for Control of Dynamic Pipes

MESSAGE TYPE	FAMILY	DESCRIPTION
PipeClose	CR	Closes a pipe because no further Resources are needed. This is typically used to terminate the producing Process.
PipePause	CR	Pauses a Process if no further Resources can be consumed or produced.
PipePull	CGR	Requests a new Resource from a pipe.
PipePush	CGR	Notifies that a new Resource is available in a pipe.

5.10.1 Common PipeControl Element

5.10.1.1 PipeParams

The ***PipeParams*** Element is also used by the messages ***PipePull***, ***PipePush*** and ***PipePause***.

Table 5.18: PipeParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
JobID ? New in JDF 1.2	string	Specifies the @*JobID* of the process at the receiving end of the message that links to the resource specified in @*PipeID*.
JobPartID ? New in JDF 1.2	string	Specifies the @*JobPartID* of the process at the receiving end of the message that links to the resource specified in @*PipeID*.
PipeID	string	Pipe ID of the **JDF** resource that defines the dynamic pipe.
ProjectID ? New in JDF 1.5	string	Specifies the @*ProjectID* of the Node at the receiving end of the message that links to the Resource specified in @*PipeID*.
Status = "InProgress"	enumeration	Process status after the request. **Allowed values are** from: ***JDF***/@*Status*(▸ Table 3.4 JDF).
UpdatedStatus ?	enumeration	This value represents the actual status of the pipe resource and MAY be used by the receiving process for process termination control. For details see ▸ Section 4.3.5.2 Formal Iterative Processing. **Allowed values are** from: ***Resource***@*Status* (▸ Table 3.10 Abstract Resource Element).
AmountPool ? New in JDF 1.5	element	Updated ***AmountPool*** for the pipe Resource. The ***AmountPool***/***PartAmount***/***Part*** MAY contain additional metadata related to the updated Resource. The ordering of the ***PartAmount*** elements in the ***AmountPool*** is relevant.

Table 5.18: PipeParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Resource *	element	Updated resources to be used by the process that receives the pipe command:***PipePull*** (the receiver creates the pipe Resource), ***PipePush*** (the receiver consumes the pipe Resource) and ***PipePause*** (the receiver only updates the inputs). Possible commands are: ***PipePull***, ***PipePush*** or ***PipePause***. In case of the ***PipeClose*** Command message, the Resources are ignored. The data type and @*Class* of ***Resource*** is derived from the Abstract Resource. See ▸ Section 3.8.3 Abstract Resource.
ResourceLink ? Deprecated in JDF 1.5	element	Updated ***ResourceLink*** to the pipe Resource: ***PipePull*** (it is an output link), ***PipePush*** (it is an input link) and ***PipePause*** (depends on the pipe end). This ***ResourceLink*** MAY be used by the Process that links to the pipe Resource. The Attributes @*rRef* and @*Usage* of a ***ResourceLink*** SHALL NOT be modified by the Agent that sends the Pipe Control message because these Attributes are used by the **JMF** receiver to identify the ***ResourceLink*** that is to be modified. In case of the ***PipeClose*** Command message, the ***ResourceLink*** is ignored. **Deprecation note:** Starting with **JDF** 1.5, ***AmountPool*** replaces ***ResourceLink***. This change allows for amounts and partitions without using @*rRef* and @*Usage*. The Resource is identified by @*PipeId*).

5.11 Queue Support

In **JMF**, a Controller or Device is assumed to have one input queue that accepts submitted Jobs. Controllers which receive submitted Jobs SHALL in turn submit these Jobs to lower level Controllers or Devices to pass the submission on. In other words, Job submission "cascades" down through Controllers until they get to the Device. Similarly, ***ReturnQueueEntry*** messages "cascade" back up through each level. If a Machine supports multiple queues, it SHALL be represented by multiple logical Devices in **JDF**. In other words, a Device SHALL NOT have more than one Queue. The simple case of a Device with no queue can be mapped to a queue with two @*Status* states: "Waiting" and "Full". **JMF** supports simple handling of priority queues. The following assumptions are made:

- Queues support priority. Priority SHALL only be changed for waiting Jobs. A queue MAY round priorities to the number of supported priorities, which MAY be one, indicating no priority handling.
- Priority is described by an integer from 0 to 100. Priority 100 defines a Job that SHOULD pause another Job that is in progress and commence immediately. If a Device does not support the pausing of running Jobs, it SHOULD queue a priority 100 Job before the last pending priority 100 Job.
- A Controller MAY control multiple Devices/Queues.
- Queue entries can be unambiguously identified by a @*QueueEntryID*.
- A Controller or Device MAY analyze a **JDF** that is submitted to a queue at submission or execution time. A Queue MAY treat a **JDF** as a closed envelope that is passed on to the Device without checking. The behavior is implementation dependent.

Some conventions used in the following sections have already been introduced in ▸ Section 5.6 Message Template. This affects the message Families and the descriptive tables at the beginning of each message section that describe the type objects related to the corresponding message. The type objects are ***QueryTypeObj***, ***CommandTypeObj*** and ***ResponseTypeObj*** (see also ▸ Figure 5-1: JMF Root Element – a diagram of its structure).

5.11.1 Queue Entry ID Generation

Queue entries are accessed using a @*QueueEntryID* Attribute, which the queue's Controller or Device generates when it receives the submitted Job, and which is returned in the ***SubmitQueueEntry*** Response message. @*QueueEntryID* SHALL uniquely identify an entry within the scope of one queue. An implementation is free to choose the algorithm that generates @*QueueEntryID* values.

5.12 Messages for Queue Entry Handling

Queue-entry handling is provided so that the state of individual Jobs within a queue can be changed. Job submission, queue-entry grouping, priorities and hold / suspend / resume of entries are all supported. The individual commands are defined in the table and explained in greater detail in the sections that follow.

Starting with **JDF** 1.5, the ***Queue*** Element is deprecated in the response to all queue entry handling messages. The ***QueueFilter*** that limits the ***Queue*** is also deprecated in the respective commands and queries. The status of the resulting queue SHOULD therefore be queried with an explicit ***QueueStatus*** message. See ▸ Section 5.41 QueueStatus.

Table 5.19: Messages for queue entry handling

MESSAGE TYPE	FAMILY	DESCRIPTION
AbortQueueEntry Modified in JDF 1.2	CR	The ***QueueEntry*** is aborted and remains in the ***Queue*** with ***QueueEntry***/@*Status* = "Aborted".
HoldQueueEntry	CR	The entry remains in queue but is not executed until a ***ResumeQueueEntry*** Command message is received.
RemoveQueueEntry	CR	A Job is removed from the queue.
RequestQueueEntry New in JDF 1.2	CR	A new Job is requested by the Device. This message is used to signal that a Device has processing capabilities available.
ResubmitQueueEntry	CR	Replaces a queue entry without affecting the entry's parameters. The command is used, for example, for late changes to a submitted **JDF**.
ResumeQueueEntry	CR	A held Job is resumed. The Job is re-queued at the position defined by its current priority. Submission time is set to the current time stamp.
ReturnQueueEntry New in JDF 1.2	CR	Returns a Job that had been submitted with a ***SubmitQueueEntry*** to the queue that represents the Controller that originally submitted the Job.
SetQueueEntryPosition	CR	Queues a Job behind a given position n, where n represents a numerical value. "0" = pole position. Priority is set to the priority of the Job at position n.
SetQueueEntryPriority	CR	Sets the priority of a queued Job to a new value. This does not apply to Jobs that are already running.
SubmitQueueEntry	CR	A Job is submitted to a queue in order to be executed.
SuspendQueueEntry New in JDF 1.2	CR	The entry is suspended if it is already running. It remains suspended until a ***ResumeQueueEntry*** Command message is received.

The following table specifies the status transitions for the respective queue entry handling messages. The error(n) indicates the ReturnCode which is returned on an illegal Status transition and the queue entry Status is unchanged. For details on error codes, see ▸ Appendix C Return Values and Error Codes.

The following are codes for the following table:

A: Aborted

C: Completed

H: Held

PR: PendingReturn New in JDF 1.4

Rm: Removed

Rn: Running

S: Suspended

W: Waiting

number: Error that specified number (e.g., "105" means "error(105)"). .

Table 5.20: Status Transitions for QueueEntry Handling Messages (Sheet 1 of 2)

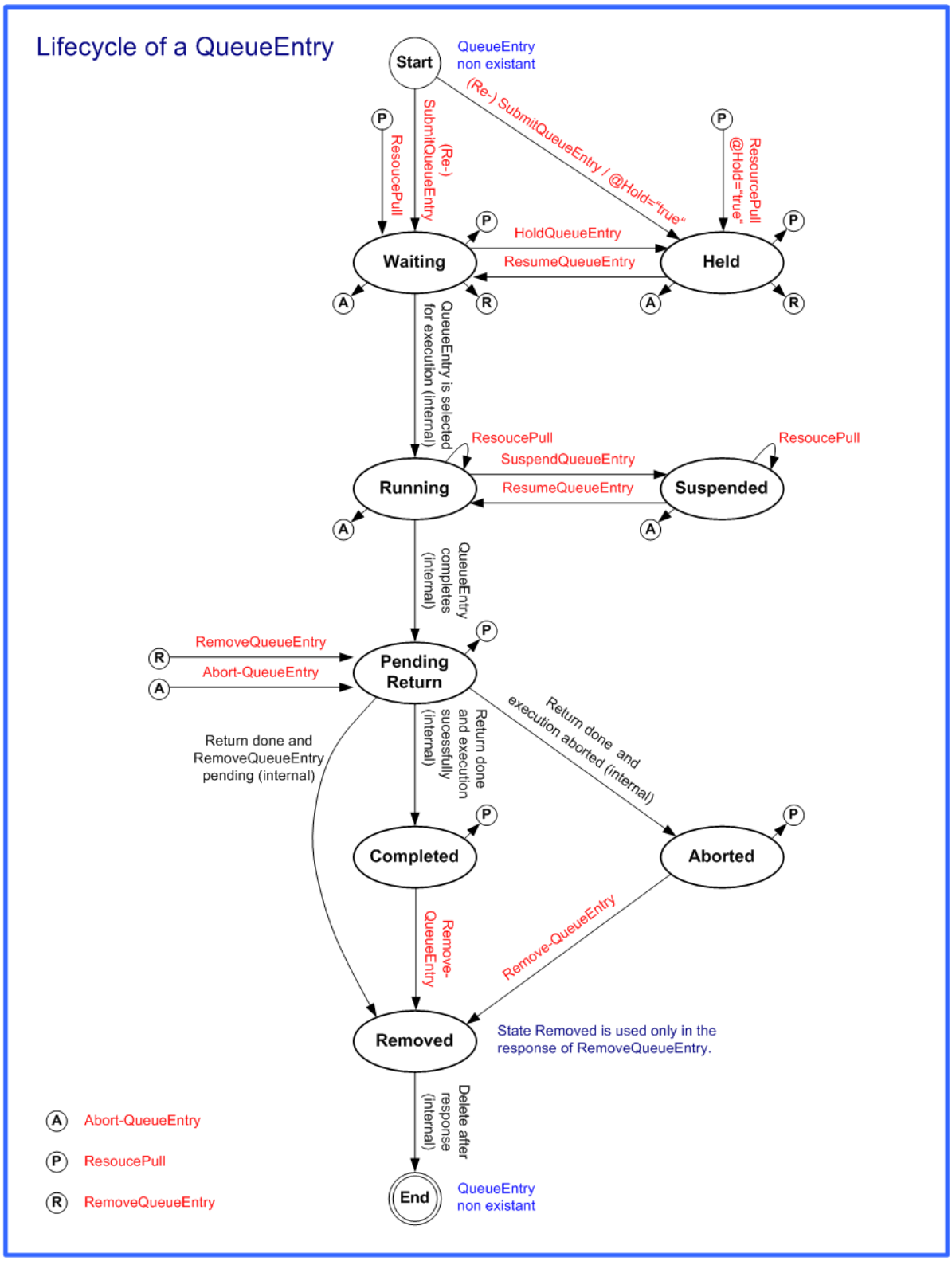
PREVIOUS STATUS MESSAGE TYPE	NON-EXISTENT	W	H	RN	S	PR	C	A
AbortQueueEntry	105	A	A	A	A	114	114	113
HoldQueueEntry	105	H	113	106	106	114	114	114
RemoveQueueEntry	105	Rm	Rm	106	106	106	Rm	Rm
ResumeQueueEntry	105	113	W	113	R/W	114	114	114
SetQueueEntryPosition	105	W	H	107	107	114	114	114

Table 5.20: Status Transitions for QueueEntry Handling Messages (Sheet 2 of 2)

PREVIOUS STATUS MESSAGE TYPE	NON-EXISTENT	W	H	RN	S	PR	C	A
SetQueueEntryPriority	105	W	H	107	107	114	114	114
SuspendQueueEntry	105	115	115	S	113	114	114	114
RequestQueueEntry	***RequestQueueEntry*** is emitted by the Controller of the queue and not sent to the queue. Therefore it is not applicable in this section.							
ResubmitQueueEntry	105	W	H	Rn + W + 107	S + 107	114	Rn + W + 114	Rn + W + 114
ReturnQueueEntry	***ReturnQueueEntry*** is emitted by the Controller of the queue and not sent to the queue. Therefore it is not applicable in this section.							
SubmitQueueEntry	W,H, Rn	A new @*QueueEntryID* is generated by the queue owner on submission. Therefore these states are not applicable.						

The following @*Status* transition diagram depicts the life cycle of a queue entry.

Figure 5-5: JMF QueueEntry Status Transition Diagram



5.13 Messages for Global Handling of Queues

Whereas the commands in the preceding section change the state of an individual queue entry, the commands in this section modify the state of an entire queue. Note that entries that are executing in a Device are not affected by the global queue-handling commands and SHALL be accessed individually. An individual queue can be selected by specifying the target Device in the @*DeviceID* Attribute of the **JMF** Root. If no @*DeviceID* is specified, the commands or queries are ap-

plied to all queues that are controlled by the Controller that received the message. The following individual messages are defined:

Table 5.21: Messages for global handling of queues

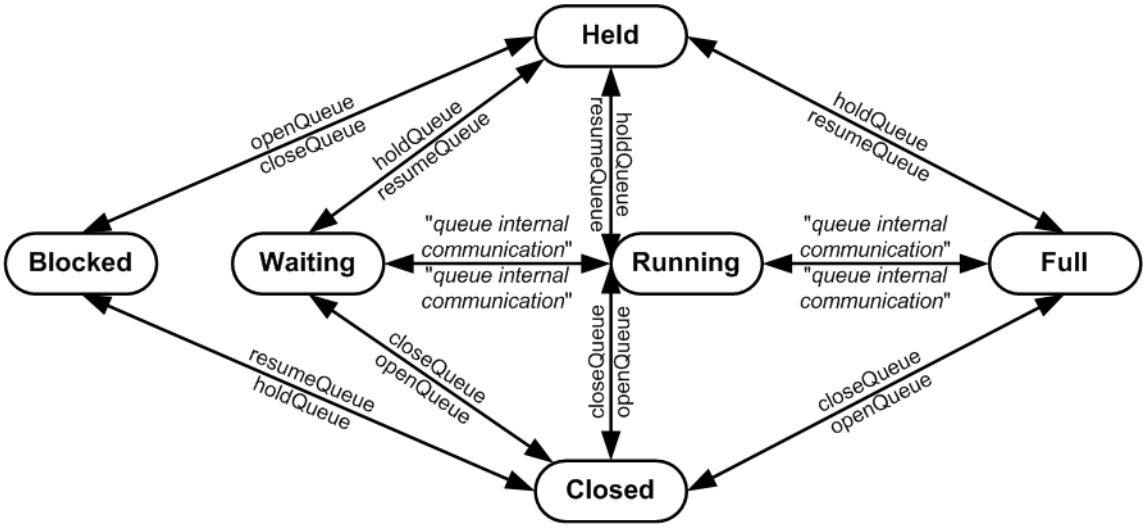
MESSAGE TYPE	FAMILY	DESCRIPTION
FlushQueue	CQRS	All entries in the queue are removed.
SubmissionMethods	QR	Queries a list of supported submission methods to the queue.
CloseQueue	CR	The queue is closed. No Jobs are to be accepted by the queue.
FlushQueue	CQRS	All entries in the queue are removed.
HoldQueue	CR	The queue is held. No Jobs within the queue are to be executed.
OpenQueue	CR	The queue is opened. Jobs are to be accepted.
QueueEntryStatus Deprecated in JDF 1.2	QRS	Returns a ***QueueEntry*** Element.
QueueStatus	QRS	Returns the ***Queue*** Element that describes a queue.
ResumeQueue	CR	The queue is activated and queue entries are to be executed.
SubmissionMethods	QR	Queries a list of supported submission methods to the queue.

The following table shows the resulting status of a ***Queue*** in dependence on global queue commands ***CloseQueue***/***OpenQueue*** and ***HoldQueue***/***ResumeQueue***as well as the load of queue and its processor. The first command pair determines the logical state of the first column “Closed” and the second of the column “Held”. The ***Queue*** is held if the ***Queue*** manager doesn't send existing entries to the ***Queue***'s processor.

Table 5.22: Definition of the Queue Status Attribute Values

CLOSED	HELD	QUEUE FULL	PROCESSOR FULL	STATUS
Yes	Yes	Any	Any	"Blocked"
Yes	No	Any	Any	"Closed"
No	Yes	Any	Any	"Held"
No	No	Any	No	"Waiting"
No	No	No	Yes	"Running"
No	No	Yes	Yes	"Full"

Figure 5-6: Effects of the global queue Messages on the queue Status



5.13.1 QueueEntryStatus

Deprecated in JDF 1.2

Deprecation note: Starting with **JDF** 1.2, use ***QueueStatus*** with an appropriate ***QueueFilter*** instead of ***QueueEntryStatus***. See ▸ Section N.4.9 QueueEntryStatus for details of this deprecated **JMF** element.

5.14 Elements for Queues

In this section Elements used by queue-handling commands are defined.

5.14.1 Queue

The Attributes in the following table are defined for ***Queue*** message Elements. ***Queue*** Elements represent the queue of a Device including ***QueueEntry*** Elements that represent both pending and running queue entries.

Table 5.23: Queue Element

NAME	DATA TYPE	DESCRIPTION
DeviceID	string	Identifies the Device that is represented by the queue.
MaxQueueSize ? New in JDF 1.6	integer	The maximum number of ***QueueEntry*** Elements excluding ***QueueEntry***[@*Status* = "Completed" or ***QueueEntry***[@*Status* ="Aborted"] Elements that can be contained in the ***Queue***.
QueueSize ? New in JDF 1.2 Modified in JDF 1.6	integer	The total number of ***QueueEntry*** Elements that are in the ***Queue***.
Status	enumeration	Status of the queue. **Allowed values are:** Blocked – Queue is completely inactive. Entries SHALL NOT be added and no entries are executed. The queue is closed and held. The queue requires an interaction like ***OpenQueue*** or ***ResumeQueue*** to reactivate it. Closed – Queue entries that are in the queue are executed, but new entries SHALL NOT be submitted. The lock SHALL be removed explicitly by the ***OpenQueue*** Command message. Full – Queue entries that are in the queue are executed but new entries SHALL NOT be submitted. The lock is removed by the queue Controller as soon as it is able to do so. Running – A Process is executing. Entries can be submitted and will be executed when they reach their turn in the queue. Waiting – Queue accepts new entries and has free Resources to immediately commence processing. Held – Entries can be submitted but will not be executed until the queue is resumed by the ***ResumeQueue*** Command message.
Device *	element	The Devices that execute entries in this queue. Only ***Device***/@*DeviceID* SHOULD be specified in these ***Device*** Elements.
QueueEntry * Modified in JDF 1.2 Modified in JDF 1.6	element	***QueueEntry*** Elements (see ▸ Table 5.24 QueueEntry Element, below). The entries SHALL be ordered in the sequence they have been or will be executed, beginning with the running entries, followed by the waiting entries, highest ***QueueEntry***/@*Priority* first, which are then followed by the completed entries, sorted beginning with the youngest ***QueueEntry***/@*EndTime*. The ***Queue*** contains a list of all ***QueueEntry*** Elements that are still accessible on the Device using the queue entry handling messages that are defined in ▸ Table 5.24 QueueEntry Element. A ***QueueEntry*** is not automatically deleted when executed or aborted, but rather it remains in the ***Queue*** and its @*Status* is changed to "Completed" or "Aborted" accordingly. ***QueueEntry***[@*Status* = "Completed" or @*Status* = "Aborted"] Elements SHALL NOT count towards determining ***Queue***/@*Status* based on the number of ***QueueEntry*** Elements versus the @*MaxQueueSize*.

Example 5.8: Queue Element

```
<Queue DeviceID="Q12345" Status="Running">
    <QueueEntry JobID="111" JobPartID="1" Priority="1" QueueEntryID="111-1"
          Status="Running"/>
    <QueueEntry JobID="111" JobPartID="2" Priority="1" QueueEntryID="111-2"
          Status="Waiting"/>
    <QueueEntry JobID="112" JobPartID="1" Priority="55" QueueEntryID="112-1"
          Status="Held"/>
    <QueueEntry JobID="111" JobPartID="0" Priority="1" QueueEntryID="111-0"
          Status="Completed"/>
</Queue>
```

5.14.2 QueueEntry

Table 5.24: QueueEntry Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Activation ? New in JDF 1.5	enumeration	Specifies the activation of the requested ***QueueEntry***. **Allowed values are** from: @*Activation* in ▸ Table 3.4 JDF.
DeviceID ? New in JDF 1.2	string	Identification of the Device that the ***QueueEntry*** will be or was executed on. If not specified, it defaults to the default Device of the queue.
EndTime ? New in JDF 1.2	dateTime	Time when the Job has been ended.
GangName ? New in JDF 1.3	NMTOKEN	Name of the gang that this ***QueueEntry*** belongs to. @*GangName* SHALL be specified, if the ***QueueEntry*** is a candidate member of a gang Job.
GangPolicy ? New in JDF 1.3	enumeration	Ganging policy for the ***QueueEntry***. **Allowed value is from:** ▸ GangPolicy
JobID ? Modified in JDF 1.1	string	The @*JobID* of the **JDF** Process.
JobPartID ?	string	The @*JobPartID* of the **JDF** Process.
Priority = "1"	integer	Priority of the ***QueueEntry***. Values are 0-100."0" is the lowest priority, while "100" is the highest priority.
QueueEntryID	string	ID of a ***QueueEntry***. This ID SHALL be generated by the queue owner.
StartTime ? New in JDF 1.1	dateTime	Time when the Job has been started.

Table 5.24: QueueEntry Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Status Modified in JDF 1.3	enumeration	Status of the individual entry. **Allowed values are:** Running – The queue entry is running on the Device. A ***QueueEntry*** is "Running" when ***JDF***/@*Status*.of any node associated to the ***QueueEntry*** is one of "Setup", "InProgress" or "Cleanup". Waiting – The queue entry is waiting and will be executed when Resources are available. Held – The queue entry is held and SHALL NOT execute until resumed. A held ***QueueEntry*** with @*GangPolicy* other than "NoGang" does not interact with its respective gang. Removed – The queue entry has been removed. This status can only be sent when a persistent channel watches a queue and the queue entry is removed. Suspended – The queue entry was running and has been held. It will not continue to execute until resumed. A ***QueueEntry*** is "Suspended" when the ***QueueEntry*** has been suspended using the ***SuspendQueueEntry***/@*Operation*="Suspend" UI equivalent on the device. New in JDF 1.2 PendingReturn – Indicates that the ***QueueEntry*** has been executed correctly, and is finished, but that the corresponding **JDF** has not yet been successfully returned to the respective Controller. New in JDF 1.3 Completed – Indicates that the Node or queue entry has been executed correctly, and is finished. For ***QueueEntry***.New in JDF 1.2 Aborted – Indicates that the Process executing the Node has been aborted, which means that execution will not be resumed again. For ***QueueEntry***.New in JDF 1.2
StatusDetails ? New in JDF 1.5	string	@*StatusDetails* provides additional details on the status of the ***QueueEntry***. **Values include:** HeldForResourcePull – When @*Status* is "PendingReturn", Job is not returned on purpose, commands ***ResourcePull***, ***RemoveQueueEntry*** or ***AbortQueueEntry*** are possible JobUserInputRequired – When @*Status* is"Waiting" or "Running", Job is not producible and waits for user input required to process further (e.g., missing parameters, decisions, etc.) JobMissResources – When @*Status* is "Waiting" or "Running", Job waits for resources to become available to process further JobReadyForStart – When is @*Status* "Waiting" or "Running", Job is ready and waits for (manual) start event to process further QueuedToRun – When @*Status* is"Waiting" or "Running", Job is queued to run and waits for device to become available (idle) to process further PendingReturn – When @*Status* is "PendingReturn", Job is currently returning (explicit "PendingReturn" to distinguish from devices/controllers that do not support @*StatusDetails*) Running – When @*Status* is "Running", Job is processing (explicit Running to distinguish from devices/controllers that do not support @*StatusDetails*)
SubmissionTime ?	dateTime	Time when the entry was submitted to the queue.
GangSource * New in JDF 1.6	element	If present, each ***GangSource*** SHALL represent the source jobs that are being processed as a gang job by this ***QueueEntry***.
JobPhase * New in JDF 1.2	element	Description of the current status of the Job that is associated with the ***QueueEntry***. Note that in **JDF** 1.3 and above, one ***QueueEntry*** MAY have multiple active ***JobPhase*** Elements.
Part * New in JDF 1.2	element	Describes which parts of a Job were submitted to the queue. The ***Part*** Elements are copies of ***AncestorPool***/***Part*** of the **JDF** Node that is executed by the Device.
Preview * New in JDF 1.2	element	Any number of ***Preview*** Elements MAY be associated with a ***QueueEntry*** and used for display purposes. ***Preview***/@*PreviewUsage* SHOULD be "ThumbNail" or "Viewable".

5.14.3 QueueEntryDef

The Element specifies a queue entry and is used to refer to a certain queue entry.

Table 5.25: QueueEntryDef Element

NAME	DATA TYPE	DESCRIPTION
QueueEntryID	string	ID of the queue entry. The ID is generated by the queue owner.

5.14.4 QueueFilter

New in JDF 1.2

The ***QueueFilter*** element defines a filter for all messages that return a queue. The supplied elements of the ***QueueFilter*** define a matching criteria that is a logical "and". Only ***QueueEntry*** elements that match all restrictions specified by the ***QueueFilter*** are included in the ***Queue*** Element that is returned by the queue-handling message. The ***QueueFilter*** Element is also used to specify the ***QueueEntry*** Elements to be removed by the ***FlushQueue*** message.

Table 5.26: QueueFilter Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Activation ? New in JDF 1.5	enumeration	The activation state of the ***QueueEntry*** elements to be returned. If not specified, there is no filtering on ***QueueEntry***/@*Activation*. **Allowed values are** from: @*Activation* in ▸ Table 3.4 JDF.
FirstEntry ? New in JDF 1.6	string	@*QueueEntryID* of the first ***QueueEntry*** that this ***QueueFilter*** applies to. Only ***QueueEntry*** elements that are behind this (including this) ***QueueEntry*** in the current queue sorting SHALL be selected
GangNames ? New in JDF 1.3	NMTOKENS	Gang names of the ***QueueEntry*** Elements to be returned. If not specified, there is no filtering on ***QueueEntry***/@*GangName*.
JobID ? New in JDF 1.4	string	Return only ***QueueEntry*** elements with specified @*JobID*. If not specified, there is no filtering on ***QueueEntry***/@*JobId*.
JobPartID ? New in JDF 1.4	string	Return only ***QueueEntry*** elements with specified @*JobPartID*. If not specified, there is no filtering on ***QueueEntry***/@*JobPartID*.
LastEntry ? New in JDF 1.6	string	@*QueueEntryID* of the last ***QueueEntry*** that this ***QueueFilter*** applies to. Only ***QueueEntry*** elements that are in front of this (including this) ***QueueEntry*** in the current queue sorting SHALL be selected
MaxEntries ?	integer	Maximum number of ***QueueEntry*** elements to provide in the ***Queue*** element. If not specified, fill in all matching ***QueueEntry*** elements.
MaxPriority ? New in JDF 1.6	integer	Only ***QueueEntry*** elements with a @*Priority* lower than or equal to the value of @*MaxPriority* SHALL be provided in the ***Queue*** element. If not specified, there is no @*Priority* upper bound on candidates.
MinPriority ? New in JDF 1.6	integer	Only ***QueueEntry*** elements with a @*Priority* higher than or equal to the value of @*MinPriority* SHALL be provided in the ***Queue*** element. If not specified, there is no @*Priority* lower bound on candidates.
NewerThan ?	dateTime	Only ***QueueEntry*** elements with a @*SubmissionTime* newer than or equal to this dateTime SHALL BE provided in the ***Queue*** Element or removed by the ***FlushQueue*** message. If not specified, there is no dateTime upper bound on candidates.
OlderThan ?	dateTime	Only ***QueueEntry*** Elements with a @*SubmissionTime* older than or equal to this dateTime SHALL BE provided in the ***Queue*** Element or removed by the ***FlushQueue*** message. If not specified, there is no dateTime lower bound on candidates.

Table 5.26: QueueFilter Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
PreviewUsages ? New in JDF 1.4	enumerations	Specifies the particular kind (or kinds) of ***Preview*** resources to return in ***QueueEntry***/***Preview***. If @*PreviewUsages* is empty or not supplied, the ***QueueEntry*** element SHALL not contain any ***Preview*** resources. The ***Preview*** resources returned in a ***QueueEntry*** are a subset of those in the actual ***QueueEntry*** defined by: ***QueueEntry***/***Preview*** [contains (***QueueFilter***/@*PreviewUsages*, @*PreviewUsage*)] **Allowed values are** from: ***Preview***/@*PreviewUsages* (▸ Table 8.213 Preview Resource).
QueueEntryDetails = "Brief" Modified in JDF 1.4	enumeration	Refines the level of provided information about the queue. **allowed values are:** None – Do not fill in the ***QueueEntry*** elements into the queue. Brief – Provide all available ***QueueEntry*** information except for the associated ***JobPhase*** element. JobPhase – Provide all available ***QueueEntry*** information including the associated ***JobPhase*** elements. ***JobPhase***/@*URL* MAY be returned when @*QueueEntryDetails* = "JobPhase". JDF – Provide all available ***QueueEntry*** information including the associated ***JobPhase*** element and the associated **JDF** element in the ***JobPhase*** element. Deprecated in JDF 1.4 **Deprecation note:** Starting with **JDF** 1.4, use the ***Status*** query to retrieve status information including information about the current **JDF**.
StatusList ?	enumerations	Only ***QueueEntry*** Elements with a @*Status* matching one of the entries in @*StatusList* SHALL be returned. considered. If not specified, there is no filtering on ***QueueEntry***/@*Status*. **Allowed values are** from: ▸ NodeStatus.
UpdateGranularity ? New in JDF 1.4	enumeration	Specifies whether all or only the updated ***QueueEntry*** elements should be included in the ***Queue***. **Allowed values are:** All – The ***Queue*** element describes all ***QueueEntry*** elements. ChangesOnly – The ***Queue*** element describes only those ***QueueEntry*** elements that have new information since the last ***Queue*** element was sent. When used in conjunction with a signal, the ***Queue*** element describes all jobs on the first instance of the signal being sent.
Device *	element	Only ***QueueEntry*** elements that match the attribute values specified in one of these ***Device*** resources SHALL be included. ***QueueEntry*** elements match the criteria if the attribute values specified in one of these ***Device*** resource match the equivalent attribute values of the devices that are processing the ***QueueEntry***. Unspecified attributes always match. If ***Device*** is not specified, all ***QueueEntry*** elements are returned. As this is a filter, only information that can be used to identify a ***Device*** SHALL be specified. This precludes use of ***DeviceCap*** and ***IconList*** in this ***Device***.
GangSource * New in JDF 1.6	element	If present, each ***GangSource*** SHALL represent the source jobs that are being processed as a gang job during this ***JobPhase***.
Part * New in JDF 1.4	element	Only ***QueueEntry*** elements with all specified ***Part*** elements SHALL BE returned. If not specified, there is no filtering on ***QueueEntry***/***Part***.
QueueEntryDef *	element	Defines an explicit list of queue entries. If not specified, all entries in the ***Queue*** are considered.

5.15 Gang Jobs

New in JDF 1.3

JMF provides a mechanism to specify groups of *QueueEntry* Elements within a queue that are processed together in a gang. A Job is submitted to a gang by specifying *QueueSubmissionParams*/@*GangPolicy*. The details of how individual job parts are ganged are device specific. For a description of planned job ganging, see also ▸ Section 6.3.38 SheetOptimizing.

Table 5.27: Messages for Gang Jobs

MESSAGE TYPE	FAMILY	DESCRIPTION
ForceGang New in JDF 1.3	CR	A gang is forced to execute.
GangStatus New in JDF 1.3	CR	The status of a gang is queried.

5.16 Extending Messages

This specification defines a set of predefined messages for general usage. Extensions to existing messages and additional message types can be defined using the standard extension rules described in ▸ Section 3.12 JDF Extensibility. Note, the generic content of ▸ Section 3.1 Generic Contents of All Elements is also valid for **JMF** Elements. It is not allowed to define message extensions which duplicate the functionality of messaging types, messaging Elements or message Attributes that are already defined in this specification.

For example the content of the @*Type* Attribute MAY be specified with a prefix that identifies the organization that defined the extension. The prefix and name SHOULD be separated by a single colon (':'). Any additional Attributes and Elements are allowed, and internal Elements MAY be declared with explicit namespaces. The official namespace of **JMF** Elements is @*xmlns* = "http://www.CIP4.org/JDFSchema_1_1". This namespace is identical to that defined for **JDF** in ▸ Section 3.12 JDF Extensibility. An example is provided:

Example 5.9: Custom Query

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Circus"
       TimeStamp="2005-07-25T12:32:48+02:00"
       MaxVersion="1.4" Version="1.4"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:Circus="Circus Schema URI">
    <Query ID="Q1" Type="Circus:IsClownHappy" xsi:type="QueryIsClownHappy">
       <Circus:ClownParams Gender="male"/>
    </Query>
</JMF>
```

Example 5.10: Custom Response

The Response message will also have the "Circus:" namespace identifier. All *Circus* Elements are explicitly declared.

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Circus 2"
       TimeStamp="2005-07-25T12:32:48+02:00"
       MaxVersion="1.4" Version="1.4"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:Circus="Circus Schema URI">
    <Response ID="M1" Type="Circus:IsClownHappy" xsi:type="ResponseIsClownHappy"
           refID="Q1">
       <Circus:Clown happy="true" name="Joe"/>
       <Circus:Clown happy="false" name="John"/>
    </Response>
</JMF>
```

5.16.1 IfraTrack Support

The extending mechanism can be used to implement compatibility with other XML-based messaging standards, for example version 3.0 of IfraTrack. The @*Type* Attribute is set to the appropriate namespace, and the foreign message is included, as demonstrated in the following example:

Note that the application is free to select the appropriate response types in order to fulfill its local (IfraTrack) protocol requirements if it uses its own namespace. In the ex-

More on IfraTrack

IfraTrack is a specification for the interchange of status and management information between local and global production management systems in newspaper production. For more information on IfraTrack, including a case study paper, please see http://www.ifra.com/WebSite/news.nsf/(Structured-SearchAll)?OpenAgent&IFRATRACK

amples below, the default namespace associated with the **JMF** Query message and Response Elements has been overwritten by the Ifra namespace.

Example 5.11: Custom Query for IfraTrack

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="IFRA"
       TimeStamp="2003-07-25T12:32:48+02:00"
       MaxVersion="1.4" Version="1.4"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:IFRA="IfraTrack URI">
    <Query ID="Q1" Type="IFRA:IMF" xsi:type="QueryIMF">
       <imf:IMF xmlns:imf="IfraTrack URI">
          Whatever you want (might be multiple top level Elements)
       </imf:IMF>
    </Query>
</JMF>
```

Example 5.12: Custom Response for IfraTrack

The legal Response message would be:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="IFRA"
       TimeStamp="2003-07-25T12:32:48+02:00"
       MaxVersion="1.4" Version="1.4"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:IFRA="IfraTrack URI">
    <Response ID="M1" Type="IFRA:IMF" xsi:type="ResponseIMF" refID="Q1">
       <imf:IMF xmlns:imf="IfraTrack URI">
          The appropriate IFRA response(s)
       </imf:IMF>
    </Response>
</JMF>
```

5.17 AbortQueueEntry

Once this command is issued, the entry specified by ***AbortQueueEntryParams***/***QueueFilter*** is stopped or aborted and remains in the ***Queue*** with ***QueueEntry***/@*Status* = "Aborted" or "Completed" depending on the value of ***AbortQueueEntryParams***/@*EndStatus*. The ***Audit*** elements and ***JDF***/@*Status* of the processing **JDF** Node are to be appropriately set to "Aborted"or "Completed" and the **JDF** Node SHALL be delivered to the URL as specified by ***SubmitQueueEntry***/@*ReturnURL*, ***SubmitQueueEntry***/@*ReturnJMF* or ***NodeInfo***/@*TargetRoute*.

Table 5.28: AbortQueueEntry Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.2 Modified in JDF 1.5	***AbortQueueEntryParams*** ? New in JDF 1.5	
	QueueEntryDef Deprecated in JDF 1.5	Defines the queue entry or set of queue entries. **Deprecation note:** Starting with **JDF** 1.5, this ***QueueEntryDef*** SHOULD be located in ***AbortQueueEntryParams***/***QueueFilter***.
	QueueFilter ? New in JDF 1.2 Deprecated in JDF 1.5	Defines a filter for the returned ***Queue*** Elements in the ***AbortQueueEntry*** message.
ResponseTypeObj Modified in JDF 1.5	***Queue*** Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.

For the definition of the Elements listed above, see ▸ Section 5.14 Elements for Queues.

5.17.1 AbortQueueEntryParams

New in JDF 1.5

Table 5.29: AbortQueueEntryParams Element

NAME	DATA TYPE	DESCRIPTION
EndStatus	enumeration	End Status of the job after completing processing. **Allowed values are:** Completed Aborted
QueueFilter ?	element	This ***QueueFilter*** selects the ***QueueEntry*** elements to apply this message to.

Example 5.13: AbortQueueEntry Command

The following example demonstrates how an ***AbortQueueEntry*** Command message causes a Job in a queue to be aborted and only return the @*Status* of the aborted ***QueueEntry*** in the response, rather than the entire ***Queue***:

```
<Command ID="M009" Type="AbortQueueEntry" xsi:type="CommandAbortQueueEntry">
    <AbortQueueEntryParams>
        <QueueFilter>
            <QueueEntryDef QueueEntryID="job-0032"/>
        </QueueFilter>
    </AbortQueueEntryParams>
</Command>
```

Example 5.14: AbortQueueEntry Response

The following example shows a possible Response message to the Command message example above:

```
<Response ID="M109" Type="AbortQueueEntry" xsi:type="ResponseAbortQueueEntry"
        refID="M009" ReturnCode="0">
</Response>
```

5.18 CloseQueue

The queue is closed. No further queue entries are accepted by the queue. The status of entries that are already in the queue remains unchanged and entries that are already in the ***Queue*** MAY be executed.

Table 5.30: CloseQueue Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.5	***QueueFilter*** ? New in JDF 1.2 Deprecated in JDF 1.5	Defines a filter for the returned ***Queue*** Element in the ***CloseQueue*** message.
ResponseTypeObj Modified in JDF 1.5	***Queue*** Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.

5.19 Events

Deprecated in JDF 1.5

Starting with **JDF** 1.5, the functionality of ***Events*** can be achieved using a subscription to ***Notification*** messages. For details of the deprecated ***Events*** message, see ▸ Section N.4.2 Events.

5.20 FlushQueue

5.20.1 FlushQueue Command

FlushQueue Command is used to remove ***QueueEntry*** Elements from the ***Queue***.
Note: A ***QueueEntry*** is not automatically deleted when executed or aborted, but rather it remains in the ***Queue*** and its @*Status* is changed to "Completed" or "Aborted" accordingly. ***FlushQueueParams*** allows the specification of which

QueueEntry Elements to remove. The *QueueFilter* in the *FlushQueue* message is applied to the *Queue* returned after the command is executed. The *QueueFilter* contained within the *FlushQueueParams* is used to specify which *QueueEntry* Elements to remove.

Table 5.31: FlushQueue Command Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.5	*FlushQueueParams* ? New in JDF 1.2	Defines the *QueueEntry* Elements to be removed. If not specified, then only pending (i.e., @*Status* = "Waiting" and @*Status* = "Held" queue entries are removed).
	QueueFilter ? New in JDF 1.2 Deprecated in JDF 1.5	Defines a filter for the returned *Queue* Element in the *FlushQueue* message.
ResponseTypeObj Modified in JDF 1.5	*FlushQueueInfo* ? New in JDF 1.2	Defines the *QueueEntry* Elements that were removed.
	Queue Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.

5.20.1.1 FlushQueueParams

New in JDF 1.2

Table 5.32: FlushQueueParams Element

NAME	DATA TYPE	DESCRIPTION
QueueFilter ?	element	Defines a *QueueFilter* that specifies the *QueueEntry* Elements to be removed. If not specified, the *Queue* is completely flushed.

5.20.2 FlushQueue Query

When used as a Signal or Query, *FlushQueue Query* allows a Controller to monitor queue flushing that is initiated by the Device (e.g., due to Resource constraints). The *QueueFilter* in the *FlushQueue* message is applied to the *Queue* returned after the command is executed. The *QueueFilter* contained within the *FlushQueueInfo* is used to specify which *QueueEntry* Elements were removed.

Table 5.33: FlushQueue Query Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj Modified in JDF 1.5	*FlushQueueParams* New in JDF 1.5	Defines a *QueueFilter* that specifies the *QueueEntry* Elements to be removed. If not specified, the *Queue* is completely flushed.
	QueueFilter ? Deprecated in JDF 1.5	Defines a filter for the returned *Queue* Element in the *FlushQueue* message.
ResponseTypeObj Modified in JDF 1.5	*FlushQueueInfo* ? New in JDF 1.2	Defines the *QueueEntry* Elements that were removed.
	Queue Deprecated in JDF 1.5	Describes the state of the queue after the Elements have been flushed.

5.20.2.1 FlushQueueInfo

New in JDF 1.2

The *QueueFilter* in *FlushQueueParams* defines the *QueueEntry* Elements to be removed by *FlushQueue*. Those *QueueEntry* Elements meeting the criteria set in the *QueueFilter* will be removed.

Table 5.34: FlushQueueInfo Element

NAME	DATA TYPE	DESCRIPTION
QueueFilter	element	Defines a *QueueFilter* that specifies the *QueueEntry* Elements that were removed. Typically *QueueFilter* contains a set of *QueueEntryDef* elements that specify the *QueueEntry* elements that were removed.

5.21 FlushResources

New in JDF 1.2

The *FlushResources* message is used to remove temporary Resources from a Device. *FlushResourceParams* specifies the Resources to remove.

The *Command* allows a Controller to Request that a Device actively Flush its resources whereas the *Query* or *Signal* allows a Device to message that it has flushed resources to a Controller.

5.21.1 FlushResources Command

The *FlushResources Command* is used to remove temporary Resources from a Device. *FlushResourceParams* allows the specification of which Resources to remove.

Table 5.35: FlushResources Command

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	*FlushResourceParams* ?	Defines the Resources to be removed.
	QueueFilter ? Deprecated in JDF 1.5	Defines a filter for the returned *Queue* Element in the *FlushResources* message.
ResponseTypeObj		This Element is a placeholder for future use.

5.21.2 FlushResources Query

The *FlushResources Query* is used to query whether temporary Resources have been removed by a Device. *FlushResourceParams* allows the specification of which Resources were removed.

Table 5.36: FlushResources Query

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	*FlushResourceParams* ?	Defines the Resources to be removed.
	QueueFilter ? Deprecated in JDF 1.5	Defines a filter for the returned *Queue* Element in the *FlushResources* message.
ResponseTypeObj		This Element is a placeholder for future use.

5.21.2.1 FlushResourceParams

Table 5.37: FlushResourceParams Element

NAME	DATA TYPE	DESCRIPTION
FlushPolicy = "QueueEntry"	enumeration	Policy that defines how much of the ***QueueEntry*** Resources is requested to be flushed. **Allowed values are:** Complete – Remove all temporary Resources belonging to the selected ***QueueEntry*** including global resources that MAY be used by other ***QueueEntry*** elements. QueueEntry – The local Resources belonging to the selected ***QueueEntry*** are completely removed and no longer available — the default. Intermediate – Remove any intermediate Resources that belong to the ***QueueEntry*** (e.g., intermediate raster files in a combined RIP and Image-Setting Process), and retain the original Input Resources. A ***ResourcePull*** message is still possible after executing ***FlushResources*** with @*FlushPolicy* = "Intermediate".
QueueFilter ?	element	Defines a ***QueueFilter*** that specifies the ***QueueEntry*** Elements to which the Resources to be removed belong. If not specified, all temporary resources on the Device are completely flushed according to the value of @*FlushPolicy*.

5.22 ForceGang

New in JDF 1.3

The ***ForceGang*** message forces all ***QueueEntry*** [@*Status* = "Waiting"] Elements that belong to a gang (as specified below) to be executed, even though the Device dependent queue entry collecting algorithm might not be completed. A ***QueueEntry*** belongs to a gang if ***QueueEntry***/@*GangName* is included in the list of ***GangCmdFilter***/@*GangNames*.

Table 5.38: Contents of the ForceGang Command Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	***GangCmdFilter***	Defines the gang(s) to be force executed.
ResponseTypeObj	—	

5.22.1 GangCmdFilter

Table 5.39: GangCmdFilter Element

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
GangNames ?	NMTOKENS	@*GangName* of the gang(s) being queried.

5.23 GangStatus

New in JDF 1.3

GangStatus returns a description of the gang(s). Details are specified in ***GangInfo*** Element.

Table 5.40: GangStatus Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***GangQuFilter*** ?	Defines a filter for the gang(s) that are queried. If ***GangQuFilter*** is not supplied, all gangs are queried.
ResponseTypeObj	***GangInfo*** *	Describes the status of the gang(s).

5.23.1 GangQuFilter

Table 5.41: GangQuFilter Element

NAME	DATA TYPE	DESCRIPTION
GangNames ?	NMTOKENS	@*GangName* of the gang(s) being queried.

5.23.2 GangInfo

Details of the gang are specified in ***GangInfo*** elements. ***GangInfo*** is a placeholder for future gang related information that only returns the gang names in **JDF** 1.3.

Table 5.42: GangInfo Element

NAME	DATA TYPE	DESCRIPTION
GangName	NMTOKEN	Name of the gang.

5.24 HoldQueue

The queue is held. No entries will start execution. Note that the status of a held entry prior to ***HoldQueue*** is retained so that held Jobs remain held after a ***ResumeQueue***. New entries can still be submitted to a held queue. ***HoldQueue*** only has effect on Jobs that have not commenced processing. Queue entries that are already running SHALL be suspended individually using the ***SuspendQueueEntry*** Command message.

Table 5.43: HoldQueue Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.5	***QueueFilter*** ? New in JDF 1.2 Deprecated in JDF 1.5	Defines a filter for the returned ***Queue*** Element in the ***HoldQueue*** message.
ResponseTypeObj Modified in JDF 1.5	***Queue*** Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.

5.25 HoldQueueEntry

The entry specified by ***HoldQueueEntryParams***/***QueueFilter*** remains in the queue but is not executed. If its @*Status* is "Waiting", its @*Status* is set to "Held". The ***HoldQueueEntry*** Command message has no effect on Jobs with a @*Status* other than "Waiting". If ***QueueEntry***/@*GangPolicy* is other than "NoGang", a held ***QueueEntry*** retains its respective gang data but does not influence execution of other Jobs that are in the gang. For details, see ▸ Table 5.20 Status Transitions for QueueEntry Handling Messages.

Table 5.44: HoldQueueEntry Message (Sheet 1 of 2)

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.5	***HoldQueueEntryParams*** ? New in JDF 1.5	
	QueueEntryDef Deprecated in JDF 1.5	Defines the queue entry.
	QueueFilter ? New in JDF 1.2 Deprecated in JDF 1.5	Defines a filter for the returned ***Queue*** Elements in the ***HoldQueueEntry*** message.

Table 5.44: HoldQueueEntry Message (Sheet 2 of 2)

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
ResponseTypeObj Modified in JDF 1.5	*Queue* Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.

5.25.1 HoldQueueEntryParams

New in JDF 1.5

Table 5.45: HoldQueueEntryParams Element

NAME	DATA TYPE	DESCRIPTION
QueueFilter ?	element	This *QueueFilter* selects the *QueueEntry* elements to apply this message to.

5.26 KnownControllers

Deprecated in JDF 1.5

Starting with **JDF** 1.5, use *KnownDevices*. For details of the deprecated *KnownControllers* message, see ▸ Section N.4.3 KnownControllers.

5.27 KnownDevices

The *KnownDevices* query message requests information about the devices that are controlled by a controller. If a high level controller controls lower level controllers, it SHOULD also list the devices that are controlled by these. The response is a *DeviceList* which is a list of *DeviceInfo* elements controlled by the controller that receives the query, as demonstrated in ▸ Example 182 KnownDevices Response.

Table 5.46: KnownDevices Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	*DeviceFilter* ?	Refines the list of Devices queried. Only Devices that match the *DeviceFilter* are listed. The default SHALL return a list of all known Devices.
ResponseTypeObj Modified in JDF 1.1A	*DeviceList* ?	The list of known Devices. **Modification note:** Before **JDF** 1.1A this was "*Device**". It was changed due to inconsistencies of the inheritance model in the **JDF** schema.

Example 5.15: KnownDevices Response

```
<Response ID="M1" refID="Q1" Type="KnownDevices" xsi:type="ResponseKnownDevices">
    <DeviceList>
        <DeviceInfo DeviceStatus="Unknown">
            <Device DeviceID="Joe SpeedMaster"
                    DeviceType="Heidelberg SM102/6 rev. 47"/>
        </DeviceInfo>
    </DeviceList>
</Response>
```

5.27.1 DeviceFilter

The ***DeviceFilter*** Element refines the list of Devices that are requested to be returned. Only Devices that match all parameters of one of the ***Device*** Resources specified in the ***DeviceFilter*** Element are included.

Table 5.47: DeviceFilter Element

NAME	DATA TYPE	DESCRIPTION
DeviceDetails = "None" New in JDF 1.1	enumeration	Refines the level of provided information about the Device. **Allowed values are:** None – Provide only ***DeviceInfo***/@*DeviceID* and ***DeviceInfo***/@*DeviceStatus*. Brief – Provide all available Device information except for ***Device*** Elements. Modules – ***ModuleStatus*** elements are to be provided without module specific status details and without module specific employee information. Details – Provide maximum available Device information excluding Device capability descriptions. Includes ***Device*** Elements which represent details of the Device. NamedFeature – Provide maximum available Device information including limited Device capability descriptions. Includes ***Device*** Elements which represent details of the Device and ***Device***/***DeviceCap***/***FeaturePool*** Subelements which represent named features of the Device. Capability – Provide ***Device***/***DeviceCap*** Subelements which represent details of the capabilities of the Device. Full – Provide maximum available device information including ***Device*** capability descriptions. Includes ***Device*** elements which represent details of the device.
Localization ? New in JDF 1.2	languages or "all"	If present, @*Localization* defines the language code(s) specifying the localization(s) to be returned for each Device (see the ***DeviceCap*** Subelement description for details of what entries are localized). If "all" is specified, then all localizations for the Device are returned. If not specified, no localizations are returned.
Device *	element	Only Devices that match the Attribute Values specified in one of these ***Device*** Resources are included. Devices match the criteria if the Attribute Values specified here in the ***Device*** Resource match the equivalent Attribute Values of the known Devices. Unspecified Attributes always match. If ***Device*** is not specified, all known ***Device*** Resources are returned. As this is a filter, only information that can be used to identify a Device SHALL be specified. This precludes use of ***DeviceCap*** and ***IconList*** in this ***Device***. The data type of ***Device*** is ***ResourceElement***. See ▸ Section 3.10.1 ResourceElement – Subelement of a Resource.

5.27.2 DeviceList

New in JDF 1.1A

The ***DeviceList*** Element contains a list of information about Devices that are returned.

Table 5.48: DeviceList Element

NAME	DATA TYPE	DESCRIPTION
DeviceInfo *	element	List of information about known devices as requested by the ***DeviceFilter*** element. For details of the ***DeviceInfo*** Element, see ▸ Table 5.104 DeviceInfo Element in the message description ▸ Section 5.55 Status.

5.28 KnownJDFServices

Deprecated in JDF 1.2

In **JDF** 1.2 and beyond, ***KnownJDFServices*** has been replaced with ***KnownDevices*** and @*DeviceDetails* = "Capabilities". See ▸ Section N.4.6 KnownJDFServices for the details of this deprecated message.

5.29 KnownMessages

The ***KnownMessages*** Query message returns a list of all message types that are supported by the controller.

Table 5.49: KnownMessages Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***KnownMsgQuParams*** ?	Refines the query for known messages. If not specified, list all supported message types.
ResponseTypeObj	***MessageService*** *	Specifies the supported messages. Multiple ***MessageService*** Elements MAY be specified for a message with a given ***JMF***/@*Type*.

5.29.1 KnownMsgQuParams

The flags of the ***KnownMsgQuParams*** Element specify the message Families to include in the response list. Multiple flags are allowed.

Table 5.50: KnownMsgQuParams Element

NAME	DATA TYPE	DESCRIPTION
ChannelMode ? New in JDF 1.4	enumerations	Limits the list based on supported channel modes for the message. **Allowed value is from:** ▸ ChannelMode **Note:** See ▸ Table 5.5 Signal Message Element.
Exact = "false" New in JDF 1.1	boolean	Requests an exact description of the known messages. If "true", the response also contains the requested ***DevCaps*** of the messages.
ListCommands = "true"	boolean	Lists all supported ***Command*** types.
ListQueries = "true"	boolean	Lists all supported ***Query*** types.
ListRegistrations = "true" New in JDF 1.3	boolean	Lists all supported ***Registration*** message types.
ListSignals = "true"	boolean	Lists all supported ***Signal*** types.
Persistent = "false"	boolean	If "true", only lists messages that can use persistent channels. If "false", ignores the ability to use persistent channels.

5.29.2 MessageService

The response is a list of ***MessageService*** elements, one for each supported message type. The flags of the ***MessageService*** response message element are set in each ***MessageService*** entry. They define the supported usage of the message by the controller. Note that no @*Response* attribute is included in the list, since the capability to process one of the other message families implies the capability to generate an appropriate ***Response*** message. Multiple flags are allowed. .

Table 5.51: MessageService Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Acknowledge = "false" New in JDF 1.1	boolean	If "true", the Device supports asynchronous ***Acknowledge*** answers to this message.
ChannelMode ? New in JDF 1.4	enumerations	Specifies the supported channel modes for the message. **Allowed value is from:** ▸ ChannelMode **Note:** See ▸ Table 5.5 Signal Message Element.
Command = "false"	boolean	If "true", the message is supported as a ***Command***.

Table 5.51: MessageService Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
GenericAttributes ? New in JDF 1.3	NMTOKENS	List of generic Attributes that are supported and unrestricted by the Device implementation. Descriptions of Attributes that appear in ***State*** Elements (see the following ▸ Section 10.2.7 State) overwrite the description in @*GenericAttributes*, which SHALL NOT be specified if ***KnownMsgQuParams***/@*Exact* = "false".
JMFRole ? New in JDF 1.3	enumeration	The role of the Device that responds with the ***MessageService*** . **Allowed values are:** Receiver – The Device that responds to ***KnownMessages*** receives and responds to the message specified in @*Type*. This ***MessageService*** specifies Query messages, Signal messages Command messages and Registration messages that the device understands. Sender – The Device that responds to ***KnownMessages*** is the originator of the message specified in @*Type*. This ***MessageService*** specifies ***Response*** Elements and ***Acknowledge*** Elements that the Device understands as a ***Response*** to the messages that it has sent.
Persistent = "false"	boolean	If "true" the message is supported as a persistent channel.
Query = "false"	boolean	If "true" the message is supported as a ***Query***.
Registration = "false" New in JDF 1.3	boolean	If "true" the message is supported as a ***Registration*** message.
Signal = "false"	boolean	If "true" the message is supported as a ***Signal***.
Type	NMTOKEN	Type of the supported message. Extension types are specified by stating the namespace prefix in @*Type* **Values include those from:** ***Message***/@*Type*.
URLSchemes ? New in JDF 1.3	NMTOKENS	List of schemes supported for the message defined by this ***MessageService***. **Values include:** file – The file scheme according to ▸ [RFC1738] and ▸ [RFC3986]. http – HTTP (Hypertext Transport Protocol) https – HTTPS (Hypertext Transport Protocol – Secure)
ActionPool ? New in JDF 1.3	element	Container for zero of more ***Action*** Elements for use as constraints. For details on ***Action*** Elements, see ▸ Section 10.2.2 ActionPool. ***ActionPool*** SHALL NOT be specified if ***KnownMsgQuParams***/@*Exact* = "false".
DevCapPool ? New in JDF 1.3	element	Pool of ***DevCap*** Elements that can be referenced from multiple Elements within the ***DeviceCap*** structure. ***DevCapPool*** SHALL NOT be specified if ***KnownMsgQuParams***/@*Exact* = "false".
DevCaps * New in JDF 1.1	element	Specifies the restrictions of the parameter space of the supported messages. For details on using ***DevCaps***, see ▸ Section 10.2.5 DevCaps. ***DevCaps*** SHALL NOT be specified if ***KnownMsgQuParams***/@*Exact* = "false".
ModulePool ? New in JDF 1.3	element	Pool of ***ModuleCap*** Elements that specify the availability of a given Module. See ▸ Section 10.2.4.1 ModuleCap for details of ***ModuleCap***. ***ModulePool*** SHALL NOT be specified if ***KnownMsgQuParams***/@*Exact* = "false".
State * New in JDF 1.4	element	***State*** Elements that define the parameter space that is covered by the Device. One ***State*** Element SHALL be defined for each supported Attribute of the **JDF** Node that is not specified @*GenericAttributes* or implied by @*TypeExpression* or @*Types*.
TestPool ? New in JDF 1.3	element	Container for zero or more ***Test*** Elements that are referenced from ***ActionPool***/***Action*** Elements. ***TestPool*** SHALL NOT be specified if ***KnownMsgQuParams***/@*Exact* = "false".

Example 5.16: KnownMessages Response

The following is an example of a response message to a ***KnownMessages*** Query message.

```
<Response ID="M1" Type="KnownMessages" xsi:type="ResponseKnownMessages" refID="Q1">
    <MessageService JMFRole="Receiver" Query="true" Type="KnownMessages"/>
    <MessageService JMFRole="Receiver" Persistent="true" Query="true" Signal="true"
          Type="Status"/>
</Response>
```

5.30 KnownSubscriptions

New in JDF 1.4

The ***KnownSubscriptions*** **JMF** enables controllers to query devices for a list of active persistent channels.

Table 5.52: KnownSubscriptions Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***SubscriptionFilter*** ?	Refines the query for known messages. If not specified, list all supported message types.
ResponseTypeObj	***SubscriptionInfo*** *	List of active persistent channels.

5.30.1 SubscriptionFilter

New in JDF 1.4

The ***SubscriptionFilter*** Element is a filter to limit the list of ***SubscriptionInfo*** Elements that are returned in the ***KnownSubscriptions*** Response.

Table 5.53: SubscriptionFilter Element

NAME	DATA TYPE	DESCRIPTION
ChannelID ?	NMTOKEN	@*ChannelID* of the persistent channel to be queried. If the channel has been created with a ***Query*** message, the @*ChannelID* specifies the @*ID* of the ***Query*** message (identical to the @*refID* of the ***Response*** message)
DeviceID ?	NMTOKEN	Only subscriptions from devices or controllers with a matching @*DeviceID* attribute are queried.
Families ?	enumerations	Only subscriptions with the family (***Signal*** or ***Command***) listed are queried
JobID ? Deprecated in JDF 1.5	string	@*JobID* of the **JDF** Node that messages are subscribed for. If not specified, Subscriptions are returned for all @*JobID* values. **Deprecation note:** Job specific subscriptions are discouraged.
JobPartID ? Deprecated in JDF 1.5	string	@*JobPartID* of the **JDF** node that messages are subscribed for. If not specified, Subscriptions are returned for all @*JobPartID* values. **Deprecation note:** Job specific subscriptions are discouraged.
MessageTypes ?	NMTOKENS	List of ***Message***/@*Type* values of the subscribed messages. If not specified, Subscriptions are returned for all message types.
QueueEntryID ? Deprecated in JDF 1.5	string	@*QueueEntryID* of the Job whose Subscriptions are queried. If @*QueueEntryID* is specified, @*JobID*, @*JobPartID* and ***Part*** are ignored. If none of @*JobID*, @*JobPartID*, ***Part*** or @*QueueEntryID* are specified, ***KnownSubscriptions*** applies to all persistent channels that were established. **Deprecation note:** Job specific subscriptions are discouraged.
URL ?	URL	URL of the receiving controller. This SHALL be identical to the @*URL* that was used to create the persistent channel. If no @*ChannelID* is specified, all persistent channels to this @*URL* are queried.
Part * Deprecated in JDF 1.5	element	***Part*** Elements that describe the Partition of the Job whose Subscriptions are queried. For details on Node Partitions, see ‣ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources. **Deprecation note:** Job specific subscriptions are discouraged.

5.30.2 SubscriptionInfo

New in JDF 1.4

A ***SubscriptionInfo*** Element describes the ***Subscription*** details of a persistent channel.

Table 5.54: SubscriptionInfo Element

NAME	DATA TYPE	DESCRIPTION
ChannelID	NMTOKEN	@*ChannelID* specifies the ID of the ***Query*** message (identical to the @*refID* of the Signal or Response message).
Family	enumeration	Specifies whether the persistent channel is a Signal or Command. **Allowed values are:** Signal Command
JobID ? Deprecated in JDF 1.5	string	@*JobID* of the **JDF** Node that this Persistent Channel applies to. If not specified, this Persistent Channel applies to all @*JobID* values. **Deprecation note:** Job specific subscriptions are discouraged.
JobPartID ? Deprecated in JDF 1.5	string	@*JobPartID* of the **JDF** Node that this Persistent Channel applies to. If not specified, this Persistent Channel applies to all @*JobPartID* values. **Deprecation note:** Job specific subscriptions are discouraged.
MessageType	NMTOKEN	***Message***/@*Type* value of the subscribed messages.@*MessageType* SHALL match the local element name (i.e. without namespace prefix) of the Signals that comprise this persistent channel.
QueueEntryID ? Deprecated in JDF 1.5	string	@*QueueEntryID* of the ***QueueEntry*** that this Persistent Channel applies to. If not, specified, this Persistent Channel applies to all @*QueueEntryID* values. **Deprecation note:** Job specific subscriptions are discouraged.
SenderID	string	Device or Controller @*SenderID*.
Part * Deprecated in JDF 1.5	element	***Part*** Elements that describe the Partition of the **JDF** Node that this Persistent Channel applies to. For details on Node Partitions, see ▸ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources. **Deprecation note:** Job specific subscriptions are discouraged.
Subscription	element	The ***Subscription*** Element that describes the persistent channel.

5.31 ModifyNode

New in JDF 1.3

This **JMF** is used to modify either the @*Activation* or @*CommentURL* Attributes of a **JDF** Node and to add or modify ***Comment*** Elements of a **JDF** Node or a Resource.

5.31.1 ModifyNode Command

The ***ModifyNode Command*** is sent by a Controller to a Device to modify the **JDF** Node on the Device.

Table 5.55: ModifyNode Command

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	***ModifyNodeCmdParams*** ?	Defines the details of the ***ModifyNode*** message.
ResponseTypeObj	–	–

5.31.2 ModifyNode Signal

The ***ModifyNode Signal*** is sent by a Device to a Control to Signal that the **JDF** Node on the Device has been modified.

Table 5.56: ModifyNode Signal

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***ModifyNodeCmdParams*** ?	Defines the details of the ***ModifyNode*** message.
ResponseTypeObj	-	-

5.31.2.1 ModifyNodeCmdParams

The ***ModifyNodeCmdParams*** specifies the details of the **JDF** Node to be modified.

Table 5.57: ModifyNodeCmdParams Element

NAME	DATA TYPE	DESCRIPTION
Activation ?	enumeration	The new value for @*Activation*. Allowed values are from: ***JDF***/@*Activation* (▸ Table 3.4 JDF).
CommentURL ?	URL	The new value for @*CommentURL*. Note that @*CommentURL* is specified in ▸ Table 3.1 Any Element (generic content) and that the semantics are overridden by the definition in this table.
JobID	string	@*JobID* of the Node to be modified. In case of adding a ***Comment*** to a ***Resource*** or ***Audit***, this @*JobID* SHALL be an Attribute of the Node where the ***AuditPool*** or ***AuditPool*** resides.
JobPartID	string	@*JobPartID* of the Node to be modified. In the case of adding a ***Comment*** to a ***Resource*** or ***Audit***, this @*JobPartID* SHALL be an Attribute of the Node where the ***AuditPool*** or ***ResourcePool*** resides.
NewComment *	element	Details of modifications of ***Comment*** Elements.

5.31.2.2 NewComment

Table 5.58: NewComment Element

NAME	DATA TYPE	DESCRIPTION
Action	enumeration	**Allowed values are:** Add – A new ***Comment*** is added. If @*refID* is specified, the Comment is stored in the Resource or ***Audit*** with @*ID* = @*refID*. Concat – ***Comment*** is concatenated to the ***Comment*** with ***Comment***/@*ID* = @*CommentID*. Replace – ***Comment*** replaces the ***Comment*** with ***Comment***/@*ID* = @*CommentID*. Remove – The Comment with ***Comment***/@*ID* = @*CommentID* is removed.
CommentID ?	NMTOKEN	@*ID* of the existing ***Comment***. SHALL be specified if @*Action* = "Concat", "Replace" or "Remove".
refID ?	NMTOKEN	@*ID* of the ***Resource*** or ***Audit*** where the ***Comment*** SHALL be added. The @*refID* SHALL NOT be set unless @*Action* = "Add".
Comment ?	element	The ***Comment*** to "Add", "Concat" or "Replace". ***Comment*** SHALL NOT be specified if @*Action* = "Remove". Note that ***Comment*** * is specified in ▸ Table 3.1 Any Element (generic content) and that the cardinality and semantics are overridden by the definition in this table.
Part ? New in JDF 1.4	element	Partition of the Resource where the ***Comment*** SHALL be added. ***Part*** SHALL NOT be specified unless @*refID* references a Resource and @*Action* = "Add".

5.32 NewJDF

New in JDF 1.2

The *NewJDF* message can be used to query and initiate the modification of **JDF** Nodes by either a subordinate Controller or a master Controller. It is mainly used to synchronize *JDF*/@*JobID* and *JDF*/@*JobPartID* between an MIS and a Device or Controller. Either side MAY initiate synchronization. A Query message or Signal message informs a Controller or MIS system that a **JDF** Node has been created. A command initiates a modification.

5.32.1 NewJDF Query

The *NewJDF Query* message is sent to a Device or Controller in order to extract information about previously unknown **JDF** Nodes. For instance, an MIS that has received a **JMF** with an unknown @*JobPartID* MAY query the **JMF** sender about details of the **JDF** with that @*JobPartID*. When used as a Signal, the Signaling Device specifies that it has created a new **JDF** with the properties defined by *IDInfo*, for instance when a Workflow Controller has instantiated an abstract Process Group Node with new Subnodes. *NewJDF* is made selective by specifying a *NewJDFQuParams* Element.

The query's Response message returns a list of *IDInfo* Elements that contains the queried information concerning the newly created Nodes.

Table 5.59: NewJDF Query Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	*NewJDFQuParams*	Specifies the details of the Nodes that information is requested about.
ResponseTypeObj	*IDInfo* *	Contains the information about the newly created Nodes.

5.32.1.0.1 NewJDFQuParams

Table 5.60: NewJDFQuParams Element

NAME	DATA TYPE	DESCRIPTION
JobID ?	string	@*JobID* of the **JDF** Node that is being queried.
JobPartID ?	string	@*JobPartID* of the **JDF** Node that is being queried.
QueueEntryID ?	string	@*QueueEntryID* of the Job that is currently being executed. If @*QueueEntryID* is specified, @*JobID* and @*JobPartID* are ignored.

5.32.2 NewJDF Command

The *NewJDF Command* message is sent to an MIS, Device or Controller to initiate creation of new **JDF** Nodes by that Device or Controller. For instance, a Workflow Controller might have received content data and now requires a **JDF** Job from an MIS to which work on the content can be booked. The *NewJDF Command* message does not imply any Job submission or request for Job submission. Job queue submission SHALL still be requested with a *RequestQueueEntry* message, and the MIS SHALL still subsequently submit the Job to the requesting Controller or Device.

Table 5.61: NewJDF Command Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	*NewJDFCmdParams*	Specifies the details of the Nodes that are to be created
ResponseTypeObj	*IDInfo* ?	Contains the information about the newly created Node.

5.32.2.1 NewJDFCmdParams

Table 5.62: NewJDFCmdParams Element

NAME	DATA TYPE	DESCRIPTION
JDFDetails = "Brief"	string	Level of detail requested for the returned ***IDInfo*** Elements. **Values include:** None – Do not return any ***IDInfo*** Elements. Brief – Return ***IDInfo*** Elements without embedded **JDF** or Device. Full – Return ***IDInfo*** Elements with embedded **JDF** and Device.
IDInfo	element	Details of the new **JDF** Node that SHALL be created.

5.32.2.2 IDInfo

Table 5.63: IDInfo Element

NAME	DATA TYPE	DESCRIPTION
Category ?	NMTOKEN	***JDF***/@*Category* of the **JDF** Node. **Values include those from:** ***JDF***/@*Category*.
JDFURL ? New in JDF 1.5	URL	URL to detailed **JDF** description. Provides a way of referencing a **JDF** Element instead of embedding it at ***IDInfo***/***JDF***. At most one of **JDF** and @*JDFURL* SHALL be specified. **Note:** The referenced ▸ JDF MAY be an ancestor ▸ JDF Node of the newly created node. In this case the recipient SHALL search the returned ▸ JDF for the ▸ JDF Node with the correct @*JobPartID*.
JobID ?	string	@*JobID* of the **JDF** Node.
JobPartID ?	string	@*JobPartID* of the **JDF** Node.
ParentJobID ?	string	@*JobID* of the parent Node of the **JDF** Node. If not specified, it defaults to the value of @*JobID*.
ParentJobPartID ?	string	Job Part ID of the parent Node of the **JDF** Node.
ProjectID ? New in JDF 1.5	string	Identification of the project context that the **JDF** described by this ***IDInfo*** belongs to. Enables usage of ***NewJDF*** in a web to print environment where @*ProjectID* represents the shopping cart.
Type ?	NMTOKEN	***JDF***/@*Type* of the **JDF** Node. **Values include those from:** ***JDF***/@*Type*.
Types ?	NMTOKENS	***JDF***/@*Types* of the **JDF** Node. **Values include those from:** ***JDF***/@*Types*.
Device ?	element	Description of the Device that the **JDF** is targeted for. The data type of ***Device*** is ResourceElement. See ▸ Section 3.10.1 ResourceElement – Subelement of a Resource.
JDF ?	element	Detailed **JDF** description. Contains information that allows the receiver of the ***NewJDF*** message to properly respond. Note that the **JDF** is not implicitly submitted. At most one of **JDF** and @*JDFURL* SHALL be specified. **Note:** This may be an ancestor ▸ JDF Node of the newly created ▸ Node. In this case the recipient SHALL search the returned ▸ JDF for the ▸ JDF Node with the correct @*JobPartID*.

5.33 NodeInfo

New in JDF 1.2

Deprecated in JDF 1.3

The ***NodeInfo*** message has been replaced with the ***Resource*** message in **JDF** 1.3. For details of the deprecated ***NodeInfo*** message, see ▸ Section N.4.5 NodeInfo.

5.34 Notification

Notification messages are generally sent as Signals. The Query is defined to allow subscriptions for ***Notification*** messages.

Notification Elements are also used to signal usual events due to any activities of a Device, operator, etc. (e.g., scanning a bar code). Such a ***Signal*** always has a @*Type* = "Notification".

Table 5.64: Notification Signal

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***NotificationFilter*** ? New in JDF 1.4	Defines the types of ***Notification*** Elements that should be returned
ResponseTypeObj	***Notification***	***Notification*** that describes the event. See ▸ Section 3.11.4.5 Notification.

5.34.1 NotificationFilter

Table 5.65: NotificationFilter Element

NAME	DATA TYPE	DESCRIPTION
Classes ?	enumerations	Defines the set of ***Notification***/@*Class* to be queried/subscribed for. **Default behavior:** All ***Notification*** Classes are subscribed to. **Allowed values are from:** ▸ Severity. **Constraint note:** If both @*Classes* and @*Types* contain lists of values, the ***NotificationFilter*** defines an OR of all combinations.
DeviceID ? Deprecated in JDF 1.3	string	ID of the Device whose messages are queried/subscribed. MAY be specified for Device selection if the Controller controls more than one Device. **Deprecation note:** Starting with **JDF** 1.3, use ***JMF***/@*DeviceID*.
JobID ? Deprecated in JDF 1.5	string	JobID of the Job whose messages are queried/subscribed. **Deprecation note:** Job specific subscriptions are discouraged.
JobPartID ? Deprecated in JDF 1.5	string	JobPartID of the Job whose messages are queried/subscribed. **Deprecation note:** Job specific subscriptions are discouraged.
MilestoneTypes ?	NMTOKENS	Matching Milestone types are returned and/or subscribed to. **Default value is:** all supported @*MilestoneType* values. **Values include those from:** ▸ Table C.12 MessageEvents ContentStatus and MilestoneType Values.
QueueEntryID ? New in JDF 1.2 Deprecated in JDF 1.5	string	@*QueueEntryID* of the Job whose messages are queried/subscribed. If @*QueueEntryID* is specified, @*JobID*, @*JobPartID* and ***Part*** are ignored. If none of @*JobID*, @*JobPartID*, ***Part*** or @*QueueEntryID* are specified, ***NotificationFilter*** applies to all Jobs. **Deprecation note:** Job specific subscriptions are discouraged.
SignalTypes = "Notification" New in JDF 1.2	NMTOKENS	***Signal***/@*Type* values of the subscribed messages. @*SignalTypes* SHOULD contain values as shown in ***Message***/@*Type*, but are restricted to the values for ***Signal*** messages. In addition **Values include:** All – **specifies that all signals, regardless of @*Type* are queried/subscribed.**
Types ?	NMTOKENS	Matching notification types are returned/subscribed. **Default value is:** all supported notification types. **Values include those from:** ▸ Table C.14 List of NotificationDetails Elements.
Part * New in JDF 1.2 Deprecated in JDF 1.5	element	***Part*** Elements that describe the Partition of the Job whose messages are queried/subscribed. For details on Job Partitions, see ▸ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources. **Deprecation note:** Job specific subscriptions are discouraged.

Example 5.17: Notification Signal

```
<Signal ID="S1" Type="Notification" xsi:type="SignalNotification">
    <Notification Class="Event" TimeStamp="2005-07-25T12:32:48+02:00"
          Type="Barcode">
       <Comment>Palette completed</Comment>
       <Barcode Code="99923AAA123"/>
    </Notification>
</Signal>
```

5.35 Occupation

Deprecated in JDF 1.5

The ***Occupation*** message has been deprecated in **JDF** 1.5. For details of the deprecated ***Occupation*** message, see ▸ Section N.4.7 Occupation.

Deprecation note: ***Activity*** elements provide the functionality that makes ***Occupation*** redundant.

5.36 OpenQueue

The queue is opened and new queue entries can be accepted by the queue. A held queue remains held. The ***OpenQueue*** Command message is the opposite of a ***CloseQueue*** Command message.

Table 5.66: OpenQueue Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.5	***QueueFilter*** ? New in JDF 1.2 Deprecated in JDF 1.5	Defines a filter for the returned ***Queue*** Element in the ***OpenQueue*** message.
ResponseTypeObj Modified in JDF 1.5	***Queue*** Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.

5.37 PipeClose

The ***PipeClose*** message notifies the Process at the other end of a dynamic pipe that the sender of this message needs no further Resources or will produce no further Resources through the pipe. The ***PipeClose*** Command message response is equivalent to the ***PipePull*** and ***PipePush*** Command message responses ***PipePull***described below.

If ***Resource***/@*PipeProtocol* = "JMFPush" the producer SHALL terminate the pipe with a ***PipeClose*** message. If ***Resource***/@*PipeProtocol* = "JMFPull" the consumer SHALL terminate the pipe with a ***PipeClose*** message.

Table 5.67: PipeClose Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	***PipeParams***	Describes the pipe Resource. The ***PipeParams*** Element is described in ▸ Section 5.39 PipePull.
ResponseTypeObj Modified in JDF 1.5	***JobPhase*** Deprecated in JDF 1.5	The status of the responding Process. The ***JobPhase*** Element is defined in ▸ Table 5.105 JobPhase Element.

5.38 PipePause

The ***PipePause*** message pauses execution of a Process that is at the other end of a dynamic pipe.

PipePause MAY be emitted by either the consumer or the producer whenever a condition exists that requires a resynchronization.

If ***Resource***/@*PipeProtocol* = "JMFPush", and the consumer sends a ***PipePause***, the producer SHALL NOT send further ***PipePush*** messages until the consumer has reopened the pipe by sending a ***PipePull*** message.

If *Resource*/@*PipeProtocol* = "JMFPull", and the producer sends a *PipePause*, the consumer SHALL NOT send further *PipePull* messages until the producer has reopened the pipe by sending a *PipePush* message.

PipePause MAY be sent by the respective other end of the pipe even if the pipe is already paused. In this case the resynchronization requirements above still apply.

The *PipePause* Command message response is equivalent to the *PipePull* Command message response described above.

Table 5.68: PipePause Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	*PipeParams*	Describes the pipe Resource. The *PipeParams* Element is described in ▸ Section 5.39 PipePull.
ResponseTypeObj Modified in JDF 1.5	*JobPhase* Deprecated in JDF 1.5	The status of the responding Process. The *JobPhase* Element is defined in ▸ Table 5.105 JobPhase Element.

5.39 PipePull

The *PipePull* message requests Resources that are described in a **JDF** dynamic pipe (see ▸ Section 3.8.7 Pipe Resources and ▸ Section 4.3.3 Overlapping Processing Using Pipes). *PipePull* messages are the **JMF** equivalent of a dynamic input *ResourceLink*. Below, depicts the mode of operation of a *PipePull* message.

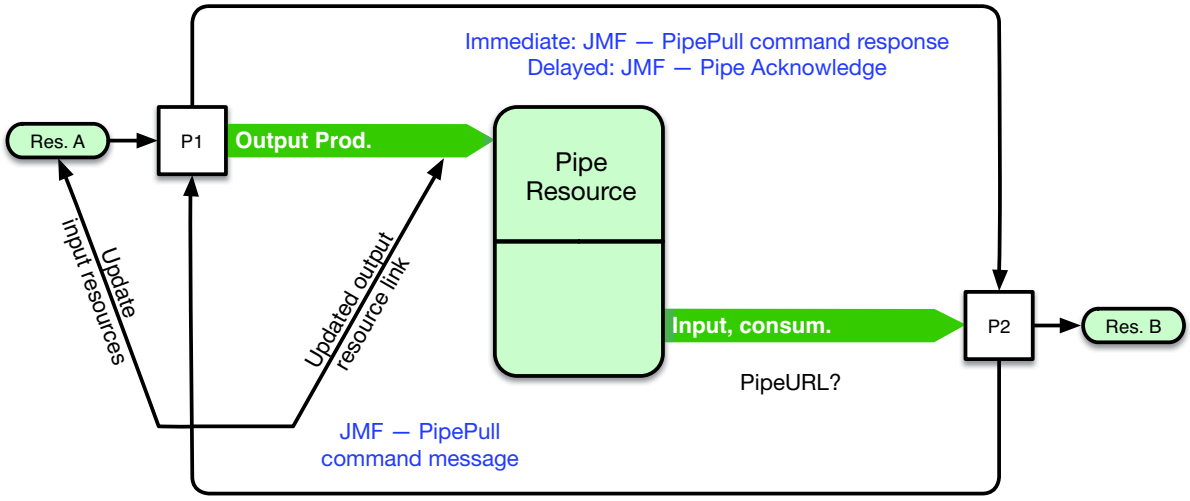
The *PipePull* Command message response returns a @*ReturnCode* of 0 if the command has been accepted by the receiving Controller. If not successful the @*ReturnCode* is one of the codes presented in ▸ Section Return Values and Error Codes. The *Response* message MAY contain a *Notification* Element. The *JobPhase* Element (see ▸ Section 5.55 Status) returned SHOULD provide only the @*Status* Attribute that describes the Job status of the responding Process after receiving the command.

If *Resource*/@*PipeProtocol* = "JMFPull", the consumer SHALL initiate the pipe with a *PipePull* Command message..

Table 5.69: PipePull Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	*PipeParams*	Describes the requested pipe Resource.
ResponseTypeObj Modified in JDF 1.5	*JobPhase* Deprecated in JDF 1.5	The status of the responding Process. The *JobPhase* Element is defined in ▸ Table 5.105 JobPhase Element.

Figure 5-7: Mechanism of a PipePull Message



5.40 PipePush

The *PipePush* message notifies the availability of pipe Resources that are described in a **JDF** dynamic pipe (see ▸ Section 3.8.7 Pipe Resources and ▸ Section 4.3.3 Overlapping Processing Using Pipes). *PipePush* messages are the **JMF** equivalent

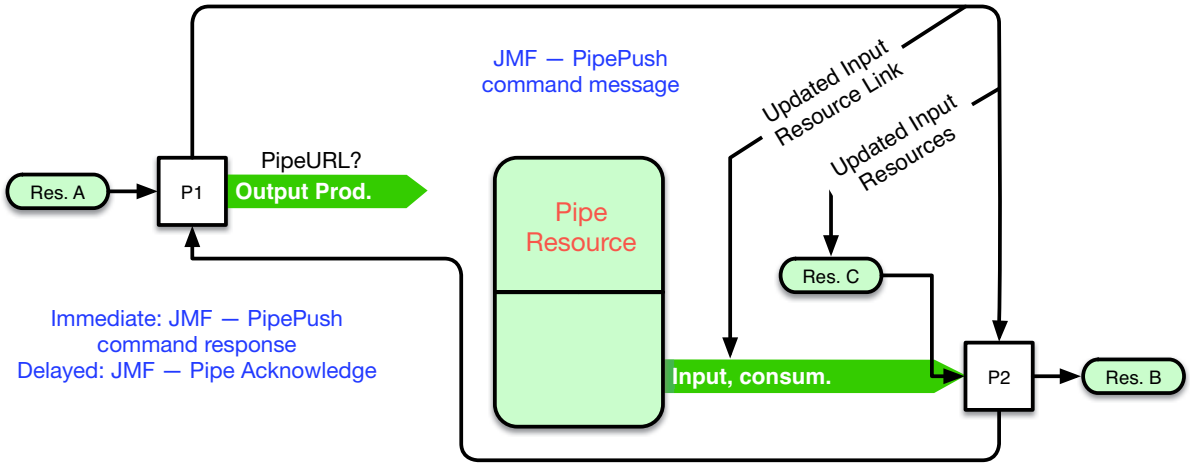
of a dynamic output ***ResourceLink***. The ▸ Figure 5-8: Mechanism of a PipePush Message depicts the mode of operation of a ***PipePush*** message. The ***PipePush*** Command message response is equivalent to the ***PipePull*** Command message Response described above.

If ***Resource***/@*PipeProtocol* = "JMFPush", the producer SHALL initiate the pipe with a ***PipePush*** message.

Table 5.70: PipePush Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	***PipeParams***	Describes the produced pipe Resource. The ***PipeParams*** Element is described in ▸ Section 5.39 PipePull.
ResponseTypeObj Modified in JDF 1.5	***JobPhase*** Deprecated in JDF 1.5	The status of the responding Process. The ***JobPhase*** Element is defined in ▸ Table 5.105 JobPhase Element.

Figure 5-8: Mechanism of a PipePush Message



5.41 QueueStatus

Returns a queue description.

Table 5.71: QueueStatus Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj Modified in JDF 1.2	***QueueFilter*** ? New in JDF 1.2	Defines a filter for the ***QueueStatus*** message.
ResponseTypeObj	***Queue***	Describes the status of the queue.

For the definition of the ***Queue*** Element, see ▸ Section 5.14 Elements for Queues.

5.42 RemoveQueueEntry

This command causes the entries specified by ***RemoveQueueEntryParams***/***QueueFilter*** to be removed from the queue. It does not affect ***QueueEntry*** [@*Status* = "Running" or @*Status* = "Suspended"]. Use ***AbortQueueEntry*** to stop a running or

suspended Job and then remove it with ***RemoveQueueEntry***. For details, see ▸ Table 5.20 Status Transitions for QueueEntry Handling Messages.

Table 5.72: RemoveQueueEntry Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.2 Modified in JDF 1.5	***QueueEntryDef*** Deprecated in JDF 1.5	Defines the queue entry.
	QueueFilter ? New in JDF 1.2 Deprecated in JDF 1.5	Defines a filter for the returned ***Queue*** Elements in the ***RemoveQueueEntry*** message.
	RemoveQueueEntryParams ? New in JDF 1.5	
ResponseTypeObj Modified in JDF 1.5	***Queue*** Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.

5.42.1 RemoveQueueEntryParams

New in JDF 1.5

Table 5.73: RemoveQueueEntryParams Element

NAME	DATA TYPE	DESCRIPTION
QueueFilter ?	element	This ***QueueFilter*** selects the ***QueueEntry*** elements to apply the ***RemoveQueueEntry*** message to.

5.43 RepeatMessages

Deprecated in JDF 1.5

The ***RepeatMessages*** message has been deprecated in **JDF** 1.5. ***RepeatMessages*** was designed to query for missed messages if Signals were required to be complete. This functionality SHOULD preferably be implemented using reliable channels (i.e., by specifying ***Subscription***/@*Channelmode* = "Reliable". See ▸ Section 5.3.3 Reliable Signalling. For details of the deprecated ***RepeatMessages*** message, see ▸ Section N.4.4 RepeatMessages.

5.44 RequestForAuthentication

New in JDF 1.4

The ***RequestForAuthentication*** message can be used as a Command to exchange certificates or as a Query to obtain the authentication status of previously exchanged certificates. Acknowledge messages SHALL NOT be used to respond to a ***RequestForAuthentication Command*** or ***RequestForAuthentication Query***. In other words, the Response element SHALL NOT specify @*Acknowledged* = "true". If it is not possible to confirm authentication before the HTTP channel times out, the @*ReturnCode* SHALL be "304", which means "Authentication pending".

5.44.1 RequestForAuthentication Command

New in JDF 1.4

The ***RequestForAuthentication Command*** Command is used to request authentication and trust of a certificate that is provided in the ***RequestForAuthentication Command***. The sender of the Command is identified by the @*SenderID* attribute in the **JMF** Element that contains the ***RequestForAuthentication Command***. The sender MAY be authenticated as both a client and as a server, and a separate certificate SHALL be provided by the sender for each role that the sender wishes to use

If a ***RequestForAuthentication Command*** is received over a secure channel, and a previous ***RequestForAuthentication Command*** has already been received, the previous ***RequestForAuthentication Command*** SHOULD be ignored, and any certificates associated with the prior Command SHOULD be considered untrusted. This allows for a party that is currently trusted to update its certificate as needed (such as when the previous certificate is about to expire),

Once authentication has been established between two parties, any ***RequestForAuthentication Command*** that is sent over a non-secure channel SHALL result in error 305, which is "Authentication already established". Other @*Reason* values MAY be supported over secure channels.

Table 5.74: RequestForAuthentication Command Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	***AuthenticationCmdParams***	Details of the certificate of the sender.
ResponseTypeObj	***AuthenticationResp*** ?	@*ReturnCode* = 0 indicates "I trust you". The initial response to a ***RequestForAuthentication Command*** SHALL include a fully specified ***AuthenticationResp*** Element.

5.44.1.1 AuthenticationCmdParams

Table 5.75: AuthenticationCmdParams Element

NAME	DATA TYPE	DESCRIPTION
AuthenticationType	enumeration	**Allowed values are:** AsClient – Sender of the message wishes to be authenticated as a client that initiates HTTP requests. Command includes the sender's client certificate, the Response will include the responders server certificate. AsServer – Sender of message wishes to be authenticated as a server that responds to HTTP requests. Command includes the sender's server certificate, the response will include the responders client certificate.
Reason	enumeration	Used to indicate the reason for sending this message. **Allowed values are:** InitiateConnection – the client wishes to exchange certificates with the server. ClientCertificateExpired – the previously-sent client certificate has expired. ServerCertificateExpired – the previously-received server certificate has expired. ClientHostnameMismatch – the client certificate's Common Name couldn't be resolved to match the IP address or domain name from which the request came. ServerHostnameMismatch – the server certificate's Common Name couldn't be resolved to match the IP address or domain name from which the response came. ClientCertificateRevoked – the previously-sent client certificate has been revoked. ServerCertificateRevoked – the previously-received server certificate has been revoked. Other – some other reason. Use @*ReasonDetails* for further explanation.
ReasonDetails ?	string	Further details on the reason for this message.
SecureURL ?	URL	URL of the port of the Command sender that will accept **JMF** messages via the HTTPS protocol. This Attribute SHALL be specified when the sender of the ***RequestForAuthentication Command*** has specified ***AuthenticationCmdParams***/@*AuthenticationType* = "AsServer".
Certificate ?	element	The requester's certificate. If @*AuthenticationType* = "AsClient", this certificate SHALL be the requester's client certificate. If @*AuthenticationType* = "AsServer", this certificate SHALL be the requester's server certificate.

5.44.1.2 Certificate

Table 5.76: Certificate Element

NAME	DATA TYPE	DESCRIPTION
	text	The certificate in PEM MD5 format. **Implementation Note:** There SHALL NOT be any whitespace between the end of the tag and the start of the certificate, or between the end of the certificate and the start of the end tag. See example below. **Note:** The certificate should only include the public key.

5.44.1.3 AuthenticationResp

Table 5.77: AuthenticationResp Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
SecureURL ?	URL	URL of the port of the command recipient that will accept **JMF** messages via the HTTPS protocol. This Attribute SHALL be specified when the sender of the ***RequestForAuthentication Command*** has specified ***AuthenticationCmdParams***/@*AuthenticationType* = "AsClient".

Table 5.77: AuthenticationResp Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Certificate ?	element	The Command recipient's certificate. If ***AuthenticationCmdParams***/@*AuthenticationType* = "AsClient", this certificate SHALL be the Command recipient's server certificate. If ***AuthenticationCmdParams***/@*AuthenticationType* = "AsServer", this certificate SHALL be the Command recipient's client certificate. When responding to a ***RequestForAuthentication Command*** over a non-secure channel with *Reason* = "InitiateConnection", this Element SHALL be specified. When responding to a ***RequestForAuthentication Query***, the ***Certificate*** Element SHALL NOT be specified. See ***AuthenticationCmdParams***/***Certificate***.

5.44.2 RequestForAuthentication Query

New in JDF 1.4

The ***RequestForAuthentication Query*** is used to determine the authentication status of a certificate that was provided in an earlier ***RequestForAuthentication Command*** or the Response to the Command. The sender of the Query is identified by the @*SenderID* Attribute in the **JMF** Element that contains the ***RequestForAuthentication Query***. The sender MAY be authenticated as both a client and as a server, and a separate certificate SHALL be provided by the sender for each role that the sender wishes to use.

If a ***RequestForAuthentication Query*** is received, and no previous ***RequestForAuthentication Command*** has been received, the Response SHALL specify a @*ReturnCode* of 306, which is "No authentication request in process". .

Table 5.78: RequestForAuthentication Query Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***AuthenticationQuParams***	Specifies the type of authentication being queried.
ResponseTypeObj	***AuthenticationResp*** ?	@*ReturnCode* = 0 indicates "I trust you".

5.44.2.1 AuthenticationQuParams

Table 5.79: AuthenticationQuParams Element

NAME	DATA TYPE	DESCRIPTION
AuthenticationType	enumeration	**Allowed values are:** AsClient – Sender of the message wishes to check the authentication status of the client certificate associated with it. AsServer – Sender of message wishes to check the authentication status of the server certificate associated with it..

Example 5.18: RequestForAuthentication Command

```
<Command ID="M001" Type="RequestForAuthentication"
       xsi:type="CommandRequestForAuthentication">
    <AuthenticationCmdParams AuthenticationType="AsClient"
          Reason="InitiateConnection">
       <Certificate>=====BEGIN CERTIFICATE=====
MIIC3jCCApwCBEIWY6YwCwYHKoZIzjgEAwUAMFUxCzAJBgNVBAYTAkNIMQ8wDQYDVQQHEwZadXJp
Y2gxDTALBgNVBAoTBENJUDQxDzANBgNVBAsTBkpNRiBXRzEVMBMGA1UEAxMMd3d3LmNpcDQub3Jn
MB4XDTA1MDIxODIxNTIzOFoXDTA1MDUyOTIxNTIzOFowVTELMAkGA1UEBhMCQ0gxDzANBgNVBAcT
Blp1cmljaDENMAsGA1UEChMEQ0lQNDEPMA0GA1UECxMGSk1GIFdHMRUwEwYDVQQDEwx3d3cuY2lw
NC5vcmcwggG3MIIBLAYHKoZIzjgEATCCAR8CgYEA/X9TgR11EilS30qcLuzk5/YRt1I870QAwx4/
gLZRJmlFXUAiUftZPY1Y+r/F9bow9subVWzXgTuAHTRv8mZgt2uZUKWkn5/oBHsQIsJPu6nX/rfG
G/g7V+fGqKYVDwT7g/bTxR7DAjVUE1oWkTL2dfOuK2HXKu/yIgMZndFIAccCFQCXYFCPFSMLzLKS
uYKi64QL8Fgc9QKBgQD34aCF1ps93su8q1w2uFe5eZSvu/o66oL5V0wLPQeCZ1FZV4661FlP5nEH
EIGAtEkWcSPoTCgWE7fPCTKMyKbhPBZ6i1R8jSjgo64eK7OmdZFuo38L+iE1YvH7YnoBJDvMpPG+
qFGQiaiD3+Fa5Z8GkotmXoB7VSVkAUw7/s9JKgOBhAACgYArHi/BVNf3OG0JIIdzWraVrx1wg9RM
do+tYRjY4bXue7LRDCvVaSX1Ddy9kTyeTTntwUrJOyx/8qEi/WmraGXhK8wGSrtE/q3S/A16DwEB
CiyeMhlCrd4QiAhp5WtR4KIMIBjq2Xn8+0MnnT1qDnmesNaSwdZ/01E0azSPTy5XnDALBgcqhkjO
OAQDBQADLwAwLAIUFZHojJvsO3+UYMBZk6yDzhdejzMCFHC0WbkDwfImQCa+dTebXZ1e1GlQ
=====END CERTIFICATE=====</Certificate>
    </AuthenticationCmdParams>
</Command>
```

Example 5.19: RequestForAuthentication Response

The form of Response that would most likely follow the above Command appears below:

```
<Response ID="M101" Type="RequestForAuthentication" refID="M001"
       xsi:type="ResponseRequestForAuthentication" ReturnCode="304">
    <AuthenticationResp SecureURL="https://printserver.mycompany.com/A3Printer">
       <Certificate>=====BEGIN CERTIFICATE=====
uYKi64QL8Fgc9QKBgQD34aCF1ps93su8q1w2uFe5eZSvu/o66oL5V0wLPQeCZ1FZV4661FlP5nEH
EIGAtEkWcSPoTCgWE7fPCTKMyKbhPBZ6i1R8jSjgo64eK7OmdZFuo38L+iE1YvH7YnoBJDvMpPG+
qFGQiaiD3+Fa5Z8GkotmXoB7VSVkAUw7/s9JKgOBhAACgYArHi/BVNf3OG0JIIdzWraVrx1wg9RM
do+tYRjY4bXue7LRDCvVaSX1Ddy9kTyeTTntwUrJOyx/8qEi/WmraGXhK8wGSrtE/q3S/A16DwEB
CiyeMhlCrd4QiAhp5WtR4KIMIBjq2Xn8+0MnnT1qDnmesNaSwdZ/01E0azSPTy5XnDALBgcqhkjO
OAQDBQADLwAwLAIUFZHojJvsO3+UYMBZk6yDzhdejzMCFHC0WbkDwfImQCa+dTebXZ1e1GlQ
MIIC3jCCApwCBEIWY6YwCwYHKoZIzjgEAwUAMFUxCzAJBgNVBAYTAkNIMQ8wDQYDVQQHEwZadXJp
Y2gxDTALBgNVBAoTBENJUDQxDzANBgNVBAsTBkpNRiBXRzEVMBMGA1UEAxMMd3d3LmNpcDQub3Jn
MB4XDTA1MDIxODIxNTIzOFoXDTA1MDUyOTIxNTIzOFowVTELMAkGA1UEBhMCQ0gxDzANBgNVBAcT
Blp1cmljaDENMAsGA1UEChMEQ0lQNDEPMA0GA1UECxMGSk1GIFdHMRUwEwYDVQQDEwx3d3cuY2lw
NC5vcmcwggG3MIIBLAYHKoZIzjgEATCCAR8CgYEA/X9TgR11EilS30qcLuzk5/YRt1I870QAwx4/
gLZRJmlFXUAiUftZPY1Y+r/F9bow9subVWzXgTuAHTRv8mZgt2uZUKWkn5/oBHsQIsJPu6nX/rfG
G/g7V+fGqKYVDwT7g/bTxR7DAjVUE1oWkTL2dfOuK2HXKu/yIgMZndFIAccCFQCXYFCPFSMLzLKS
=====END CERTIFICATE=====</Certificate>
    </AuthenticationResp>
</Response>
```

Example 5.20: Follow up RequestForAuthentication Query

Next, the original command sender would send a follow up ***RequestForAuthentication Query***:

```
<Query ID="M004" Type="RequestForAuthentication"
       xsi:type="QueryRequestForAuthentication">
    <AuthenticationQuParams AuthenticationType="AsClient"/>
</Query>
```

Example 5.21: RequestForAuthentication Response from Follow Up Query

If authentication has been confirmed, the following Response would be sent to the ***RequestForAuthentication Query***:

```
<Response ID="M102" Type="RequestForAuthentication" refID="M004"
      xsi:type="ResponseRequestForAuthentication" ReturnCode="0">
</Response>
```

5.45 RequestQueueEntry

New in JDF 1.2

This command requests a new queue entry from a potential submitting Agent. The actual submission is still handled by the standard queue entry handling parameters. Note that this command is emitted from the Device that is represented by the queue to a Controller or Device and not to the queue, as is the case with most other queue handling commands.

Whereas **JDF** generally assumes a "Push" workflow, where a controller or MIS assigns a task to a given Device, ***RequestQueueEntry*** allows a "Pull" workflow to be implemented, where a Device with free processing capabilities dynamically requests a new task .

Table 5.80: RequestQueueEntry Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	***RequestQueueEntryParams***	Defines the specifics for the requested Job.
ResponseTypeObj	—	The response to this message contains no ***ResponseTypeObj***, only an empty Response element that specifies the *@ReturnCode*. Any Job submission is handled using hot folders or the standard ***SubmitQueueEntry*** message.

5.45.1 RequestQueueEntryParams

Table 5.81: RequestQueueEntryParams Element

NAME	DATA TYPE	DESCRIPTION
Activation ? New in JDF 1.5	enumeration	Specifies the activation of the requested ***QueueEntry***. **Allowed values are** from: *@Activation* in ▸ Table 3.4 JDF.
JobID ?	string	*@JobID* of the requested ***QueueEntry***.
JobPartID ?	string	*@JobPartID* of the requested ***QueueEntry***.
QueueURL	URL	URL of the ***Queue*** Controller that is requesting the ***QueueEntry*** and will accept ***Queue*** manipulation messages. If URL specifies a File, this is the hot folder for **JDF** submission.
SubmitPolicy ? New in JDF 1.3	enumeration	Defines the requested policy for submitting the Node. If not specified, the submission policy is dependent on the Controller implementation. *@SubmitPolicy* allows a Device to request a Node that would otherwise not be submitted by the Controller due to missing Resources. **Allowed values are:** Standard – All linked Resources SHALL have a ***Resource***@*Status* as defined by ***ResourceLink***/@*MinStatus*. Late – All linked Resources SHALL have a ***Resource***@*Status* as defined by ***ResourceLink***/@*MinLateStatus*. Force – The Node SHALL be submitted regardless of the values of linked ***Resource***@*Status*.
Part *	element	Partition parts of the requested ***QueueEntry***.
Queue ?	element	Representation of the current status of the Device's ***Queue***.

5.46 Resource

The ***Resource*** message can be a ***Command*** message or a ***Query*** message to modify or to query **JDF** resources. In both cases (query and command), it is possible to address either global Device Resources, such as Device settings or Job-specific Resources. The Query message retrieves information about the resources without modifying them, while the ***Command*** message modifies those settings within the resource that is specified. Settings that are not specified remain unchanged.

5.46.1 Resource Query

The ***Resource Query*** can be made selective by specifying a ***ResourceQuParams*** element. The presence of the @*JobID* attribute determines whether global device resources or job-related resources are returned. If no ***ResourceQuParams*** element is specified, only the global device resources are returned.

The query's Response message returns a list of ***ResourceInfo*** Elements that contains the queried information concerning the Resources. If the list is empty because the selective query parameters of the ***ResourceQuParams*** lead to a null selection of the known Device/Job Resources, then the @*ReturnCode* is 103 (@*JobID* unknown), 104 (@*JobPartID* unknown) or 108 (empty list) and SHOULD be flagged as a warning with ***Notification*** [@*Class* = "Warning" and @*Type* = "Error"].

Table 5.82: Resource Query Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***ResourceQuParams*** ?	Specifies the Resources queried.
ResponseTypeObj	***ResourceInfo*** *	Contains the amount data of Resources and if requested, the Resources itself.

5.46.1.1 ResourceQuParams

Table 5.83: ResourceQuParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Classes ?	enumerations	List of the Resource Classes to be queried. For example, in order to query the actual level of consumables in a Device outside of any Job context, specify @*Classes* = "Consumable" in the query without a @*JobID* Attribute. **Default value is:** all Classes (if @*Classes* is empty or not specified). Allowed values are from: ***Resource***/@*Classes* (▸ Table 3.10 Abstract Resource Element).
Context ? New in JDF 1.5	enumeration	Specifies the job context of the queried resources. **Allowed values are:** Job – the query is for all resources in the context of the specified job. Global – the query is for a catalog of all known resources.
Exact = "false"	boolean	Requests an exact description of the **JDF** Resource. If "true", the response will also return the requested **JDF** Resource.
JobID ? Modified in JDF 1.4	string	@*JobID* of the **JDF** Node that is being queried. If no @*JobID* is specified, the request applies to the currently running Job or global resources, depending on the value of @*Context*.
JobPartID ?	string	@*JobPartID* of the **JDF** Node that is being queried. If no @*JobPartID* is specified, all resources related to @*JobID* are queried.
Location ?	string	Identifies the location of a Resource, such as paper tray, ink container or thread holder. The name is the same name used in the Partition Key @*Location* of distributed Resources (see also ▸ Section 3.10.6.4 Locations of PhysicalResources). **Default value is:** the location will be selected by the Device **Values include those from:** ▸ Table C.10 Input Tray and Output Bin Names. **Note:** The specified values are for printer locations.
LotDetails = "Brief" New in JDF 1.4	enumeration	Refines the level of information provided about individual lots of the Resources. This attribute is most useful when querying an MIS, and SHOULD NOT be specified when querying a device. **Allowed values are:** Brief – Provides only the @*LotControlled* Attribute in the Response indicating whether or not the Resources are lot controlled. Full – Provides ***Lot*** Elements related to the Resources. Amount – Same as "Full", but with the addition of the @*Amount* Attribute so the MIS can indicate what the current "on hand" balance for the ***Lot*** is in the MIS.
LotID ? New in JDF 1.4	string	@*LotID* of the individual lot of the Resource that is queried.
ProcessUsage ?	string	Selects a Resource in which ***ResourceLink***/@*ProcessUsage* matches the token specified. Only necessary if a Resource name is used more than once by one Node. For example, the ***Component*** input ***ExposedMedia*** of a **ConventionalPrinting** Process SHALL be distinguished by specifying@*ProcessUsage* = "Plate" and @*ProcessUsage* = "Proof", respectively. The @*ResourceName*, @*Usage* and @*ProcessUsage* Attributes are combined by a logical AND conjunction to select the Resource to be queried. **Values include those from:** ***ResourceLink***/@*ProcessUsage* (▸ Table 3.16 ResourceLink Element).
ProductID ? New in JDF 1.2	string	@*ProductID* of the Resource that is queried.

Table 5.83: ResourceQuParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
QueueEntryID ? New in JDF 1.2	string	@*QueueEntryID* of the Job that is currently being executed. If @*QueueEntryID* is specified, @*JobID*, @*JobPartID* and ***Part*** SHALL be ignored. If none of @*JobID*, @*JobPartID*, ***Part*** or @*QueueEntryID* are specified, ***ResourceQuParams*** applies to all Jobs.
ResourceDetails = "Full" New in JDF 1.4	enumeration	Refines the level of information provided about the resources. **Allowed values are:** Brief – Provides appropriate ID information specific to the type of resource and @*DescriptiveName* attributes only. For example, @*ProductID* would be included for ***Consumable Resource*** elements that represent consumables, @*PersonalID* for ***Employee*** resources. Full – Provides all of the attributes of the resources.
ResourceID ? New in JDF 1.3 Deprecated in JDF 1.5	NMTOKEN	***Resource***/@*ID* of the Resource that is queried. **Note:** The data type is NMTOKEN and not IDREF because the referenced @*ID* need not be present in the **JMF**. **Deprecation note:** Starting with **JDF** 1.5, Resources SHOULD be identified by @*ProductID* in ***Resource*** **JMF** messages.
ResourceName ? Modified in JDF 1.4	NMTOKENS	Name of the Resource(s) being queried. **Values include those from:** ▸ Section Resources. **Modification note:** Starting with **JDF** 1.4, the data type was expanded from NMTOKEN to NMTOKENS.
Scope ? New in JDF 1.4	enumeration	Specifies whether the query refers to a complete list of all potential resources or to the currently loaded resources. **Allowed values are:** Allowed – All known resources SHALL be returned, including those that are not currently available. Present – Currently available resources SHALL be returned.
Usage ?	enumeration	Selects a resource in which the value of the ***ResourceLink***/@*Usage* attribute matches the token specified here in this attribute. Only necessary if a resource with a given name is used both as input and output by one node. **Allowed values are** from: ***ResourceLink***/@*Usage* (▸ Table 3.16 ResourceLink Element).
Part * New in JDF 1.2	element	***Part*** elements that describe the resource whose messages are queried.

Example 5.22: Resource Query about Paper

The following is an example of a press system sending a ***Resource Query*** to another MIS to get information on all paper known by the MIS:

```
<Query ID="M170" Type="Resource" xsi:type="QueryResource" >
<ResourceQuParams ResourceDetails="Brief" LotDetails="Full" Scope="Allowed"/>
</Query>
```

Example 5.23: Resource Response about Paper

The following is an example of a ***Resource*** Response to the previous ***Resource Query***

```
<Response ID="M1001" Type="Resource" xsi:type="ResponseResource" refID="M170">
   <ResourceInfo LotControlled="false">
      <Media ID="R01" Class="Consumable" Status="Available"
             ProductID="9902-1" DescriptiveName="60 lb #3 Gloss Book"/>
   </ResourceInfo>
   <ResourceInfo LotControlled="true">
      <Media ID="R01" Class="Consumable" Status="Available"
             ProductID="9903-1" DescriptiveName="80 lb #3 C1S Cover"/>
      <Lot LotID="LN8845739CN7787399-03"/>
      <Lot LotID="LN8845739CN7787399-04"/>
      <Lot LotID="LN8845739CN7787399-06"/>
      <Lot LotID="LN8845739CN7787399-10"/>
   </ResourceInfo>
   <!-- ... -->
   <ResourceInfo LotControlled="false">
      <Media ID="R01" Class="Consumable" Status="Available"
             ProductID="9989-5" DescriptiveName="110 lb #1 Coated Cover"/>
   </ResourceInfo>
</Response>
```

Example 5.24: Resource Query about Employees

The following is an example of a press system sending a ***Resource Query*** to an MIS to get a list of all known employees in the MIS:

```
<Query ID="M170" Type="Resource" xsi:type="QueryResource">
   <ResourceQuParams ResourceName="Employee" ResourceDetails="Brief"/>
</Query>
```

Example 5.25: Resource Response about Employees

The following is an example of a ***Resource*** Response to the previous ***Resource Query***

```
<Response ID="M1001" Type="Resource" xsi:type="ResponseResource"
       refID="M170">
   <ResourceInfo>
      <Employee ID="E01" Class="Implementation" Status="Available"
             PersonalID="1034" DescriptiveName="John Allen"/>
   </ResourceInfo>
   <ResourceInfo>
      <Employee ID="E02" Class="Implementation" Status="Available"
             PersonalID="1057" DescriptiveName="Sally Brown"/>
   </ResourceInfo>
   <ResourceInfo>
      <Employee ID="E03" Class="Implementation" Status="Available"
             PersonalID="2105" DescriptiveName="Mike Davison"/>
   </ResourceInfo>
   <!-- ... -->
   <ResourceInfo>
      <Employee ID="E04" Class="Implementation" Status="Available"
             PersonalID="6410" DescriptiveName="Will Smith"/>
   </ResourceInfo>
</Response>
```

Example 5.26: Resource Signal about Consumed Resources

The following is an example of a ***Resource*** Signal used to report the inventory identification of the Resources that were used:

```
<Signal ID="P172" Type="Resource" xsi:type="SignalResource">
    <ResourceQuParams JobID="34028" JobPartID="_F05A84BD"/>
    <ResourceInfo>
        <Media PartIDKeys="SheetName" ID="RI007" Class="Consumable"
               ProductID="3002" Brand="Roll Stock"
               Dimension="2520 8640000" MediaType="Paper">
            <Media SheetName="1"/>
            <Media SheetName="2"/>
        </Media>
        <AmountPool>
            <PartAmount ActualAmount="9700">
                <Part SheetName="1"/>
                <Lot ActualAmount="4850" Consumption="Full"
                       LotID="LN1040788312RN20050917-04"/>
                <Lot ActualAmount="4850" Consumption="Partial"
                       LotID="LN1040788339RN20050919-01"/>
            </PartAmount>
            <PartAmount ActualAmount="5027">
                <Part SheetName="2"/>
                <Lot ActualAmount="5027" Consumption="Partial"
                       LotID="LN1040788319RN20050917-04"/>
            </PartAmount>
        </AmountPool>
    </ResourceInfo>
</Signal>
```

5.46.2 Resource Command

The ***Resource Command*** message is used to modify or create either global Device settings or resources of a running Job. It can be made selective by specifying the OPTIONAL Attributes in the ***ResourceCmdParams*** Element. The presence of ***ResourceCmdParams***/@*JobID* determines whether global Device Resources or Job-related Resources are modified. If no Resource exists in the target **JDF** that matches the filter settings in ***ResourceCmdParams***, and ***ResourceCmdParams***/@*JobID* is present, then the specified Resource SHALL be created as an Input Resource to the **JDF** Node.

The ***Resource*** message contains a list of ***ResourceInfo*** Elements with all Resources and private extensions of the Device after the changes have been applied. The type of the Resource that is given as a response depends on the type of the Resource given in the command.

If the ***Resource Command*** message was successful, the value of @*ReturnCode* is "0". If it is not successful, the value of @*ReturnCode* is one of those that have been described in the above section about the ***Resource Query*** message; or it is "200" (invalid Resource parameters) or "201" (insufficient Resource parameters). Partial application of the Resource SHOULD also be flagged as a warning with ***Notification***[@*Class* = "Warning" and @*Type* = "Error"]. If the value of @*ReturnCode* is larger than "0", the Controller that issued the command can evaluate the returned Resource in order to find the setting that could not be applied.

Table 5.84: Resource Command Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	***ResourceCmdParams***	Specifies the Resources to be modified.
ResponseTypeObj	***ResourceInfo*** *	Contains information about the resources after modification.

5.46.2.1 ResourceCmdParams

Table 5.85: ResourceCmdParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Activation = "Active" New in JDF 1.1	enumeration	Describes the activation status of the uploaded Resource. Allows for a range of activity, including deactivation and test running of a Resource prior to actually committing the change to the Device. Values are in order of involvement from least to most active. **Allowed values are:** Held – Used for uploading a Resource that requires operator intervention before being applied. TestRun – Used for a test run check by the Controller or a Device. This does not imply that the update is automatically applied when the check is completed. TestRunAndGo – Similar to "TestRun", but requests a subsequent automatic update of the Resource if the test run has been completed successfully. Active – Used for applying the update immediately. **Note:** @*Activation* uses an identical syntax to ***JDF***/@*Activation*, but that is does not explicitly set ***JDF***/@*Activation* in the **JDF** on the Device. The "Inactive" value defined in ***JDF***/@*Activation* is not a valid value in this list.
Exact = "false"	boolean	Requests an exact description of the **JDF** Resource. If "true", the Response message will also return the requested **JDF** Resource.
JobID ?	string	@*JobID* of the **JDF** Node that the Resource being modified is linked to. If no @*JobID* is specified, global Resource settings are modified.
JobPartID ?	string	@*JobPartID* of the **JDF** Node that the Resource being modified is linked to.
ProcessUsage ?	NMTOKEN	Selects a Resource in which the value of the ***ResourceLink***/@*ProcessUsage* Attribute matches the token specified here in this Attribute. Only necessary if a Resource name is used more than once by one Node. For example, the ***ExposedMedia*** Input Resources of a **ConventionalPrinting** Process can be distinguished by specifying @*ProcessUsage* = "Plate" and @*ProcessUsage* = "Proof", respectively. The @*ResourceName*, @*Usage* and @*ProcessUsage* Attributes are combined by a logical AND conjunction to select the Resource to be modified. **Values include those from:** ***ResourceLink***/@*ProcessUsage* (▸ Table 3.16 ResourceLink Element).
ProductID ? New in JDF 1.2	string	@*ProductID* of the Resource that is updated.
ProductionAmount ?	double	New requested amount of Resource production. This value replaces the ***ResourceLink***/@*Amount* of the selected Resource.
QueueEntryID ? New in JDF 1.2	string	@*QueueEntryID* of the Job that is currently being executed. If @*QueueEntryID* is specified, @*JobID*, @*JobPartID* and ***Part*** are ignored. If none of @*JobID*, @*JobPartID*, ***Part*** or @*QueueEntryID* are specified, ***ResourceCmdParams*** applies to global resources.
ResourceID ? New in JDF 1.3 Deprecated in JDF 1.5	NMTOKEN	***Resource***/@*ID* the Resource that is modified. If both @*ResourceID* and ***Resource*** are specified, Resources with a non-matching ***Resource***/@*ID* SHALL NOT be updated. **Note:** The data type is NMTOKEN and not IDREF because the referenced @*ID* NEED NOT be present in the **JMF**. **Deprecation note:** Starting with **JDF** 1.5, resources SHOULD be identified by @*ProductID* in ***Resource*** **JMF** messages.
ResourceName ?	NMTOKEN	Name of the resource whose production amount will be modified. **Values include those from:** ▸ Chapter 8 Resources.
Status ? New in JDF 1.2	enumeration	Updated @*Status* of the selected resource. **Values include those from:** ***Resource***/@*Status* (▸ Table 3.10 Abstract Resource Element).

Table 5.85: ResourceCmdParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
UpdateIDs ? New in JDF 1.1 Deprecated in JDF 1.3	NMTOKENS	The @*UpdateID* attributes of one or more ***ResourceUpdate*** that are defined in resources known to the recipient. The data type is NMTOKENS and not IDREFS because no matching IDs exist within this message. The order of tokens in defines the order in which the updates are applied.
UpdateMethod = "Complete" New in JDF 1.3 Modified in JDF 1.4	enumeration	Method how the resource is updated. Attributes that are required to correctly identify the resource SHALL be specified, even if @*UpdateMethod*="Remove" or @*UpdateMethod*="Incremental". These attributes include @*ProductID*, @*Class*, @*PartIDKeys*, and any partition keys. **Allowed values are:** Complete – The Resource Partitions defined by ***Part*** are completely overwritten by ***Resource*** in this message. Incremental – The Resource Partitions defined by ***Part*** are incrementally updated by the values that are explicitly set in ***Resource*** in this message. Remove – The Resources or Resource Partitions are removed. New in JDF 1.4
Usage ? New in JDF 1.4	enumeration	Selects a Resource in which the value of the ***ResourceLink***/@*Usage* Attribute matches the token specified here in this Attribute. Only necessary if a Resource name is used both as input and output by one Node. **Allowed values are** from: ***ResourceLink***/@*Usage* (▸ Table 3.16 ResourceLink Element).
MISDetails ? New in JDF 1.2	element	Definition how the costs for the modification of the ***Resource*** are to be charged.
Part * New in JDF 1.2	element	***Part*** Elements that describe the Partitions of the Resource that is being modified. If not specified, the entire Resource is selected. If a Resource is the final instance of set of Partitioned Resources, and thus the properties of the Partition that represents the set are modified in addition to the properties of the instance, then the ***Part*** that represents the set SHOULD also be specified explicitly. For example, if the fourth plate of a four color process set is now available, and thus the entire surface is now available, ***Part*** Elements for both the fourth plate and for the entire surface SHOULD be specified. If the other surface is also available, then a ***Part*** Element for the sheet SHOULD be specified as well.
Resource *	element	Resources to be uploaded to the Device. They replace the original Resources according to the policy specified in @*UpdateMethod*. The Resource SHOULD be identified by ***ResourceCmdParams***/@*ResourceName*, ***ResourceCmdParams***/@*Usage*, ***ResourceCmdParams***/@*ProcessUsage* or ***ResourceCmdParams***/@*ProductID*. The data type and @*Class* of ***Resource*** MAY be derived from the ***Abstract Resource***. See ▸ Section 3.8.3 Abstract Resource.

Example 5.27: Resource Command: Single Resource is Available

The following is an example for specifying that the Cyan, Front plate of *Sheet2*, signature 1 has become available.

```
<Command ID="C1" Type="Resource" xsi:type="CommandResource">
    <ResourceCmdParams JobID="MakeBrochure 012" ResourceID="ExposedMediaID"
        Status="Available">
        <Part SignatureName="Sig1" SheetName="Sheet2" Side="Front"
              Separation="Cyan"/>
    </ResourceCmdParams>
</Command>
```

Example 5.28: Resource Command: Multiple Resources are Available

The following is an example for specifying that the Black, Front plate of *Sheet2*, signature 1 has become available and is also the last plate of Sheet 2.

```
<Command ID="C2" Type="Resource" xsi:type="CommandResource">
    <ResourceCmdParams JobID="MakeBrochure 012" ResourceID="ExposedMediaID"
        Status="Available">
        <Part SignatureName="Sig1" SheetName="Sheet2" Side="Front"
              Separation="Black"/>
        <!-- the entire front of Sheet2 is also available -->
        <Part SignatureName="Sig1" SheetName="Sheet2" Side="Front"/>
        <!-- the entire Sheet2 is also available -->
        <Part SignatureName="Sig1" SheetName="Sheet2"/>
    </ResourceCmdParams>
</Command>
```

5.46.2.2 ResourceInfo

Table 5.86: ResourceInfo Element (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
ActualAmount ? New in JDF 1.2	double	When querying a Device, this Attribute reflects the current accumulated amount of the Resource that has been consumed (input) or produced (output) by the Process. This corresponds to the current value of ***ResourceLink***/@*ActualAmount* if it would be written now. When querying an MIS, this Attribute SHOULD NOT be specified.
Amount ?	double	When querying a Device, this Attribute reflects the intended accumulated amount of the Resource that will be consumed (input) or produced (output) by the Process. This corresponds to the current value of ***ResourceLink***/@*Amount* if it would be written now. When querying an MIS, this Attribute specifies the amount of the ***Consumable Resource*** that is available in inventory.
AvailableAmount ?	double	When querying a Device, this Attribute specifies the Device-specific amount of the ***Consumable Resource*** that is available in the Device. When querying an MIS, this Attribute specifies the amount of the ***Consumable Resource*** that is available in inventory
CommandResult ? New in JDF 1.4	enumeration	Result of a ***Resource Command***. **Allowed values are:** Rejected – the ***Resource Command*** was not applied to this Resource. Removed – An existing Resource was removed completely by a Resource specified in ***ResourceCmdParams***. New – A new Resource with the values specified in ***ResourceCmdParams*** was created. Merged – Values from the Resource in ***ResourceCmdParams*** were merged into an existing Resource. See the ***ResourceInfo***/***Resource*** for the merged result. Replaced – An existing Resource was replaced completely by a Resource specified in ***ResourceCmdParams***.
DeviceID ? New in JDF 1.5	NMTOKEN	Used to disambiguate the location of a Resource when a Controller is returning cumulative Resource information from its controlled Devices.
Level ? Modified in JDF 1.4	enumeration	Level of consumable resources in the device. A Device MAY specify a level status that describes a low or empty consumable level. **Allowed values are:** Empty – Specification is left to the Device manufacturer. Low – Specification is left to the Device manufacturer. OK – Specification is left to the Device manufacturer. **Modification note:** Starting with **JDF** 1.4, the default of "OK" is removed to allow job independent Resource information.

Table 5.86: ResourceInfo Element (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
Location ?	string	Device-specific string to identify the location of a given consumable, such as paper tray, ink container or thread holder. The name is the same name used in the Partition Key @*Location* of distributed Resources (see also ▸ Section 3.10.6.4 Locations of PhysicalResources). **Default value is:** the location will be selected by the Device **Values include those from:** ▸ Table C.10 Input Tray and Output Bin Names. **Note:** The specified values are for printer locations.
LotControlled ? New in JDF 1.4	boolean	Indicates that the Resource is lot controlled.
ModuleID ? New in JDF 1.3	string	@*ModuleID* of the Module that the ***Resource*** is consumed or produced by. If neither of @*ModuleID* or @*ModuleIndex* are specified, defaults to the entire Device specified by ***JMF***/@*SenderID*.
ModuleIndex ? New in JDF 1.3	IntegerRangeList	The 0-based indices of the module or modules that the ***Resource*** is consumed or produced by. If neither of @*ModuleID* or @*ModuleIndex* are specified, defaults to the entire Device specified by ***JMF***/@*SenderID*.
Orientation ? New in JDF 1.5	Orientation	Named orientation describing the transformation of the orientation of the resource relative to the ideal process coordinate that uses the resource. this attribute can be used to describe orientation dependent resources such as paper in a paper tray.
ProcessUsage ?	NMTOKEN	Selects a Resource in which the value of the ***ResourceLink*** /@*ProcessUsage* Attribute matches the token specified here in this Attribute. Only necessary if a Resource name is used more than once by one Node. For example, the ***ExposedMedia***v Input Resources of a **ConventionalPrinting** Process can be distinguished by specifying @*ProcessUsage* = "Proof" and @*ProcessUsage* = "Plate", respectively. The @*ResourceName* and @*ProcessUsage* Attributes are combined by a logical AND conjunction to select the Resource to be queried. **Values include those from:** ***ResourceLink***/@*ProcessUsage* (▸ Table 3.16 ResourceLink Element).
ProductID ? New in JDF 1.2	string	@*ProductID* of the Resource.
ResourceID ? New in JDF 1.3 Deprecated in JDF 1.5	NMTOKEN	***Resource***/@*ID* of the Resource. **Note:** The data type is NMTOKEN and not IDREF because the referenced @*ID* NEED NOT be present in the **JMF**. **Deprecation note:** Starting with **JDF** 1.5, Resources SHOULD be identified by @*ProductID* in ***Resource*** **JMF** messages.
ResourceName ?	NMTOKEN	Name of the Resource if @*Exact* = "false" in the query. @*ResourceName* specifies the primary Resource that this ***ResourceInfo*** applies to. Additional Resources MAY be specified to ensure complete references from the primary Resource. **Values include those from:** ▸ Chapter 8 Resources.
Scope ? New in JDF 1.5	enumeration	Specifies whether the query refers to a complete list of all potential Resources or to the currently loaded Resources. **Allowed values are:** Allowed – All known Resources SHALL be returned, including those that are not currently available. Present – Currently available Resources SHALL be returned.
Speed ? New in JDF 1.6	double	The current speed at which the Resource that this ***ResourceInfo*** describes is being consumed or produced. @*Speed* is defined in the units specified by @*Unit* / hour.

Table 5.86: ResourceInfo Element (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
Status ? New in JDF 1.2	enumeration	Updated @*Status* of the selected Resource. **Allowed values are** from: ***Resource***/@*Status* (▸ Table 3.10 Abstract Resource Element).
TotalAmount ? New in JDF 1.6	double	@*TotalAmount* specifies the job independent total counter setting for a given type of resource. **Note:** This allows tracking of power consumption without requiring a device to track it individually for each job.
Transformation ? New in JDF 1.6	matrix	@*Transformation* describes the transformation of the orientation of the resource (described by this ***ResourceInfo***) relative to the ideal process coordinate that uses this resources as an input or output.
Unit ?	string	Unit of the amount Attributes. In a Job context it is strongly discouraged to specify a unit other than the unit defined in the respective **JDF** Resource, although this might be necessary due to technical considerations, such as when ink is specified in weight (g) and ink measurement is specified in volume (liter). **Values include those from:** ▸ Table 1.7 Units Used in JDF**.**
Usage ? New in JDF 1.3	enumeration	Specifies a Resource in which the value of the ***ResourceLink***/@*Usage* Attribute matches the value of this Attribute. Only required if a Resource name is used both as input and output by one Node. **Allowed values are from:** ***ResourceLink***/@*Usage***.** See ▸ Table 3.16 ResourceLink Element.
AmountPool ? New in JDF 1.3	element	Definition of partial amounts and pipe parameters for this ***Resource***. The contents of the ***AmountPool*** are described for the various types of ***ResourceLink*** Elements in ▸ Table 3.18 AmountPool Element. If ***AmountPool*** is specified, the ***ResourceInfo*** SHALL NOT contain any of the amount related Attributes defined in ***AmountPool***/***PartAmount*** .
CostCenter ?	element	Cost center to which the Resource consumption is allocated.
Lot * New in JDF 1.4	element	Used when a Device is querying a Controller to determine what lots exist for the Resource being queried. When a Device is the sender of this message, lot information is specified in the ***AmountPool***, and SHALL NOT be specified here.
MISDetails ? New in JDF 1.2	element	Definition how the costs for the production of the ***Resource*** are to be charged.
Part * New in JDF 1.4	element	***Part*** Elements that describe the Resource. **Creation note:** Starting with **JDF** 1.4, ***Part*** is back after being deprecated in **JDF** 1.3.
Resource * Modified in JDF 1.4	element	**JDF** description of the resource. If the query or command leading to this Response message Element contains ***Part*** elements, the Resource SHALL contain only the appropriate matching Partitions. The data type and @*Class* of ***Resource*** is derived from the abstract resource. See ▸ Section 3.8.3 Abstract Resource. **Modification note:** Starting with **JDF** 1.4, there can be multiple occurrences of ***Resource*** elements. See @*ResourceName* for the reason.

Example 5.29: Resource Query for Consumables

The following is an example for retrieving settings:

```
<Query ID="Q1" Type="Resource" xsi:type="QueryResource">
    <ResourceQuParams Classes="Consumable" Exact="true"/>
</Query>
```

Example 5.30: Resource Response about Consumables

The following is a possible Response message to the Query message above:

```
<Response ID="M1" Type="Resource" xsi:type="ResponseResource" refID="Q1">
    <ResourceInfo AvailableAmount="2120" Location="Paper Tray 1">
        <Media ID="ID123" Class="Consumable" Status="Available">
            <!-- Media resource defined in JDF -->
        </Media>
    </ResourceInfo>
    <ResourceInfo AvailableAmount="0" Level="Empty" Location="Ink1" Unit="l">
        <Ink ID="ID124" Class="Consumable" Status="Available">>
            <!-- Ink description resource defined in JDF -->
        </Ink>
    </ResourceInfo>
</Response>
```

Example 5.31: Resource Command for Changing Amount

The following is an example for modifying the production amount of a specific Job to produce brochures.

```
<Command ID="C1" Type="Resource" xsi:type="CommandResource">
    <ResourceCmdParams JobID="MakeBrochure 012" ProductionAmount="7500"
        ResourceName="Component"/>
</Command>
```

Example 5.32: Resource Response for Changing Amount

The following is a possible response to the ***Resource Command*** message above.

```
<Response ID="M2" Type="Resource" xsi:type="ResponseResource" refID="C1">
    <ResourceInfo Amount="7500" ResourceName="Component"/>
</Response>
```

5.47 ResourcePull

New in JDF 1.2

The ***ResourcePull*** message requests a Resource from a Controller or Device. The Resource is specified as the Output Resource of a **JDF** Node. The requested Resource MAY be a subset of the Resource specified in the original **JDF**. The ***ResourcePullParams*** Element provides the parameters. The command can be used to regenerate the output of a ***QueueEntry*** or **JDF** Node with any *@Status*.

If the ***ResourcePull*** is accepted, the respective ***QueueEntry*** is requeued with ***QueueEntry***/*@Status* = "Waiting". After processing, the processing result SHALL be sent to the original submitter of the ***QueueEntry*** that is being repeated using a ***ReturnQueueEntry*** message. The sender of the ***ResourcePull*** message SHOULD be informed of the completion of the ***ResourcePull*** message with a ***Resource Command***.

Workflow Integration with ResourcePull

When ***ResourcePull*** is submitted directly to a Device in a workflow that is monitored by an MIS system, the MIS system SHALL be informed about the re-execution of the **JDF** Node, so that it can update the state of the entire Job appropriately.

Note: It is preferred to pull a Resource from a Device in a workflow that is monitored by an MIS system by sending the ***ResourcePull*** message to the MIS. The MIS can then control the Device in the standard manner and also maintain consistency of its internal Job representation.

Table 5.87: ResourcePull Message (Sheet 1 of 2)

OBJECT TYPE	ELEMENT	DESCRIPTION
CommandTypeObj	***QueueFilter*** Deprecated in JDF 1.5	Defines a filter for the returned ***Queue*** Element in the ***ResourcePull*** message.
	ResourcePullParams	Defines the parameters of the repeated Job.

Table 5.87: ResourcePull Message (Sheet 2 of 2)

OBJECT TYPE	ELEMENT	DESCRIPTION
ResponseTypeObj	***Queue*** Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.
	QueueEntry	Provides the queue entry of the repeated Job.

5.47.1 ResourcePullParams

The ***ResourcePullParams*** MAY contain queue-ordering Attributes equivalent to those used by the ***SetQueueEntryPriority*** and ***SetQueueEntryPosition*** messages. The OPTIONAL list of ***Part*** Elements refers to the Output Resource that is produced by the **JDF** Node.

Table 5.88: ResourcePullParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Amount ?	double	The @*Amount* Attribute identifies the amount of the Output Resource to be created by the **JDF** Node that is executed by the cloned ***QueueEntry***. This @*Amount* is the amount to be produced by the Process that is executed due to the ***ResourcePull***. Thus if 200 copies had been created previously and 100 copies are requested by the ***ResourcePull***, @*Amount* = "100" and not "300".
Hold = "false"	boolean	If "true", the entry is submitted as held.
JobID ?	string	@*JobID* of the **JDF** Node that creates the requested Resource. If @*QueueEntryID* is specified, @*JobID* is ignored. Exactly one of @*JobID* or @*QueueEntryID* SHALL be specified.
NextQueueEntryID ?	string	ID of the queue entry that SHALL be positioned directly behind the entry.
PrevQueueEntryID ?	string	ID of the queue entry that SHALL be positioned directly in front of the entry.
Priority = "1"	integer	Number from 0 to 100, where 0 is the lowest priority and 100 is the maximum priority.
QueueEntryID ?	string	@*QueueEntryID* of the **JDF** Node that creates the requested Resource. If @*QueueEntryID* is specified, @*JobID* and ***Part*** are ignored. Exactly one of @*JobID* or @*QueueEntryID* SHALL be specified.
RepeatPolicy ?	enumeration	Policy that defines how to reuse intermediate Resources that were generated in the original processing step (e.g., intermediate raster files in a combined RIP and ImageSetting Process). **Allowed values are:** Complete – Restart from the original Input Resources if they are available. The Process can run based on intermediate Resources if any original Resources are not available. CompleteOnly – Restart from the original Input Resources. The Process SHALL NOT run if any original Resources are not available. Fast – Reuse as many intermediate Resources as possible (e.g., restart ImageSetting from stored intermediate raster files and do not reRIP if possible).
ResourceID	string	ID Attribute of the Resource requested.
ReturnURL ? Deprecated in JDF 1.4	URL	URL where the **JDF** file SHALL be written when the Job is completed or aborted. If not specified, the **JDF** SHALL placed in the default output hot folder of the queue Controller. If @*ReturnURL* is specified with the "file" scheme, @*ReturnURL* SHALL specify an individual file. @*ReturnURL* takes precedence when ***NodeInfo***/@*TargetRoute* is specified in the previously submitted **JDF**.
WatchURL ? Deprecated in JDF 1.4	URL	URL of the Controller that SHALL be notified when the status of the ***QueueEntry*** or the underlying Job changes. Specifying @*WatchURL* is equivalent to sending a ***Subscription*** for an ***Events*** message with @*SignalTypes* = "All".

Table 5.88: ResourcePullParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Disposition ?	element	Specifies how long the ***QueueEntry*** SHOULD be retained in the queue. If not specified, the ***QueueEntry*** MAY be removed from the queue immediately after Process completion of the ***QueueEntry***.
MISDetails ?	element	Definition how the costs for the production of the ***Resource*** are to be charged.
Part *	element	The ***Part*** Elements identify the parts of a Partitioned Output Resource to be created by the **JDF** Node. The structure of the ***Part*** Element is defined in ‣ Table 3.26 Part Element. For details on Partitioned Resources, see ‣ Section 3.10.5 Description of Partitioned Resources. For details on Node Partitions, see ‣ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources.

Example 5.33: ResourcePull Command

For example, if an **ImageSetting** Process produces a Partitioned set of plates, the following example message would request only the yellow plate of the "Front" *@Surface* of *Sheet1*.

```
<Command ID="C3" Type="ResourcePull" xsi:type="CommandResourcePull">
   <ResourcePullParams QueueEntryID="AllPlates" Priority="100" ResourceID="R42">
      <Part SheetName="Sheet1" Side="Front" Separation="Yellow"/>
   </ResourcePullParams>
</Command>
```

5.48 ResubmitQueueEntry

A Job is resubmitted to a queue using the ***ResubmitQueueEntry*** message. This allows late changes to be made to a Job without affecting queue parameters and without exporting the internal structure of a queue. Resubmission overwrites the Job with **JDF** information specified in ***ResubmissionParams***/*@URL*. If ***QueueEntry***/*@Status* is neither "Waiting" nor "Held", resubmitting a queue entry MAY fail because a Device NEED NOT implement ***ResubmitQueueEntry*** for running queue entries. Resubmission does not affect other queue parameters as specified. For example, resubmission does not affect queue ordering. For details, see ‣ Table 5.20 Status Transitions for QueueEntry Handling Messages.

Table 5.89: ResubmitQueueEntry Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.2	***QueueFilter*** ? New in JDF 1.2 Deprecated in JDF 1.5	Defines a filter for the returned ***Queue*** Element in the ***ResubmitQueueEntry*** message.
	Resubmission Params	Defines the Job resubmission.
ResponseTypeObj Modified in JDF 1.5	***Queue*** Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.

5.48.1 ResubmissionParams

Table 5.90: ResubmissionParams Element

NAME	DATA TYPE	DESCRIPTION
QueueEntryID	string	ID of the queue entry to be replaced.
URL	URL	Location of the **JDF** to be submitted. It MAY be a URL with a "cid" scheme in the case of MIME Multipart/Related.

5.49 ResumeQueue

The queue is activated and queue entries can be executed. The ***ResumeQueue*** Command message is the opposite of a ***HoldQueue*** Command message.

Table 5.91: ResumeQueue Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.5	***QueueFilter*** ? New in JDF 1.2 Deprecated in JDF 1.5	Defines a filter for the ***ResumeQueue*** message.
ResponseTypeObj Modified in JDF 1.5	***Queue*** Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.

5.50 ResumeQueueEntry

The hold status of the queue entries specified by ***ResumeQueueEntryParams***/***QueueFilter***/***QueueEntryDef*** is removed. A ***QueueEntry*** with @*Status* = "Held" gets a @*Status* of "Waiting". A ***QueueEntry*** with @*Status* = "Suspended" gets a @*Status* of "Running". If ***QueueEntry***/@*GangPolicy* is other than "NoGang", a resumed ***QueueEntry*** joins its respective gang. For details, see ‣ Table 5.20 Status Transitions for QueueEntry Handling Messages.

Table 5.92: ResumeQueueEntry Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.5	***QueueEntryDef*** Deprecated in JDF 1.5	Defines the queue entry. **Deprecation note:** Starting with **JDF** 1.5, this ***QueueEntryDef*** SHOULD be located in ***ResumeQueueEntryParams***/***QueueFilter***.
	QueueFilter ? New in JDF 1.2 Deprecated in JDF 1.5	Defines a filter for the returned ***Queue*** Element in the ***ResumeQueueEntry*** message.
	ResumeQueueEntryParams ? New in JDF 1.5	
ResponseTypeObj Modified in JDF 1.5	***Queue*** Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.

5.50.1 ResumeQueueEntryParams

New in JDF 1.5

Table 5.93: ResumeQueueEntryParams Element

NAME	DATA TYPE	DESCRIPTION
QueueFilter ?	element	This ***QueueFilter*** selects the ***QueueEntry*** elements to apply the ***ResumeQueueEntry*** message to.

5.51 ReturnQueueEntry

New in JDF 1.2

The ***ReturnQueueEntry*** message returns a Job that had been submitted with a ***SubmitQueueEntry*** to the queue that represents the Controller that originally submitted the Job. The ***ReturnQueueEntryParams*** Element provides the parameters. Note that this command is emitted from the Device that is represented by the queue to a Controller or dispatcher and not to the queue, as is the case with most other queue handling commands.

Table 5.94: ReturnQueueEntry Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	***ReturnQueueEntryParams***	Defines the Job being returned from Device to Controller after processing is completed or aborted.
ResponseTypeObj	-	

5.51.1 ReturnQueueEntryParams

The *URL* Attribute specifies the location where the **JDF** file to be submitted can be retrieved by the Controller. The scheme of the *URL* Attribute (such as "file", "http" or "cid") defines the retrieval method to be used to retrieve the **JDF** .

Table 5.95: ReturnQueueEntryParams Element

NAME	DATA TYPE	DESCRIPTION
Aborted ?	NMTOKENS	ID of the **JDF** Nodes that have been executed and aborted or failed test running. If @*Aborted* and @*Completed* are empty, no executable Node was found. Note that the data type of this Attribute was erroneously specified as IDFREFS in **JDF** 1.2. and **JDF** 1.3.
Completed ?	NMTOKENS	ID of the **JDF** Nodes that have been executed and completed or succeeded in test run. Note that the data type of this Attribute was erroneously specified as IDFREFS in **JDF** 1.2. and **JDF** 1.3.
Priority ?	integer	The priority of the ***QueueEntry*** when it was executed on the Device. The Controller receiving this message MAY prioritize this Job for continued processing based on this value.
QueueEntryID	string	***QueueEntry***/@*QueueEntryID* of the returned queue entry. Note that this Attribute was erroneously omitted in **JDF** 1.2. and **JDF** 1.3.
URL	URL	Location of the **JDF** to be returned. Note that the @*URL* SHOULD be queried with a ***SubmissionMethods*** Query message to determine whether MIME Multipart/Related is supported

5.52 SetQueueEntryPosition

The position of the queue entry is modified. The ***QueueEntryPosParams*** Element provides the parameters. The position of a queue entry SHALL NOT be modified unless @*Status* = "Waiting" or @*Status* = "Held". For details, see ▸ Table 5.20 Status Transitions for QueueEntry Handling Messages.

Table 5.96: SetQueueEntryPosition Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.2	***QueueEntryPosParams***	Defines the queue entry.
	QueueFilter ? New in JDF 1.2 Deprecated in JDF 1.5	Defines a filter for the returned ***Queue*** Element in the ***SetQueueEntryPosition*** message.
ResponseTypeObj Modified in JDF 1.5	***Queue*** Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.

5.52.1 QueueEntryPosParams

@*QueueEntryID* specifies the queue entry to be moved. Jobs can either be set to a specific position within the queue or positioned next to an existing queue entry. The priority of the entry matches the priority of the entry that precedes it, after it has been repositioned.

Table 5.97: QueueEntryPosParams Element

NAME	DATA TYPE	DESCRIPTION
NextQueueEntryID ?	string	ID of the queue entry that SHALL be positioned directly behind the entry. Exactly one of @*NextQueueEntryID*, @*PrevQueueEntryID* or @*Position* SHALL be specified.
PrevQueueEntryID ?	string	ID of the queue entry that SHALL be positioned directly in front of the entry. Exactly one of @*NextQueueEntryID*, @*PrevQueueEntryID* or @*Position* SHALL be specified.
Position ?	integer	Position in the queue. "0" = pole position. Note that the position is based on the queue before modification. Thus if a queue entry is moved back in the queue, its final position is one lower than specified in @*Position*. Exactly one of @*NextQueueEntryID*, @*PrevQueueEntryID* or @*Position* SHALL be specified.
QueueEntryID	string	ID of a queue entry.

5.53 SetQueueEntryPriority

The priority of the queue entry is modified. The ***QueueEntryPriParams*** Element provides the parameters. For details, see ▸ Table 5.20 Status Transitions for QueueEntry Handling Messages.

Table 5.98: SetQueueEntryPriority Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.5	***QueueEntryPriParams***	Defines the queue entry.
	QueueFilter ? New in JDF 1.2 Deprecated in JDF 1.5	Defines a filter for the returned ***Queue*** Element in the ***SetQueueEntryPriority*** message.
ResponseTypeObj Modified in JDF 1.5	***Queue*** Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.

5.53.1 QueueEntryPriParams

@*QueueEntryID*, described in the table below, specifies the queue entry that has its priority modified.

Table 5.99: QueueEntryPriParams Element

NAME	DATA TYPE	DESCRIPTION
Priority	integer	Number from 0 to 100, where "0" = lowest priority and "100" = maximum priority. The priority from ***QueueSubmissionParams***/@*Priority* and ***QueueEntryPriParams***/@*Priority* takes precedence over ***NodeInfo***/@*JobPriority*.
QueueEntryID Deprecated in JDF 1.5	string	ID of a queue entry.
QueueFilter ? New in JDF 1.5	element	This ***QueueFilter*** selects the ***QueueEntry*** elements to apply the ***SetQueueEntryPriority*** message to.

5.54 ShutDown

New in JDF 1.2

The ***ShutDown*** Command message shuts down a Controller or Device. A Device SHALL use the ***Status*** message if it signals its own shutdown.

Table 5.100: ShutDown Message

OBJECT TYPE	ELEMENT	DESCRIPTION
CommandTypeObj	***QueueFilter*** Deprecated in JDF 1.5	Defines a filter for the returned ***Queue*** Element in the ***ShutDown*** message.
	ShutDownCmdParams	Defines the details of a shutdown.
ResponseTypeObj	***DeviceInfo***	Describes the Device status as anticipated after the shut-down.
	Queue Deprecated in JDF 1.5	Provides information about the queue and all its entries as anticipated after the shutdown. This Element will only be provided if the Device has queue capabilities. The Queue Element is described in ▸ Section 5.14 Elements for Queues.

5.54.1 ShutDownCmdParams

Table 5.101: ShutDownCmdParams Element

NAME	DATA TYPE	DESCRIPTION
ShutDownType = "StandBy"	enumeration	Defines the device shutdown method. **Allowed values are:** StandBy – The device is set to standby mode. It can be restarted with a ***WakeUp*** **JMF** message. ***DeviceInfo***/@*DeviceStatusDetails* SHOULD be "StandBy". If the device requires a ***WakeUp*** **JMF** prior to accepting new jobs, ***DeviceInfo***/@*DeviceStatus* SHALL be "Down", else it SHALL be "Idle". Full – Completely shut down the device. It is no longer accessible via **JMF** after the shutdown. ***DeviceInfo***/@*DeviceStatusDetails* SHOULD be "Shut-Down". DeviceInfo/@*DeviceStatus* SHALL be "Down".
FlushQueueParams ?	element	Defines the policy for flushing the queue upon shutdown. If not specified, the queue is not flushed. The behavior of a queue after shutdown is system specific.

5.55 Status

The ***Status*** message queries the general status of a device or a controller and the status of jobs associated with this device or controller. No job context is needed to issue a ***Status*** message. The response contains one or more ***DeviceInfo*** elements, which contain the device specific information and which MAY contain other ***JobPhase*** elements that in turn contain the job specific information .

Table 5.102: Status Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***StatusQuParams***	Refines the query to include various aspects of the device and job states.
ResponseTypeObj	***DeviceInfo*** +	Describes the actual device Status. If multiple ***DeviceInfo*** Elements are specified, these describe multiple Devices. A sequential state change of an individual Device SHALL be encoded as 2 separate Signals.
	Queue ? Deprecated in JDF 1.5	Provides information about the queue and all its entries. This Element will only be provided if the Device has queue capabilities. The ***Queue*** Element is described in ▸ Section 5.14 Elements for Queues.

Example 5.34: Status Signal

New in JDF 1.4

Example of a Status Signal for a phase switch from setup to running

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
       SenderID="MIS master A" TimeStamp="2007-08-09T11:35:41+02:00"
       MaxVersion="1.4" Version="1.4"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Signal ID="m18" Type="Status" xsi:type="SignalStatus">
       <DeviceInfo DeviceStatus="Running">
          <JobPhase JobID="jID" JobPartID="jpID"
                PhaseStartTime="2007-08-09T11:35:40+02:00" Status="Setup"/>
       </DeviceInfo>
    </Signal>
    <Signal ID="m19" Type="Status" xsi:type="SignalStatus">
       <DeviceInfo DeviceStatus="Running">
          <JobPhase JobID="jID" JobPartID="jpID"
                PhaseStartTime="2007-08-09T11:35:41+02:00" Status="InProgress"/>
       </DeviceInfo>
    </Signal>
</JMF>
```

5.55.1 StatusQuParams

StatusQuParams is a filter that refines the level of information that SHALL be returned in the response or signals.

Table 5.103: StatusQuParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
DeviceDetails = "None"	enumeration	Refines the provided status information about the Device. **Allowed values are:** None – ***DeviceInfo***/@*DeviceID* and ***DeviceInfo***/@*DeviceStatus*. shall be provided. DevInfo/@* MAY be provided. ***DeviceInfo***/***Employee***, ***DeviceInfo***/***Device*** and ***DeviceInfo*** /***ModuleStatus*** elements SHALL NOT be provided. Brief – Provide all available Device information except for ***Device*** Elements. The provided information includes ***JobPhase*** and ***Activity*** elements. Modules – Provide ***ModuleStatus*** Elements with module specific status details. Details – Provide maximum available Device information excluding Device capability descriptions. Includes ***Device*** Elements which represent details of the Device. Capability – Provide ***Device*** Elements with ***DeviceCap*** Subelements which represent details of the capabilities of the Device. Full – Provide maximum available Device information including Device capability descriptions. Includes ***Device*** elements which represent details of the Device.
EmployeeInfo = "false"	boolean	If "true", ***Employee*** elements are to be provided in the response. Those Elements describe the employees which are associated to the Device independent on any Job.
JobDetails = "None"	enumeration	Refines the provided status information about the Jobs associated with the Device. Each higher entry includes the values specified in the lower entries. **Allowed values are:** None – ***JobPhase***/@*JobID*, ***JobPhase***/@*JobPartID* SHALL be specified if a job is running. ***JobPhase***/***Part*** SHOULD be specified if a job is running. At least one of ***JobPhase***/@*Amount* and ***JobPhase***/@*PercentCompleted* SHALL be specified if a job is running. Additional job related data SHOULD NOT be specified. MIS – Provide business with the relevant information contained in the ***CostCenter*** Element and the @*DeadLine*, @*DeviceStatus*, @*Status*, @*StatusDetails* and the various @*Counter* Attributes. In **JDF** 1.2 and beyond, this value is identical to "Brief". Deprecated in JDF 1.2 Brief – Provide all available status information including ***JobPhase*** and ***Activity*** Elements except for **JDF**. Full – Provide maximum available status information. Includes a URL reference to an actual **JDF** which represents a snapshot of the current job state.

Table 5.103: StatusQuParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
JobID ?	string	@*JobID* of the **JDF** Node whose status is being queried. The @*JobID* SHALL be unique within the workflow. If not specified, list all known Jobs.
JobPartID ?	string	@*JobPartID* of the **JDF** Node whose status is being queried.
QueueEntryID ? New in JDF 1.2	string	@*QueueEntryID* of the job that is being queried. If @*QueueEntryID* is specified, @*JobID*, @*JobPartID* and ***Part*** are ignored. If none of @*JobID*, @*JobPartID*, ***Part*** or @*QueueEntryID* are specified, ***StatusQuParams*** applies to all jobs.
QueueInfo Deprecated in JDF 1.5	boolean	If "true", a ***Queue*** element is requested to be provided. This is analogous to a ***QueueStatus*** Query message. **Deprecation note:** In **JDF** 1.6 and beyond, use a ***QueueStatus*** message.
Part * New in JDF 1.2	element	***Part*** elements that describe the partition of the job whose status is queried. For details on node partitions, see ▸ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources.

5.55.2 DeviceInfo

The response message returns a ***DeviceInfo*** element for the queried device.

Table 5.104: DeviceInfo Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
CounterUnit ?	string	The unit of the @*ProductionCounter*, the @*TotalProductionCounter* and numerator unit of @*Speed*. The default unit is the default unit defined by **JDF** for the Output Resource of the Node executed by the Device. For example, in case of a Sheet-Fed printer, it is the number of Sheets; in case of a Web Printer, it is the length of printed Web in meters. **Values include those from:** ▸ Table 1.7 Units Used in JDF.
DeviceCondition ? New in JDF 1.2	enumeration	The general condition of a Device. **Allowed values are:** OK – The Device is in working condition. NeedsAttention – The Device is still in working condition but requires attention. Failure – The Device is not in working condition. OffLine – The Device is off line and its condition is unknown.
DeviceID ? New in JDF 1.3	string	@*DeviceID* of the ***Device*** that this ***DeviceInfo*** describes. @*DeviceID* SHALL match ***Device***/@*DeviceID* if ***Device*** is specified in this ***DeviceInfo***.
DeviceStatus	enumeration	The status of a Device. **Allowed values are:** Unknown – No Device is known or the Device cannot provide a @*DeviceStatus*. Idle – No Job is being processed and the Device is accepting new Jobs. Down – No Job is being processed and the Device currently cannot execute a Job. The Device might be broken, switched off, etc. Setup – The Device is currently being set up. This state is allowed to occur also during the execution of a Job. Running – The Device is currently executing a Job. Cleanup – The Device is currently being cleaned. This state is allowed to occur also during the execution of a Job. Stopped – The Device has been stopped, probably temporarily. This status indicates some kind of break, including a pause, maintenance or a breakdown, as long as execution has not been aborted.
HourCounter ?	duration	The total integrated time (life time) of Device operation in hours.

Table 5.104: DeviceInfo Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
IdleStartTime ? New in JDF 1.4	dateTime	Specifies the beginning of the last phase with no ***JobPhase*** entries. A Device is idle when no active Jobs are being processed. Multiple phases with different status values and no active Job phases MAY be specified, for instance a maintenance phase followed by an idle phase. @*IdleStartTime* SHALL NOT be specified if ***JobPhase*** Elements are present in the ***DeviceInfo*** or @*DeviceStatus* != "Idle", "Down" or "Stopped".
PowerOnTime ?	dateTime	Date and time when the device was switched on.
ProductionCounter ?	double	The current machine production counter. This counter can be reset. Typically, it starts counting at power-on time. The reset of this counter MAY be signaled by a ***Notification*** [@*Class*="Event",@*Type* ="CounterReset"] message (see ▸ Section C.8 NotificationDetails).
Speed ?	double	The current Machine speed. @*Speed* is defined in the same units as @*ProductionCounter* / hour.
StatusDetails ?	string	String that defines the Device state more specifically. **Values include those from:** ▸ Section C.11 StatusDetails Supported Strings.
TotalProductionCounter ?	double	The current total Machine production counter since the Machine was produced.
Activity * New in JDF 1.5	element	Device and operator activities that are related to the device and are unrelated to a specific job.
Device ?	element	A ***Device*** Resource that describes details of the Device. The data type of ***Device*** is ResourceElement. See ▸ Section 3.10.1 ResourceElement – Subelement of a Resource.
Employee *	element	***Employee*** Resources that describe which employees are currently working at the Device. The data type of ***Employee*** is ResourceElement. See ▸ Section 3.10.1 ResourceElement – Subelement of a Resource.
JobPhase *	element	Each ***JobPhase*** SHALL describe the actual status of jobs in the device. All jobs that are active on the device SHALL be specified. Supplying no ***JobPhase*** specifies that no Job is currently active on the Device. Active Jobs have ***JDF***/@*Activation* = "Active", "TestRun" or "TestRunAndGo" and ***JDF***/@*Status* or ***JDF***/***StatusPool***/***PartStatus***/@*Status* = "TestRunInProgress", "Setup", "InProgress", "Cleanup" or "Stopped". Multiple ***JobPhase*** Elements specify that multiple Job phases are active simultaneously on the Device. For details on using ***JobPhase*** Elements, see ▸ Table 5.105 JobPhase Element.
ModuleStatus *	element	Status of individual modules that are in use independent of a Job. ***ModuleStatus*** SHALL not be specified for modules that are specified in ***JobPhase***/***ModuleStatus***. For details on using ***ModuleStatus*** Elements, see ▸ Table 5.106 ModuleStatus Element.

5.55.3 JobPhase

A ***Status*** response message MAY provide ***JobPhase*** elements. The ***JobPhase*** element represents the actual state of a job. The ***JobPhase*** Element is an analogue to the ***PhaseTime*** Audit Element described in ▸ Section 3.11.4.6 PhaseTime. The main difference between a ***JobPhase*** Element and a ***PhaseTime*** Audit Element is that a ***JobPhase*** message Element reflects a snapshot of the current Job status whereas the ***PhaseTime*** Audit Element reflects a time span bordered by two (sub-) status transitions.

For exact information about the Job phase, a ***JobPhase*** Element MAY include a URL reference to a copy of the current state of the Job described as **JDF**. If ***Part*** Elements are specified, all Attributes in ***JobPhase*** apply only to the specified

parts. If an actual **JDF** is not supported by the Controller, the same rules apply for the ***Status*** Response message as those which apply for the ***Resource*** Response message.

Table 5.105: JobPhase Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Activation ? New in JDF 1.1	enumeration	The activation of the **JDF** Node. **Allowed values are** from: ***JDF***/@*Activation* (▸ Table 3.4 JDF).
Amount ?	double	Sum of actual @*Amount* that the node defined in this ***JobPhase*** produced since @*StartTime*. If @*Waste* is also specified, the value SHALL be without waste. The unit MAY be specified in the @*CounterUnit* attribute of the parent element ***DeviceInfo***.
DeadLine ?	enumeration	Scheduling state of the Job. **Allowed values are:** InTime – The Job or Job Part will probably not miss the deadline. Warning – The Job or Job Part could miss the deadline. Late – The Job or Job Part will miss the deadline. **Note:** For more details on scheduling, see ***NodeInfo***.
JobID ?	string	@*JobID* of the **JDF** node that is executing.
JobPartID ?	string	@*JobPartID* of the **JDF** node that is executing.
PercentCompleted ?	double	Node processing progress in percent (%) completed.
PhaseAmount ? New in JDF 1.2	double	Actual amount that the Node defined in this ***JobPhase*** produced during this ***JobPhase***. If @*PhaseWaste* is also specified, the value is without waste. The unit is specified in the @*CounterUnit* Attribute of the parent Element ***DeviceInfo***.
PhaseStartTime ? New in JDF 1.2	dateTime	Time that this ***JobPhase*** started.
PhaseWaste ? New in JDF 1.2	double	Actual amount of waste that the Node defined in this ***JobPhase*** produced during this ***JobPhase***. The unit is specified in the @*CounterUnit* Attribute of the parent Element ***DeviceInfo***.
QueueEntryID ?	string	If the job was submitted to a ***Queue*** and the @*QueueEntryID* is known, this attribute SHOULD be provided.
RestTime ? New in JDF 1.1	duration	Estimated duration of time to finishing processing this Node.
SpawnID ? New in JDF 1.5	NMTOKEN	@*SpawnID* allows distinguishing multiple spawned Jobs with the same @*JobID*.
Speed ?	double	The current job speed. @*Speed* is defined in the same units as @*ProductionCounter* / hour. Defaults to the speed specified in the ***DeviceInfo*** element.
StartTime ? New in JDF 1.1	dateTime	Time when execution of the Node that is described by this ***JobPhase*** has been started, defined by the transition of ***JDF***/@*Status* from "Waiting" or "Ready" to any active value.
Status	enumeration	The status of the **JDF** node. **Allowed values are** from: ***JDF***/@*Status* see ▸ Table 3.4 JDF).
StatusDetails ?	string	String that defines the Job state more specifically. Values include those from: ▸ Section C.11 StatusDetails Supported Strings.
TotalAmount ? New in JDF 1.1	double	Planned amount that will be produced when this Job phase is 100% completed. The unit is specified in the @*CounterUnit* Attribute of the parent Element ***DeviceInfo***.

Table 5.105: JobPhase Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
URL ? New in JDF 1.4	URL	URL of a copy of the complete **JDF** that represents a snapshot of the Job that is currently being processed. The **JDF** is for reference only and SHALL not be merged with the main **JDF** of the Job using spawning and merging methods. ***JDF***/@*Activation* SHOULD be set to "Informative" in this **JDF** Element. The URL SHOULD reference a MIME part using a "cid" URL scheme.
Waste ? New in JDF 1.1	double	Total @*Amount* of waste that the node defined in this ***JobPhase*** produced since @*StartTime*. The unit MAY be specified in the @*CounterUnit* attribute of the parent element ***DeviceInfo***.
Activity * New in JDF 1.5	element	Device and Operator activities that are related to a specific job or job phase.
CostCenter ?	element	The cost center that the Job is currently being charged to. Defaults to the cost center specified in the ***DeviceInfo*** Element.
GangSource * New in JDF 1.6	element	If present, each ***GangSource*** SHALL represent the source jobs that are being processed as a gang job by this ***QueueEntry***.
JDF ? Deprecated in JDF 1.4	element	Complete **JDF** Node that represents a snapshot of the Job that is currently being processed. This Element is for reference only and SHALL NOT be merged with the main **JDF** of the Job using spawning and merging methods. ***JDF***/@*Activation* SHALL be set to "Informative" in this **JDF** Element. **Deprecation note:** Starting with **JDF** 1.4, **JDF** has been replaced by @*URL*. This avoids clashes of identical @*ID* Attributes when multiple ***JobPhase*** Elements from the same **JDF** are specified.
MISDetails ? New in JDF 1.2	element	Definition how the costs for this ***JobPhase*** are to be charged.
ModuleStatus * New in JDF 1.3	element	Status of individual modules that are used to execute this ***JobPhase***. ***ModuleStatus*** SHALL NOT be specified for modules that are specified in ***DeviceInfo***/***ModuleStatus***. For details on using ***ModuleStatus*** Elements, see ▸ Table 5.106 ModuleStatus Element.
Part * Modified in JDF 1.1	element	Describes which parts of a job are currently being processed. For details on node partitions, see ▸ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources.

5.55.4 ModuleStatus

The ***ModuleStatus*** Element restricts the scope of a ***JobPhase*** or ***DeviceInfo*** Element to apply only to the device modules that are selected by the list of ***ModuleStatus*** Elements. The ***ModuleStatus*** Element is similar to the ***ModulePhase*** Element of the ***PhaseTime*** Audit Element (see ▸ Table 3.39 ModulePhase Element). ***ModulePhase***/@*DeviceID* Attribute is not specified because it is already uniquely identified in ***DeviceInfo***/@*DeviceID*. The ***ModuleStatus*** Element is described in the following table.

Table 5.106: ModuleStatus Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
CombinedProcessIndex ? New in JDF 1.3	IntegerList	@*CombinedProcessIndex* Attribute specifies the indices of individual Processes in the @*Types* Attribute to which a ***ModuleStatus*** that describes a Combined Process Node or Process Group Node belongs. Multiple entries in @*CombinedProcessIndex* specify that the Module specified by ***ModuleStatus*** is executing the respective multiple Processes in the Combined Process Node.

Table 5.106: ModuleStatus Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
DeviceStatus ?	enumeration	Status of the module. **Allowed values are:** Unknown – The module status is unknown. Idle – The module is not used. An example is a color print module that is inactive during a black-and-white print. Down – The module cannot be used. It might be broken, switched off etc. Setup – The module is currently being set up. Running – The module is currently executing. Cleanup – The module is currently being cleaned. Stopped – The module has been stopped, but running might be resumed later. This status can indicate any kind of break, including a pause, maintenance or a breakdown, as long as running can be easily resumed.
ModuleID ? New in JDF 1.3	string	@*ModuleID* of the Module that ***ModuleStatus*** refers to. If not specified, the module is specified in @*ModuleIndex*. At least one of @*ModuleID* or @*ModuleIndex* SHALL be specified.
ModuleIndex ? Modified in JDF 1.3	IntegerRangeList	The 0-based indices of the module or modules. If multiple module types are available on one Machine, indices SHALL also be unique. @*ModuleIndex* is unique within the machine.
ModuleType ? Modified in JDF 1.5	NMTOKEN	Module description **Values include those from:** ▸ Section C.2 DeviceClass and ModuleType Supported Strings. **Note:** The allowed values depend on the type of Device. Each type of Device has a separate table of values. **Modification note:** Starting with **JDF** 1.5, @*ModuleType* is optional.
StatusDetails ?	string	Description of the module status phase that provides details beyond the enumerative values given by the @*DeviceStatus* Attribute. **Values include those from:** ▸ Section C.11 StatusDetails Supported Strings.
Employee * Deprecated in JDF 1.5	element	***Employee*** Resource(s) that represent the employee(s) that are working at this module (the module is specified by the Attributes @*ModuleIndex* and @*ModuleType*). The data type of ***Employee*** is ResourceElement. See ▸ Section 3.10.1 ResourceElement – Subelement of a Resource.

Example 5.35: Status Response to Query

The following is an example of a ***Response*** message to a ***Status*** Query message. The device in this example holds one job and executes another job that is currently printed duplex (each side) on four-color modules for the front and three-color modules for the back, with one idle:

```
<Response ID="M1" refID="Q1" Type="Status" xsi:type="ResponseStatus">
   <DeviceInfo DeviceStatus="Running" StatusDetails="Waste">
      <JobPhase Amount="2560" DeadLine="InTime" JobID="678" JobPartID="01"
            PercentCompleted="52" QueueEntryID="Job-05" Status="InProgress"
            StatusDetails="Waste"/>
      <JobPhase Amount="0" DeadLine="Warning" JobID="679" JobPartID="01"
            PercentCompleted="0" QueueEntryID="Job-06" Status="Ready"/>
      <ModuleStatus ModuleIndex="0~3 6~8" ModuleType="PrintModule"
            DeviceStatus="Running"/>
      <ModuleStatus ModuleIndex="4" ModuleType="PrintModule" DeviceStatus="Idle"/>
      <ModuleStatus ModuleIndex="5" ModuleType="PerfectingModule"
            DeviceStatus="Running"/>
   </DeviceInfo>
</Response>
```

5.56 StopPersistentChannel

The ***StopPersistentChannel*** Command message unregisters a listening Controller from a persistent channel. No more s are sent to the Controller once the command has been issued. A certain subset of signals MAY be addressed for unsubscription by specifying a ***StopPersChParams*** Element.

Table 5.107: StopPersistentChannel Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	***StopPersChParams***	Specifies the persistent channel and the message types to be unsubscribed.
ResponseTypeObj	***SubscriptionInfo***	One ***SubscriptionInfo*** element SHALL be returned for every Persistent Channel that was removed.

5.56.1 StopPersChParams

StopPersChParams provides a filter that selects persistent channels that SHALL be unregistered.

Table 5.108: StopPersChParams Element

NAME	DATA TYPE	DESCRIPTION
ChannelID ? Deprecated in JDF 1.5	NMTOKEN	@*ChannelID* of the persistent channel to be deleted. If the channel has been created with a ***Query*** message, the @*ChannelID* specifies the @*ID* of the ***Query*** message (identical to the @*refID* of the ***Response*** message).
MessageType ?	NMTOKEN	Only messages with a matching message type are suppressed. **Default value is:** all message types **Values include those from:** ***Message***/@*Type*.
DeviceID ?	NMTOKEN	Only messages from Devices or Controllers with a matching @*DeviceID* Attribute are suppressed.
JobID ? Deprecated in JDF 1.5	string	Only messages with a matching @*JobID* Attribute are suppressed. **Deprecation note:** Job specific subscriptions are discouraged.
JobPartID ? Deprecated in JDF 1.5	string	Only messages with a matching @*JobPartID* Attribute are suppressed. **Deprecation note:** Job specific subscriptions are discouraged.
QueueEntryID ? New in JDF 1.2 Deprecated in JDF 1.5	string	@*QueueEntryID* of the Job whose messages are queried/subscribed. If @*QueueEntryID* is specified, @*JobID*, @*JobPartID* and ***Part*** are ignored. If none of @*JobID*, @*JobPartID*, ***Part*** or @*QueueEntryID* are specified, ***StopPersChParams*** applies to all Jobs that will be processed by the receiver. **Deprecation note:** Job specific subscriptions are discouraged.
URL	URL	URL of the receiving controller. This SHALL be identical to the @*URL* that was used to create the persistent channel. If no @*ChannelID* is specified, all persistent channels to this @*URL* are deleted.
Part * New in JDF 1.2 Deprecated in JDF 1.5	element	***Part*** Elements that describe the Partition of the Job whose messages are suppressed. For details on Node Partitions, see ▸ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources. **Deprecation note:** Job specific subscriptions are discouraged.

5.57 SubmissionMethods

The ***SubmissionMethods*** message returns information about the ***QueueEntry*** submission and return formats that are supported by a Device or Controller. Thus, it can be used to determine the details of how a ***SubmitQueueEntry*** message can be sent to a Device, or the details of a ***ReturnQueueEntry*** message that will be returned by the Device.

Table 5.109: SubmissionMethods Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	—	—
ResponseTypeObj	***SubmissionMethods*** ?	Describes the submission methods supported by the queue.

5.57.1 SubmissionMethods

The Response message Element MAY contain multiple Attributes, as defined below. If an Attribute is not specified, the corresponding submission method is not supported.

Table 5.110: SubmissionMethods Element

NAME	DATA TYPE	DESCRIPTION
File ? Deprecated in JDF 1.2	boolean	Can retrieve a **JDF** from a File specified in the URL In **JDF** 1.2 and beyond, include "file" in @*URLSchemes*.
HotFolder ? Deprecated in JDF 1.4	URL	URL specification of a hot folder location. **Deprecation note:** Starting with **JDF** 1.4, use the ***KnownDevices*** Response: /***JMF***/***Response***/***DeviceInfo***/***Device***/@*JDFInputURL*
HttpGet ? Deprecated in JDF 1.2	boolean	Can retrieve a **JDF** via HTTP **get** commands. In **JDF** 1.2 and beyond, include "http" in @*URLSchemes*.
MIME ? Deprecated in JDF 1.2	boolean	Accepts MIME Multipart/Related submission messages via a message post. In **JDF** 1.2 and beyond, use @*Packaging* = "MIME".
Packaging ? New in JDF 1.2 Modified in JDF 1.4	enumerations	List of packaging methods supported. **Default behavior:** the Controller does not support receiving packaged messages and SHALL retrieve **JDF** files using a URL with a scheme other than "cid". **Allowed values are:.** MIME – Accepts MIME Multipart/Related packaging of **JMF**, **JDF** and digital assets. None – no form of packaging is supported. New in JDF 1.4
URLSchemes ? New in JDF 1.2	NMTOKENS	List of schemes supported in for retrieving **JDF** files. If not specified, the Controller does not support retrieving **JDF** files from remote URLs. **Values include:** file – The file scheme according to ▸[RFC1738] and ▸[RFC3986]. ftp – FTP (File Transfer Protocol) http – HTTP (Hypertext Transport Protocol) https – HTTPS (Hypertext Transport Protocol — Secure)

Example 5.36: SubmissionMethods Response

The following is an example of a Response message to a ***SubmissionMethods*** Query message:

```
<Response ID="M1" Type="SubmissionMethods"
       xsi:type="ResponseSubmissionMethods" refID="Q1">
    <SubmissionMethods HotFolder="file://MyDevice/HotFolder" Packaging="MIME"
          URLSchemes="http file ftp"/>
</Response>
```

5.58 SubmitQueueEntry

SubmitQueueEntry submits a Job to a queue of a Device or Controller ***QueueSubmissionParams*** provides the parameters of the submission.

Table 5.111: SubmitQueueEntry Message (Sheet 1 of 2)

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.5	***QueueFilter*** ? New in JDF 1.2 Deprecated in JDF 1.5	Defines a filter for the returned ***Queue*** Element in the ***SubmitQueueEntry*** message.
	QueueSubmissionParams	Defines the Job submission.

Table 5.111: SubmitQueueEntry Message (Sheet 2 of 2)

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
ResponseTypeObj Modified in JDF 1.5	***Queue*** Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed.
	QueueEntry ? Modified in JDF 1.2	Provides the queue entry of the submitted Job. ***QueueEntry*** SHALL be specified if the submission was successful and SHALL be omitted in case the submission was rejected.
Definition of the ***QueueEntry*** Elements, see ▸ Section 5.14 Elements for Queues.		

5.58.1 QueueSubmissionParams

The job submission can contain queue-ordering attributes equivalent to those used by the ***SetQueueEntryPriority*** and ***SetQueueEntryPosition*** messages. The *@URL* attribute specifies the location where the **JDF** file to be submitted can be retrieved by the queue controller. The location type in the *@URL* attribute (such as "file", "http" or "cid") defines the submission method. *@ReturnURL* or *@ReturnJMF* may specify the location where the modified **JDF** shall be sent after the job is completed or aborted.

The *@URL* attribute specifies the location where the queue controller can retrieve the **JDF** file to be submitted.

Table 5.112: QueueSubmissionParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Activation ?	enumeration	Activation of the submitted **JDF**. **Allowed values are from:** *@Activation* in ▸ Table 3.4 JDF.
GangName ? New in JDF 1.3	NMTOKEN	Name of the Gang for the Job. If *@GangName* is specified, the ***QueueEntry*** SHOULD be executed together with other ***QueueEntry*** Elements that share a common value of *@GangName*. If *@GangName* is not known, the receiving Device MAY either return an error 131 or create the gang with *@GangName* on the fly.
GangPolicy ? New in JDF 1.3	enumeration	Ganging policy for the ***QueueEntry***. **Allowed value is from:** ▸ GangPolicy
Hold = "false"	boolean	If "true", the entry is submitted as with ***QueueEntry***/*@Status*="Held". If a ***QueueEntry*** is submitted with *@Hold* = "true" and *@GangPolicy* is other than "NoGang", the ***QueueEntry*** retains its respective gang data but does not influence execution of other Jobs that are in the gang.
NextQueueEntryID ?	string	ID of the queue entry that SHALL be positioned directly behind the entry. At most one of *@NextQueueEntryID*, *@PrevQueueEntryID* or *@Priority* SHALL be specified.
PrevQueueEntryID ?	string	ID of the queue entry that SHALL be positioned directly in front of the entry. At most one of *@NextQueueEntryID*, *@PrevQueueEntryID* or *@Priority* SHALL be specified.
Priority = "1"	integer	Number from 0 to 100, where "0" = lowest priority and "100" = maximum priority. Exactly one of *@NextQueueEntryID*, *@PrevQueueEntryID* or *@Priority* SHALL be specified. Note that ***QueueSubmissionParams***/*@Priority* is not the same as ***NodeInfo***/*@Priority*. ***QueueSubmissionParams***/*@Priority* specifies the priority in the context of the device queue whereas ***NodeInfo***/*@Priority* specifies the priority of the task in general. ***QueueSubmissionParams***/*@Priority* MAY be modified due to additional scheduling information (e.g., ***NodeInfo***/*@FirstStart*). The priority from ***QueueSubmissionParams***/*@Priority* and ***QueueEntryPriParams****@Priority* takes precedence over ***NodeInfo***/*@JobPriority*.
refID ? New in JDF 1.2	NMTOKEN	Copy of the *@ID* attribute of the initiating ***RequestQueueEntry*** message.

Table 5.112: QueueSubmissionParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ReturnJMF ? New in JDF 1.2	URL	Address of a **JMF** queue where a ***ReturnQueueEntry*** message SHALL be sent when the ***QueueEntry*** is completed or aborted. Note that the @*ReturnJMF* queue SHOULD be queried with a ***SubmissionMethods*** Query message to determine whether MIME Multipart/Related is supported by the return queue. @*ReturnJMF* SHALL NOT be specified if @*ReturnURL* is present.
ReturnURL ? Modified in JDF 1.2	URL	URL where the **JDF** file SHALL be written when the ***QueueEntry*** is completed or aborted. A Controller SHALL write only a **JDF** document to the URL and SHALL NOT write a MIME Multipart package to the URL. If @*ReturnURL* is specified with the "file" scheme, @*ReturnURL* SHALL specify an individual file. @*ReturnURL* SHALL take precedence when ***NodeInfo***/@*TargetRoute* is specified in the submitted **JDF**. **Note:** A controller SHALL NOT return a **JDF** file or MIME Multipart/Related file by performing a ***SubmitQueueEntry*** or ***ReturnQueueEntry***to the @*ReturnURL* URL. The controller specified by @*ReturnURL* SHALL NOT accept **JMF** messages. See instead @*ReturnJMF*. @*ReturnURL* SHALL NOT be specified if @*ReturnJMF* is present.
URL Modified in JDF 1.2	URL	Location of the **JDF** to be submitted. In the case of MIME Multipart/Related, the URL MAY have a "cid" scheme.
WatchURL ? Modified in JDF 1.2 Deprecated in JDF 1.5	URL	URL of the controller that SHALL be notified when the status of the ***QueueEntry*** or the underlying Job changes.
Disposition ? New in JDF 1.2	element	Definition how long the ***QueueEntry*** SHOULD be retained in the queue. If not specified, the ***QueueEntry*** MAY be removed from the queue immediately after Process completion of the ***QueueEntry***.

URL with "file" Scheme

If the URL has a "file" scheme, the Device retrieves the file at the location specified in the @*URL* Attribute. The following example declares a file on the network:

Example 5.37: SubmitQueueEntry Command with "file" Scheme

```
<Command ID="M1" Type="SubmitQueueEntry" xsi:type="CommandSubmitQueueEntry">
   <QueueSubmissionParams URL="File://MyNetWorkShare/AnyDirectory/job1.jdf"/>
</Command>
```

URL with "http" Scheme

In this example, the queue controller retrieves the file with a standard HTTP **get** command from a host that MAY be remote. The job delivered as a response to the HTTP **get** command MAY be a MIME Multipart/Related entity. The HTTP server MAY retrieve a file or it MAY generate the response dynamically with a CGI script or other such tool.

Example 5.38: SubmitQueueEntry Command with "http" Scheme

```
<Command ID="M2" Type="SubmitQueueEntry" xsi:type="CommandSubmitQueueEntry" >
   <QueueSubmissionParams URL="http://JobServer.JDF.COM?job1"/>
</Command>
```

JDF Package Submission

If a Controller is capable of decoding MIME, it is legal to submit a MIME Multipart/Related message. See ▸ Section 11.3 JDF Packaging for details of MIME Multipart/Related packaging.

5.59 SuspendQueueEntry

New in JDF 1.2

The entry specified by ***QueueEntryDef*** is suspended if its @*Status* is "Running". Its @*Status* is set to "Suspended". Whether other queue entries can be run while the queue entry remains suspended depends on implementation. The ***Suspend-***

QueueEntry Command message has no effect on Jobs with a @*Status* other than "Running". For details, see ▸ Table 5.20 Status Transitions for QueueEntry Handling Messages.

Table 5.113: SuspendQueueEntry Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj Modified in JDF 1.5	*QueueEntryDef* Deprecated in JDF 1.5	Defines the queue entry.
	QueueFilter ? Deprecated in JDF 1.5	Defines a filter for the returned *Queue* Element in the *SuspendQueueEntry* message.
	SuspendQueueEntryParams ? New in JDF 1.5	
ResponseTypeObj Modified in JDF 1.5	*Queue* Deprecated in JDF 1.5	Describes the state of the queue after the command has been executed. See ▸ Section 5.14 Elements for Queues for the definition of the Elements listed above. The entry specified by *QueueEntryDef* remains in the queue but moved into the "Suspended" state.

5.59.1 SuspendQueueEntryParams

New in JDF 1.5

Table 5.114: SuspendQueueEntryParams Element

NAME	DATA TYPE	DESCRIPTION
QueueFilter ?	element	This *QueueFilter* selects the *QueueEntry* elements to apply to

5.60 Track

Deprecated in JDF 1.5

The *Track* message has been deprecated in **JDF** 1.5. For details of the deprecated *Track* message, see ▸ Section N.4.8 Track.

5.61 UpdateJDF

New in JDF 1.3

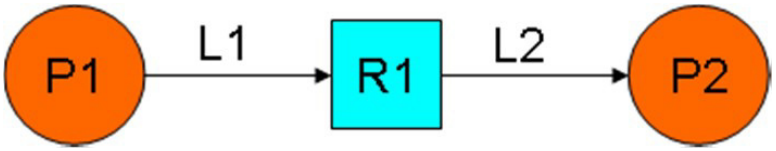
This **JMF** is used to synchronize a **JDF** Node that has been submitted by a Controller to a Device.

5.61.1 UpdateJDF Command

The *UpdateJDF Command* will be sent from a Controller (e.g., an MIS) to a Device (e.g., a Workflow System) which received the original Job. The changes SHALL be applied to Processes that have not started yet. If the MIS tries to do update a running Job, the Controller or Device MAY return an error 107.

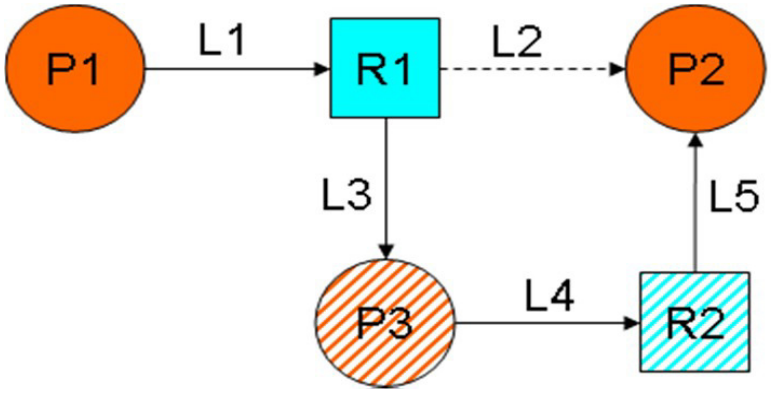
Any *JDF*/@*Type* value MAY be added to the original **JDF** with this message.

Figure 5-9: Without UpdateJDF Message



The **JDF** submitted to the Controller contains the two Processes P1 and P2. They are linked using *Resource* R1 and the *ResourceLink*Elements L1 and L2

Figure 5-10: With UpdateJDF Message



The ***Resource*** R1 is first processed by Process P3 whose Output Resource R2 is then consumed by Process P2, which has been waiting for R2 to become Available.

The ***UpdateJDF*** message contains the new Process P3, the Resource R2 and the three new ***ResourceLink*** Elements L3, L4 and L5. The ***ResourceLink*** L2 SHALL be removed from the **JDF**.

Table 5.115: UpdateJDF Command

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	***UpdateJDFCmdParams*** ?	Defines the details of the ***UpdateJDF*** message.
ResponseTypeObj	–	–

5.61.2 UpdateJDF Signal

New in JDF 1.4

The ***UpdateJDF Signal*** will be sent from the Device to a Controller. It notifies the Controller about modifications that have occurred on the Device.

Table 5.116: UpdateJDF Signal

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***UpdateJDFCmdParams*** ?	Defines the details of the ***UpdateJDF*** message.
ResponseTypeObj	–	–

5.61.2.1 UpdateJDFCmdParams

The ***UpdateJDFCmdParams*** specifies a **JDF** Node, new ***Resource*** Elements and new ***ResourceLink*** Elements to add to existing Nodes.

Table 5.117: UpdateJDFCmdParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ParentJobID	string	@*JobID* of the Node in which the new Node SHALL be inserted.
ParentJobPartID	string	@*JobPartID* of the Node in which the new Node SHALL be inserted.
CreateLink *	element	New ***ResourceLink*** Elements to be added to the previously submitted **JDF** Nodes.
CreateResource *	element	Newly created Resources to be added to previously submitted **JDF** Nodes. The Resources are used to link the new Node to existing Nodes. Resources that are linked only internally within the new Node SHOULD be in the new Node and SHOULD NOT be placed in a another ***ResourcePool*** using ***CreateResource*** Elements.
JDF	element	The new **JDF** Node to become a child of the parent Node. It is an error (204 - Cannot create Node) to specify a **JDF** with a combination of @*JobID* and @*JobPartID* that matches an existing **JDF** Node in the **JDF** ticket in which the parent Node resides.

Table 5.117: UpdateJDFCmdParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
MoveResource *	element	Specifies Resources in previously submitted **JDF** Nodes that are to be moved to another ***ResourcePool*** so that they are accessible for all new **JDF** Nodes that link to the Resources. Note: ***MoveResource*** does not create new Partitions in existing Resources.
RemoveLink *	element	***ResourceLink*** Elements in the previously submitted Job that are no longer in use and are to be removed.

5.61.2.2 CreateLink

Table 5.118: CreateLink Element

NAME	DATA TYPE	DESCRIPTION
JobID	string	@*JobID* of the Node in which the new ***ResourceLink*** is inserted.
JobPartID	string	@*JobPartID* of the Node in which the new ***ResourceLink*** is inserted.
ResourceLink +	element	The new ***ResourceLink*** Elements which link the new Node to the existing Nodes. If the Node already has a link to this Resource with a different ***Part*** Element, the Part Elements that are specified in this ***ResourceLink*** SHALL be added to the existing ***ResourceLink***.

5.61.2.3 CreateResource

Table 5.119: CreateResource Element

NAME	DATA TYPE	DESCRIPTION
JobID	string	@*JobID* of the Node in which the new ***Resource***s are to be inserted.
JobPartID	string	@*JobPartID* of the Node in which the new ***Resource***s are to be inserted.
Resource +	element	The new ***Resource*** Elements. In general, these are created to link the new Node to existing Nodes. The data type and @*Class* of ***Resource*** is derived from the Abstract Resource. See ‣ Section 3.8.3 Abstract Resource.

5.61.2.4 MoveResource

Table 5.120: MoveResource Element

NAME	DATA TYPE	DESCRIPTION
JobID	string	@*JobID* of the Node to which the new ***Resource*** SHALL be moved.
JobPartID	string	@*JobPartID* of the Node in which the new ***Resource*** SHALL be moved.
ResourceID	NMTOKEN	***Resource***/@*ID* of the ***Resource*** that is moved. Note: If the ***Resource*** has been spawned, an error MAY be reported back.

5.61.2.5 RemoveLink

Table 5.121: RemoveLink Element

NAME	DATA TYPE	DESCRIPTION
JobID	string	@*JobID* of the Node from which the ***ResourceLink*** Elements are to be removed.
JobPartID	string	@*JobPartID* of the Node from which the ***ResourceLink*** Elements are to be removed.
ResourceLink +	element	The ***ResourceLink*** Elements to be removed. Note: If this ***ResourceLink*** contains fewer ***Part*** Elements than the corresponding ***ResourceLink*** in the **JDF**, only the ***Part*** Elements specified in this ***ResourceLink*** are to be removed.

Note: This message might not work:

- if one of the Resources or Links have references to a Pipe.
- if the Controller has submitted parts of the Job to a second Controller or a Device.

The **JDF** after executing the message is valid

- on a Job which is waiting.
- if all Nodes, to which the new Node is linked are waiting.
- if the link to a running Node is not using a pipe.

Example 5.39: UpdateJDF Command

```
<Command ID="ID1" Type="UpdateJDF" xsi:type="CommandUpdateJDF">
    <UpdateJDFCmdParams ParentJobID="ID100" ParentJobPartID="ID112">
        <CreateLink JobID="ID100" JobPartID="ID111">
            <MediaLink Usage="Input" rRef="link001111"/>
        </CreateLink>
        <CreateResource JobID="100" JobPartID="110">
            <Component rRef="link001112"/>
        </CreateResource>
        <RemoveLink JobID="100" JobPartID="111">
            <MediaLink Usage="Input" rRef="link001113"/>
        </RemoveLink>
        <MoveResource JobID="100" JobPartID="101" ResourceID="link000004"/>
        <JDF JobPartID="200" Type="Cutting">
            <AuditPool>
                <Created AgentName="MIS" TimeStamp="2005-06-02T09:01:45+01:00"
                        AgentVersion="1.0"/>
            </AuditPool>
            <ResourcePool>
                <Component ID="link000002" Class="Quantity" Status="Available"
                        ComponentType="Sheet"/>
                <CuttingParams ID="link000007" Class="Parameter" Status="Available"/>
            </ResourcePool>
            <ResourceLinkPool>
                <ComponentLink Usage="Output" rRef="link000002"/>
                <CuttingParamsLink Usage="Input" rRef="link000007"/>
            </ResourceLinkPool>
        </JDF>
    </UpdateJDFCmdParams>
</Command>
```

5.62 WakeUp

New in JDF 1.2

The ***WakeUp*** Command message activates a Controller or Device that has been in stand-by mode. All queues that belong to the Device are held upon its receiving a ***WakeUp*** and SHALL be resumed with an explicit ***ResumeQueue*** message. All Jobs that were running on the Device at shutdown are also in a held state and SHALL be explicitly resumed with a ***ResumeQueueEntry*** message. A Device SHALL use the ***Status*** message if it signals its own awakening

Table 5.122: WakeUp Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	***WakeUpCmdParams*** ?	Defines the details of the ***WakeUp*** message.
ResponseTypeObj	***DeviceInfo***	Describes the Device status immediately after the ***WakeUp*** message has been sent. The Device SHOULD also send an ***Acknowledge***/***WakeUp*** message after its warm up cycle has been completed if applicable.

5.62.1 WakeUpCmdParams

WakeUpCmdParams is a placeholder for future use and for extensions to the ***WakeUp*** message.

Table 5.123: WakeUpCmdParams Element

NAME	DATA TYPE	DESCRIPTION
—	—	—

Processes

The following chapter describes the processes that are defined in detail for **JDF**.

The JDF Cookbook

Chapter 6 and following are "the list of ingredients" in the JDF "cookbook." The following Processes and Elements are fairly exhaustive. You can choose to use only what fits your workflow.

6.1 Process Template

Processes are defined by their input and output resources. All relevant resource information is provided in tables for each process. Furthermore, although they are not listed for each process, additional, OPTIONAL input resources (as defined in ▸ Table 6.1 Template for Input Resources) are valid for all processes defined in this chapter.

Note: Regarding the Templates for tables for Input Resources and Output Resources

- the *italicized* text describes the actual text that would be in its place in an actual process definition
- *Cardinality* in the Name column refers to a cardinality symbol, which is either empty or consists of a symbol, such as "?". Examples described by the Name column include: "***Media****" and "***Component*** (Proof) ?". For further details, see ▸ Section 1.3.5 Specification of Cardinality.
- The text following a "Note:" in a table field gives further information about the specified table row.
- Each of the first two rows of each table represents zero or more of what it describes. Each of the remaining rows in the Input Resource Template describes an Input Resource that is OPTIONAL for any process, even though it doesn't appear in the process's Input Resources table.

Table 6.1: Template for Input Resources (Sheet 1 of 2)

NAME	DESCRIPTION
Resource-Name Cardinality	*Information about the Input Resource.* Note: The resource represents any input resource. If an OPTIONAL resource is not specified in a **JDF** instance, the **JDF** Consumer MAY make its own assumption regarding attributes and subelements of the resource. Specification-defined attribute defaults cannot be guaranteed.
Resource-Name (someValue) *Cardinality*	*Information about the Input Resource* Note: @*ProcessUsage* attribute of the specified resource SHALL match the "someValue" value specified in the parentheses. When a process potentially contains multiple input resources of the same type, the value of @*ProcessUsage* distinguishes the resources.
ApprovalSuccess *	Any number of ***ApprovalSuccess*** Resources MAY be appended to processes in order to model proofing and verification requirements. This is implied and not specified explicitly in the tables in the following section. For more information on the **Approval** process, see ▸ Section 6.2.1 Approval.
CustomerInfo ? New in JDF 1.3	Specifies information about the customer. Prior to **JDF** 1.3 ***CustomerInfo*** was not a resource, but rather a direct child element of the **JDF** node.
Employee *	***Employee*** that is associated with processing this node.
Device *	***Device*** that is associated with processing this node.
MiscConsumable * New in JDF 1.3	Generic consumable resources that are associated with processing this node.
NodeInfo ? New in JDF 1.3	Specifies information about the node. Prior to **JDF** 1.3 ***NodeInfo*** was not a resource, but rather a direct child element of the **JDF** node.

Table 6.1: Template for Input Resources (Sheet 2 of 2)

NAME	DESCRIPTION
PreflightReport * New in JDF 1.2	Any number of ***PreflightReport*** Resources MAY be appended to processes in order to convey the results of previous preflighting steps. This is implied and not specified explicitly in the tables in the following section. For more information on the **Preflight** process, see ▸ Section 6.3.27 Preflight.
Preview * New in JDF 1.1A Deprecated in JDF 1.4	Any number of previews MAY be associated with a process and used for display purposes. ***Preview***/@*PreviewUsage* SHOULD be "ThumbNail" or "Viewable". **Deprecation note:** Starting with **JDF** 1.4, a ***Preview*** MAY be a member of any element. See ▸ Table 3.1 Any Element (generic content).
Tool * New in JDF 1.4	Miscellaneous reusable tool required for a process.
UsageCounter * New in JDF 1.3	Devices MAY use counters, called "usage counters", to track equipment utilization or work performed, such as impressions produced or documents generated.

Table 6.2: Template for Output Resources

NAME	DESCRIPTION
Resource-Name Cardinality	*Information about the Output Resource.*
Resource-Name (someValue) *Cardinality*	*Information about the Output Resource* Note: @*ProcessUsage* attribute of the specified resource SHALL match the "someValue" value specified in the parentheses. When a process potentially contains multiple output resources of the same type, the value of @*ProcessUsage* distinguishes the resources.

6.2 General Processes

6.2.1 Approval

The **Approval** process can take place at various steps in a workflow. For example, a resource (e.g., a printed sheet or a finished book) is used as the input to be approved, and an ***ApprovalSuccess*** (given, for example, by a customer or foreman) is produced. Combining the process with any other process can be used to represent a request for a receipt. The process that follows the **Approval** process in the workflow chain will most often require the ***ApprovalSuccess*** as input.

Resources typically have a @*Status* = "Draft" before the **Approval**. After a successful **Approval**, Resources have a @*Status* = "Available" and after an unsuccessful **Approval**, they have a @*Status* = "Rejected".

Table 6.3: Approval – Input Resources

NAME	DESCRIPTION
ApprovalParams	Details of the approval process.
Resource *	The resources to be approved. The input will most often be a resource of Class "Handling" or "Quantity". When the input resource of an **Approval** process is a ***ByteMap***, it is assumed that it will be displayed on a viewing device.

Table 6.4: Approval – Output Resources (Sheet 1 of 2)

NAME	DESCRIPTION
ApprovalSuccess	Result of any proofing process given, for example, by a customer or foreman. Note that ***ApprovalSuccess*** Resources are only available on success.
Resource (Accepted) *	Represents the input resources that have been accepted for further processing by the **Approval** process as output resources. This is typically used to transfer the resource @*Status* of "Draft" to "Available" (see also ▸ Section 4.3.5.2 Formal Iterative Processing).

Table 6.4: Approval – Output Resources (Sheet 2 of 2)

NAME	DESCRIPTION
Resource (Rejected) *	Represents the input resources that have been rejected for further processing by the **Approval** process as output resources. This can be used to define additional processing for rejected resources. ***Resource***/@*Status* SHOULD be set to "Rejected".

6.2.2 Buffer

New in JDF 1.1

The **Buffer** process is used to buffer a ***Resource*** for a certain time period. This can be buffering of a complete resource or of a partial ***Resource*** (e.g., in a pipe). The @*Amount* of the input and output of resources SHALL be equal. Waiting for printed material to dry before finishing is an example of the **Buffer** process.

Table 6.5: Buffer – Input Resources

NAME	DESCRIPTION
BufferParams	The parameters (e.g., times and locations of the **Buffer** process).
Resource	The ***Resource*** element to be buffered.

Table 6.6: Buffer – Output Resources

NAME	DESCRIPTION
Resource	The same ***Resource*** after buffering.

6.2.3 Combine

The **Combine** process is used to combine multiple ***PhysicalResource***s or logical resources (e.g., ***RunList*** Resources of the same content to form one resource). The sum of @*Amount* of the input and output of resources SHALL be equal. The ordering of the input ***ResourceLink*** elements SHALL be honored.

Table 6.7: Combine – Input Resources

NAME	DESCRIPTION
Resource +	The resources to be combined.

Table 6.8: Combine – Output Resources

NAME	DESCRIPTION
Resource	Result of combining. The resource formed as a result of the **Combine** process.

6.2.4 Delivery

This process can be used to describe the delivery of a ***PhysicalResource***s to or from a location. This delivery can be internal – meaning within the company – or to an external company or customer. The ***CustomerInfo*** element of the **JDF** node can also be used if the delivery to is to be made to only one customer. Note that a delivery receipt can be requested by combining the **Delivery** process with an **Approval** process.

Table 6.9: Delivery – Input Resources

NAME	DESCRIPTION
DeliveryParams	Necessary information about the physical item or items to be delivered is stored here.
Resource ? Deprecated in JDF 1.2	Any resource delivered to a location. This can be a ***PhysicalResource*** or a ***Parameter Resource*** that is delivered electronically. **Modification Note:** In **JDF** 1.2 and beyond the delivered resources are defined as refelements in elements of ***DeliveryParams***/***Drop***/***DropItem***.

Table 6.10: Delivery – Output Resources

NAME	DESCRIPTION
Resource + Modified in JDF 1.2	These SHALL be ***PhysicalResource***s.

6.2.5 ManualLabor

New in JDF 1.1

This process can be used to describe any process where resources are handled manually. The **ManualLabor** process is designed to monitor any type of non-automated labor from an MIS system.

Table 6.11: ManualLabor – Input Resources

NAME	DESCRIPTION
ManualLaborParams	Details on the **ManualLabor** process.
Resource *	Resources that are used to create the output resource.

Table 6.12: ManualLabor – Output Resources

NAME	DESCRIPTION
Resource * Modified in JDF 1.4	The resources that were created by manual work. In general these will be ***Component*** resources, but ***Handling Resource***s MAY also be processed manually. If no output resource is specified, the **ManualLabor** process describes incidental work. **Modification note:** Starting with **JDF** 1.4, multiple resources are allowed.

6.2.6 Ordering

Deprecated in JDF 1.5

See ▸ Section N.5.4 Ordering for details of this deprecated process.

6.2.7 Packing

Deprecated in JDF 1.1

See ▸ Section N.5.5 Packing for details of this deprecated process.

6.2.8 QualityControl

New in JDF 1.2

This process defines the setup and frequency of quality controls for a process. **QualityControl** is generally performed on ***Component*** Resources produced as intermediate or final output of a process.

Table 6.13: QualityControl – Input Resources

NAME	DESCRIPTION
QualityControlParams	Detailed definition of the **QualityControl** process.
Resource	The resource to be quality controlled. In general this will be a ***Component*** Resource.

Table 6.14: QualityControl – Output Resources

NAME	DESCRIPTION
QualityControlResult	Results of the process (e.g., measurement statistics).
Resource	This resource describes the resource after **QualityControl** has been applied. Note that this resource will generally be partitioned by *@Condition* to track the amount of accepted and rejected resources. This ***Resource*** SHOULD reference the ***QualityControlResult*** Output resource.

6.2.9 ResourceDefinition

This process can be used to describe the interactive or automated process of defining resources such as set-up information. This process creates output resources or modifies input resources of the same type as the output resources. The **ResourceDefinition** process is designed to monitor interactive work such as creating imposition templates. It can also be used to model a hot folder process that accepts resources from outside of a **JDF** based workflow.

Table 6.15: ResourceDefinition – Input Resources

NAME	DESCRIPTION
Resource * Modified in JDF 1.1	Any type of resource. Generally these will be templates.
ResourceDefinitionParams ?	Details on how to handle defaults.

Table 6.16: ResourceDefinition – Output Resources

NAME	DESCRIPTION
Resource + Modified in JDF 1.1	The same type of resource as one of the input resources.

6.2.10 Split

This process is used for splitting one physical or logical resource into multiple physical or logical resources containing the same content as the original. The sum of @*Amount* of the input and output of resources SHALL be equal.

Table 6.17: Split – Input Resources

NAME	DESCRIPTION
Resource	The resource to be split.

Table 6.18: Split – Output Resources

NAME	DESCRIPTION
Resource +	The resources formed as a result of splitting.

6.2.11 Verification

The **Verification** process is used to confirm that a process has been completely executed. In the case of variable data printing in which every document is unique and validated individually, database access is required. Verification in this situation can involve scanning the physical sheet and interpreting a bar code or alphanumeric characters. The decoded data can then be either recorded in a database to be later cross referenced with a verification list, or cross referenced and validated immediately in real time.

Verification differs from **QualityControl** in that **Verification** verifies the existence of a given set of resources, whereas **QualityControl** verifies that the existing resources fulfill certain quality criteria.

Table 6.19: Verification – Input Resources (Sheet 1 of 2)

NAME	DESCRIPTION
DBSchema ? Deprecated in JDF 1.5	Schema description of the cross-reference database.
DBSelection ? Deprecated in JDF 1.5	Database link that defines the database that contains cross-reference data.
FileSpec (Verification) ? New in JDF 1.5	Reference to a file that contains implementation specific descriptions of the resources to be verified.

Table 6.19: Verification – Input Resources (Sheet 2 of 2)

NAME	DESCRIPTION
IdentificationField * Deprecated in JDF 1.5	Identifies the position and type of data for an automated, OCR-based verification process. **Deprecation note:** Starting with **JDF** 1.5, use ***Component***/***IdentificationField***.
Resource ? New in JDF 1.2	The resource to be verified. The input will most often be a resource with @*Class* = "Quantity" (e.g., ***Component***) or @*Class* = "Parameter" (e.g., ***RunList***).
VerificationParams	Controls the verification requirements.

Table 6.20: Verification – Output Resources

NAME	DESCRIPTION
ApprovalSuccess ?	Signature file that defines verification success.
DBSelection ? Deprecated in JDF 1.5	Database link where the verification data SHALL be recorded.
FileSpec (Accepted) ? New in JDF 1.5	Reference to a file that contains implementation specific descriptions of the resources that were correctly verified.
FileSpec (Rejected) ? New in JDF 1.5	Reference to a file that contains implementation specific descriptions of the resources that were NOT correctly verified.
FileSpec (Unknown) ? New in JDF 1.5	Reference to a file that contains implementation specific descriptions of the resources that were scanned but NOT in the explicit or implied list of known resources.
Resource ? New in JDF 1.2	The resource after verification. Most often the ***Resource*** will not be modified by **Verification**. It has been added here to allow modeling of **Verification** in a combined processes.

6.3 Prepress Processes

This section lists all processes that are performed prior to printing. This includes processes that are performed to make digital assets press ready and the creation of physical assets such as plates or cut dies that are required for printing or converting.

6.3.1 AssetListCreation

New in JDF 1.2

The purpose of this process is to provide a listing of all assets and their dependent assets that are required in order to use the input assets. This process analyzes the input ***RunList*** to find dependent assets to provides a complete listing of files in the output ***RunList***. **AssetListCreation** does not package, encode or compress the list of files.

Table 6.21: AssetListCreation – Input Resources

NAME	DESCRIPTION
AssetListCreationParams	Parameters of the **AssetListCreation** process
RunList	List of assets used to create a listing of dependent assets.

Table 6.22: AssetListCreation – Output Resources

NAME	DESCRIPTION
RunList	A listing of all assets that the assets listed in the input ***RunList*** are dependent on including the input assets. The dependent assets are to be inserted into the output ***RunList*** as ***RunList***/***LayoutElement***/***Dependencies***/***LayoutElement***.

6.3.2 Bending

New in JDF 1.3

The **Bending** device consumes a printing plate and bends and/or punches it. In contrast to commercial printing, for newspaper printing this process is not integrated into the **ImageSetting** process. In **JDF** 1.3 and above **ImageSetting** does not imply **Bending**. An inline plate puncher SHOULD be modelled as a combined process consisting of **ImageSetting** and **Bending** processes.

Table 6.23: Bending – Input Resources

NAME	DESCRIPTION
BendingParams	List of assets used to create a listing of dependent assets.
ExposedMedia ?	The ***ExposedMedia*** resource to be bent/punched.
Media ?	In a newspaper environment, Dummy forms might be needed. In this case, a ***Media*** with *@MediaType* = "Plate" serves as an input resource.

Table 6.24: Bending – Output Resources

NAME	DESCRIPTION
ExposedMedia	The bent/punched ***ExposedMedia*** resource.

6.3.3 ColorCorrection

ColorCorrection is the process of modifying the specification of colors in documents to achieve some desired visual result. The process might be performed to ensure consistent colors across multiple files of a job or to achieve a specific design intent (e.g., "brighten the image up a little").

ColorCorrection is distinct from **ColorSpaceConversion**, which is the process of changing how the colors specified in the job will be produced on paper. Rather, **ColorCorrection** is the process of modifying the desired result, whatever the specified color space might be.

The **ColorCorrection** process MAY be part of a combined process with the **ColorSpaceConversion** process, in which case the source and destination profiles used by the **ColorSpaceConversion** process would be supplied from ***ColorSpaceConversionParams***. Either the direct *@Adjustment* attribute or the ICC profile attribute ***ColorCorrectionOp***/***FileSpec*** with *@ResourceUsage* = "AbstractProfile" can be used in this scenario to apply color corrections in the device independent ICC Profile Connection Space interpreted from the ICC source profile before the ICC destination profile is applied.

Alternatively, a **ColorCorrection** process MAY occur after a **ColorSpaceConversion** process. In this scenario only the ***ColorCorrectionOp***/***FileSpec*** with *@ResourceUsage*="DeviceLinkProfile" supplied in ***ColorCorrectionOp*** is used.

Table 6.25: ColorCorrection – Input Resources

NAME	DESCRIPTION
ColorantControl ? Modified in JDF 1.1A	Identifies the assumed color model for the job.
ColorCorrectionParams New in JDF 1.1	Parameters of the **ColorCorrection** process
RunList	List of content elements that are to be operated on.

Table 6.26: ColorCorrection – Output Resources

NAME	DESCRIPTION
RunList	List of color-corrected content elements.

6.3.4 ColorSpaceConversion

ColorSpaceConversion is the process of converting colors that are provided in a PDL to an output color space. There are two ways in which a controller can use this process to accomplish the color conversion. It can simply order the colors to

be converted by the device assigned to the task, or it can request that the process simply tag the input data for eventual conversion. Additionally, the process can remove all tags from the PDL.

Like all other color manipulation supported in **JDF** , the color conversion controls are based on the use of ICC profiles. While the assumed characterization of input data can take many forms, each can internally be represented as an ICC profile. In order to perform the transformations, input profiles SHALL be paired with the identified final target device profile to create the transformation.

The target profile for color space conversion selection should be based on ***ColorSpaceConversionParams***/@*ICCProfileUsage* in the following order of precedence.

1 UsePDL – If present, the embedded target profile SHALL be used.

2 UseSupplied – The embedded target profile SHALL NOT be used.

In order to avoid the loss of black color fidelity resulting from the transformation from a four-component CMYK to a three-component interchange space, the agent MAY provide a ***DeviceLink***[1] transform as the transform that SHALL applied when converting from a specific source colorspace to the final target device colorspace specified for the **ColorSpaceConversion** operation being applied. In these instances, the final target profile SHALL NOT be specified.

Table 6.27: ColorSpaceConversion – Input Resources

NAME	DESCRIPTION
ColorantControl ? Modified in JDF 1.1A	Identifies the assumed color model for the job.
ColorSpaceConversionParams	Parameters that define how color spaces will be converted in the file.
RunList	List of pages, sheets or byte maps on which to perform the selected operation.

Table 6.28: ColorSpaceConversion – Output Resources

NAME	DESCRIPTION
ColorantControl ?	Identifies the assumed color model for the job. The ***ColorantControl*** resource MAY be modified by a **ColorSpaceConversion** process.
RunList	List of pages, sheets or byte maps on which the selected operation has been performed.

6.3.5 ContactCopying

New in JDF 1.1

ContactCopying is the process of making an analog copy of a film onto a another film or plate. It includes **FilmToPlateCopying** as defined in **JDF** 1.0 and deprecated in **JDF** 1.1.

Table 6.29: ContactCopying – Input Resources

NAME	DESCRIPTION
ContactCopyParams	The settings of the contact copying task.
DevelopingParams ?	Controls the physical and chemical specifics of the media development process.
ExposedMedia +	The film or films to be copied onto the film or plate.
Media ?	The unexposed film or plate.
TransferCurvePool ?	Area coverage correction and coordinate transformations of the device.

1. A ***DeviceLink*** transform is a transform that is defined in an ICC profile file [ICC.1] that maps directly from one specific source color space to a specific destination device color space. An example of this is a transform that maps directly from PDL source objects defined using sRGB directly to SWOP CMYK

Table 6.30: ContactCopying – Output Resources

NAME	DESCRIPTION
ExposedMedia	The resulting exposed contact copy.

6.3.6 ContoneCalibration

This process specifies the process of contone calibration. It consumes contone raster data such as that output from a **Rendering** process. It produces contone raster data which has been calibrated to a press the information about the intended screening to correctly calibrate the contone data.

Table 6.31: ContoneCalibration – Input Resources

NAME	DESCRIPTION
RunList	Ordered list of rasterized byte maps representing pages or surfaces.
ScreeningParams ? Modified in JDF 1.1	Metadata specifying which halftoning mechanism it is intended to be applied in a subsequent **Screening** process.
TransferFunctionControl ? Modified in JDF 1.1	Specifies which calibration to apply.

Table 6.32: ContoneCalibration – Output Resources

NAME	DESCRIPTION
RunList	Ordered list of rasterized byte maps representing pages or surfaces.

6.3.7 CylinderLayoutPreparation

New in JDF 1.3

CylinderLayoutPreparation specifies where to mount a single form in a newspaper-Web Press. This information might be needed by printers as human-readable text on the surface of the form. Usually, the information is shown in the non-printable area of it.

The REQUIRED color information for each plate layout is addressed from *Layout*/*ContentObject*/@*Ord*. The attribute points to *RunList* (Document). *RunList*/@*PageListIndex* points to detailed *PageData*, including individual color information.

Table 6.33: CylinderLayoutPreparation – Input Resources

NAME	DESCRIPTION
CylinderLayoutPreparationParams ?	Set of parameters for **CylinderLayoutPreparation**.
Layout	Definition of the *Layout* of the individual plates. The resulting *CylinderLayout* references plate layouts.
RunList	The document *RunList*.

Table 6.34: CylinderLayoutPreparation – Output Resources

NAME	DESCRIPTION
CylinderLayout	*CylinderLayout* specifies where to mount a single form in a newspaper-Web Press. If requested by the printer, this information can be indicated as human-readable text on the surface of the physical plate.

6.3.8 DBDocTemplateLayout

Deprecated in JDF 1.5

See ▸ Section N.5.1 DBDocTemplateLayout for details of this deprecated process.
Deprecation note: Starting with **JDF** 1.5, use **LayoutElementProduction** instead.

6.3.9 DBTemplateMerging

Deprecated in JDF 1.5

See ▸ Section N.5.2 DBTemplateMerging for details of this deprecated process.
Deprecation note: Starting with **JDF** 1.5, use **LayoutElementProduction** instead.

6.3.10 DieDesign

New in JDF 1.4

This process describes the design of a die tool set with one or more stations starting from a ***DieLayout*** that describes the layout of the one-up designs on a die. The output of this process is a ***DieLayout*** resource, describing a tool set for the die cutter machine that can be used in a following **DieMaking** process. **DieDesign** typically follows **DieLayoutProduction**.

Table 6.35: DieDesign – Input Resources

NAME	DESCRIPTION
DieLayout	A resource describing the die cutter layout.

Table 6.36: DieDesign – Output Resources

NAME	DESCRIPTION
DieLayout +	A set of resources describing the die cutter tool set.

6.3.11 DieLayoutProduction

New in JDF 1.4

This process describes the layout of one or more structural designs for a given ***Media***. The output of this process is a ***DieLayout*** resource describing the positioning of the individual one-ups on the die. The **DieLayoutProduction** process can be performed by a human operator using a CAD application. In some cases it can be an automated process. The process can be run in estimation mode in which case multiple solutions are returned that can then be used as input of a cost estimation module to determine the optimal layout.The **DieLayoutProduction** process is the packaging equivalent of a **Stripping** process in conventional printing. The output ***DieLayout*** of **DieLayoutProduction** is typically the input of a subsequent **DieDesign** process.

Table 6.37: DieLayoutProduction – Input Resources

NAME	DESCRIPTION
DieLayoutProductionParams	The parameters for the **DieLayoutProduction**.
ShapeDef +	***ShapeDef*** resources describing the different 1-up structural designs to be stepped and repeated on the ***Media***.

Table 6.38: DieLayoutProduction – Output Resources

NAME	DESCRIPTION
DieLayout +	A resource describing the die cutter tool set. When the process is run in estimation mode, multiple alternative ***DieLayout*** elements are returned, otherwise a single ***DieLayout*** is generated.

Example 6.1: DieLayoutProduction: Single Shape and Two Sheet Sizes

Example of **DieLayoutProduction** of a single shape on 2 stock sheet sizes

```
<!-- DieLayoutProduction Sample
       Date:Sept 2007 Version: 1.00
       Single Shape is repeated on a range of alternative sheet sizes.
-->
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n001"
       Type="DieLayoutProduction" Status="Waiting" JobPartID="ID234"
       DescriptiveName="Single shape versus a set of sheet sizes"
       Version="1.4">
    <ResourcePool>
       <ShapeDef Class="Parameter" ID="Shape1Up" Status="Available">
          <FileSpec URL="file://myserver/myshare/olive.dd3"/>
       </ShapeDef>
       <!-- Layout can chose from 2 stock sheet sizes. Nesting with 2nd row
              rotated and secondary gutters. Rotate against grain/flute
              is not allowed.
       -->
       <DieLayoutProductionParams Class="Parameter" ID="LayParam"
              Status="Available">
          <ConvertingConfig SheetWidth="2834.64 ~ 2834.64"
                 SheetHeight="2267.72 ~ 2267.72"/>
          <ConvertingConfig SheetWidth="3401.57 ~ 3401.57"
                 SheetHeight="2834.64 ~ 2834.64"/>
          <RepeatDesc GutterY="0.0" GutterY2="14.17" AllowedRotate="None"
                 LayoutStyle="Reverse2ndRow"/>
       </DieLayoutProductionParams>
       <!-- The layout with minimum waste will be returned as the final result. -->
       <DieLayout Class="Parameter" ID="DieLay" Status="Unavailable"/>
    </ResourcePool>
    <ResourceLinkPool>
       <ShapeDefLink rRef="Shape1Up" Usage="Input"/>
       <DieLayoutProductionParamsLink rRef="LayParam" Usage="Input"/>
       <DieLayoutLink rRef="DieLay" Usage="Output"/>
    </ResourceLinkPool>
</JDF>
```

Example 6.2: DieLayoutProduction: Single Shape and Range of Sheet Sizes

Example of **DieLayoutProduction** of a single shape on a range of sheet sizes. The sheet sizes have defined minimum and maximum width and height. The layout is optimized for a particular order quantity.

```
<!-- DieLayoutProduction Sample
       Date:Sept 2007 Version: 1.00
       Single Shape is repeated on a continuous range of sheet sizes. -->
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n001"
       Type="DieLayoutProduction" Status="Waiting"
       DescriptiveName="Single shape versus a set of sheet sizes"
       JobPartID="ID400" Version="1.4">
    <ResourcePool>
       <ShapeDef Class="Parameter" ID="Shape1Up" Status="Available">
          <FileSpec URL="file://myserver/myshare/olive.dd3"/>
       </ShapeDef>
       <!-- Layout can choose sheet sizes between 1200mm-1000mm wide and
          1000mm-800mm high. The layout will be optimized for order quantities
          of 1 million boxes. Gutters are 5mm and cross flute/grain rotation
          is not allowed.
       -->
       <DieLayoutProductionParams Class="Parameter" ID="LayParam"
             Status="Available">
          <ConvertingConfig SheetWidth="3401.57 ~ 2834.64"
                SheetHeight="2834.64 ~ 2267.72"/>
          <RepeatDesc OrderQuantity="1000000" GutterX="14.17" GutterY="14.17"
                AllowedRotate="None"/>
       </DieLayoutProductionParams>
       <!-- The layout with minimum waste will be returned as the
             final result. -->
       <DieLayout Class="Parameter" ID="DieLay" Status="Unavailable"/>
    </ResourcePool>
    <ResourceLinkPool>
       <ShapeDefLink rRef="Shape1Up" Usage="Input"/>
       <DieLayoutProductionParamsLink rRef="LayParam" Usage="Input"/>
       <DieLayoutLink rRef="DieLay" Usage="Output"/>
    </ResourceLinkPool>
</JDF>
```

Example 6.3: DieLayoutProduction: Two Shapes and Range of Sheet Sizes

Example of **DieLayoutProduction** of 2 shapes on a range of sheet sizes. The sheet sizes have defined minimum and maximum width and height. The layout is optimized for a particular order quantity of 2 boxes.

```
<!-- DieLayoutProduction Sample
       Date:Sept 2007 Version: 1.00
       2 Shapes is repeated on a continuous range of sheet sizes.
-->
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n001"
       Type="DieLayoutProduction"
       Status="Waiting"
       DescriptiveName="Single shape versus a set of sheet sizes"
       Version="1.4">
    <ResourcePool>
       <ShapeDef Class="Parameter" ID="Shape1Up" Status="Available">
          <FileSpec URL="file://myserver/myshare/beef.dd3"/>
       </ShapeDef>
       <ShapeDef Class="Parameter" ID="Shape1Up2" Status="Available">
          <FileSpec URL="file://myserver/myshare/chicken.dd3"/>
       </ShapeDef>
       <!-- Layout can chose sheetsizes between 1200mm-1000mm wide and
              1000mm-800mm high. Layout is optimized for an order
              quantity of 300k boxes for beef and 700k boxes for chicken.
              Gutters are 5mm and cross flute/grain rotation is not allowed.
       -->
       <DieLayoutProductionParams Class="Parameter" ID="LayParam"
              Status="Available">
          <ConvertingConfig SheetWidth="3401.57 ~ 2834.64"
                 SheetHeight="2834.64 ~ 2267.72"/>
          <RepeatDesc OrderQuantity="300000" GutterX="14.17" GutterY="14.17"
                 AllowedRotate="None"/>
          <RepeatDesc OrderQuantity="700000" GutterX="14.17" GutterY="14.17"
                 AllowedRotate="None"/>
       </DieLayoutProductionParams>
       <!-- The layout with minimum waste will be returned as the final
              result. -->
       <DieLayout Class="Parameter" ID="DieLay" Status="Unavailable"/>
    </ResourcePool>
    <ResourceLinkPool>
       <ShapeDefLink rRef="Shape1Up" Usage="Input"/>
       <ShapeDefLink rRef="Shape1Up2" Usage="Input"/>
       <DieLayoutProductionParamsLink rRef="LayParam" Usage="Input"/>
       <DieLayoutLink rRef="DieLay" Usage="Output"/>
    </ResourceLinkPool>
</JDF>
```

6.3.12 DigitalDelivery

New in JDF 1.2

This process specifies the delivery of digital assets in any stage of the flow. It could be images, documents, layout, text files, ready to print raster files or any other file type. When ***ArtDeliveryIntent***/***ArtDelivery***/@*ArtDeliveryType* is "DigitalNetwork" or "DigitalFile" the corresponding process will be **DigitalDelivery** unless ***ArtDeliveryIntent***/@*Method* = "local".

It is not necessary to use the **DigitalDelivery** process to describe informal delivery of files during the workflow although **DigitalDelivery** can be used for asset collection purposes (i.e., defining how an input ***RunList*** will be collected in the output ***RunList*** describing the packing containers of compression or encoding). See example in ▸ Appendix O Examples.

Table 6.39: DigitalDelivery – Input Resources

NAME	DESCRIPTION
DigitalDeliveryParams	Parameter specifying the artwork files delivery characteristics.
RunList * Modified in JDF 1.3	The list of digital files to be delivered.

Table 6.40: DigitalDelivery – Output Resources

NAME	DESCRIPTION
RunList + Modified in JDF 1.3	The list of digital files which were actually delivered to the destination.

6.3.13 FilmToPlateCopying

Deprecated in JDF 1.1

FilmToPlateCopying has been replaced by the more generic **ContactCopying**. See ▸ Section N.5.6 FilmToPlateCopying for details of this deprecated process.

6.3.14 FormatConversion

New in JDF 1.1

Deprecated in JDF 1.5

Deprecation note: Starting with **JDF** 1.5, use a Combined process of **RasterReading** and **Rendering**.

For details see ▸ Section N.5.3 FormatConversion.

6.3.15 ImageEnhancement

New in JDF 1.5

The **ImageEnhancement** process describes generic image data processing.

Note: The source MAY be any image, but also text or vector graphics.

Table 6.41: ImageEnhancement – Input Resources

NAME	DESCRIPTION
ImageEnhancementParams	Describes the controls selected for the manipulation of images.
RunList	List of content data elements on which to perform the selected operations.

Table 6.42: ImageEnhancement – Output Resources

NAME	DESCRIPTION
RunList	List of page contents with images that have been manipulated as indicated by the *ImageEnhancementParams* resource.

6.3.16 ImageReplacement

This process provides a mechanism for manipulating documents that contain referenced image data. It allows for the "fattening" of files that simply contain a reference to external data or contain a low resolution proxy. Additionally, the resource can be specified so that this process generates proxy images from referenced data. **ImageReplacement** is intentionally neutral of the conventions used to identify the externally referenced image data.

Table 6.43: ImageReplacement – Input Resources

NAME	DESCRIPTION
ImageCompressionParams ? New in JDF 1.1	This resource provides a set of controls that determines how images will be compressed in the resulting "fat" PDL pages.
ImageReplacementParams ?	Describes the controls selected for the manipulation of images.
RunList ?	List of page contents on which to perform the selected operation.

Table 6.44: ImageReplacement – Output Resources

NAME	DESCRIPTION
RunList	List of page contents with images that have been manipulated as indicated by the ***ImageReplacementParams*** resource.

6.3.17 ImageSetting

The **ImageSetting** process is executed by an imagesetter or platesetter that images a bitmap onto the film or plate media. The **ImageSetting** process can also be used to describe hard copy proofing (see ▸ Section 6.2.1 Approval).

Table 6.45: ImageSetting – Input Resources

NAME	DESCRIPTION
ColorantControl ? New in JDF 1.2	The ***ColorantControl*** resources that define the ordering and usage of inks during marking on the imagesetter.
DevelopingParams ? New in JDF 1.1	Controls the physical and chemical specifics of the media development process.
ExposedMedia ? New in JDF 1.3	When imaging to reusable media, ***ExposedMedia*** MAY also be used as input to **ImageSetting**. **Constraint:** exactly one of ***Media*** or ***ExposedMedia*** SHALL be specified.
ImageSetterParams ? Modified in JDF 1.1	Controls the device specific features of the imagesetter.
Media ?	The unexposed media. **Constraint:** exactly one of ***Media*** or ***ExposedMedia*** SHALL be specified.
RunList ?	Identifies the set of bitmaps to image. MAY contain bytemaps or images.
TransferCurvePool ? New in JDF 1.1	Area coverage correction and coordinate transformations of the device.

Table 6.46: ImageSetting – Output Resources

NAME	DESCRIPTION
ExposedMedia	The exposed media resource. In case of film setting, this is the exposed set of films.

6.3.18 Imposition

Modified in JDF 1.4

Modification note: Starting with **JDF** 1.4, automated imposition is added.

The **Imposition** process is responsible for combining pages of input graphical content onto surfaces whose dimensions are reflective of the physical output media. Static or dynamic printer's marks can be added to the surface in order to facilitate various aspects of the production process. Among other things, these marks are used for press alignment, color calibration, job identification and as guides for cutting and folding.

Note that the **Imposition** process specifies the task of combining pages and marks on sheets. The task of setting up the parameters needed for **Imposition** (e.g., creating the ***Layout*** resource) is defined either by **LayoutPreparation**, **Stripping** or by the generic **ResourceDefinition** process.

Table 6.47: Imposition – Input Resources (Sheet 1 of 2)

NAME	DESCRIPTION
Layout	A ***Layout*** resource that indicates how the content pages from the Document ***RunList*** and marks from the Marks ***RunList*** (see below) are combined onto imposed surfaces.
RunList (Document)	Structured list of incoming page contents which is transformed to produce the imposed surface images.

Table 6.47: Imposition – Input Resources (Sheet 2 of 2)

NAME	DESCRIPTION
RunList (Marks) ?	Structured list of incoming marks. These are typically printer's marks such as fold marks, cut marks, punch marks or color bars.

Table 6.48: Imposition – Output Resources

NAME	DESCRIPTION
RunList	Structured list of imposed surfaces. The @*ElementType* of the ***LayoutElement*** resource SHALL be "Surface". Typically the output ***RunList*** will be partitioned by @*PartIDKeys* = "SheetName Side Separation". If the **Imposition** process is executed before **RIPing**, this will generally be consumed by an **Interpreting** process. In the case of where **Imposition** is executed after **RIPing**, it will be consumed by **DigitalPrinting** or **ImageSetting**.

There are two mechanisms provided for controlling the flow of page images onto sheet surfaces:

The default mechanism is for non-automated (e.g., fully-specified) **Imposition**, which originally derived from ***Layout*** in PJTF. Fully-specified imposition explicitly identifies all page content for each sheet imaged and references these pages by means of the order in which they are defined in the input ***RunList*** (Document) resource. Static printer's marks are referenced in a similar fashion from the input ***RunList*** (Marks) resource.

Setting the @*Automated* attribute of the ***Layout*** resource to "true" activates a template approach to imposition and relies upon the full hierarchy structure of the document (as specified by the ***RunList*** (Document) and referenced ▸ Structured PDL data) to specify the page content to be imposed.

In **JDF** , there is a single ***Layout*** resource definition. Its structure is broad enough to encompass the needs of both fully specified and template-driven imposition. When described fully (@*Automated* = "false"), the ***Layout*** resource partition structure defines the imposition to take place. The highest level of each partition defines a signature. The children of each of the signatures in turn specifies an array of sheets, and each sheet MAY have up to two surfaces (Front and/or Back), on which the page images and any printer's marks are to be placed using ***PlacedObject*** elements. A sheet that specifies no surface content SHALL be interpreted as blank. Pages that are to be printed SHALL be placed onto surfaces using ***ContentObject*** subelements which explicitly identify the page (Typically done using the ***ContentObject***/@*Ord* attribute which specifies an index into the document ***RunList***). Thus, the ***Layout*** partition hierarchy SHALL explicitly specify which pages are to be imaged onto each surface.

For **JDF** 1.3, automated imposition was originally defined such that ***Layout*** resource partitions specified a single signature of sheet(s) upon which page content was to be imposed. The sequence of pages to be imaged via automated imposition was defined by the document ***RunList***. The pages were pulled from this sequence as needed in order to satisfy the ***ContentObject*** elements defined for each sheet surface in the signature of the ***Layout*** resource. The signature was repeated as necessary until all pages available in the document ***RunList*** had been used.
Note: The XML order in which the partitions of the ***Layout*** resource are defined is significant for both automated and non-automated imposition and defines the order in which the imposition engine processes the ***RunList*** (Document).

6.3.18.1 Glossary for Automated Imposition

This table below introduces terms and concepts necessary for understanding automated imposition processing.

Table 6.49: Glossary for Automated Imposition (Sheet 1 of 4)

TERM	DEFINITION
Base Index	When processing an ▸ Imposition Template, the imposition engine maintains an internal **Base Index** into the ▸ Page Pool being processed. That **Base Index** is added to the ***ContentObject***/@*Ord* value, resulting in an index into the ▸ Page Pool for referencing the page to be placed, and is updated for each ▸ Imposition Template iteration. Both positive and negative base indices are maintained for use when ***ContentObject***/@*Ord* has either a negative or positive value. For an example, see ▸ Example 1329 Algorithm for Processing an Imposition Template.
Base Ord	Same as ▸ Base Index.
Collect	Set of sheets that are collected together prior to gathering.

Table 6.49: Glossary for Automated Imposition (Sheet 2 of 4)

TERM	DEFINITION
Document Major Processing Order	**Document Major Processing Order** refers to the scenario wherein all instances of a given document class (across all sets to be processed) SHALL be produced before starting processing for the next document class. For instance, the production requirements may state that all brochures SHALL be produced for each set, followed by all cover letters and then all postcards. This processing order is an example of Document Major.
Imposed Sheet Set	Describes a single set of sheet definitions generated by the imposition engine containing imposed content. Note that this may represent a precut set of sheets in a cut-and-stack workflow (where the maximum number of sheets in the ▸ Imposed Sheet Set is defined by ***Layout***/***LogicalStackParams***/@*MaxStackDepth*), or a collect when no ▸ Logical Stacks are defined.
Imposition Template	A first-level branch of a partitioned ***Layout*** resource having @*Automated* = "true" that describes a single set of sheets with a common imposition layout that accommodates very specific production characteristics. A single ***Layout*** resource defines a collection of one or more **Imposition Templates**.
Instance Document	The imposition engine treats each immediate child node of a set in a ▸ Structured PDL as an **Instance Document**. This is used as the basis for generating @*EndOfDocument* breaks in the resulting ***RunList*** (Surface), and for processing ***RunList***/@*DocCopies* attributes (see ▸ Section 8.131 RunList). If a set has only pages as its children, then a single ▸ Instance Document is assumed to exist.
Logical Sheet	One or more pages placed onto a sheet definition within a ▸ Logical Stack (i.e., a sheet definition within a ▸ Logical Stack).
Logical Stack	When ***Layout***/***LogicalStackParams***/@*MaxStackDepth* is specified in the root of the ***Layout*** resource, then the imposition engine is configured for imposition onto multiple **Logical Stacks**. These stacks are described through the use of adding ***Layout***/***PlacedObject***/@*LogicalStackOrd* to stack-specific descriptions for each placed object. For more information, see ▸ Section 6.3.18.4.1 Using Logical Stacks. For example usage, see ▸ Example 1335 Booklet Using Automated Imposition.
Logical Stack Set	The set of ▸ Logical Stacks described by an ▸ Imposed Sheet Set.
Page Pool	A **Page Pool** refers to a delimited sequence of pages defined within the ***RunList*** (Document) input to the **Imposition** process. A **Page Pool** MAY encompass all pages of the ***RunList*** (Document) as in the case of ▸ Unstructured PDLs. In the case of ▸ Structured PDLs, a **Page Pool** is defined to be that set of pages represented by a leaf node of the document structure. For example, a brochure which has a sub-structure of cover and body has two leaf nodes, cover and body, respectively. If body were further divided into chapter sections, then the leaf nodes of the brochure would be the cover and each body chapter. ***LayoutElement***/@*ElementType* may be used to demote an already ▸ Structured PDL to be treated as an ▸ Unstructured PDL. Examples of ▸ Structured PDL formats include PPML, PPML/VDX, and ISO 16612-2 PDF/VT. ▸ Imposition Templates select **Page Pools** to be processed based on their partition keys whose values are derived from metadata present in the PDL data (e.g., ***Layout*** partitioned by @*DocTags* = "Letter" would process all **Page Pools** of the current set whose metadata derived partition key @*DocTags* matches "Letter"). See below for more detail. It is important to note that the pages in a **Page Pool** SHALL be presented to the imposition engine in a well defined order known to the ***Layout*** resource creator (typically reader order) in order for them to be processed correctly.

Table 6.49: Glossary for Automated Imposition (Sheet 3 of 4)

TERM	DEFINITION
Page Pool List	A **Page Pool List** refers to a sequence of one or more ▸ Page Pools (contiguous or disjoint in the ***RunList*** (Document)) aggregated together and treated as a single ▸ Page Pool for processing by a selected ▸ Imposition Template. For example, if a **Page Pool List** is constructed from the ▸ Page Pools: Chapter1, Chapter2, and Chapter4 as defined in an input ***RunList*** (Document), then the aggregate result is a single pool of pages consisting of the pages from Chapter1, Chapter2 and Chapter4. The order of the pages of the **Page Pool List** SHALL be processed in the order in which the ▸ Page Pools are defined in the ***RunList*** (Document). The boundaries between ▸ Page Pools in a **Page Pool List** are implicitly maintained for use by the imposition processor for making page level sheet surface mapping decisions during processing (e.g., specifying a right side facing pages start at the beginning of each chapter). ▸ Page Pools are aggregated into **Page Pool Lists** through the use of the ***Layout***/@*BaseOrdReset* attribute. If @*BaseOrdReset* = "PagePoolList" then all ▸ Page Pools processed by the ▸ Imposition Template are aggregated. If @*BaseOrdReset* = "PagePool", then each ▸ Page Pool is processed separately. It is important to note that the pages in a **Page Pool List** SHALL be presented to the imposition engine in a well defined order known to the ***Layout*** resource creator (typically reader order) in order for them to be processed correctly.
PDL Metadata	Various PDL formats provide for the definition of key/value pairs within the PDL that MAY be treated as metadata for the purpose of process parameterization. For example, the metadata key/value pairs specified in the PDL data may identify the type of finished document using @*DocumentType* = "PostCard" or "Booklet", which would then affect the selection of the ▸ Imposition Template to be applied. The **Imposition** process makes use of metadata to make decisions as to which ▸ Page Pools should be processed through an ▸ Imposition Template. These decisions are performed by comparing the explicit partition key settings for each ▸ Imposition Template to the partition key/value settings mapped from the PDL for each ▸ Page Pool in the current set, and each matching ▸ Page Pool is processed through the corresponding ▸ Imposition Template(s). Within an ▸ Imposition Template, metadata associated with individual pages MAY also be used to parameterize dynamic mark and slug-line content generation (see example below). Refer to the ***RunList***/***MetadataMap*** element definition for information on how to specify the mapping of PDL specified metadata values for use by **JDF** (e.g., using partition keys or ***GeneralID*** keys). The ▸ PDL Processor SHALL make use of the ***RunList***/***MetadataMap*** to generate partition keys, ***GeneralID*** and other values during the course of imposition processing. These values SHALL be regenerated as necessary, as the metadata key/value pairs in the PDL change based on which portion of the PDL is being processed.
PDL Processor	A PDL interface that hides details of a particular PDL and syntax, etc. from the imposition engine itself. Its role is to present the structure of the PDL and pools of pages within the PDL structure to the imposition engine in a PDL independent way.
Recipient Set	Set of finished pages produced for a single recipient.
Set Major Processing Order	**Set Major Processing Order** refers to the scenario when all documents of a set instance are produced before starting on the next set instance; this is the typical processing order for most VDP applications.
Sheet Definition	A branch of an ▸ Imposition Template that describes the imposition to be performed for a sheet. ▸ Sheet Definitions for automated imposition SHALL be partitioned by @*SheetName* and @*Side*.
Structured PDL	A **Structured PDL** defines sequences of groupings of pages. These groupings may be as simple as specifying the set of pages belonging to a chapter or cover of a booklet where such a group is a ▸ Page Pool. In the case of Variable Document Printing (VDP) ▸ Structured PDLs, there are often multiple sets of content where typically a set instance comprises the content to be delivered to a single recipient. Each set has one or more documents, and documents may be further subdivided into subdocuments in hierarchical fashion. The imposition engine processes each set individually in the sequence specified in the interpretation specified by the ***RunList*** that references the ▸ Structured PDL data file. The general structure of a **Structured PDL** is identified by the PDL (PDL specification or PDL instance) itself or the value of the ***LayoutElement***/@*ElementType* attribute.

Table 6.49: Glossary for Automated Imposition (Sheet 4 of 4)

TERM	DEFINITION
Structured PDL – MultiDocument	For MultiDocument PDL files, the PDL processor supplies the context to the imposition processor that represents the PDL's document structure. This context is defined as **Set** – represents a single set containing all of the documents in the PDL file, therefore the value of @*SetIndex* SHALL always be 0. **Document** – is always the first hierarchical level in the file. **SubDoc0** to **SubDoc9** – represent consecutive levels of the hierarchy below the **Document** level in the file not including the level representing individual pages. If any level of the hierarchy is not defined, the value of the corresponding @*SubDocIndexn* is undefined. **Pages** – represent individual pages in the PDL.
Structured PDL – MultiSet	For MultiSet PDL files, the PDL processor supplies the context to the imposition processor that represents the PDL's set and document structure. This context is defined as **Set** – represents a set of related documents. **Document** – is always the first hierarchical level below the **Set** level. If a MultiSet file contains only Sets with no document or sub-document breaks (no levels are defined below the Set level), all of the pages of the set are considered to be included in a single document therefore the @*DocIndex* is always 0. **SubDoc0** to **SubDoc9** – represent consecutive levels of the hierarchy below the **Document** level in the file not including the level representing individual pages. If any level of the hierarchy is not defined, the value of the corresponding @*SubDocIndexn* is undefined. **Pages** – represent individual pages in the PDL. **Note:** The lowest level of the **JDF** hierarchy (**Set**, **Document**, **SubDoc***n*) mapped by the PDL processor represents a ▸ Page Pool context.
Unstructured PDL	An **Unstructured PDL** is a content file consisting of a single set of one or more pages. Typically such a PDL file is considered to be a single document and a single ***Layout*** ▸ Imposition Template would be applied to the entire set of pages. When a **JDF** imposes structure on such a file either using direct @*Page* indices or a partitioned ***RunList*** pointing to different page ranges of the file using @*EndOfSet*, @*EndOfDocument* attributes, then the imposition engine will treat the input ***RunList*** resource as a ▸ Structured PDL.

6.3.18.2 Variables for Automated Imposition

The imposition engine maintains a set of locally scoped variables that may be referenced during imposition processing. The values of these variables reflect the current context of processing during execution of the Imposition process. These variables include those described in ▸ Section Dynamic String Generation, as well as those described in bulleted items below. All variables below are integer variables.

Table 6.50: Variables for Automated Imposition (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
CollectIndex ?	integer	Represents a zero based index of the current collect of sheets being generated by an automated ▸ Imposition Template from the current ▸ Page Pool or ▸ Page Pool List being processed. May be greater than zero if ***Layout***/@*MaxCollect* is specified and is greater than 1.
CollectSheetIndex ?	integer	Is a zero-based index of the current physical or ▸ Logical Sheet of the current collect generated by an automated ▸ Imposition Template from the current ▸ Page Pool or ▸ Page Pool List being processed. ▸ Logical Sheets are used when ▸ Logical Stacks are defined.
ImposedSheetSetIndex ?	integer	Is the 0-based ▸ Imposed Sheet Set index.
PoolSheetIndex ?	integer	Is a zero-based index of the current physical or logical sheet generated from the current ▸ Page Pool or ▸ Page Pool List within an automated ▸ Imposition Template. ▸ Logical Sheets are used when ▸ Logical Stacks are defined. The value of this variable is independent of the number of collects generated by the same automated ▸ Imposition Template.

Table 6.50: Variables for Automated Imposition (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
SheetCount ?	integer	Is the current number of physical or logical sheets generated during the processing of the automated ***Layout*** resource. ▸ Logical Sheets are used when ▸ Logical Stacks are defined. At the beginning of processing of the ***Layout*** resource, the value of this variable is set to zero. The value of this variable may be reset to zero in later ***Layout*** partitions using the ***Layout***/@*SheetCountReset* attribute. @*SheetCount* is always reset to zero at the beginning of processing of a set regardless of the value of ***Layout***/@*SheetCountReset*.
*SubDocIndex**n*** ?	integer	Where *n* represents any hierarchical structure levels below the level of the current document present in the ▸ Structured PDL data to be processed. For example, @*SubDocIndex0* might represent a collection of chapters in a brochure where its containing parent is at the document level (@*DocIndex* is used to indicate the position (index) of the document in its containing Set).
TotalCollects ?	integer	Is the total number of collects generated by an automated ▸ Imposition Template from the current ▸ Page Pool or ▸ Page Pool List being processed.
TotalImposedSheetSets ?	integer	Is the total number of ▸ Imposed Sheet Sets defined for the job.
TotalSets ?	integer	Is the total number of recipient sets generated for the job. Note that in cases where it is used before the end of content imposition, it is necessary for the imposition processor to count the number of sets in the PDL content.
TotalSheetCount ?	integer	Is the total number of physical or ▸ Logical Sheets generated during the processing of the automated ***Layout*** resource. ▸ Logical Sheets are used when ▸ Logical Stacks are defined. The value of this variable may be recalculated in later ***Layout*** partitions using the ***Layout***/@*SheetCountReset* attribute. @*TotalSheetCount* is always reset to zero at the beginning of processing of a set regardless of the value of ***Layout***/@*SheetCountReset*.
TotalSheetsInCollect ?	integer	Is the total number of physical or ▸ Logical Sheets that make up the current collect generated by an automated ▸ Imposition Template from the current ▸ Page Pool or ▸ Page Pool List being processed. ▸ Logical Sheets are used when ▸ Logical Stacks are defined.
TotalSheetsInPool ?	integer	Is the total number of physical or ▸ Logical Sheets generated from the current page pool or page pool list within an automated ▸ Imposition Template. ▸ Logical Sheets are used when ▸ Logical Stacks are defined.

The above variables MAY be used for controlling the activation of printer's marks (See ***Layout***/***MarkObject***/***MarkActivation***). For example:

Example 6.4: Automated Imposition: MarkObject

This example causes a slug line to be imaged on the bottom center of the first sheet of the set of sheets comprising a signature instance. Here are the details. For ***MarkActivation***/@*Context*, its value of "CollectIndex" specifies that the value of @*CollectIndex* is the index used with ***MarkActivation***/@*Index*. For ***Layout***/@*Index*, its value of 0 specifies that the sheet receive the specified slug line if the value of @*CollectIndex* is 0 (i.e., if it is first sheet of the signature instance).
Note: If @*Index* were "1 4 6", then the slug line would go on the second, fifth and seventh sheets.

```
<MarkObject Anchor="BottomCenter" CTM="1 0 0 1 0 0">
    <DeviceMark FontSize="8" Font="MySlugLineFont"/>
    <!--Result: Gender=male -->
    <JobField JobFormat="Gender=%s" JobTemplate="GeneralID:Gender"/>
    <RefAnchor Anchor="BottomCenter" AnchorType="Sibling" rRef="1000006"/>
    <MarkActivation Context="CollectIndex" Index="0"/>
</MarkObject>
```

6.3.18.3 Execution Model for Automated Imposition

The **Imposition** process transforms the sequences of pages contained within a ▸ Page Pool or ▸ Page Pool List to a specific sequence of imposed sheet surfaces. The ▸ Imposition Templates and the order of the ▸ Imposition Templates defined by the ***Layout*** resource explicitly define the page to sheet surface mapping transformation applied by the imposition engine.

The pseudo-code below describes the processing performed by the imposition engine at a high level:

```
For each Set in the order specified in the input RunList (Document)
   For each ▸ Imposition Template
      For each ▸ Page Pool in the Set
         If the ▸ Partition Key conditions for the ▸ Imposition Template are satisfied
         then process the ▸ Page Pool through the ▸ Imposition Template.
```

Thus, each *Layout* Resource ▸ Imposition Template is processed in the XML structure order specified. Every ▸ Page Pool belonging to the current set is then evaluated against the partition keys specified for that ▸ Imposition Template to determine if it SHALL be processed by that ▸ Imposition Template.

Since each ▸ Page Pool is evaluated for each ▸ Imposition Template, it is possible to reuse the same ▸ Page Pool with multiple ▸ Imposition Templates. For an example algorithm for processing ▸ Page Pools through an ▸ Imposition Template, see ▸ Example 1329 Algorithm for Processing an Imposition Template.

The *RunList* resource output from the **Imposition** process represents a sequence of imposed sheet surfaces where each surface may be represented either by pointing to PDL content where all the input pages are imposed onto single PDL pages, or, when used with a combined process may refer to the page set along with imposition instructions to the interpreter using an exchange resource. The structure of the *Layout* resource affects the partition keys conserved by its output *RunList* (and its referenced content), by conserving all partition keys specified in the *Layout* along with generating all of the appropriate partition keys, such as @*SetIndex*, @*DocIndex*, @*SheetIndex*. The output *RunList* can be viewed conceptually as a collection of sheet surface pairings (front and back) that conserves information about which *Layout* ▸ Imposition Template and ▸ Page Pool metadata that was in scope at the time the sheets were generated.

Note: @*DocIndex* is always generated even if every set contains only a single document; a set that contains only pages is treated as a set with a single document.

Note: *MarkObject*/@*Ord* works in the same way for automated imposition as for non-automated imposition. In other words, the @*Ord* value corresponds to the page entry described by that absolute @*Ord* position in the *RunList* (Marks).

Example 6.5: Imposition Template: Layout

Thus, if the ▸ Imposition Template (*Layout*) in this example is applied, then the resulting *RunList* resource conceptually conserves the following partition keys: @*SetIndex*, @*SheetIndex*, @*DocTags*, @*DocIndex*, @*SheetName* and @*Side* along with any other in-scope partition keys.

Note that in this example, @*SetIndex* and @*DocIndex* are conserved by setting @*EndOfSet* and @*EndOfDocument* respectively in the output *RunList* (Surface). In a *Layout* that defines ▸ Logical Stacks containing multiple documents or sets within ▸ Imposed Sheet Sets, @*SetIndex* and @*DocIndex* would need to be conserved by explicitly setting the value of the @*SetIndex* and @*DocIndex* partition keys. The *RunList* is expected to be partitioned by @*Run*, where each @*Run* represents one or more sheets, each having at least one surface either implied by *RunList*/@*SheetSides*, or explicitly partitioned by @*Side*.

```
<Layout Class="Parameter" ID="L1" Status="Available"
        PartIDKeys="DocTags SheetName Side" Automated="true">
    <Layout DocTags="CoverLetter">
        <Layout SheetName="CoverLetterSheets">
            <Layout Side="Front">
                <ContentObject Ord="0" CTM="1 0 0 1 0 0"/>
            </Layout>
        </Layout>
    </Layout>
    <Layout DocTags="Booklet">
        <Layout SheetName="BookletSheets">
            <Layout Side="Front">
                <ContentObject Ord="0" CTM="1 0 0 1 0 0"/>
                <ContentObject Ord="-1" CTM="1 0 0 1 0 0"/>
            </Layout>
            <Layout Side="Back">
                <ContentObject Ord="1" CTM="1 0 0 1 0 0"/>
                <ContentObject Ord="-2" CTM="1 0 0 1 0 0"/>
            </Layout>
        </Layout>
    </Layout>
</Layout>
```

6.3.18.4 Configuration for Various Automated Impositions

6.3.18.4.1 Using Logical Stacks

An ▸ Imposed Sheet Set output by the imposition engine can describe multiple ▸ Logical Stacks. Each of these ▸ Logical Stacks is placed onto a well-defined section of the sheet definitions, and after printing will typically be cut in a postpress finishing operation, generating the representative physical stacks.

▸ Logical Stacks are configured through the use of two mechanisms:

- ***Layout***/***LogicalStackParams*** element specifies the control for each ▸ Logical Stack including how ▸ Logical Sheets are sequenced onto a ▸ Logical Stack, and restrictions on how ▸ Logical Sheets of ▸ Recipient Sets can span ▸ Logical Stacks and ▸ Imposed Sheet Sets.
- The abstract ***PlacedObject***/@*LogicalStackOrd* is used to assign individual placed object definitions to a ▸ Logical Stack. Each ***PlacedObject*** defines the CTM for placing that object onto the ▸ Logical Stack. Each of the ***PlacedObject*** elements will have the same @*Ord* value across the ▸ Logical Stacks.

To define a ▸ Logical Stack, the ***Layout***/***LogicalStackParams*** element SHALL be present in the root of the ***Layout*** resource. This element configures the imposition engine to place ▸ Logical Sheets within ▸ Logical Stacks. The maximum number of sheets that can make up an ▸ Imposed Sheet Set is specified by ***LogicalStackParams***/@*MaxStackDepth*. Stacks are identified through the use of ***LogicalStackParams***/***Stack***/@*LogicalStackOrd*; the first ▸ Logical Stack is @*LogicalStackOrd* = "0", the 2nd is "1", etc.

All ▸ Logical Stacks defined by ***Layout***/***LogicalStackParams*** SHALL be used in all ▸ Imposition Templates, with the exception of an optional sheet (see ***Layout***/***SheetCondition*** in ▸ Section 8.84.15 SheetCondition) having a @*Condition* of "LogicalStackSetBegin" or "LogicalStackSetEnd" – these optional ▸ Logical Sheets are placed into a specific ▸ Logical Stack as specified by the ***PlacedObject***/@*LogicalStackOrd* in the optional sheet.

The imposition works by traversing each ▸ Logical Stack (in the sequence specified by ***LogicalStackParams***/***Stack***/@*LogicalStackSequence*). Each ▸ Imposition Template is processed where ***PlacedObject*** elements are evaluated for one of two cases:

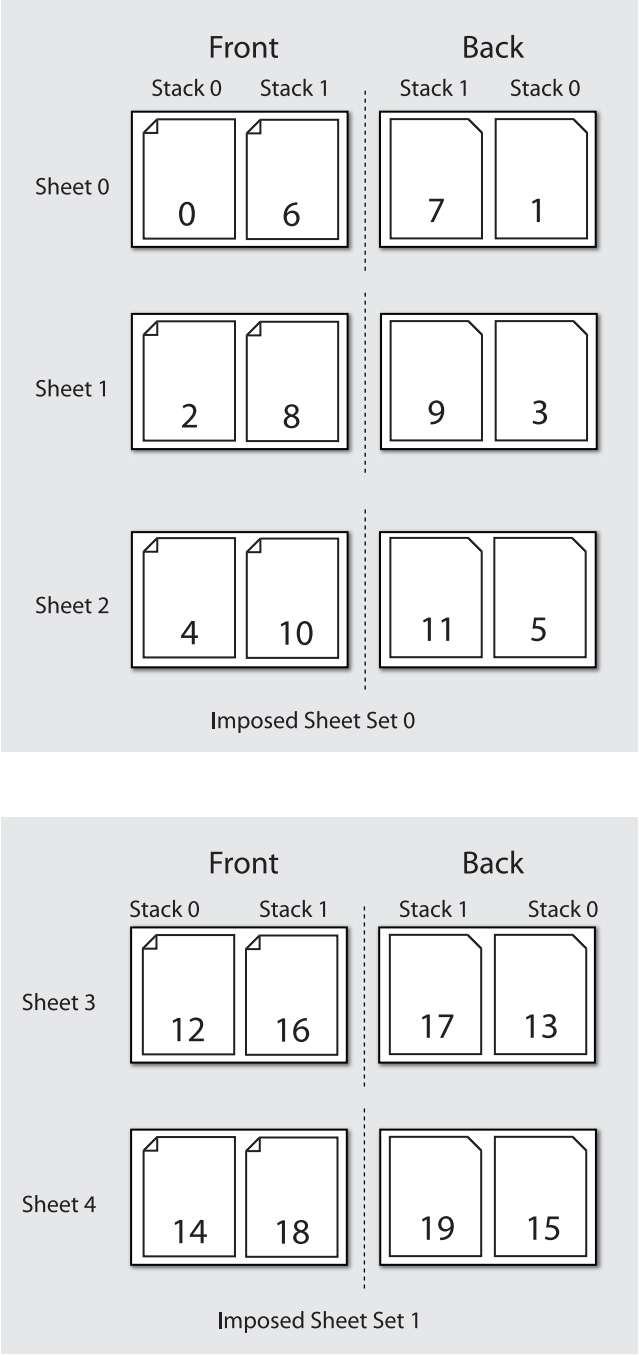
1. The ***PlacedObject*** has no @*LogicalStackOrd*. In this case, the ***PlacedObject*** is considered to be a physical sheet-level object, and is placed once at the start of processing for a physical sheet. Note that only information relevant to a physical sheet (such as @*SheetIndex*) is in scope for use in generating dynamic marks. An example of a physical sheet-level mark is a cut mark for where to cut the stacks.
2. The ***PlacedObject*** has a @*LogicalStackOrd*. In this case, only ***PlacedObject*** elements that have a matching @*LogicalStackOrd* for the current ▸ Logical Stack being processed are placed. Note that information relevant to documents and pages (such as @*CollectIndex* or @*TotalSheetsInPool*) is in scope for use in generating dynamic marks.

When insufficient number of pages remain to complete all ▸ Logical Stacks in an ▸ Imposed Sheet Set, the imposition engine SHALL distribute all content evenly across ▸ Logical Stacks in order to minimize the number of sheets in that ▸ Imposed Sheet Set, while still honoring any restrictions specified in ***Layout***/***SheetCondition***, ***LogicalStackParams***/@*Restrictions* or ***Layout***/***PageCondition***.

6.3.18.4.1.1 Imposition for Cut and Stack

This example shows how to configure for cut and stack imposition. Cut and stack produces a sequence of ▸ Imposed Sheet Sets, where each ▸ Imposed Sheet Set is cut into separate physical stacks, then each physical stack is restacked into a larger stack. This simple example is configured for 2 ▸ Logical Stacks with a @*MaxStackDepth* = "3", and is filled with 20 pages. Content on the back of the sheet is placed head-to-head with the front content.
Note: The 2nd ▸ Imposed Sheet Set has distributed the remaining 8 pages onto 2 sheets.

Figure 6-1: Imposition for Cut and Stack



6.3.18.4.2 Imposition for Signatures with Saddle Stitching

Saddle stitched booklets typically contain pages selected from the front of the reader ordered list of pages and pages selected from the back of the reader ordered list of pages on the same sheet. For instance the outside cover of a 16 page booklet will contain the first page (@*Ord* = "0") on the right of the sheet and the last page (@*Ord* ="15") on the left of the sheet. The pagination for the inner sheets is calculated by adding to the page number from the front and by subtracting from the back. The next page inside the cover of a booklet printed in duplex will typically contain the third page (@*Ord* = "2") on the right and the third from last page (@*Ord* = "13") on the left. This behavior is described by specifying negative @*Ord* values for the ***ContentObject*** elements that are filled with pages from the back of the ***RunList*** in automated imposition. The following code illustrates how absolute @*Ord* values are assigned based on sheet iterations.

Note: ***Layout***/@*MaxCollect* specifies the maximum number of sheets per signature (e.g., in a perfect bound book). @*MaxCollect* specifies the maximum number of loops prior to restarting the signature.

Example 6.6: Automated Imposition: Ord Values

```
/*
* calculates a "real" ord value in an automated layout
*
* @param ord the Value of Ord in the layout
* @param nPages the total number of pages that are consumed by the Layout, if
*    frontOffset!=0 the pages before frontOffset are NOT counted
* @param loop which sheet loop are we on?
* @param maxOrdFront number of pages consumed from the front of the list
* @param maxOrdBack positive number of pages consumed from the back of the list
* @param frontOffset page number of the first page to be placed on ord 0 in loop 0
* @return the pge to assign in this Ord, -1 if no page fits
*/

public static int calcOrd(int ord, int nPages, int loop, int maxOrdFront,
       int maxOrdBack, int frontOffset){
    final int maxOrd = maxOrdFront + maxOrdBack;
    if(maxOrd*loop >= nPages){
       return -1; // we are in a loop that has no remaining pages
    }
    int page;
    if(ord >= 0){ // count from front
       page = ord + loop*maxOrdFront;
    } else { // the page to put on -1
       int end = nPages + maxOrd - 1 -((nPages +maxOrd - 1)%maxOrd);
       page = end - loop*maxOrdBack+ord;
    }
    // if a page evaluates to e.g. 10 and we only have 9 pages, ciao
    return page< nPages? page+frontOffset : -1;
}
```

6.3.18.4.3 Selecting from Multiple Imposition Templates When Processing an unstructured PDL

In this case, the imposition engine optionally selects between ▸ Imposition Templates based on the quantity of pages present in the ▸ Page Pool:

Layout/@*OrdsConsumed* restricts the pages of a ▸ Page Pool to which a given ▸ Imposition Template of an automated layout is applied. It is designed for use with ▸ Unstructured PDLs that only allow access to pages by index. For instance, a wraparound cover might be specified as page 0 and therefore a special cover sheet with only one ***ContentObject*** can be defined whereas the body sheets might contain 2 ***ContentObject*** elements per surface.

@*OrdsConsumed* is only used when you have one ▸ Page Pool and you want to restrict the number of pages to be processed for a given ▸ Imposition Template.

6.3.18.4.4 Imposition for Start of a Chapter

The ***Layout***/***PageCondition*** element may be used to specify where on a sheet a first page of a chapter (▸ Page Pool) starts. It does this by specifying which ***ContentObject*** elements on a sheet may not be used to place the first page of a chapter. An example may be found after ▸ Table 8.145 PageCondition Element.

6.3.18.4.5 Imposition for Regenerating Sheet Surfaces

There are two methods to configure the imposition engine for re-imposing sheet surfaces:

1. **Re-imposition by sheet or sheet surface:** A specific selection of sheets or surfaces imposed by the imposition engine may be selected using the controls of the ***RunListLink*** to the ***RunList*** (Surface) output from the **Imposition** process.
2. **Re-imposition of sheets from content:** Alternatively the ***RunListLink*** to the ***RunList*** (Document) input to the imposition engine may be partitioned to select specific content to be re-imposed.

For example, if the @*Metadata0* partition key has been configured to represent a recipient record number in a VDP job, that partition keys can be used to select a specific recipient record(s) for which to re-impose sheet surfaces.

Details on how to configure ***ResourceLink***/***Part*** elements for sheet re-imposition including how to correctly regenerate dynamic sheet marks may be found at ▸ Section 3.10.7 Linking to Resources and @*IgnoreContext* in ▸ Table 8.241 RunList Resource.

6.3.18.4.6 Imposition for Document-Major Processing of a VDP ▸ Structured PDL

To process a ▸ Structured PDL in ▸ Document Major Processing Order, the ***RunList*** (Document) input ***ResourceLink*** SHALL contain ***Part*** elements specifying the order in which documents SHALL be processed. This effects a virtual reor-

dering of the content present in the PDL. Details on how to configure *ResourceLink*/*Part* elements for content reordering may be found at ▸ Section 3.10.7 Linking to Resources and *@IgnoreContext* in ▸ Table 8.241 RunList Resource.

6.3.19 InkZoneCalculation

The **InkZoneCalculation** process takes place in order to preset the ink zones before printing. The *Preview* data are used to calculate a coverage profile that represents the ink distribution along and perpendicular to the ink zones within the printable area of the preview. The *InkZoneProfile* can be combined with additional, vendor-specific data in order to preset the ink zones and the oscillating rollers of an offset printing press.

Table 6.51: InkZoneCalculation – Input Resources

NAME	DESCRIPTION
InkZoneCalculationParams ? Modified in JDF 1.3	Specific information about the printing press geometry (e.g., the number of zones) to calculate the *InkZoneProfile*.
Layout ? New in JDF 1.1	Specific information about the *Media* (including type and color) and about the sheet (placement coordinates on the printing cylinder).
Preview	A low to medium resolution bitmap file representing the content to be printed.
Sheet ? Deprecated in JDF 1.1	Specific information about the *Media* (including type and color) and about the sheet (placement coordinates on the printing cylinder). Replaced by *Layout* in **JDF** 1.1.
TransferCurvePool ?	Function to apply **ContactCopying DigitalPrinting** and **ConventionalPrinting** process characteristics (e.g., press, climate and substrate) under certain standardized circumstances. This function can be used to generate an accurate *InkZoneProfile*.

Table 6.52: InkZoneCalculation – Output Resources

NAME	DESCRIPTION
InkZoneProfile	Contains information about ink coverage along and perpendicular to the ink zones for a specific press geometry.

6.3.20 Interpreting

The interpreting device consumes page descriptions and instructions for controlling the marking device (e.g., imagesetter, digital printers, CTP, digital printing combined processes, etc.). The parsing of graphical content in the page descriptions produces a canonical display list of the elements to be drawn on each page.

The interpreter SHALL act upon any device control instructions that affect the physical functioning of the marking device such as media selection and page delivery and implied *ColorSpaceConversion*. *Media* selection determines which type of medium is used for printing and where that medium can be obtained. Page delivery controls the location, orientation and quantity of physical output.

The interpreter is also responsible for resolving all system resource references. This includes handling font substitutions and dealing with resource aliases. However, the interpreter specifically does not get involved with any functions of the device that could be considered finishing features such as stapling, duplexing and collating.

Table 6.53: Interpreting – Input Resources

NAME	DESCRIPTION
ColorantControl ? Modified in JDF 1.1	Identifies the color model used by the job.
FontPolicy ?	Describes the behavior of the font machinery in absence of requested fonts.
InterpretingParams	Provides the parameters needed to interpret the PDL pages specified in the *RunList* resource.
PDLResourceAlias *	These resources allow a **JDF** to reference resources which are defined in a Page Description Language (PDL). For example, a *PDLResourceAlias* resource could refer to a font embedded in a PostScript file.
RunList	This resource identifies a set of PDL pages or surfaces which will be interpreted.

Table 6.54: Interpreting – Output Resources

NAME	DESCRIPTION
InterpretedPDLData ? Deprecated in JDF 1.2	Pipe of streamed data which represents the results of **Interpreting** the pages in the *RunList*. In **JDF** 1.2 and beyond, a *RunList* with *InterpretedPDLData* subelements describes the output content data for **Interpreting**.
RunList ? New in JDF 1.2	Pipe of streamed data which represents the results of **Interpreting** the pages in the *RunList*. The data is specified in *InterpretedPDLData* subelements. The format and detail of these is implementation specific. In general, it is assumed that the **Interpreting** and **Rendering** processes are tightly coupled and that there is no value in attempting to develop a general specification for the format of this data.

6.3.21 LayoutElementProduction

This process describes the creation of page elements. It also explains how to create a layout that can put together all of the necessary page elements, including text, bitmap images, vector graphics, PDL or application files such as Adobe In-Design®, Adobe PageMaker® and Quark XPress®. The elements might be produced using any of a number of various software tools. This process is often performed several times in a row before the final *LayoutElement*, representing a final layout file, is produced.

Table 6.55: LayoutElementProduction – Input Resources

NAME	DESCRIPTION
LayoutElement *	Metadata about the PDL or application file, bitmap image file, text file, vector graphics file, etc.
LayoutElementProduction Params ? New in JDF 1.3	The parameters for the **LayoutElementProduction** process.

Table 6.56: LayoutElementProduction – Output Resources

NAME	DESCRIPTION
LayoutElement ?	A URL of the PDL or application file is produced by this process. Exactly one of *LayoutElement* or *RunList* SHALL be specified.
RunList ?	A *RunList* of a *LayoutElement* resource of *@ElementType* "Page" or "Document" is produced if this **LayoutElementProduction** task is the last process of type **LayoutElementProduction**. Exactly one of *LayoutElement* or *RunList* SHALL be specified.

6.3.22 LayoutPreparation

New in JDF 1.1

The **LayoutPreparation** process specifies the process of defining the *Layout* resource for the **Imposition** process. Note that it is possible to create a combined process that includes both **LayoutPreparation** and **Imposition**. In this case, the *Layout* and *RunList* (Marks) resource would not be explicitly defined, since they are exchange resources between the two processes.

Table 6.57: LayoutPreparation – Input Resources

NAME	DESCRIPTION
LayoutPreparationParams	Set of parameters needed to control the **LayoutPreparation** process.
RunList (Document) ? Modified in JDF 1.2	List of documents and/or pages that will be input into the layout. Note that this *RunList* is for information only and not modified by the **LayoutPreparation** process.
RunList (Marks) ?	List of marks that will be input into the layout. These are typically printer's marks such as fold marks, cut marks, punch marks or color bars.

Table 6.58: LayoutPreparation – Output Resources

NAME	DESCRIPTION
Layout	The layout of the document to be imposed.
RunList (Marks) ?	List of marks that SHALL be used as input of the following **Imposition** process.
TransferCurvePool ?	Definition of the transfer curves and coordinate systems of the devices.

6.3.23 LayoutShifting

New in JDF 1.4

LayoutShifting specifies how to apply separation dependent shifts on a flat or objects on a press sheet.

The exact ordering of the process within the **RIPing** and **ImageSetting** and the elements referenced by input and output ***RunList*** elements are not defined by the specification since it is implementation dependent. **LayoutShifting** MAY occur on display lists, raster data or in the image setting hardware.

Table 6.59: LayoutShifting – Input Resources

NAME	DESCRIPTION
LayoutShift	Parameters for the **LayoutShifting**
RunList	References the input objects/flats to apply shifting to.

Table 6.60: LayoutShifting – Output Resources

NAME	DESCRIPTION
RunList	The output ***RunList*** references the image data that the separation dependent layout shifts applied to.

6.3.24 PageAssigning

New in JDF 1.4

This process sorts the possibly-unordered pages from one or more input ***RunList*** resources into reader's order and places the result in the output ***RunList***.

Table 6.61: PageAssigning – Input Resources

NAME	DESCRIPTION
PageAssignParams ?	Container for future or proprietary extensions.
RunList +	One or more ***RunList*** resources with potentially unsorted pages

Table 6.62: PageAssigning – Output Resources

NAME	DESCRIPTION
RunList	***RunList*** with pages sorted in reader's order so that it can be input to an **Imposition** process (i.e., the sequence of pages in ***RunList*** corresponds to ***Layout***/***ContentObject***/@*Ord*).

6.3.25 PDFToPSConversion

The **PDFToPSConversion** process controls the generation of PostScript from a single PDF document. This process MAY be used at any time in a host-based PDF workflow to exit to PostScript for use of tools that consume such data. Additionally, it MAY be used to actively control the physical printing of data to a device that consumes PostScript data. The **JDF** model of this MAY include a **PDFToPSConversion** process in a combined process node with a **PDFToPSConversion** process.

It is RECOMMENDED to replace **PDFToPSConversion** with the combination of **Interpreting** and **PDLCreation** processes.

Table 6.63: PDFToPSConversion – Input Resources

NAME	DESCRIPTION
PDFToPSConversionParams	Set of parameters needed to control the generation of PostScript.
RunList	List of documents and pages to be converted to PostScript.

Table 6.64: PDFToPSConversion – Output Resources

NAME	DESCRIPTION
RunList	Stream or streams of resulting PostScript code. This PostScript code can end up physically stored in a file or be piped to another process. ***PDFToPSConversionParams***/@*GeneratePageStreams* determines whether there is a single stream generated for all pages in the ***RunList*** or whether each page is generated in to a separate consecutive stream.

6.3.26 PDLCreation

New in JDF 1.3

The **PDLCreation** device consumes the display list of graphical elements generated by an **Interpreting**, **RasterReading** or a ***ByteMap*** and produces a new PDL output ***RunList*** based on the selected output resource parameters.

Table 6.65: PDLCreation – Input Resources

NAME	DESCRIPTION
ImageCompressionParams ?	This resource provides a set of controls that determines how images will be compressed in the resulting PDL pages.
PDLCreationParams ?	These parameters control the operation of the process that interprets the display list and produces the resulting PDL pages.
RunList	This resource is a pipe of streamed data that represents a device independent display list structure. The ***RunList*** SHALL specify either an ***InterpretedPDLData*** or ***ByteMap*** element, but not both.

Table 6.66: PDLCreation – Output Resources

NAME	DESCRIPTION
RunList	This resource identifies the location of the resulting PDL file(s). If the ***FileSpec***/@*MimeType* is specified, then the value SHALL match ***PDLCreationParams***/@*MimeType*. If not specified, then ***PDLCreationParams***/@*MimeType* is inserted.

6.3.27 Preflight

Preflighting is the process of examining the components of a print job to ensure that the job will print successfully and with the expected results. Preflight checks can be performed on each document or page identified within the associated ***RunList*** resource.

Preflighting a file is generally a two-step process. First, the documents are analyzed and compared to the set of tests. Then, a preflight report is built to list the encountered issues (according to the tests).

Agents record the instructions for, and devices record the results of, preflight operations in **JDF** jobs, using two types of resources: ***PreflightParams*** and ***PreflightReport***.

Table 6.67: Preflight – Input Resources (Sheet 1 of 2)

NAME	DESCRIPTION
PreflightParams	A specified list of tests against which documents and/or pages are to be tested.

Table 6.67: Preflight – Input Resources (Sheet 2 of 2)

NAME	DESCRIPTION
PreflightReportRulePool ? Modified in JDF 1.4	A list of rules used to build the ***PreflightReport***. Those rules are attached to actions in the ***ActionPool***. **Modification Note:** Starting with **JDF** 1.4, this resource becomes optional.
RunList	The list of documents and/or pages to be preflighted.

Table 6.68: Preflight – Output Resources

NAME	DESCRIPTION
PreflightReport	***PreflightReport*** is a container for logging information that is generated by the **Preflight** process.

6.3.28 PreviewGeneration

The **PreviewGeneration** process produces a low resolution ***Preview*** of each separation that will be printed. The ***Preview*** can be used in later processes such as **InkZoneCalculation**. The **PreviewGeneration** process typically takes place after **Imposition** or **RIPing**.

The **PreviewGeneration** can be performed in one of the following two ways: 1) the imaged printing plate is scanned by a conventional plate scanner or 2) medium to high resolution digital data are used to generate the ***Preview*** for the separation(s). The extent of the PDL coordinate system (as specified by the **MediaBox** attribute, the resolution of the preview image, and width and height of the image) SHALL fulfill the following requirements:

MediaBox-length / 72 * x-resolution = width ± 1

MediaBox-height / 72 * y-resolution = height ± 1

A gray value of 0 represents full ink, while a value of 255 represents no ink (see the DeviceGray color model in ▸ [PS] Chapter 4.8.2.

6.3.28.1 Rules for the Generation of the Preview Image

To be useful for the ink consumption calculation, the preview data SHALL be generated with an appropriate resolution. This means not only spatial resolution, but also color or tonal resolution. Spatial resolution is important for thin lines, while tonal resolution becomes important with large areas filled with a certain tonal value. The maximum error caused by limited spatial and tonal resolution SHOULD be less than 1%.

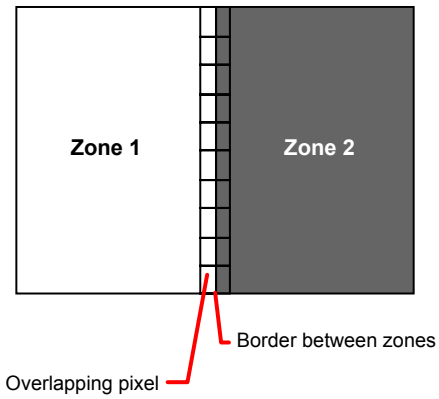
6.3.28.2 Spatial Resolution

Since some pixel of the preview image might fall on the border between two zones, their tonal values SHALL be split up. In a worst case scenario, the pixels fall just in the middle between a totally white and a totally black zone. In this case, the tonal value is 50%, but only 25% contributes to the black zone. With the resolution of the preview image and the zone width as variables, the maximum error can be calculated using the following equation:

$$error[\%] = \frac{100}{4 * resolution[L/mm] * zone_width[mm]}$$

For zone width broader than 25 mm, a resolution of 2 lines per mm will always result in an error less than 0.5%. Therefore, a resolution of 2 lines per mm (equal to 50.8 dpi) is suggested.

Figure 6-2: Worst case scenario for area coverage calculation



6.3.28.3 Tonal Resolution

The kind of error caused by color quantization depends on the number of shades available. If the real tonal value is rounded to the closest (lower or higher) available shade, the error can be calculated using the following equation:

$$error[\%] = \frac{100}{2 * number_of_shades}$$

Therefore, at least 64 shades SHOULD be used.

6.3.28.4 Line Art Resolution

When rasterizing line art elements, the minimal line width is 1 pixel, which means 1/resolution. Therefore, the relationship between the printing resolution and the (spatial) resolution of the preview image is important for these kind of elements. In addition, a specific characteristic of PostScript RIPs adds another error: within PostScript, each pixel that is touched by a line is set. Tests with different PostScript jobs have shown that a line art resolution of more than 300 dpi is normally sufficient for ink-consumption calculation.

6.3.28.5 Conclusion

There are quite a few different ways to meet the requirements listed above. The following list includes several examples:

- The job can be RIPed with 406.4 dpi monochrome.
- With anti-aliasing, the image data can be filtered down by a factor of 8 in both directions. This results in an image of 50.8 dpi with 65 color shades.
- High resolution data can also be filtered using anti-aliasing. First, the RIPed data, at 2540 dpi monochrome, are taken and filtered down by a factor of 50 in both directions. This produces an image of 50.8 dpi with 2501 color shades. Finally those shades are mapped to 256 shades, without affecting the spatial resolution.

Rasterizing a job with 50.8 dpi and 256 shades of gray is not sufficient. The problem in this case is the rendering of thin lines (see Line Art Resolution above).

6.3.28.6 Recommendations for Implementation

The following three guidelines are strongly RECOMMENDED:

- The resolution of RIPed line art SHOULD be at least 300 dpi.
- The spatial resolution of the preview image SHOULD be approximately 20 pixel/cm (= 50.8 dpi).
- The tonal resolution of the preview image SHOULD be at least 64 shades.

Table 6.69: PreviewGeneration – Input Resources (Sheet 1 of 2)

NAME	DESCRIPTION
ColorantControl ? New in JDF 1.1	The ***ColorantControl*** resources that define the ordering and usage of inks in print modules. Needed for generating thumbnails.
ExposedMedia ?	The **PreviewGeneration** process can use an exposed printing plate to produce a ***Preview*** resource. This task is performed using an analog plate-scanner. Exactly one of ***ExposedMedia***, ***Preview*** or ***RunList*** SHALL be specified in any **PreviewGeneration** process.

Table 6.69: PreviewGeneration – Input Resources (Sheet 2 of 2)

NAME	DESCRIPTION
Preview ? New in JDF 1.1	Medium or low resolution bitmap file that can be used for calculation of overviews and thumbnails. Exactly one of *ExposedMedia*, *Preview* or *RunList* SHALL be specified in any PreviewGeneration process.
PreviewGenerationParams	Parameters specifying the size and the type of the preview.
RunList ?	High resolution bitmap data are consumed by the PreviewGeneration process. These data represent the content of a separation that is recorded on a printing plate or other such item. Exactly one of *ExposedMedia*, *Preview* or *RunList* SHALL be specified in any PreviewGeneration process.
TransferCurvePool ? New in JDF 1.1	Area coverage correction and coordinate transformations of the device.

Table 6.70: PreviewGeneration – Output Resources

NAME	DESCRIPTION
Preview	The *Preview* data are comprised of low to medium resolution bitmap files representing, for example, the content of a separation that is recorded on a printing plate or other such item. A *Preview* can also be used to visualize resources, such as thumbnail images.

6.3.29 Proofing

Deprecated in JDF 1.2

The Proofing process is deprecated in **JDF** 1.2. Instead, use a combined process to produces the hard proof (e.g., one that includes the ImageSetting, ConventionalPrinting or DigitalPrinting process). Then input the hard proof to a separate Approval process. See ▸ Section N.5.10 Proofing for details of this deprecated process. In **JDF** 1.2 and beyond, proofing is a combined process.

6.3.30 PSToPDFConversion

This section defines the controls needed to invoke a device that accepts a PostScript stream and produces a set of PDF pages as output.

It is RECOMMENDED to replace PSToPDFConversion with the combination of Interpreting and PDLCreation processes.

Table 6.71: PSToPDFConversion – Input Resources

NAME	DESCRIPTION
FontParams ?	These parameters determine how the conversion process will handle font errors encountered in the PostScript stream.
ImageCompressionParams ?	This resource provides a set of controls that determines how images will be compressed in the resulting PDF pages.
PSToPDFConversionParams ?	These parameters control the operation of the process that interprets the PostScript stream and produces the resulting PDF pages.
RunList	This resource specifies where the PostScript stream SHALL be found.

Table 6.72: PSToPDFConversion – Output Resources

NAME	DESCRIPTION
RunList	This resource identifies the location of the resulting PDF pages.

6.3.31 RasterReading

New in JDF 1.3

The RasterReading device consumes raster graphic formatted files into a display list structure as the principal element to be drawn on each page. The RasterReading process is not a stand-alone process but is used in conjunction with processing and rendering processes in a combined process such as Rendering or PDLCreation.

Table 6.73: RasterReading – Input Resources

NAME	DESCRIPTION
RasterReadingParams ?	Additional parameters for reading raster files.
RunList	This resource identifies a set of raster pages or surfaces that will be inserted into the display list. This resource SHALL reference *ByteMap* images.

Table 6.74: RasterReading – Output Resources

NAME	DESCRIPTION
RunList	Pipe of streamed data that represents the results of RasterReading the pages in the input *RunList*. The format and detail are implementation dependent. The *RunList* SHALL specify an *InterpretedPDLData* element that describes the output content data for RasterReading.

6.3.32 Rendering

The Rendering process consumes the display list of graphical elements generated by the Interpreting or RasterReading process. It converts the graphical elements according to the geometric and graphic state information contained within the display list, combined with the *RenderingParams* information to produce binary rasterized data suitable for processes which consume *ByteMap* information.

Table 6.75: Rendering – Input Resources

NAME	DESCRIPTION
ImageCompressionParams ? New in JDF 1.5	Allows definition of compressed Raster Images
InterpretedPDLData ? Deprecated in JDF 1.2	Pipe of streamed data that represents the results of Interpreting the pages in the *RunList*. In **JDF** 1.2 and beyond, a *RunList*/*InterpretedPDLData* subelement describes the input content data for Rendering.
Media ? Deprecated in JDF 1.1	This resource provides a description of the physical media which will be marked. The physical characteristics of the media can affect decisions made during Rendering.
RenderingParams ?	This resource describes the format of the byte maps to be created and other specifics of the Rendering process.
RunList ? New in JDF 1.2	Pipe of streamed data that represents the results of Interpreting or RasterReading the pages in the input *RunList*. The data is specified in *InterpretedPDLData* subelements. The format and detail of these is implementation specific. In general, it is assumed that the Interpreting, RasterReading, Rendering and PDLCreation are tightly coupled and that there is no value in attempting to develop a general specification for the format of this data. **Modification note:** Starting with **JDF** 1.4, all text replaced by text from *RunList* in output resource

Table 6.76: Rendering – Output Resources

NAME	DESCRIPTION
RunList	Pipe of streamed data that represents the results of Rendering. This *RunList* MAY be consumed by any following process that consumes raster data, including PDLCreation, ImageSetting or DigitalPrinting. The data MAY be specified in *ByteMap* sub-elements. In general, it is assumed that the Interpreting, RasterReading, Rendering and PDLCreation are tightly coupled and that there is no value in attempting to develop a general specification for the format of this data. **Modification note:** Starting with **JDF** 1.4, first half of text is modified.

6.3.33 RIPing

RIPing is a Gray Box (see ▸ Section 3.3.2.1 Use of the Types Attribute in Process Group Nodes – Gray Boxes) that is a combination of at least two processes. Most often it includes **Interpreting** and **Rendering**, but it may also include **ColorSpaceConversion**, **Trapping**, **Separation**, **Imposition** and **Screening**. Thus one typical **RIPing** node is with ***JDF***/@*Type* = "ProcessGroup" and ***JDF***/@*Category* = "RIPing" as shown in the following example:

Example 6.7: RIPing

```
<JDF Type="ProcessGroup" Types="RIPing" Category="RIPing"
      ID="ID100" JobPartID="ID23" Status="Ready" Version="1.4"/>
```

The **RIPing** process consumes page descriptions and instructions for producing the graphical output. It parses the graphical contents in the page descriptions, renders the contents, and produces a rasterized image of the page. This raster MAY contain contone data and be represented upon output as a ***ByteMap***. Alternatively, the **RIPing** process MAY also perform halftone screening, in which case the output is in the form of a bitmap. It is also responsible for resolving all system resource references that include font handling and resource aliasing.

Instructions read by the RIP include information about the media, halftoning, color transformations, colorant controls and other items that affect that rasterized output. They do not, however, represent any specific controls for the physical output device, nor do they deal with any instructions intended for the finishing device.

In most cases, RIPing will be part of a combined process with a process that specifies physical marking (e.g., **DigitalPrinting** or **ImageSetting**). In this case, the interpreter SHOULD be able to act upon device control instructions that affect the physical functioning of the printing device such as media selection and page delivery. ***Media*** selection determines which type of medium is used for marking and where that medium can be obtained. Page delivery controls the location, orientation and quantity of physical output. The RIP is also responsible for resolving all system resource references. This includes handling font substitutions and dealing with resource aliases. However, the RIP specifically does not get involved with any functions of the device that could be considered finishing features such as stapling, duplexing and collating.

When a **RIPing** process is comprised of only the **Interpreting** and **Rendering** processes, various intermediary steps are needed before the output can be run through a **ConventionalPrinting** process. In theory, however, a workflow could include no intermediary steps between a **RIPing** process and a **DigitalPrinting** process. The following workflow scenarios represent possible process chains in each circumstance:

RIPing → **Screening** → **ImageSetting** → **ContactCopying** → **ConventionalPrinting**

RIPing → (**Screening**) → **DigitalPrinting**

Since **RIPing** is not a predefined **JDF** process, see the processes that contribute to the RIP for input and output resources.

6.3.34 Scanning

The **Scanning** process creates bitmaps from analog images using a scanner.

Table 6.77: Scanning – Input Resources

NAME	DESCRIPTION
ExposedMedia	Description of the media to be scanned. The ***ExposedMedia*** SHOULD be partitioned by @*RunIndex*, in order to provide unique mapping from ***ExposedMedia*** to the output ***RunList***.
ScanParams	High level scanner settings. These settings are specifically not intended as a replacement for low-level device interfaces such as TWAIN.

Table 6.78: Scanning – Output Resources

NAME	DESCRIPTION
RunList	List of a ***ByteMap*** resource or a ***LayoutElement*** resource of @*ElementType* = "Image".

6.3.35 Screening

This process specifies the process of halftone screening. It consumes contone raster data (e.g., the output from an **Interpreting** and **Rendering** process). It produces monochrome which has been filtered through a halftone screen to identify which pixels are needed to approximate the original shades of color in the document.

This process definition includes capabilities for halftoning after raster image processing according to the PostScript definitions. Alternatively it allows for the selection of FM screening/error diffusion techniques. In general, an actual screening process will be a combined process of **ContoneCalibration** and **Screening** processes.

Table 6.79: Screening – Input Resources

NAME	DESCRIPTION
RunList	Ordered list of rasterized ***ByteMap*** or ***InterpretedPDLData*** representing pages or surfaces.
ScreeningParams	Parameters specifying which halftone mechanism SHALL be applied and with what specific controls.

Table 6.80: Screening – Output Resources

NAME	DESCRIPTION
RunList	Ordered list of rasterized and screened output pages. Assumes that the resolution remains the same and that resulting data are one bit per component. Furthermore, the organization of planes within the data does not change.

6.3.36 Separation

The **Separation** process specifies the controls associated with the generation of color-separated data. It is designed to be flexible enough to allow a variety of possible methods for accomplishing this task. First of all, it sponsors host-based PDF separating operations, in which a ***RunList*** of pre-separated PDF data is generated. It can also be combined with a RIP to allow control of In-RIP separations. In this scenario a ***RunList*** containing ***ByteMap*** resources generated as the output. Yet another anticipated combination is with the process to deal with incoming device-dependent data. And finally, it MAY be part of a combined process with an **ImageReplacement** process in order to do image substitution for omitted or proxy images.

Table 6.81: Separation – Input Resources

NAME	DESCRIPTION
ColorantControl ? Modified in JDF 1.1A	Identifies which colorants in the job are to be output.
RunList	List of pages that are to be operated on.
SeparationControlParams	Controls for the separation process.

Table 6.82: Separation – Output Resources

NAME	DESCRIPTION
RunList	List of separated pages or separated raster bytemaps.

6.3.37 ShapeDefProduction

New in JDF 1.4

This process describes the structural design of a one-up product (e.g., a non rectangular label, a box, a display, a bag, a pouch, etc.). This is a description of the unprinted blank box as it will be available after **ShapeCutting** and before **BoxFolding**. Also, this process typically (but not exclusively) describes the process of designing the shape of a new box using a CAD application. See **DieLayoutProduction** for the process of designing a die for multiple one-up products. The output of the **ShapeDefProduction** process can be multiple ***ShapeDef*** resources (e.g., when the design of the box results in multiple pieces, such as a box, an object and an insert piece, where the insert piece is fixed to the object to be packed in the box). Another example would be a multi-piece display. The **ShapeDefProduction** process can be performed by a human operator using a CAD application. In some cases it can be an automated process. Note that **ShapeDefProduction** needs information stored in both ***ShapeDefProductionParams*** and ***ShapeDef*** to make a new structural design.

Table 6.83: ShapeDefProduction – Input Resources

NAME	DESCRIPTION
LayoutElement ?	A rough drawing or outline (e.g., an EPS) of the ***ShapeDef*** that serves as the input for structural design.
ShapeDefProductionParams	Parameters for the structural design.

Table 6.84: ShapeDefProduction – Output Resources

NAME	DESCRIPTION
ShapeDef +	A resource describing the shape of the product to be produced.

6.3.38 SheetOptimizing

New in JDF 1.5

SheetOptimizing describes ganging of multiple sections onto one or more printed sheets. Sections MAY be parts of unrelated customer jobs. This process is also referred to as job ganging.

SheetOptimizing MAY be used together with ***QueueSubmissionParams***/@*GangName* and the ***ForceGang*** command. In this case, individual jobs with identical ***QueueSubmissionParams***/@*GangName* are collected with each job submission. A ***ForceGang*** command instructs the ganging engine to process the waiting ***GangInfo*** elements.

SheetOptimizing is a further definition of the concepts first described in version 1.0 of **JDF** [**JDF** 1.0] and are found in ▸ Section 4.4.5 Case 5: Spawning and Merging of Independent Jobs above. **SheetOptimizing** MAY be defined either using Intent based job descriptions, as recommended in ▸ Section 4.4.5 Case 5: Spawning and Merging of Independent Jobs, or using process based job descriptions.

Table 6.85: SheetOptimizing – Input Resources

NAME	DESCRIPTION
Assembly *	Input ***Assembly*** to specify the binding order e.g. for creep calculation. These assemblies MAY contain sections that are not included in this sheet optimization (e.g., when only covers are optimized and the bodies are produced individually).
BinderySignature ?	List of ***BinderySignature*** elements that describe the individual gang candidate signatures. This ***BinderySignature*** SHALL at least be partitioned by @*BinderySignatureID*.
SheetOptimizingParams ?	Parameters specifying details that allow individual sections to be distributed on the printed sheets.

Table 6.86: SheetOptimizing – Output Resources

NAME	DESCRIPTION
StrippingParams	The ***StrippingParams*** resource that will be populated by the **SheetOptimizing** process. The resource MAY be partially populated by the submitter with restrictions on what the **SheetOptimizing** is allowed to do.

6.3.39 SoftProofing

Deprecated in JDF 1.2

The **SoftProofing** process is deprecated in **JDF** 1.2. Instead, use a combined process to produce the soft proof in which the last process is the **Approval** process that approves the soft proof. See ▸ Section N.5.11 SoftProofing for details of this deprecated process. In **JDF** 1.2 and beyond, soft proofing is a combined process.

6.3.40 Stripping

New in JDF 1.2

An important aspect of the interface between an MIS system and a prepress workflow system is imposition. When an order is accepted or even during the estimation phase, the MIS system determines how the product will be produced using the available equipment (e.g., presses, folders, cutters, etc.) in the most cost-efficient way. The result of this exercise has a large impact on imposition in prepress.

The **Stripping** process specifies the process of translating a high level structured description of the imposition of one or multiple job parts or part versions represented by the ***StrippingParams*** resource into a ***Layout*** resource for the **Imposition** process. Note that the **Stripping** process can generate all resources needed for the **Imposition** process, thus also the ***RunList*** (Marks).

The ***Assembly*** resource is often referred to as the product view, while the ***BinderySignature*** is referred to as the production view. In this way, ***Assembly***/@*BindingSide* typically refers to the bound side of the final product, while ***BinderySignature***/@*BindingEdge* refers to the bound side during production.

When both attributes are not equal, it is up to the stripping device to modify the orientation and/or sequence of the content pages to synchronize product and production view.

.

Table 6.87: Stripping – Input Resources

NAME	DESCRIPTION
Assembly +	***Assembly*** describes how the sections of the different job parts imposed together are combined. If multiple ***Assembly*** resources are defined, mapping between ***StrippingParams*** and ***Assembly*** is achieved by matching the respective @*JobID* and @*AssemblyIDs* attributes.
BinderySignature ?	List of ***BinderySignature*** elements that describe the individual signatures that are combined to produce a final product. This ***BinderySignature*** SHALL at least be partitioned by @*BinderySignatureID*.
ColorantControl ? New in JDF 1.3	Contains information on the colors and separations. Useful when creating marks that need color information.
RunList (Document) ?	List of documents. When available, this list can be used to generate a ***Layout*** and populated ***RunList*** (no ***LayoutElement***[@*ElementType* = "Reservation"]) which can be fed into a subsequent **Imposition** process.
StrippingParams	High level structured description of the imposition of one or multiple job parts or part versions.
TransferCurvePool ?	Definition of the transfer curves and coordinate systems of the devices. The coordinate system of the ***StrippingParams*** coincides with the ***Layout*** coordinate system specified in the ***TransferCurvePool***.

Table 6.88: Stripping – Output Resources

NAME	DESCRIPTION
Layout	The layout of the document to be imposed.
RunList (Document) ?	List of documents that are to be used as input of the following **Imposition** process.
RunList (Marks) ?	List of marks that are to be used as input of the following **Imposition** process.

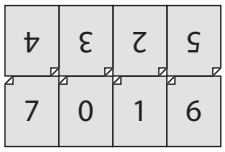
Example 6.8: Stripping: Simple Example

This simple example specifies three 16 page bindery signatures using folding catalog scheme F16-6.

```
<StrippingParams ID="FoldCatalogSample" Class="Parameter" Status="Available"
                 WorkStyle="WorkAndBack" PartIDKeys="SheetName">
    <BinderySignature FoldCatalog="F16-6"/>
    <StrippingParams SheetName="Sheet1"/>
    <StrippingParams SheetName="Sheet2"/>
```

WorkAndBack B2x4

WorkAndTurn B2x2



are based on a *B2x4* ***BinderySignature*** using the "Wor-
B2x2 using the "WorkAndTurn" WorkStyle.

```
<BinderySignature ID="B2x4" Class="Parameter" Status="Available"
       NumberUp="4 2">
    <SignatureCell FrontPages="15 0 3 12" BackPages="14 1 2 13"
          Orientation="Up"/>
    <SignatureCell FrontPages="8 7 4 11" BackPages="9 6 5 10"
          Orientation="Down"/>
</BinderySignature>
<BinderySignature ID="B2x2" Class="Parameter" Status="Available"
       NumberUp="2 2">
    <SignatureCell FrontPages="7 0" BackPages="6 1" Orientation="Up"/>
    <SignatureCell FrontPages="4 3" BackPages="5 2" Orientation="Down"/>
</BinderySignature>
<StrippingParams ID="L1" Class="Parameter" Status="Available"
       WorkStyle="WorkAndBack" PartIDKeys="SheetName">
    <StrippingParams SheetName="Sheet1">
       <BinderySignatureRef rRef="B2x4"/>
    </StrippingParams>
    <StrippingParams SheetName="Sheet2">
       <BinderySignatureRef rRef="B2x4"/>
    </StrippingParams>
    <StrippingParams WorkStyle="WorkAndTurn" SheetName="Sheet3">
       <BinderySignatureRef rRef="B2x2"/>
       <Position RelativeBox="0 0 0.5 1"/>
       <Position RelativeBox="0.5 0 1 1" Orientation="Flip180"/>
    </StrippingParams>
</StrippingParams>
```

6.3.41 Tiling

The **Tiling** process allows the contents of surfaces to be imaged onto separate pieces of media. Note that many different workflows are possible. **Tiling** SHALL always follow **Imposition**, but it can operate on imposed PDL page contents or on contone or halftone data. **Tiling** will generally be part of a combined process. For example, **Tiling** might be part of a combined process with **ImageSetting**. In that case, the input would be a ***RunList*** that contains ***ByteMap*** resources for each surface.

Table 6.89: Tiling – Input Resources

NAME	DESCRIPTION
RunList (Marks) ?	Structured list of incoming marks. These are typically printer's marks that provide the information needed to combine the tiles.
RunList (Surface)	Structured list of imposed page contents or Byte Maps that are to be decomposed to produce the images for each tile. The @*ElementType* value of the ***LayoutElement*** resource SHALL be "Surface".
Tile	A partitioned ***Tile*** resource that describes how the surface contents are to be decomposed.

Table 6.90: Tiling – Output Resources

NAME	DESCRIPTION
RunList	Structured list of portions of the decomposed surfaces. The value of the @*ElementType* attribute of the ***LayoutElement*** resource SHALL be "Tile".

6.3.42 Trapping

Trapping is a prepress process that modifies PDL files to compensate for a type of error that occurs on presses. Specifically, when more than one colorant is applied to a piece of media using more than one inking station, the media might not stay in perfect alignment when moving between inking stations. Any misalignment will result in an error called misregistration. The visual effect of this error is either that inks are erroneously layered on top of one another, or, more seriously, that gaps occur between inks that are intended to abut. In this second case, the color of the media is revealed

in the gap and is frequently quite noticeable. **Trapping**, in short, is the process of modifying PDL files so that abutting colorant edges intentionally overlap slightly, in order to reduce the risk of gaps.

The **Trapping** process modifies a set of document pages to reduce or (ideally) eliminate visible misregistration errors in the final printed output. The process MAY be part of a combined process with **RIPing** or specified as a stand-alone process.

Table 6.91: Trapping – Input Resources

NAME	DESCRIPTION
ColorantControl ? Modified in JDF 1.1A	Identifies color model used by the job.
FontPolicy ? New in JDF 1.1	Describes the behavior of the font machinery in absence of requested fonts.
RunList	Structured list of incoming page contents that are to be trapped.
TrappingDetails	Describes the general setting needed to perform trapping.

Table 6.92: Trapping – Output Resources

NAME	DESCRIPTION
RunList	Structured list of the modified page contents after **Trapping** has been executed.

6.4 Press Processes

Press processes are various technological procedures involving the transfer of ink to a substrate. From a technical standpoint they are often classified in impact and non-impact printing technologies. The impact printing class can be further subdivided into relief, intaglio, planograph or screen technologies, which in turn can be divided in further subparts. Because of the way a workflow is constructed in **JDF**, however, a different approach to classification was used. All of the various printing technologies are gathered into two categories:

1 **ConventionalPrinting**, which involves printing from a physical master,

2 **DigitalPrinting**, which involves generic commercial printing from a digital master.

The **ConventionalPrinting** and **DigitalPrinting** processes can be applied to either web or sheetfed printing.

The most prominent physical, planographic printing technologies are offset lithography and electrophotography. They are also the printing processes with the highest adoption in today's graphic arts industry. Consequently, the **ConventionalPrinting** process in **JDF** takes them as models. That does not mean, however, that other printing techniques can not make use of the **ConventionalPrinting** process and its resources. The extensibility features of **JDF** can be used to fill other requirements related to printing technology.

6.4.1 ConventionalPrinting

This process covers several conventional printing tasks, including sheet-fed printing, web printing, web/ribbon coating, converting and varnishing. Typically, each takes place after prepress and before postpress processes. Direct imaging technology on press is modelled as a combined process of **ImageSetting** and **ConventionalPrinting**. Press machinery often includes postpress processes (e.g., **WebInlineFinishing**, **Folding** and **Cutting**) as in-line finishing operations. The **ConventionalPrinting** process itself does not cover these postpress tasks.

Using a conventional printing press for producing a pressproof can be performed in the following two ways:

- A proof of type ***Component*** is produced with a **ConventionalPrinting** process. The result of this process is then sent to the **Approval** process, which in turn produces an ***ApprovalSuccess*** resource. That resource is then passed on to a second **ConventionalPrinting** process, which requires that the press be set up a second time.
- The *@DirectProof* attribute of the ***ConventionalPrintingParams*** can be used to specify the proof if it is produced during the **ConventionalPrinting** process. In this case, the press need only be set up once.

Note that the definition and ordering of separations is specified by ***ColorantControl***/***ColorantOrder*** of the appropriate resource.

In the context of web printing, the **ConventionalPrinting** process SHALL be in a combined process with the **WebInlineFinishing** process. The following drawing gives an overview about web printing in general.

Figure 6-3: Overview of Web Printing

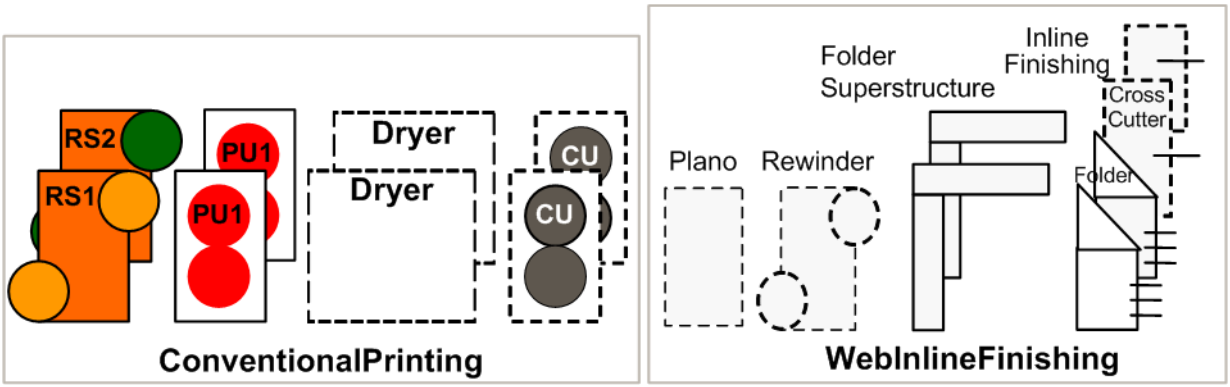


Table 6.93: ConventionalPrinting – Input Resources (Sheet 1 of 2)

NAME	DESCRIPTION
ColorantControl ?	The ***ColorantControl*** resources that define the ordering and usage of inks in print modules. The ***ColorantControl*** resource specifies the complete set of colors that will be printed on a sheet.
Component ? Modified in JDF 1.4	Various components in the form of preprints can be used in **ConventionalPrinting** in lieu of ***Media***. Examples include waste or a set of preprinted sheets. **Modification note:** Starting with **JDF** 1.4, the input ***ComponentLink*** NEED NOT have *@ProcessUsage*= "Input".
Component (Proof) ?	A proof component is used if a proof was produced during an earlier print run. Note that the proof MAY be a ***Component*** produced in a previous run and has not necessarily been produced explicitly as a proof. In general, at most one of ***Component*** (Proof) or ***ExposedMedia*** (Proof) SHOULD be specified.
ConventionalPrintingParams	Specific parameters to set up the press.
ExposedMedia (Cylinder) ? New in JDF 1.3	***ExposedMedia*** (Cylinder) is used to describe direct imaging on reusable cylinders. ***ExposedMedia*** (Cylinder) defines the set of cylinders to be used in the press run that is described by this node. Both ***ExposedMedia*** (Cylinder) and ***ExposedMedia*** (Plate) MAY occur in the same device. At least one of ***ExposedMedia*** (Cylinder) or ***ExposedMedia*** (Plate) SHALL be specified.
ExposedMedia (Plate) ? Modified in JDF 1.3	The printing plates and information about them are used to set up the press. The ***ExposedMedia*** (Plate) resource defines the set of plates to be used in the press run that is described by this node. Both ***ExposedMedia*** (Cylinder) and ***ExposedMedia*** (Plate) MAY occur in the same device. At least one of ***ExposedMedia*** (Cylinder) or ***ExposedMedia*** (Plate) SHALL be specified.
ExposedMedia (Proof) ?	A proof is used to compare color and content during **ConventionalPrinting**. This proof is produced by a prepress proofing device. At most one of ***ColorantControl*** (Proof) or ***ExposedMedia*** (Proof) SHOULD be specified.
ExposedMedia (Sleeve) ? New in JDF 1.4	Description of a sleeve.
Ink ? Modified in JDF 1.1	Information about the ink (e.g., brand, color) is useful to set up the press.
InkZoneProfile ?	The ***InkZoneProfile*** contains information about how much ink is needed along the printing cylinder of a specific printing press. It is only useful for Offset Lithography presses with ink key adjustment functions.

Table 6.93: ConventionalPrinting – Input Resources (Sheet 2 of 2)

NAME	DESCRIPTION
Layout ? New in JDF 1.1	Sheet and surface elements from the ***Layout*** tree (e.g., ***CIELABMeasuringField***, ***DensityMeasuringField*** or ***ColorControlStrip***) can be used for quality control at the press. The quality control field value and position can be of interest for automatic quality control systems. ***RegisterMark*** can be used to line up the printing plates for the press run, and its position can in turn be used to position items such as a camera.
Media ?	The physical substrate (e.g., paper or foil) and information about the ***Media*** (e.g., thickness, type and size) are useful in setting up paper travel in the press. This resource SHALL be present if no preprinted ***Component*** (Input) resource is used.
Media (MountingTape) ? New in JDF 1.4	Description of a mounting tape for a sleeve.
PrintCondition ? New in JDF 1.2	Used to control the use of colorants when printing pages on a specific media. The attributes and elements of the ***PrintCondition*** resource describe the aim values for a given printing process.
Sheet ? Deprecated in JDF 1.1	Specific information about the ***Media*** (including type and color) and about the sheet (e.g., placement coordinates on the printing cylinder). Replaced by ***Layout*** in **JDF** 1.1.
TransferCurvePool ? New in JDF 1.1	Area coverage correction and coordinate transformations of the device.

Table 6.94: ConventionalPrinting – Output Resources

NAME	DESCRIPTION
Component Modified in JDF 1.2	Describes the printed sheets, ribbons or webs which can be used by another printing process or postpress processes. Note that the @*Amount* attribute of the ***ResourceLink*** to this resource indicates the number of copies of the entire job which will be produced. **Modification note:** Prior to **JDF** 1.2 this ***Component*** was marked with a @*ProcessUsage* = "Good", which is OPTIONAL, but supported in **JDF** 1.2 and beyond.
Component (Waste) ? Deprecated in JDF 1.2	Produced waste of printed sheets or ribbons. In **JDF** 1.2 and beyond, **ConventionalPrinting** produces one ***Component*** that MAY be partitioned by @*Condition* in order to distinguish waste ***Component*** resources from good ***Component*** resources.

6.4.2 DigitalPrinting

DigitalPrinting is a direct printing process that, like **ConventionalPrinting**, occurs after prepress processes but before postpress processes. In **DigitalPrinting**, the data to be printed are not stored on an extra medium (e.g., a printing plate or a printing foil), but instead are stored digitally. The printed image is generated for every output using the digital data. Electrophotography, inkjet, and other technologies are used for transferring ink (both liquid ink and dry toner) onto the substrate. Furthermore, both Sheet-Fed and Web presses can be used as machinery for **DigitalPrinting**.

DigitalPrinting is often used to image a small area on preprinted ***Component*** resources to perform actions such as addressing or numbering another ***Component***. This kind of process can be executed by imaging with an inkjet printer during press, postpress or packaging operations. Therefore, **DigitalPrinting** is not only a press or prepress operation but sometimes also a postpress process.

Digital printing devices which provide some degree of finishing capabilities (e.g., collating and stapling) as well as some automated layout capabilities (e.g., N-up and duplex printing) MAY be modeled as a combined process which includes **DigitalPrinting**. Such a combined process MAY also include other processes (e.g., **Approval**, **ColorCorrection**, **ColorSpaceConversion**, **ContoneCalibration**, **Cutting**, **Folding**, **HoleMaking**, **ImageReplacement**, **Imposition**, **Interpreting**, **LayoutPreparation**, **Perforating**, **Rendering**, **Screening**, **Stacking**, **Stitching**, **Trapping** or **Trimming**).

Controls for **DigitalPrinting** are provided in the ***DigitalPrintingParams*** resource. The set of input resources of a combined process which includes **DigitalPrinting** MAY be used to represent an Internet Printing Protocol (IPP) Job or a PPML Job. See Application Notes for IPP and Variable Data printing. Note that putting a label on a product or ***DropItem*** is not **DigitalPrinting** but **Inserting**.

Table 6.95: DigitalPrinting – Input Resources

NAME	DESCRIPTION
ColorantControl ?	The ***ColorantControl*** resources that define the ordering and usage of inks in print modules.
Component * Modified in JDF 1.4	Various components can be used in **DigitalPrinting** instead of ***Media***. Examples include preprinted covers, waste, precut ***Media***, or a set of preprinted sheets or webs. If multiple ***Component*** (Input) resources are linked to one process, the mapping of media to content is defined in the partitions of ***DigitalPrintingParams***. At least one of ***Component*** or ***Media*** SHALL be specified as input **Modification note:** Starting with **JDF** 1.4, the input ***ComponentLink*** NEED NOT have @*ProcessUsage*= "Input".
Component (Proof) ?	A proof component is used if a proof was produced during an earlier print run (see description in ▸ Section 6.4.1 ConventionalPrinting). Note that the proof MAY be a ***Component*** produced in a previous run and has not necessarily been produced explicitly as a proof. In general, at most one of ***Component*** (Proof) or ***ExposedMedia*** SHOULD be specified.
DigitalPrintingParams	Specific parameters to set up the machinery.
ExposedMedia ?	A proof is useful for comparisons (completeness, color accuracy) with the print out of the **DigitalPrinting** process. In general, at most one of ***Component*** (Proof) or ***ExposedMedia*** SHOULD be specified
Ink ?	Ink or toner and information that is needed for **DigitalPrinting**.
Layout ? New in JDF 1.1	Sheet and surface elements from a ***Layout*** (e.g., the ***CIELABMeasuringField***, ***DensityMeasuringField*** or ***ColorControlStrip***) can be used for quality control at the press. The value and position of the quality can be of interest for automatic quality control systems. ***RegisterMark*** resources can be used to line up the printing registration during press run, and its position can in turn be used to position an item such as a camera.
Media *	The physical ***Media*** and information about the ***Media*** (e.g., thickness, type and size) is used to set up paper travel in the press. This has to be present if no preprinted ***Component*** (Input) resource is present. Unprinted ***Media*** used for covers are also defined as ***Media***. At least one of ***Component*** or ***Media*** SHALL be specified as Input **Note:** Printing a job on more than one web or sheet at the same time is parallel processing.
PrintCondition ?	Used to control the use of colorants when printing pages on a specific media. The attributes and elements of the ***PrintCondition*** resource describe the aim values for a given printing process.
RunList	Rendered data in Byte Maps that will be printed on the digital press are needed for **DigitalPrinting**. The ***RunList*** contains only ***ByteMap*** elements.
Sheet ? Deprecated in JDF 1.1	Specific information about the ***Media*** (including type and color) and about the sheet (placement coordinates on the printing cylinder). Replaced by ***Layout*** in **JDF** 1.1.
TransferCurvePool ? New in JDF 1.1	Area coverage correction and coordinate transformations of the device.

Table 6.96: DigitalPrinting – Output Resources (Sheet 1 of 2)

NAME	DESCRIPTION
Component Modified in JDF 1.2	Components are produced for other printing processes or postpress processes. Note that the @*Amount* attribute of the ***ResourceLink*** to this resource indicates the number of copies of the entire job which SHALL be produced. Prior to **JDF** 1.2 this ***Component*** was marked with a @*ProcessUsage* = "Good", which is OPTIONAL, but supported in **JDF** 1.2 and beyond MAY be specified. **Note:** When processing a PDL with multiple documents or sets, such as pdf/vt, the amount is defined in the scope of the entire document. If one copy of the number of copies defined within the PDL file of each record is requested, the ***Component***/@*Amount* SHALL be set to 1.

Table 6.96: DigitalPrinting – Output Resources (Sheet 2 of 2)

NAME	DESCRIPTION
Component (Waste) ? Deprecated in JDF 1.2	Produced waste, MAY be used by other processes. In **JDF** 1.2 and beyond, **DigitalPrinting** produces one ***Component*** that MAY be partitioned by @*Condition* in order to distinguish waste ***Component*** resources from good ***Component*** resources.

6.4.3 Varnishing

New in JDF 1.4

Varnishing is the process of varnishing a ***Component***. Spot varnishing with a ripped image or a printing plate from ***ExposedMedia*** SHALL be described as **DigitalPrinting** or **ConventionalPrinting** with ***Ink***/@*Family* = "Varnish". All types of all-over (flood) varnishing or spot varnishing applied without a ripped image or a printing plate from ***ExposedMedia*** SHALL be described with the **Varnishing** process.

Offline coatings are typically intended to be protective. They can increase water-resistance, scuff-resistance, and even food resistance in the case of restaurant menus.

Common coating types requested by customers include UV coatings (Ultra Violet cured polymers) which provide higher durability, and aqueous coatings that are viewed as greener and typically more easily recycled at end-of-life. Both types of overall coating protect the printed image as well as the substrate.

Table 6.97: Varnishing – Input Resources

NAME	DESCRIPTION
Component ?	The ***Component*** to be varnished. Exactly one of ***Component*** or ***Media*** SHALL be specified.
ExposedMedia *	Various types of ***ExposedMedia*** MAY be specified for varnishing. See ***VarnishingParams***/@*VarnishMethod* for details
Ink ?	Details of the colorant that is used for **Varnishing**. ***Ink***/@*Family* SHOULD be "Varnish".
Media ?	The ***Media*** to be varnished. Exactly one of ***Component*** or ***Media*** SHALL be specified.
VarnishingParams ?	Details of the setup of the varnishing device.

Table 6.98: Varnishing – Output Resources

NAME	DESCRIPTION
Component	The varnished ***Component***.

6.4.4 IDPrinting

Deprecated in JDF 1.1

The IDPrinting process was deprecated in **JDF** 1.1. Instead, implementations SHOULD use a combined process that includes the **DigitalPrinting** process, thus improving interoperability by reducing one of the combinations of processes. Also the **IDPrinting** process defined a number of resources and subelements which are deprecated since they duplicate other resources. See ▸ Section N.5.12 IDPrinting for details of this deprecated process.

6.5 Postpress Processes

Postpress is the most flexible and varied area that is covered by this specification. The individual postpress processes are provided in alphabetical order.

6.5.1 AdhesiveBinding

Deprecated in JDF 1.1

The **AdhesiveBinding** process has been split into the following individual processes:

- **CoverApplication**
- **Gluing**
- **SpinePreparation**
- **SpineTaping**

Note that the parameters of the *GlueApplication* for adhesive-binding operations have been moved into *CoverApplicationParams* and *SpineTapingParams* as *GlueApplication* subelements. The generic *GlueApplication* for adhesive binding is now described by the Gluing process.

6.5.2 BlockPreparation

New in JDF 1.1

As there are many options for a hardcover book, the block preparation is more complex than what has already been described for other types of binding above. Those options are the ribbon band (numbers of bands, materials and colors), gauze (material and glue), headband (material and colors), kraft paper (material and glue) and tightbacking (different geometry and measurements).

Table 6.99: BlockPreparation – Input Resources

NAME	DESCRIPTION
Component	The BlockPreparation process consumes one *Component* and creates a book block.
BlockPreparationParams	Specific parameters to set up the machinery.

Table 6.100: BlockPreparation – Output Resources

NAME	DESCRIPTION
Component	One *Component* is produced: the prepared book block. The value of *Component*/@*ProductType* SHALL be "BookBlock".

6.5.3 BoxFolding

New in JDF 1.3

BoxFolding defines the process of folding and gluing blanks into folded flat boxes for packaging.

Table 6.101: BoxFolding – Input Resources

NAME	DESCRIPTION
BoxFoldingParams	Specific parameters to set up the folder gluer.
Component	The BoxFolding process consumes one *Component*, the folding blank. Its @*ProductType* = "BlankBox".
Component (Application) * Deprecated in JDF 1.4	This process MAY consume additional *Component* resources, such as windows, handles or inlets. These *Component* resources SHALL additionally be referenced from *BoxFoldingParams*/*BoxApplication* elements. **Deprecation note:** Starting with **JDF** 1.4, a combined process that includes the BoxFolding and Inserting processes replaces *BoxApplication*.

Table 6.102: BoxFolding – Output Resources

NAME	DESCRIPTION
Component	One *Component* is produced: the folded flat box. Its @*ProductType* = "FlatBox".

6.5.4 BoxPacking

New in JDF 1.1

A pile, stack or bundle of products can be packed into a box or carton.

Table 6.103: BoxPacking – Input Resources (Sheet 1 of 2)

NAME	DESCRIPTION
BoxPackingParams	Specific parameters to set up the machinery.

Table 6.103: BoxPacking – Input Resources (Sheet 2 of 2)

NAME	DESCRIPTION
Component +	The **BoxPacking** process puts a set of ***Component*** resources into the box ***Component***. If more than one ***Component*** is specified, a ***Component***/***Bundle*** resource SHALL also be specified for each ***Component***. **Modification note:** Starting with **JDF** 1.4, ***Component*** can occur more than once.
Component (Box) ?	Details of the box or carton.
Media (Tie) ? New in JDF 1.3	Protective ***Media*** can be placed between individual rows of ***Component*** resources.
Media (Underlay) ? New in JDF 1.3	Protective ***Media*** can be placed between individual layers of ***Component*** resources.

Table 6.104: BoxPacking – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the boxed ***Component***.

6.5.5 Bundling

New in JDF 1.2

JDF 1.1 contains no process for bundling products. The **Bundling** process normally will be followed by a **Strapping** process. In a **Bundling** process, single products like sheets or signatures are bundled. The bundle is the output ***Component*** of the process and is used to store the products. As input a ***Component*** to a consuming or subsequent process (e.g., **Gathering**, **Collecting** or **Inserting**), the single components of a bundle are used.

Figure 6-4: Bundle Creation

Figure 6-5: Bundle Transport

Table 6.105: Bundling – Input Resources (Sheet 1 of 2)

NAME	DESCRIPTION
BundlingParams	Bundling parameters.

Table 6.105: Bundling – Input Resources (Sheet 2 of 2)

NAME	DESCRIPTION
Component	Component to be bundled
Media ?	End boards to protect the bundle. For each bundle a pair of end boards is needed.

Table 6.106: Bundling – Output Resources

NAME	DESCRIPTION
Component	The completed bundle.

Parameters like manufacturer and device type are defined in the ***Device*** element.

6.5.6 CaseMaking

New in JDF 1.1

Case making is the process where a hard case is produced. As there are many different kinds of hardcover cases, they will be described in a later version of the **JDF** specification.

Table 6.107: CaseMaking – Input Resources

NAME	DESCRIPTION
CaseMakingParams	Specific parameters to set up the machinery.
Component (CoverMaterial) ?	The cover material is either a preprinted or processed sheet of paper. Exactly one of ***Media*** (CoverMaterial) or ***Component*** (CoverMaterial) SHALL be specified.
Media (CoverBoard) Modified in JDF 1.1A	The cardboard ***Media*** used for the cover board.
Media (CoverMaterial) ?	The **CaseMaking** process MAY also consume unprocessed ***Media*** as cover material. Exactly one of ***Media*** (CoverMaterial) or ***Component*** (CoverMaterial) SHALL be specified.
Media (SpineBoard) ?	The cardboard ***Media*** used for the spine board. If not specified, the ***Media*** (CoverBoard) SHALL be used for the spine board.

Table 6.108: CaseMaking – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the book case. Its *@ProductType* = "BookCase".

6.5.7 CasingIn

New in JDF 1.1

The hard cover book case and the book block are joined in the **CasingIn** process.

Table 6.109: CasingIn – Input Resources

NAME	DESCRIPTION
CasingInParams	Specific parameters to set up the machinery.
Component	The prepared book block.
Component (Case)	The hard cover book case.

Table 6.110: CasingIn – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the completed hard cover book.

6.5.8 ChannelBinding

Various sizes of metal clamps can be used in **ChannelBinding**. The process can be executed in two ways. In the first, a pile of single sheets – sometimes together with a front and back cover – is inserted into a U-shaped clamp and crimped in special machinery. In the second, a pre-assembled cover that includes the open U-shaped clamp is used instead of the U-shaped clamp alone. The thickness of the pile of sheets determines in both cases the width of the U-shaped clamp to be used for forming the fixed document, which is not meant to be reopened later.

Table 6.111: ChannelBinding – Input Resources

NAME	DESCRIPTION
ChannelBindingParams	Specific parameters to set up the machinery.
Component	The operation requires one component: the block of sheets to be bound. If ***Component*** (Cover) is NOT provided and there is a cover, this ***Component*** SHALL be partitioned, and the first partition of this ***Component*** SHALL specify the cover **Modification note:** Starting with **JDF** 1.4, the input ***ComponentLink*** NEED NOT have *@ProcessUsage*= "BookBlock".
Component (Cover) ?	The empty cover with the U-shaped clamp that might, for example, have been printed before it is used during the **ChannelBinding** process.

Table 6.112: ChannelBinding – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the channel-bound component forming an item such as a brochure.

6.5.9 CoilBinding

Another name for **CoilBinding** is *spiral binding*. Metal wire, wire with plastic or pure plastic is used to fasten prepunched sheets of paper, cardboard or other materials. First, automated machinery forms a spiral of proper diameter and length. The ends of the spiral are then "tucked-in". Finally, the content is permanently fixed. Note that every time a coil-bound book is opened, a vertical shift occurs as a result of the coil action. This is a characteristic of the process.

Table 6.113: CoilBinding – Input Resources

NAME	DESCRIPTION
CoilBindingParams	Specific parameters to set up the machinery.
Component	The operation requires one component: the pile of prepunched sheets often including a top and button cover.

Table 6.114: CoilBinding – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the coil-bound component forming an item such as a calendar.

6.5.10 Collecting

This process collects folded sheets or partial products, some of which might have been cut. The first ***Component*** to enter the workflow lies at the bottom of the pile collected on a saddle, and the sequence of the input components that follows depends upon the produced component. The figure to the right shows a typical collected pile.

The operation coordinate system is defined as follows: The y-axis is aligned with the binding edge. It increases from the registered edge to the edge opposite to the registered edge. The x-axis is aligned with the registered edge. It increases from the binding edge to the edge opposite to the binding edge (i.e., the product front edge).

Table 6.115: Collecting – Input Resources

NAME	DESCRIPTION
Assembly ? New in JDF 1.3	***Assembly*** explicitly describes the sequence of the ***Component*** resources to be collected. If ***Assembly*** is not specified, the sequence is defined by the sequence of the ***Component***. Caution: ***Assembly*** has the first on the outside, whereas the ***Component*** resources are listed from inside to outside.
CollectingParams ?	Specific parameters to set up the machinery.
Component +	Variable amount of sheets to be collected.
DBRules * Deprecated in JDF 1.5	Database input that describes which sheets are to be collected for a particular instance component. In this version the schema is only human readable text. One rule is applied for each individual component.
DBSelection ? Deprecated in JDF 1.5	Database input that describes which sheets are to be collected for a particular instance component.
IdentificationField ? Deprecated in JDF 1.2	Information about identification marks on the component. In **JDF** 1.2 and beyond, this information is defined in the ***Component*** itself.

Table 6.116: Collecting – Output Resources

NAME	DESCRIPTION
Component	A block of collected sheets is produced. This ***Component*** can be joined in further post-press processes.

6.5.11 CoverApplication

New in JDF 1.1

CoverApplication describes the process of applying a soft cover to a book block.

Table 6.117: CoverApplication – Input Resources

NAME	DESCRIPTION
Component	The book block on which the cover is applied. If ***Component*** (Cover) is NOT provided, this ***Component*** SHALL be partitioned, and the first partition of this ***Component*** SHALL specify the cover.
Component (Cover) ? Modified in JDF 1.4	The soft cover that is applied. **Modification note:** Starting with **JDF** 1.4, this ***Component*** is optional because of the new rule about partitioning the main ***Component*** specified above.
CoverApplicationParams	Specific parameters to set up the machinery.

Table 6.118: CoverApplication – Output Resources

NAME	DESCRIPTION
Component	The book block with the applied soft cover.

6.5.12 Creasing

New in JDF 1.1

Sheets are creased or grooved to enable folding or to create even, finished page delimiters.

Table 6.119: Creasing – Input Resources

NAME	DESCRIPTION
Component Modified in JDF 1.2	This process consumes one *Component*: the printed sheets. Note that prior to **JDF** 1.2 this *Component* was OPTIONAL, which was clearly a typing mistake in the specification.
CreasingParams	Details of the Creasing process.

Table 6.120: Creasing – Output Resources

NAME	DESCRIPTION
Component	One creased *Component* is produced.

6.5.13 Cutting

Sheets are cut using a guillotine Cutting machine. Before Cutting, the sheets might be jogged and buffered. *CutBlock* resources and/or *CutMark* resources can be used for positioning the knife. After the Cutting process is performed, the blocks are often again buffered on a pallet.

Since Cutting is described here in a way that is machine independent as much as possible, the specified *CutBlock* elements do not directly imply a particular cutting sequence. Instead, a specialized agent SHALL determine the sequence.

Media might also be cut in a precutting step. In this case, Cutting MAY deliver *Media* as the output resource.

Cutting MAY also be used to describe cutting of a web into multiple ribbons on a web press. This process is commonly referred to as "Slitting".

Table 6.121: Cutting – Input Resources

NAME	DESCRIPTION
Component ?	This process consumes one *Component*: the printed sheets. Exactly one of *Component* or *Media* SHALL be specified as input.
CutBlock * Deprecated in JDF 1.1	One or more *CutBlock* resources can be used to define the Cutting sequence. Either *CutBlock* or *CuttingParams*/*Cut* SHALL be specified, but not both.
CutMark * Deprecated in JDF 1.1	*CutMark* resources can be used to adapt the theoretical cut positions to the real positions of the corresponding blocks on the *Component* to be cut.
CuttingParams New in JDF 1.1	Details of the Cutting process.
Media ?	Cutting can be applied to *Media* in order to adjust size or shape. Exactly one of *Component* or *Media* SHALL be specified as input.

Table 6.122: Cutting – Output Resources

NAME	DESCRIPTION
Component * Modified in JDF 1.3	One or several blocks of cut *Component* resources are produced. When an input *Component* is cut, the output SHALL be a set of *Component* resources. Either *Component* or *Media* SHALL be specified as output, but not both.
Media * Modified in JDF 1.3	When *Media* are cut, the output SHOULD also be a set of *Media*. Either *Component* or *Media* SHALL be specified as output, but not both.

6.5.14 DieMaking

New in JDF 1.4

This process describes the production of tools for a die cutter (e.g., in a die maker shop).

Table 6.123: DieMaking – Input Resources

NAME	DESCRIPTION
DieLayout	A resource describing the die cutter tool set.

Table 6.124: DieMaking – Output Resources

NAME	DESCRIPTION
Tool +	The set of tools for the die cutter.

6.5.15 Dividing

Deprecated in JDF 1.1

Dividing has been replaced by **Cutting**. See ▸ Section N.5.14 Dividing for details of this deprecated process.

6.5.16 Embossing

New in JDF 1.1

The **Embossing** process is performed after printing to stamp a raised or depressed image (artwork or typography) into the surface of paper using engraved metal embossing dies, extreme pressure and heat. Embossing styles include blind, deboss and foil-embossed.

Table 6.125: Embossing – Input Resources

NAME	DESCRIPTION
Component	This process consumes one ***Component*** which is embossed by the process.
EmbossingParams	Parameters to setup the machinery.
Media * Modified in JDF 1.4	If foil stamping or foil embossing, the stamping foil materials are REQUIRED. **Modification note:** Starting with **JDF** 1.4, ***Media*** can occur more than once.
Tool * Modified in JDF 1.4	The embossing stamps or calenders. **Modification note:** Starting with **JDF** 1.4, ***Tool*** can occur more than once.

Table 6.126: Embossing – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is created.

6.5.17 EndSheetGluing

EndSheetGluing finalizes the folded sheet or book block in preparation for case binding. It requires three ***Component*** resources – the back-end sheet, the book block and the front-end sheet – and information about how they are merged together. Back-end sheets and front-end sheets are in most cases sheets folded once before **EndSheetGluing** takes place. The end sheets serve as connections between the book block and the cover boards.

Table 6.127: EndSheetGluing – Input Resources (Sheet 1 of 2)

NAME	DESCRIPTION
Component Modified in JDF 1.5	A back-end sheet and a front-end sheet are glued onto the book block. At least one of ***Component***, ***Component*** (BackEndSheet) or ***Component*** (FrontEndSheet) SHALL be present. **Modification note:** Starting with **JDF** 1.4, the input ***ComponentLink*** NEED NOT have *@ProcessUsage*= "BookBlock".

Table 6.127: EndSheetGluing – Input Resources (Sheet 2 of 2)

NAME	DESCRIPTION
Component (BackEndSheet) ? Modified in JDF 1.5	A back-end sheet to be mounted on the book block. **Modification note:** Starting with **JDF** 1.5, this element is optional.
Component (FrontEndSheet) ? Modified in JDF 1.5	A front-end sheet to be mounted on the book block. **Modification note:** Starting with **JDF** 1.5, this element is optional.
EndSheetGluingParams	Specific parameters to set up the machinery.

Table 6.128: EndSheetGluing – Output Resources

NAME	DESCRIPTION
Component	A book block is produced that includes the end sheets.

6.5.18 Feeding

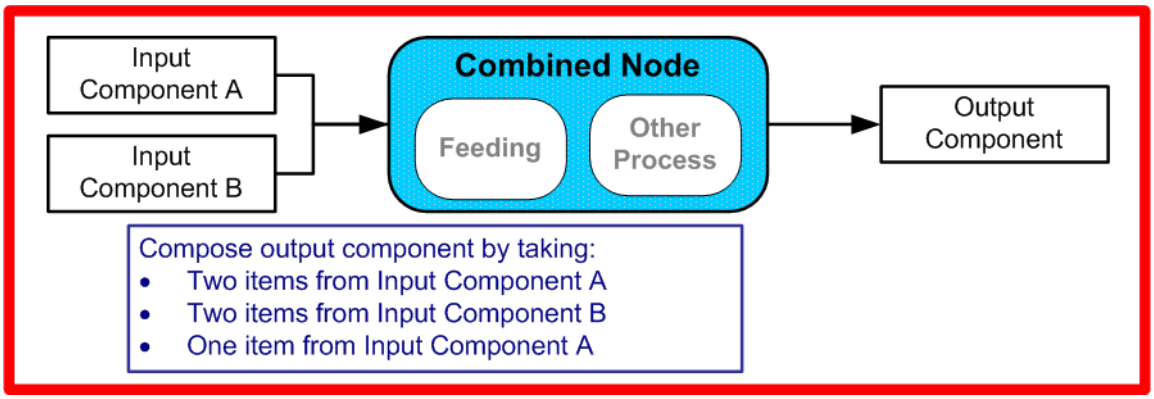
New in JDF 1.2

The **Feeding** process separates sheets or signatures from a stack, roll or stream and feeds single ***Components*** to processes such as **Folding**, **Gathering**, **Collecting**, **ConventionalPrinting**, etc. In general, the **Feeding** process will be part of a combined process with processes that consume the feed of ***Components*** or ***Media***.

When used in a combined process with feed consuming process (e.g., **Gathering**), the **Feeding** process allows an arbitrary complex selection of input ***Component*** elements in any number, and in any order, as long as elements are consumed consecutively (i.e., no random access within a single input component).

When specified for a web press or web finishing device, **Feeding** describes the process of unwinding ***Media*** or ***Components*** from a roll.

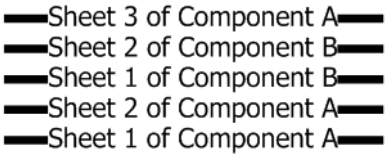
Figure 6-6: Combined Process with Feeding Process



In our example above, one input component (Component A) is a bundle component (@*BundleType* = "Stack") consisting of a collated set of three sheets, the other one (Component B) is a collated set consisting of two sheets per set. Both sets are oriented face-up (See ▸ Figure 6-7: Input Components). ▸ Figure 6-8: Output Component shows the output for the case of **Gathering**.

Figure 6-7: Input Components

Figure 6-8: Output Component



Note that, by default, none of the sheets is flipped, so surfaces of sheet 1 of ***Component*** A do not show in a different direction. To flip sheets, ***FeedingParams***/***CollatingItem***/@*Orientation* MAY be specified.

Table 6.129: Feeding – Input Resources

NAME	DESCRIPTION
Component *	Sheets or signatures to be fed to the machinery. The @*ProcessUsage* of the ***Component*** MAY be specified as any valid @*ProcessUsage* of the feed consuming process.
FeedingParams	Specific parameters to set up the **Feeding** process
Media *	Media to be fed to the feeder machinery.

Table 6.130: Feeding – Output Resources

NAME	DESCRIPTION
Component *	***Component***(s) fed to the consuming process.
Media *	***Media*** fed to the consuming process.

6.5.19 Folding

Buckle folders or knife folders are used for **Folding** sheets. One or more sheets can be folded at the same time. Web presses often provide in-line **Folding** equipment. Longitudinal **Folding** is often performed using a former, a plow folder or a belt. While jaw folding, chopper folding or drum folding equipment is used for folding the sheets that have been divided.

The **JDF** **Folding** process covers both operations done in stand-alone **Folding** machinery – typically found for processing printed materials from Sheet-Fed presses – and in-line equipment of Web presses. Creasing and/or slot perforating are sometimes necessary parts of the **Folding** operation that guarantee exact process execution. They depend on the folder used, the ***Media*** and the folding layout. These operations are specified in the **Creasing** and **Perforating** processes respectively.

Table 6.131: Folding – Input Resources

NAME	DESCRIPTION
Component	***Component*** resources, including a printed sheet or a pile of sheets, are used in the **Folding** process.
FoldingParams	Specific parameters to set up the machinery.

Table 6.132: Folding – Output Resources

NAME	DESCRIPTION
Component Modified in JDF 1.1	The process produces a ***Component***, which in most cases is a folded sheet.

6.5.20 Gathering

In the **Gathering** process, ribbons, sheets or other ***Component*** resources are accumulated on a pile that will eventually be stitched or glued in some way to create an individual ***Component***. The input ***Component*** resources MAY be output resources of a Web-Printing machine used in **Collecting** or of any machine that executes a **ConventionalPrinting** or **DigitalPrinting**

process. In sheet applications, a moving gathering channel is used to transport the pile. But no matter what the inception of the **Gathering** process, the sequence of the input components dictates the produced component. ▸ Figure 6-9: Gathering shows typical gathered piles.

Figure 6-9: Gathering

Table 6.133: Gathering – Input Resources

NAME	DESCRIPTION
Assembly ? New in JDF 1.3	Explicitly describes the sequence of the ***Component*** resources to be gathered. If ***Assembly*** is not specified, the sequence is defined by the sequence of the ***Component***. Caution: ***Assembly*** has the first on the top, whereas the ***Component*** resources are listed from bottom to top.
Component +	Variable amount of components including single sheets or folded sheets are used in the **Gathering** process. The first ***Component*** in the list lies at the bottom of the gathered pile.
DBRules * Deprecated in JDF 1.5	Database input that describes which sheets are to be gathered for a particular instance component. The schema are only in the form of human-readable text. One rule is applied for each individual component.
DBSelection ? Deprecated in JDF 1.5	Database input that describes which Sheets are to be gathered for a particular instance component.
GatheringParams	Specific parameters to set up the machinery.
IdentificationField ? Deprecated in JDF 1.2	Information about identification marks on the component. In **JDF** 1.2 and beyond, this information is defined in the ***Component*** itself.

Table 6.134: Gathering – Output Resources

NAME	DESCRIPTION
Component	Components gathered together (e.g., a pile of folded sheets).

6.5.21 Gluing

New in JDF 1.1

Gluing describes arbitrary methods of applying glue to a ***Component***.

Table 6.135: Gluing – Input Resources

NAME	DESCRIPTION
Component	This process consumes one ***Component***: the printed sheets.
GluingParams	Details of the **Gluing** process.

Table 6.136: Gluing – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced, the input ***Component*** with glue applied to it.

6.5.22 HeadBandApplication

New in JDF 1.1

Head bands are applied to the hard cover book block.

Table 6.137: HeadBandApplication – Input Resources

NAME	DESCRIPTION
Component	The prepared book block.
HeadBandApplicationParams	Specific parameters to set up the machinery.

Table 6.138: HeadBandApplication – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the hard cover block with head bands.

6.5.23 HoleMaking

A variety of machines (e.g., those responsible for stamping and drilling) can perform the **HoleMaking** process. This postpress process is needed for different binding techniques (e.g., spiral binding). One or several holes with different shapes can be made that are later on used for binding the book block together.

Table 6.139: HoleMaking – Input Resources

NAME	
Component	One ***Component*** (e.g., a printed sheet or a pile of sheets) are modified in the **HoleMaking** process.
HoleMakingParams	Specific parameters, including hole diameter and positions, used to set up the machinery.

Table 6.140: HoleMaking – Output Resources

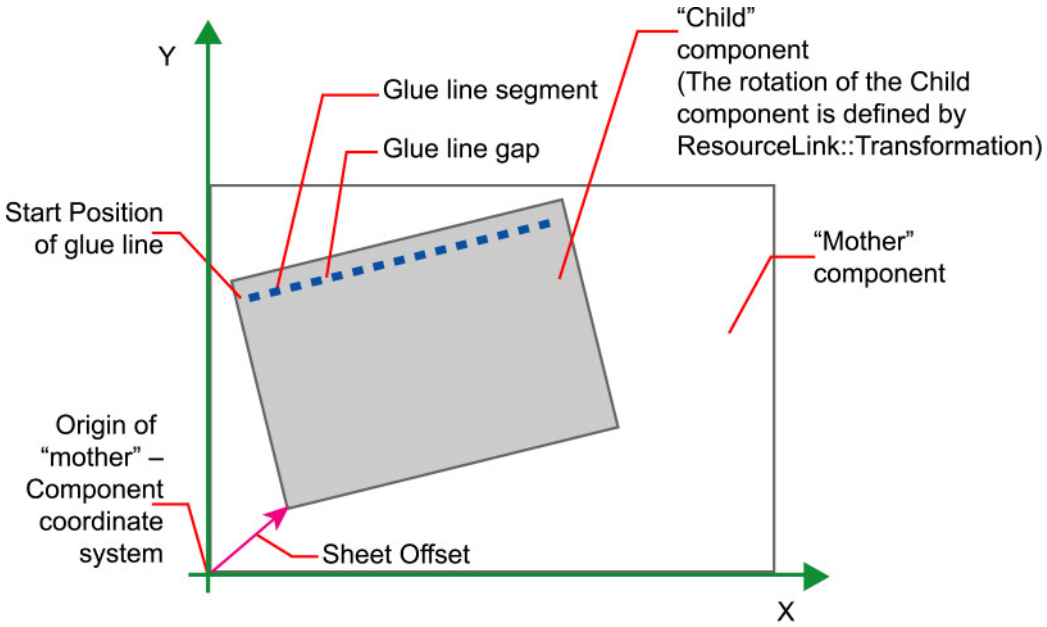
NAME	DESCRIPTION
Component	A ***Component*** with holes (e.g., a book block or a single sheet) is produced for further postpress processes.

6.5.24 Inserting

This process can be performed at several stages in postpress. The process can be used to describe the labeling of products, labeling of packages or the gluing-in of a ***Component*** (e.g., a card, sheet or CD-ROM). Two ***Component*** resources are required for the **Inserting** process: the "mother" ***Component*** and the "child" ***Component***. ***Inserting*** can be a selective process by means of inserting different "child" ***Component*** resources. Information about the placement is needed to perform

the process. Inserting multiple child components is specified as a combined process with multiple individual **Inserting** steps.

Figure 6-10: Parameters and coordinate system used for Inserting



The process coordinate system is defined as follows: The Y-axis is aligned with the binding edge and increases from the registered edge to the edge opposite the registered edge. The X-axis, meanwhile, is aligned with the registered edge. It increases from the binding edge to the edge opposite the binding edge, which is the product front edge.

Table 6.141: Inserting – Input Resources

NAME	DESCRIPTION
Component Modified in JDF 1.4	Designates where to insert the child ***Component***. **Modification note:** Starting with **JDF** 1.4, the input ***ComponentLink*** NEED NOT have *@ProcessUsage*= "Mother".
Component (Child)	The ***Component*** to be inserted in the mother ***Component***.
DBRules ? Deprecated in JDF 1.5	Database input that describes whether the child is to be inserted for a particular instance ***Component***. In this version the schema is only human readable text.
DBSelection ? Deprecated in JDF 1.5	Database input that describes whether the child is to be inserted for a particular instance ***Component***.
IdentificationField ? Deprecated in JDF 1.2	Information about identification marks on the ***Component***. In **JDF** 1.2 and beyond, this information is defined in the ***Component*** itself.
InsertingParams	Specific parameters (e.g., placement) to set up the machinery.

Table 6.142: Inserting – Output Resources

NAME	DESCRIPTION
Component	A mother ***Component*** is produced containing the inserted child ***Component***.

6.5.25 Jacketing

New in JDF 1.1

Jacketing is the process where the book is wrapped by a jacket that needs to be folded twice. As long as the book is specified and the jacket dimensions are known, there are just a few important details. If the jacketing device also creases the jacket, this can be described with a combined process of **Jacketing** and **Creasing**.

Table 6.143: Jacketing – Input Resources

NAME	DESCRIPTION
Component (Book)	The book that the jacket is wrapped around.
Component (Jacket)	The description of the jacket.
JacketingParams	Specific parameters to set up the machinery.

Table 6.144: Jacketing – Output Resources

NAME	DESCRIPTION
Component	The jacketed book.

6.5.26 Labeling

New in JDF 1.1

A label can be attached to a bundle. The label can contain information on the addressee, the product, the product quantities, etc., which can be different for each bundle.

Table 6.145: Labeling – Input Resources

NAME	DESCRIPTION
Component	The **Labeling** process labels one ***Component*** with a set of labels.
Component (Label) ?	The label to be attached to the ***Component***.
LabelingParams	Specific parameters to set up the machinery.

Table 6.146: Labeling – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the labeled ***Component***.

6.5.27 Laminating

In the **Laminating** process, a plastic film is bonded to one or both sides of a ***Component*** resource's media, and adhered under pressure with either a thermal setting or pressure sensitive adhesive.

Table 6.147: Laminating – Input Resources

NAME	DESCRIPTION
Component	A ***Component*** is REQUIRED for **Laminating**.
LaminatingParams	Specific parameters to set up the machinery.
Media?	The laminating foil material.

Table 6.148: Laminating – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the laminated component.

6.5.28 LongitudinalRibbonOperations

Deprecated in JDF 1.1

In version 1.1 of **JDF** and beyond, in-line finishing is described using the "standard" finishing processes (e.g., **Creasing**, **Cutting**, **Folding**) or in a combined process node with **ConventionalPrinting**. See ▸ Section N.5.15 LongitudinalRibbonOperations for details of this deprecated process.

6.5.29 Numbering

Deprecated in JDF 1.5

Starting with **JDF** 1.5, use **LayoutElementProduction**. For details of the deprecated ***Numbering*** process, see ▸ Section N.5.16 Numbering.

6.5.30 Palletizing

New in JDF 1.1

Bundles, stacks, piles or boxes can be loaded onto a pallet.

Table 6.149: Palletizing – Input Resources

NAME	DESCRIPTION
Component +Modified in JDF 1.4	The **Palletizing** process describes placing the bundle that is represented by the ***Component*** onto a pallet. If more than one ***Component*** is specified, a ***PalletizingParams***/***Bundle*** resource SHALL also be specified. **Modification note:** Starting with **JDF** 1.4, ***Component*** can occur more than once.
Pallet	The pallet.
PalletizingParams	Specific parameters to set up the machinery.

Table 6.150: Palletizing – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced. It represents the loaded pallet. If more than one input ***Component*** is supplied, a ***Component***/***Bundle*** resource SHALL also be supplied in the output ***Component***

6.5.31 Perforating

New in JDF 1.1

Perforating describes any process where a ***Component*** is perforated. **Perforating** includes production perforation applied as a preparation for **Folding**.

Table 6.151: Perforating – Input Resources

NAME	DESCRIPTION
Component	This process consumes one ***Component***: the printed sheets.
PerforatingParams	Details of the **Perforating** process.

Table 6.152: Perforating – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced.

6.5.32 PlasticCombBinding

In the **PlasticCombBinding** process, a plastic insert wraps through prepunched holes in the substrate. Most often, these holes are rectangular and elongated. After the plastic comb is opened with a special tool, the prepunched block of sheets – often together with a top and button cover – is inserted onto the "teeth" of the plastic comb. When released from the machine, the teeth return to their original cylindrical positions with the points tucked into the backside of the spine area. Special machinery can be used to reopen the plastic comb binding.

Table 6.153: PlasticCombBinding – Input Resources

NAME	DESCRIPTION
Component	The operation requires one component: the pile of sheets often including a top and button cover.
PlasticCombBindingParams	Specific parameters to set up the machinery.

Table 6.154: PlasticCombBinding – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the plastic-comb-bound component forming an item such as a calendar.

6.5.33 PrintRolling

New in JDF 1.2

The single products like sheets, signatures or partial products are rolled onto a roll stand. The roll is the output component of the process and is used to store the products. The single components of a roll are used as input component of a consuming process (e.g., **Collecting**, **Gathering** or **Inserting**). See ‣ Figure 6-11: Print Roll

Figure 6-11: Print Roll

Table 6.155: PrintRolling – Input Resources

NAME	DESCRIPTION
Component ?	***Component*** to be rolled.
PrintRollingParams ?	Print rolling parameters.
RollStand ?	Roll stand to store the component(s) as rolls.

Table 6.156: PrintRolling – Output Resources

NAME	DESCRIPTION
Component ?	The print roll.

6.5.34 RingBinding

In this process, prepunched sheets are placed in a ring binder. Ring binders have different numbers of rings that are fixed to a metal backbone. In most cases, two, three or four metal rings hold the sheets together as long as the binding is closed. Depending on the amount of sheets to be bound together, ring binders of different thickness SHALL be used.

Table 6.157: RingBinding – Input Resources

NAME	DESCRIPTION
Component Modified in JDF 1.4	The operation requires one component: the pile of prepunched sheets to be inserted into the ring binder. **Modification note:** Starting with **JDF** 1.4, the input ***ComponentLink*** NEED NOT have *@ProcessUsage*= "BookBlock".
Component (RingBinder) ?	The empty ring binder that might have been printed, for example, before it is used during the **RingBinding** process.
RingBindingParams	Specific parameters to set up the process/machinery.

Table 6.158: RingBinding – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the ring-bound component forming an item such as a calendar.

6.5.35 SaddleStitching

Deprecated in JDF 1.1

SaddleStitching has been replaced by **Stitching** in **JDF** 1.1. See ▸ Section N.5.17 SaddleStitching for details of this deprecated process.

6.5.36 ShapeCutting

New in JDF 1.1

The **ShapeCutting** process can be performed using tools such as hollow form punching, perforating or die-cutting equipment.

Table 6.159: ShapeCutting – Input Resources

NAME	DESCRIPTION
Component	This process consumes one ***Component***: The sheets to be cut.
ShapeCuttingParams ? Modified in JDF 1.3	Details of the **ShapeCutting** process.
Tool * Modified in JDF 1.3	The set of tools (die, counter, blankers, strippers, etc.).

Table 6.160: ShapeCutting – Output Resources

NAME	DESCRIPTION
Component +Modified in JDF 1.3	One or more ***Component*** resources are produced by the **ShapeCutting** process.

6.5.37 Shrinking

New in JDF 1.1

The **Shrinking** process shrinks the shrink-wrap that is wrapped around a bundle. Shrink-wrap foil SHALL be treated in order to shrink.

Note: **Shrinking** does NOT include the wrapping of the ***Component*** with foil. The actual wrapping is described by the **Wrapping** process. See ▸ Section 6.5.52 Wrapping

Table 6.161: Shrinking – Input Resources

NAME	DESCRIPTION
Component	The *Bundle* including the shrink-wrap media is represented by this *Component*.
ShrinkingParams	Specific parameters to set up the machinery.

Table 6.162: Shrinking – Output Resources

NAME	DESCRIPTION
Component	One *Component* is produced: the bundle including the shrunk shrink-wrap media.

6.5.38 SideSewing

Deprecated in JDF 1.1

Replaced by **ThreadSewing**. See ▸ Section N.5.18 SideSewing for details of this deprecated process.

6.5.39 SpinePreparation

New in JDF 1.1

The **SpinePreparation** process describes the preparation of the spine of book blocks for hard and soft cover book production (e.g., milling and notching).

Table 6.163: SpinePreparation – Input Resources

NAME	DESCRIPTION
Component	The raw book block.
SpinePreparationParams	Specific parameters to set up the machinery.

Table 6.164: SpinePreparation – Output Resources

NAME	DESCRIPTION
Component	The book block with a processed spine.

6.5.40 SpineTaping

New in JDF 1.1

SpineTaping describes the process of applying a tape strip to the spine of a book block. It also describes the process of applying kraft paper to a hard cover book block.

Table 6.165: SpineTaping – Input Resources

NAME	DESCRIPTION
Component	The book block that the spine is taped to.
SpineTapingParams	Specific parameters to set up the machinery.

Table 6.166: SpineTaping – Output Resources

NAME	DESCRIPTION
Component	The book block with the spine.

6.5.41 Stacking

New in JDF 1.1

The **Stacking** process collects ***PhysicalResource***s ***Component***(products) and produces a pile, stack or bundle for delivery. In a standard production each bundle consists of the same amount of identical products, possibly followed by one or more odd-count bundles. In a production with variable data (e.g., newspaper dispatch, demographic production or individual addressed products), each bundle has a variable amount of products, and, in the worst case, each product can be different from the others. The input components are single products; the output components are stacks of this product.

Table 6.167: Stacking – Input Resources

NAME	DESCRIPTION
Component	The **Stacking** process consumes one ***Component*** and stacks it onto a stack.
StackingParams	Specific parameters to set up the machinery.

Table 6.168: Stacking – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the stack of input components.

6.5.42 StaticBlocking

New in JDF 1.4

The **StaticBlocking** process puts an electrical charge on a stack in order to hold it together for shipping.

Table 6.169: StaticBlocking – Input Resources

NAME	DESCRIPTION
Component	The **StaticBlocking** process puts an electrical charge on the specified ***Component***.
StaticBlockingParams	Specific parameters for the electrical charging.

Table 6.170: StaticBlocking – Output Resources

NAME	DESCRIPTION
Component	The resulting electrically charged ***Component***.

6.5.43 Stitching

Gathered or collected sheets or signatures are stitched together with a cover.

Table 6.171: Stitching – Input Resources

NAME	DESCRIPTION
Component	A ***Component*** is REQUIRED that represents the pile of gathered or collected sheets, including the cover.
StitchingParams	Specific parameters to set up the machinery.

Table 6.172: Stitching – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the gathered or collected sheets including the cover stitched together.

Example 6.10: Stitching: Combined Process

Components containing staples of different characteristics like shape, width, etc. are defined by a combined process.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="CombinedStitch"
       JobID="Stitching special" JobPartID="ID123" Type="Combined"
       Types="Stitching Stitching" Status="Ready" Version="1.4">
    <ResourcePool>
       <StitchingParams Class="Parameter" ID="Stitch1" NumberOfStitches="2"
              StapleShape="Butted" Status="Available" StitchPositions="100 700"
              StitchWidth="28.3" WireBrand="Steel" WireGauge="2.3"/>
       <StitchingParams Class="Parameter" ID="Stitch2" NumberOfStitches="2"
              StapleShape="Eyelet" Status="Available" StitchPositions="300 500"
              StitchWidth="42.5" WireBrand="Steel" WireGauge="2.3"/>
       <Component Class="Quantity" ID="Comp1" Status="Available"
              ComponentType="Sheet"/>
       <Component Class="Quantity" ID="Comp2" Status="Unavailable"
              ComponentType="Sheet"/>
    </ResourcePool>
    <ResourceLinkPool>
       <StitchingParamsLink CombinedProcessIndex="0" Usage="Input" rRef="Stitch1"/>
       <StitchingParamsLink CombinedProcessIndex="1" Usage="Input" rRef="Stitch2"/>
       <ComponentLink Usage="Input" rRef="Comp1"/>
       <ComponentLink Usage="Output" rRef="Comp2"/>
    </ResourceLinkPool>
</JDF>
```

6.5.44 Strapping

New in JDF 1.1

A bundle MAY be strapped. There are different kinds of strapping (e.g., single (one strap around the bundle), double (two parallel straps) and cross (two crossed straps)).

Table 6.173: Strapping – Input Resources

NAME	DESCRIPTION
Component	The **Strapping** process puts straps around a bundle that is represented by a ***Component***.
Strap ?	The straps used.
StrappingParams	Specific parameters to set up the machinery.

Table 6.174: Strapping – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the strapped ***Component***.

6.5.45 StripBinding

New in JDF 1.1

Hard plastic strips are held together by plastic pins, which in turn are bound to the strips with heat. The sheets to be bound SHALL be prepunched so that the top strip with multiple pins fits through the assembled material. It is then connected to the bottom strip with matching holes for the pins. The binding edge is often compressed in a special machine before the excess pin length is cut off. The backstrip is permanently fixed with plastic clamping bars and cannot be removed without a special tool.

Table 6.175: StripBinding – Input Resources

NAME	DESCRIPTION
Component	The operation requires one component: the block of sheets to be bound.
StripBindingParams	Specific parameters to set up the machinery.

Table 6.176: StripBinding – Output Resources

NAME	DESCRIPTION
Component	One *Component* is produced: the strip-bound component forming an item such as a book.

6.5.46 ThreadSealing

New in JDF 1.1

Similar to Smythe sewing, **ThreadSealing** involves sewing the signatures at the spine of the book. After the signatures are sewn, they are gathered and run through the perfect binder. The perfect binder however does not grind the spine. Instead the binding adhesive (which attaches the cover) envelops the thread that holds the book together. This special thread holds to the glue to create a sewn book with most of the same properties as Smythe sewing.

Table 6.177: ThreadSealing – Input Resources

NAME	DESCRIPTION
Component	This process consumes one *Component*: the printed sheets.
ThreadSealingParams	Details of the **ThreadSealing** process.

Table 6.178: ThreadSealing – Output Resources

NAME	DESCRIPTION
Component	One *Component* is produced.

6.5.47 ThreadSewing

This process might include a gluing application, which would be used principally between the first and the second sheet or the last and the last sheet but one. **Gluing** might also be necessary if different types of paper are used.

Table 6.179: ThreadSewing – Input Resources

NAME	DESCRIPTION
Component	The operation requires one component: the gathered sheets.
ThreadSewingParams	Specific parameters to set up the machinery.

Table 6.180: ThreadSewing – Output Resources

NAME	DESCRIPTION
Component	One *Component* is produced: the thread-sewn components forming an item such as a raw book block.

6.5.48 Trimming

The **Trimming** process is performed to adjust a book block or sheet to its final size. In most cases, it follows a block joining process, and the process is often executed as an in-line operation of a production chain. For example, the binding station might deliver the book blocks to the trimmer. A combined process in the trimming machinery would then execute a cut at the front, head and tail in a cycle of two operations. Closed edges of folded signatures would then be opened while the book block is trimmed to its predetermined dimensions.

The separation of N-up multiple products is specified with a **Cutting** process in front of a **Trimming** process.

Table 6.181: Trimming – Input Resources

NAME	DESCRIPTION
Component Modified in JDF 1.2	The bound book block or sheet that will be trimmed.
TrimmingParams	Specific parameters (e.g., trim size) to set up the machinery.

Table 6.182: Trimming – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the trimmed component.

6.5.49 WebInlineFinishing

New in JDF 1.3

The **WebInlineFinishing** process combines all additional information about inline finishing functionality in connection with Web printing. In order to describe the **WebInlineFinishing** functionality fully, it is necessary to combine additional processes like **Stitching**, **Trimming**, **Gluing**, etc.

Table 6.183: WebInlineFinishing – Input Resources

NAME	DESCRIPTION
Assembly ?	In context of newspaper printing, ***Assembly*** describes how the newspaper job is sub-divided in physical sections and bound together.
Component	Printed webs or ribbons, which will be processed by the **WebInlineFinishing** process
ProductionPath ?	***ProductionPath*** describes the paper path that is used through the press and describes exactly one particular product which has to be produced.
StrippingParams ?	Defines how the surfaces of the bindery signatures of a single job or jobs are placed onto the web(s) or sheet(s) This information MAY be used for counting the amount of components produced.
WebInlineFinishingParams ?	Additional parameters for production are described by ***WebInlineFinishingParams***

Table 6.184: WebInlineFinishing – Output Resources

NAME	DESCRIPTION
Component	Describes the finished printed ***Component*** out of web inline finishing equipment. This could be printed and / or folded sheets or rolls. With one production run, it is possible to produce more than one product / order.***Component*** MAY be partitioned by @*WebProduct*

6.5.50 Winding

New in JDF 1.5

The **Winding** process describes the winding of continuous media or processed components onto a core. The setup is defined in ***WindingParams***. The final orientation of the labels on the output roll is specified in ***Component***/@*WindingResult*.

Table 6.185: Winding – Input Resources (Sheet 1 of 2)

NAME	DESCRIPTION
Component	Ribbon or web to be wound. Exactly one of ***Media*** or ***Component*** SHALL be specified.

Table 6.185: Winding – Input Resources (Sheet 2 of 2)

NAME	DESCRIPTION
Media ?	Unprocessed ***Media*** MAY be wound. Exactly one of ***Media*** or ***Component*** SHALL be specified.
Media (Core) ?	Core that the input ***Component*** is wound around.
WindingParams ?	Setup parameters of the winding process.

Table 6.186: Winding – Output Resources

NAME	DESCRIPTION
Component	The roll including the core and the wound products. ***Component***/@*WindingResult* SHALL be evaluated to determine the winding orientation.

6.5.51 WireCombBinding

In **WireCombBinding** metal wire, wire with plastic or pure plastic is used to fasten prepunched sheets of paper, cardboard or other such materials. The wire – often formed as a double wire – is inserted into the holes, then curled to create a circular enclosure.

Table 6.187: WireCombBinding – Input Resources

NAME	DESCRIPTION
Component	The operation requires one component: the pile of preprinted sheets often including a front and back cover.
WireCombBindingParams	Specific parameters to set up the machinery.

Table 6.188: WireCombBinding – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the wire-comb bound component forming an item such as a calendar.

6.5.52 Wrapping

New in JDF 1.1

Single products, bundles or pallets can be wrapped by film or paper.

Table 6.189: Wrapping – Input Resources

NAME	DESCRIPTION
Component	The **Wrapping** process wraps a bundle that is represented by a ***Component***.
Component (Wrapper) ? New in JDF 1.6	If the wrapping material is preprinted, then ***Component*** (Wrapper) represents the wrapping material. Rubber bands and other non-printed material SHOULD be represented as ***MiscConsumable***.
Media ? Deprecated in JDF 1.6	The wrapping material. **Note:** From version 1.6 use a ***Component***(Wrapper) and/or a ***MiscConsumable***.
MiscConsumable (Wrapper) ? New in JDF 1.6	Additional details of the wrapper material. Non-printed material SHOULD be represented as ***MiscConsumable***. ***MiscConsumable***(Wrapper) SHALL NOT be present if ***Component*** (Wrapper) is provided.
WrappingParams	Specific parameters to set up the machinery.

Table 6.190: Wrapping – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the wrapped ***Component***.

6.6 Postpress Processes Structure

6.6.1 Block Production

This subcategory of the postpress processes merges together all the processes for making a book block. First the block is compiled using the **Collecting** and **Gathering** processes. After that, it is combined using one or several of the block joining processes, including **CoverApplication**, **SpineTaping**, **Stitching** and **ThreadSewing**. The workflow using these processes eventually produces a ***Component*** that can be trimmed.

6.6.1.1 Block Compiling

The **Gathering** and **Collecting** processes are used to position unfolded sheets and/or folded sheets in a planned order. These operations set a fixed page sequence in preparation for three-side trimming and binding. Block compiling includes:

- **Collecting**
- **Gathering**
- **PrintRolling**
- **Feeding**
- **Winding**

6.6.1.2 Block Joining

The block joining processes can be grouped into two major subcategories: conventional binding methods, which includes the processes of **Stitching**, **CoverApplication**, **SpinePreparation**, **SpineTaping**, **ThreadSealing** and **ThreadSewing**; and single-leaf binding methods, which are listed in ▸ Section 6.6.1.3.1 Single-Leaf Binding Methods. Together they form a subcategory of block-production processes. All of these processes, which are known as block joining processes, unite sheets and/or folded sheets lying loose on top of each other.

There are numerous possible binding methods. The most prominent ones are modeled by the processes described in the following sections. Many of them can be part of a combined production chain being performed as in-line tasks. Block joining includes:

- **CoverApplication**
- **EndSheetGluing**
- **Gluing**
- **SpinePreparation**
- **SpineTaping**
- **Stitching**
- **ThreadSewing**

6.6.1.3 Binding Methods

6.6.1.3.1 Single-Leaf Binding Methods

Besides the conventional binding methods, there is a multifaceted group of binding methods for single-leaf bindings. This group can again be subdivided into two subtypes: loose-leaf binding and mechanical binding, each of which is described in the sections that follow.

6.6.1.3.2 Loose-Leaf Binding Method

This binding techniques allow contents to be changed, inserted or removed at will. There are two essential groups of loose-leaf binding systems: those that require the paper to be punched or drilled and those that do not. The **RingBinding** method, described in the next section, is the most prominent binding in the loose-leaf binding category. Loose-leaf binding methods include:

- **RingBinding**

6.6.1.3.3 Mechanical Binding Methods

Single leafs are fastened into what is essentially a permanent system that is not meant to be reopened. However, special machinery can be used to reopen some of the mechanical binding systems described below.

In mechanical binding, printing and folding can be done in a conventional manner. The gathered sheets, however, often require the back to be trimmed, as well as the other three sides. Mechanical bindings are often used for short-run jobs

such as ones that have been printed digitally. The most prominent mechanical binding processes are described in the sections that follow. Mechanical binding methods include:

- **ChannelBinding**
- **CoilBinding**
- **PlasticCombBinding**
- **RingBinding**
- **StripBinding**
- **WireCombBinding**

6.6.2 HoleMaking

- **HoleMaking**

6.6.3 Laminating

- **Laminating**

6.6.4 Numbering

- **Numbering**

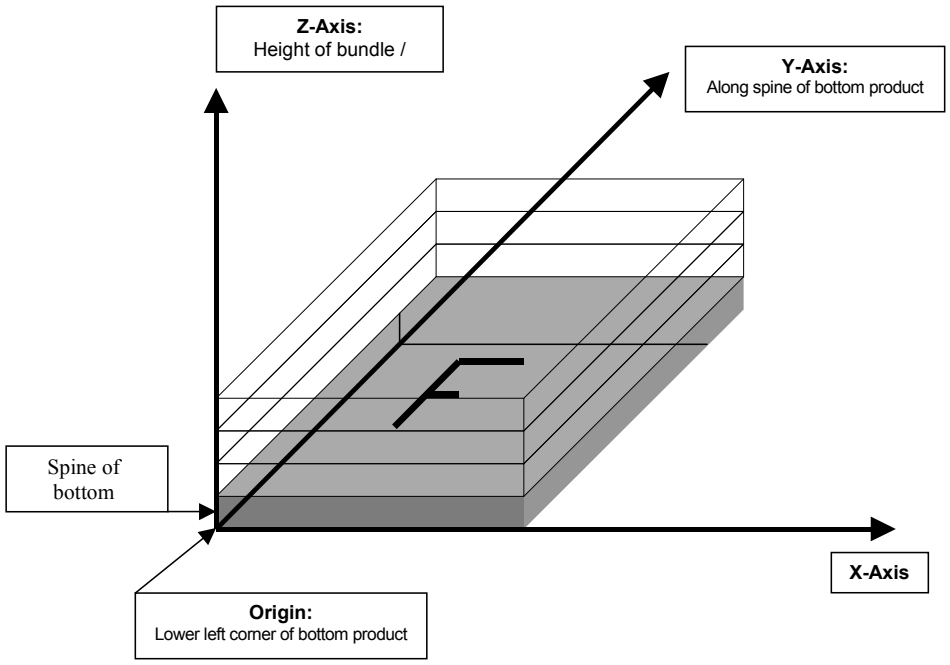
6.6.5 Packaging Processes

The individual processes defined in this section replace the deprecated **Packing** process. Packaging processes include:

- **BoxPacking**
- **Bundling**
- **Labeling**
- **Palletizing**
- **Shrinking**
- **Stacking**
- **Strapping**
- **Wrapping**

Each of these processes share a common coordinate system as depicted below:

Figure 6-12: Packaging Process Coordinate System



6.6.6 Processes in Hardcover Book Production

The following processes refer to the production of hard cover books. Several processes are needed to produce a hardcover book. Some of them are essential and others are optional. The processes are:

CaseMaking:	Production of hard cover book cases.
BlockPreparation:	The optional hardcover design elements (e.g., rounding and backing, ribbon band, headband, side gluing and tightbacking) are described in this process. Application of kraft paper to the book block is described in the **SpineTaping** process.
CasingIn:	In this process, the case and the prepared book block are brought together.
Jacketing:	In the **Jacketing** process, the jacket is wrapped around the hardcover book.

Processes in hardcover book production include:

- **BlockPreparation**
- **CaseMaking**
- **CasingIn**
- **Collecting**
- **Gluing**
- **HeadBandApplication**
- **Jacketing**
- **SpinePreparation**
- **SpineTaping**
- **ThreadSealing**
- **ThreadSewing**

6.6.7 Sheet Processes

Many printing processes produce sheets that are processed further in finishing operations. The web processes presented in the preceding sections result in sheets that are treated in much the same way as sheets produced by Sheet-Fed printing presses. The following processes describe these sheet finishing operations. Sheet processes include:

- **Creasing**
- **Cutting**
- **Embossing**
- **Feeding**
- **Folding**
- **Gathering**
- **Gluing**
- **Palletizing**
- **Perforating**
- **PrintRolling**
- **ShapeCutting**
- **ThreadSealing**
- **Winding**

6.6.8 Tip-on/in

The following processes, ***EndSheetGluing***, **Inserting**, are part of the postpress operations. They can be grouped together as the tip-on/in processes. Both processes can be performed by hand, tip-on/in machine or by a press. Tip-on/in includes:

- **EndSheetGluing**
- **Inserting**

6.6.9 Trimming

- **Trimming**.

6.6.10 Web Processes

This subchapter of the postpress processes is dedicated to web and ribbon operations (i.e., operations that require a web or a ribbon to execute). In essence, a ribbon is a web that has been slit or cross-cut. More specifically, a web is a continuous strip of ***Media*** to be used for printing (e.g., paper or foil). This substrate is called "Web" while it is threaded through the printing machinery, but once it has run through the **Cutting** process and been slit, the web no longer exists. In its place are ribbons or sheets.

A ribbon, then, is the part of the web that enters the folder. If the web is never slit, however, the web and the ribbon are identical. Slitting and salvage-trim operations on a web can result in one or more ribbons. A ribbon can be further subdivided after it has been slit. After the **Cutting** process, sheets are treated further. The **Gathering** process and **Folding** process also handle web and ribbon applications.

Product Intent

As was described in ▸ Section 4.1.1 Product Intent Constructs, ***Intent Resource***s are designed to narrow down the available options when defining a **JDF** job. Many of the elements in ***Intent Resource***s are OPTIONAL. If an OPTIONAL element of an ***Intent Resource*** is omitted and no additional information is specified in the description, the value defaults to "don't care". If an entire ***Intent Resource*** that specifies a given product feature is omitted, then that feature is not requested. For instance, if a product intent description has no ***ResourceLink*** to ***BindingIntent***, then no binding is requested. The characteristics of the product that are not specified through the use of ***Intent Resource***s will be selected by the system that processes the ***Intent Resource***s. The system that processes the product intent data in a **JDF** job ticket MAY insert the details of its selection into the **JDF** data for the job. See ▸ Section 1.5.2.1 Conformance Requirements for Support of Attributes and Attribute Values for more information on the handling and processing of systems-specified default values.

All ***Intent Resource***s share a set of subelements that allow a 'Request for Quote' to describe a range of acceptable values for various aspects of the product. These elements, taken together, allow an administrator to provide a specific value for the quote. The section below () describes these elements.

7.0.1 Product Intent Descriptions

Product intent is also described as a **JDF** node. The following table defines the list of **JDF** ***Intent Resource***s used to describe product intent.

Table 7.1: Product Intent – Input Resources (Sheet 1 of 2)

NAME	DESCRIPTION
ArtDeliveryIntent ?	This resource specifies the prepress art delivery intent for a **JDF** job.
BindingIntent ?	This resource specifies the binding intent for a **JDF** job.
ColorIntent ?	This resource specifies the type of ink to be used for a **JDF** job.
Component *	Components that are partial products of the product described by this node. If input ***Component*** resources are specified, at least one of ***BindingIntent*** or ***InsertingIntent*** is REQUIRED.
DeliveryIntent ?	Summarizes the options that describe pickup or delivery time and location of the ***PhysicalResource***s of a job.
EmbossingIntent ?	This resource specifies the embossing and/or foil stamping intent for a **JDF** job.
FoldingIntent ?	This resource specifies the fold intent for a **JDF** job using information that identifies the number of folds, the height and width of the folds, and the folding catalog number.
HoleMakingIntent ?	This resource specifies the holemaking intent for a **JDF** job.
InsertingIntent ?	This resource specifies the placing or inserting of one component within another, using information that identifies page location, position and attachment method.
LaminatingIntent ?	This resource specifies the laminating intent for a **JDF** job using information that identifies whether or not the product is laminated.
LayoutIntent ?	This resource records the size of the finished pages for the product component.
MediaIntent ?	This resource describes the media to be used for the product component.
NumberingIntent ?	This resource describes the parameters of stamping or applying variable marks in order to produce unique components, for items such as lottery notes or currency.
PackingIntent ?	This resource specifies the packaging intent for a **JDF** job, using information that identifies the type of package, the wrapping used and the shape of the package.
ProductionIntent ?	This resource specifies the manufacturing intent and considerations for a **JDF** job using information that identifies the desired result or specified manufacturing path.

Table 7.1: Product Intent – Input Resources (Sheet 2 of 2)

NAME	DESCRIPTION
ProofingIntent ?	This resource specifies the prepress proofing intent for a **JDF** job, using information that identifies the type, quality, brand name and overlay of the proof.
PublishingIntent ?	This resource specifies publishing metadata that are of general interest for prepress, press and postpress. The data include details on the general structure of product being published.
ScreeningIntent ?	This resource specifies the screening intent parameters desired for a **JDF** job.
ShapeCuttingIntent ?	This resource specifies form and line cutting for a **JDF** job.
SizeIntent ? Deprecated in JDF 1.2	This resource records the size of the finished pages for the product component. ***SizeIntent*** has been deprecated in **JDF** 1.1. All contents have been moved to ***LayoutIntent***.

Table 7.2: Product Intent – Output Resources

NAME	DESCRIPTION
Component +	Resource representation of the output this product intent node. Multiple ***Component*** resources SHALL be specified in a root node that contains a ***DeliveryIntent*** that references multiple ***Component*** resources as delivery end products.

7.1 Intent Properties Template

Each of the following sections begins with a brief narrative description of the resource. Following that is a list containing details about the properties of the resource, as shown below. The first item in the list provides the class of the resource, which, in this section is always "Intent". For more information on resource class, see ▸ Section 3.8.5 Resource Classes. A template of this list is shown below.

After the list describing the resource properties, each section contains tables that outline the structure of each resource and, when applicable, the abstract or subelement information that pertains to the resource structure. The first column contains the name of the attribute or element. A template of these tables is also provided below.

Note: For the resource properties template below, the *italicized* text describes the actual text that would be in its place in an actual resource definition.

Note: For the resource structure template table below: *Cardinality* in the Name column of the resource structure template table refers to a cardinality symbol, which is either empty or consists of a symbol, such as "?". Examples described by the Name column include: "***Ink*** *" and "***FileSpec*** ("DeviceLinkProfile") ?". For further details, see ▸ Section 1.3.5 Specification of Cardinality.

Intent Properties Template

Resource Class: Defines the Resource Class.

Process Resource Pairing: List of process resources with which an intent resource is generally identified with. In practice, the process resources will contain the data with which the customer's intent is fulfilled in production and distribution of the product. This is a list of the primary resources and not a complete list.

Example Partition: List of recommended partition keys: For a complete list of partition keys, see the description of @*PartIDKeys* in ▸ Table 3.24 Partitionable Resource Element.

Note: Resources may be partitioned by keys that are not specified in this list.

Table 7.3: Template for Intent Resources (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Attribute-Name *Cardinality*	*Attribute-data-type*	*Information about the attribute.*
Element-Name *Cardinality*	element	*Information about the element.* Note: The "element" data type means that the specified element SHALL be an in-line subelement within the resource.

Table 7.3: Template for Intent Resources (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Element-Name Cardinality	refelement	*Information about the element* Note: The "refelement" data type means that the specified element is based on other atomic resources or resource elements. The specified element SHALL be either an in-line element or an instance of a resourceRef element (see ▸ Section 3.10.2 ResourceRef – Element for Inter-Resource Linking and refelement). In case of a ***ResourceRef*** element, a "Ref" SHALL be appended to the name specified in the table column entitled "Name".

Intent Resources contain subelements that allow spans of values to be specified. These subelements also provide mechanisms to select a set of values from the provided range and map them to a set of quotes. These subelements are called span elements. The span element to use is determined by the data type of the values to be recorded. Span elements are defined to facilitate negotiation between buyer and provider.

7.1.1 Abstract Span Element

Span elements of ***Intent Resource***s have a common set of attributes that define the priority, data type and requested identity of the element. These common attributes are described in ▸ Table 7.4 Abstract Span Element. In addition, abstract span elements have at least four attributes that define the data type dependent aspects of the span. The data type of these values depends on the data type of the span and is defined in the following sections:

@*Actual* –The intended value agreed to by the producer of the product.

@*OfferRange* – A proposed range of equivalent values in cost that are defined by the producer of the product.

@*Preferred* – A preferred value defined by the recipient of the product.

@*Range* – A proposed range of values defined by the recipient of the product.

Table 7.4: Abstract Span Element

NAME	DATA TYPE	DESCRIPTION
DataType	enumeration	Describes the data type of the span element within an ***Intent Resource***. This attribute is provided for applications that do not have access to schema validation. **Allowed values are:** DurationSpan, EnumerationSpan, IntegerSpan, NameSpan, NumberSpan, OptionSpan, ShapeSpan, StringSpan, TimeSpan, XYPairSpan
Priority ? Deprecated in JDF 1.2	enumeration	Indicates the importance of the specific intent. **Allowed values are:** None Suggested – The customer will accept a value of @*Actual* that is different than the value of @*Preferred* or outside of @*Range*. Required – The customer expects the @*Actual* to be equal to @*Preferred* or within @*Range*. Note that the attribute @*Preferred* is available in the data types which inherit from this abstract type. **Deprecation note:** Starting with **JDF** 1.2, use @*SettingsPolicy*.

7.1.2 Span Elements

The Data Type column of tables for ***Intent Resource***s (below) can contain the same data types as non-***Intent Resource***s (namely data types defined in the ▸ Section 1.6 Data Structures) as well as span elements that are listed in the ▸ Table 7.5 List of Span Elements. In ***Intent Resource*** tables, *XXXSpan* elements are treated as attribute-like data types even though span elements are technically XML elements because the semantic usage of the span elements is equivalent to the usage of attributes in process resources.

Each span element contains attributes or subelements listed in ▸ Table 7.4 Abstract Span Elementand in the pertinent span element listed in ▸ Table 7.5 List of Span Elements.

Table 7.5: List of Span Elements

NAME	PAGE	DESCRIPTION
DurationSpan New in JDF 1.1	page 304	Describes a set of duration values.
EnumerationSpan	page 304	Describes a set of enumeration values.
IntegerSpan	page 305	Describes a numerical range of integer values.
NameSpan	page 305	Describes a set of NMTOKEN values.
NumberSpan	page 306	Describes a numerical range of values.
OptionSpan	page 306	Describes an intent in which the principal information is that a specific option is requested.
ShapeSpan New in JDF 1.1	page 306	Describes a set of shape values.
StringSpan	page 307	Describes a set of string values.
TimeSpan	page 307	Describes a set of dateTime values.
XYPairSpan	page 307	Describes a set of XYPair values.

7.1.2.1 DurationSpan

New in JDF 1.1

This span subelement is used to describe a selection of instances in time. It inherits from the abstract span element described in ▸ Section 7.1.1 Abstract Span Element.

Table 7.6: DurationSpan Element

NAME	DATA TYPE	DESCRIPTION
Actual ?	duration	The actual value selected for the quote.
OfferRange ? New in JDF 1.3	DurationRange	Provides an offered range of time durations. If not specified, it defaults to the value of @*Actual*.
Preferred ?	duration	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of @*Preferred* SHALL fall within the range of values specified in @*Range*.
Range ?	DurationRange	Provides a valid range of time durations. If not specified, it defaults to the value of @*Preferred*.

7.1.2.2 EnumerationSpan

This span subelement is used to describe ranges of enumerative values. It inherits from the abstract span element described in ▸ Section 7.1.1 Abstract Span Element. It is identical to the ***NameSpan*** element except for the fact that it describes a closed list of enumeration values.

Table 7.7: EnumerationSpan Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Actual ?	enumeration	The actual value selected for the quote.
OfferRange ? New in JDF 1.3	enumerations	Provides an offered range of values. **Default value is from:** @*Actual*.
Preferred ?	enumeration	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of @*Preferred* SHALL fall within the range of values specified in @*Range*.

Table 7.7: EnumerationSpan Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Range ?	enumerations	Provides a set of discreet enumeration values. **Default value is from:** @*Preferred*.

Example 7.1: EnumerationSpan

```
<BindingIntent Class="Intent" ID="BI1" Status="Available">
    <BindingType DataType="EnumerationSpan" Actual="Ring"/>
    <RingBinding>
        <HoleType DataType="EnumerationSpan" Range="R4m-DIN-A5 R6m-DIN-A5">
            <Comment Name="R4m-DIN-A5">
                4 equidistant holes on each side of a hexagonal piece of paper
            </Comment>
            <Comment Name="R6m-DIN-A5">
                6 equidistant holes on each side of a hexagonal piece of paper
            </Comment>
        </HoleType>
    </RingBinding>
</BindingIntent>
```

7.1.2.3 IntegerSpan

This span subelement is used to describe ranges of integer values. It inherits from the abstract span element described in ▸ Section 7.1.1 Abstract Span Element.

Table 7.8: IntegerSpan Element

NAME	DATA TYPE	DESCRIPTION
Actual ?	integer	The actual value selected for the quote.
OfferRange ? New in JDF 1.3	IntegerRangeList	Provides either a set of discreet values, a range of values or a combination of the two that comprise all offered values for the span. **Default value is from:** @*Actual*.
Preferred ?	integer	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of @*Preferred* SHALL fall within the range of values specified in @*Range*.
Range ?	IntegerRangeList	Provides either a set of discreet values, a range of values or a combination of the two that comprise all allowed values for the span. **Default value is from:** @*Preferred*.

7.1.2.4 NameSpan

This span subelement is used to describe name ranges. It inherits from the abstract span element described in ▸ Section 7.1.1 Abstract Span Element. It is identical to the ***EnumerationSpan*** element except for the fact that it describes an extensible list of NMTOKEN values.

Table 7.9: NameSpan Element

NAME	DATA TYPE	DESCRIPTION
Actual ?	NMTOKEN	The actual value selected for the quote.
OfferRange ? New in JDF 1.3	NMTOKENS	Provides a set of discreet values that comprise all offered values for the span. **Default value is from:** @*Actual*.
Preferred ?	NMTOKEN	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of @*Preferred* SHALL fall within the range of values specified in @*Range*.
Range ?	NMTOKENS	Provides a set of discreet values that comprise all allowed values for the span. **Default value is from:** @*Preferred*.

7.1.2.4.1 Specifying New Values in a NameSpan Subelement

NameSpan elements generally define an open list of predefined values. If a custom value is specified, a ***Comment*** element in the ***NameSpan*** defines the value with a @*Name* attribute in the ***Comment***, as demonstrated in the following example:

7.1.2.5 NumberSpan

This span subelement is used to describe a numerical range of values. It inherits from the abstract span element described in ▸ Section 7.1.1 Abstract Span Element.

Table 7.10: NumberSpan Element

NAME	DATA TYPE	DESCRIPTION
Actual ?	double	The actual value selected for the quote.
OfferRange ? New in JDF 1.3	DoubleRangeList	Provides either a set of discreet values, a range of values or a combination of the two that comprise all offered values for the span. **Default value is from:** @*Actual*.
Preferred ?	double	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of @*Preferred* SHALL fall within the range of values specified in @*Range*.
Range ?	DoubleRangeList	Provides either a set of discreet values, a range of values or a combination of the two that comprise all allowed values for the span. **Default value is from:** @*Preferred*.

7.1.2.6 OptionSpan

This span subelement is used to describe a range of options or Boolean values. It inherits from the abstract span element described in ▸ Section 7.1.1 Abstract Span Element.

Table 7.11: OptionSpan Element

NAME	DATA TYPE	DESCRIPTION
Actual ?	boolean	The actual value selected for the quote. If the option is included = "true".
Detail ? Deprecated in JDF 1.2	string	@*Detail* provides information about the option. **Deprecation note:** Starting with **JDF** 1.2, use @*DescriptiveName*.
OfferRange ? New in JDF 1.3	enumerations	Provides a set of the discreet Boolean values. **Default value is from:** @*Actual*. **Allowed values are:** true false
Preferred ?	boolean	Provides a value specified by the person submitting the request, indicating what that person prefers.
Range ? New in JDF 1.2	enumerations	Provides a set of the discreet Boolean values. **Allowed values are:** true false

7.1.2.7 ShapeSpan

New in JDF 1.1

This span subelement is used to describe ranges of numerical value pairs. It inherits from the abstract span element described in ▸ Section 7.1.1 Abstract Span Element.

Table 7.12: ShapeSpan Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Actual ?	shape	The actual value selected for the quote.

Table 7.12: ShapeSpan Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
OfferRange ? New in JDF 1.3	ShapeRange-List	Provides either a set of discreet values, a range of values or a combination of the two that comprise all offered values for the span. **Default value is from:** @*Actual*.
Preferred ?	shape	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of @*Preferred* SHALL fall within the range of values specified in @*Range*.
Range ?	ShapeRange-List	Provides either a set of discreet values, a range of values or a combination of the two that comprise all allowed values for the span. **Default value is from:** @*Preferred*.

7.1.2.8 StringSpan

This span subelement is used to describe string ranges. It inherits from the abstract span element described in ▸ Section 7.1.1 Abstract Span Element.

Table 7.13: StringSpan Element

NAME	DATA TYPE	DESCRIPTION
Actual ?	string	The actual value selected for the quote.
Preferred ?	string	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of @*Preferred* SHALL fall within the range of values specified in @*Range*.
OfferRange * New in JDF 1.3	text element	Provides a set of discreet values that comprise all offered values for the span. **Default value is from:** @*Actual*.
Range *	text element	Provides a set of discreet values that comprise all allowed values for the span. **Default value is from:** @*Preferred*.

7.1.2.9 TimeSpan

This span subelement is used to describe a selection of instances in time. It inherits from the abstract span element described in ▸ Section 7.1.1 Abstract Span Element.

Table 7.14: TimeSpan Element

NAME	DATA TYPE	DESCRIPTION
Actual ?	dateTime	The actual value selected for the quote.
OfferRange ? New in JDF 1.3	DateTimeRange	Provides a range of values that comprise all offered values for the span. **Default value is from:** @*Actual*.
Preferred ?	dateTime	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of @*Preferred* SHALL fall within the range of values specified in @*Range*.
Range ?	DateTimeRange	Provides a range of values that comprise all allowed values for the span. **Default value is from:** @*Preferred*.

7.1.2.10 XYPairSpan

This span subelement is used to describe ranges of numerical value pairs. It inherits from the abstract span element described in ▸ Section 7.1.1 Abstract Span Element.

Table 7.15: XYPairSpan Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Actual ?	XYPair	The actual value selected for the quote.

Table 7.15: XYPairSpan Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
OfferRange ? New in JDF 1.3	XYPair-RangeList	Provides either a set of discreet values, a range of values or a combination of the two that comprise all offered values for the span. **Default value is from:** @*Actual*.
Preferred ?	XYPair	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of @*Preferred* SHALL fall within the range of values specified in @*Range*.
Range ?	XYPair-RangeList	Provides either a set of discreet values, a range of values or a combination of the two that comprise all allowed values for the span. **Default value is from:** @*Preferred*.

7.2 ArtDeliveryIntent

This resource specifies the prepress art delivery intent for a **JDF** job and maps the items to the appropriate reader pages and separations. Art delivery refers to any physical or electronic asset that is needed for processing the job.

Resource Properties

Resource Class Intent

Process Resource Pairing: ***DeliveryParams***, ***DigitalDeliveryParams***

Example Partition: "Option"

Table 7.16: ArtDeliveryIntent Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ArtDeliveryDate ? New in JDF 1.1	TimeSpan	Specifies the latest time by which the transfer of the artwork will be made.
ArtDeliveryDuration ? New in JDF 1.1	DurationSpan	Specifies the latest time by which the transfer will be made relative to the date of the purchase order. Within an RFQ or a Quote, at most one of either ***ArtDeliveryDate*** or ***ArtDeliveryDuration*** SHALL be specified. Within a purchase order, only ***ArtDeliveryDate*** is allowed.
ArtHandling ? New in JDF 1.1	EnumerationSpan	Describes what SHALL happen to the artwork after usage. The address for the "Return" and "Pickup" values SHALL be specified by a ***Contact***[contains (@*ContactTypes*, "ArtReturn")]/***Address***. **Allowed values are:** ReturnWithProof – The artwork is delivered back to the customer together with the proof if there is any. ReturnWithProduct – The artwork is delivered back to the customer together with the final product. Return – The artwork is delivered back independently directly after usage. Pickup – The customer picks up the artwork. Destroy – The printer destroys the artwork. PrinterOwns – The artwork belongs to the printer. Store – The printer has to store the artwork for future purposes.
DeliveryCharge ? New in JDF 1.1 Modified in JDF 1.3	EnumerationSpan	Specifies who pays for a delivery being made by a third party. **Allowed values are from:** ***DeliveryIntent***/***DeliveryCharge***.
Method ? Modified in JDF 1.5	NameSpan	Specifies the delivery method, which can be a generic method. **Values include those from:** ***Drop***/@*Method* **Modification note:** Starting in **JDF** 1.5, values have changed.

Table 7.16: ArtDeliveryIntent Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
PreflightStatus = "NotPerformed" New in JDF 1.1 Modified in JDF 1.2	enumeration	Information about a **Preflight** process probably applied to the artworks before being submitted. **Allowed values are:** NotPerformed – No preflighting was applied. WithErrors – Preflighting resulted in error messages and possibly warning messages. WithWarnings – Preflighting resulted in warning messages and no errors. WithoutErrors – Preflighting was successful. No errors and no warnings occurred.
ReturnList = "None" New in JDF 1.1	NMTOKENS	Type of printer created intermediate materials that are to be sent to the customer after usage. **Values include:** DigitalMedia – Digital data on media (e.g., a CD). DigitalNetwork – Digital data via network. ExposedPlate – Pre-exposed press plates, usually used for a rerun. ImposedFilm – Film of the imposed surfaces. LooseFilm – Film of individual pages or sections. OriginalPhysicalArt – Analog artwork (e.g., reflective or transparencies). Tool – Tools needed for processing the job (e.g., a die for die cutting or embossing stamp). None – No intermediate materials are to be returned to the customer.
ReturnMethod ? New in JDF 1.1	NameSpan	Specifies a delivery method for returning the artwork if ***ArtHandling***/@*Actual* = "Return" and for the printer created materials listed in ***ReturnList***. **Values include those from: *Method***
ServiceLevel ? New in JDF 1.2	StringSpan	The service level of the specific carrier. **Values include those from: *Drop*/*ServiceLevel***
Transfer ? New in JDF 1.1	Enumera-tionSpan	Describes the responsibility of the transfer. **Allowed values are:** BuyerToPrinterDeliver – The buyer delivers the artwork to the printer. The printer MAY specify in the quote a special ***Contact***[contains (@*ContactTypes*, "Delivery")] to specify where the buyer SHALL send the artwork. BuyerToPrinterPickup – The printer picks up the artwork. The ***Contact***[contains (@*ContactTypes*, "Pickup")] specifies where the printer has to pick up the artwork.
ArtDelivery + Modified in JDF 1.1	element	Individual delivery.
Company ? Deprecated in JDF 1.1	refelement	Address and further information of the art delivery. ***Company*** SHALL NOT be specified unless the printer is expected to pick up the art delivery at this address. In **JDF** 1.1 and beyond, ***Company*** is a subelement of ***Contact***.
Contact * New in JDF 1.1	refelement	Address and further information about the transfer of the artwork. The actual delivery address SHALL be specified by ***Contact***[contains (@*ContactTypes*, "Delivery")]/***Address***. At most one such ***Contact*** SHALL be specified. The actual pickup address SHALL be specified by ***Contact***[contains (@*ContactTypes*, "Pickup")]/***Address***. At most one such ***Contact*** SHALL be specified.

7.2.1 ArtDelivery

Each ***ArtDelivery*** element defines a set of existing products that are needed to create the specified product. Attributes that are specified in an ***ArtDelivery*** element overwrite those that are specified in their parent ***ArtDeliveryIntent*** element. If OPTIONAL attributes are not specified, their values default to the values specified in ***ArtDeliveryIntent***.

Table 7.17: ArtDelivery Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Amount ? Modified in JDF 1.2	integer	Number of physical objects to be delivered. Only valid if no detailed resource description (e.g., ***ExposedMedia***, ***RunList***, ***ScanParams***, ***DigitalMedia*** or ***Tool***) is specified.
ArtDeliveryDate ? New in JDF 1.1	TimeSpan	Specifies the latest time by which the transfer of the artwork will be made.
ArtDeliveryDuration ? New in JDF 1.1	DurationSpan	Specifies the latest time by which the transfer will be made relative to the date of the purchase order. Within an RFQ or a Quote, at most one of either ***ArtDeliveryDate*** or ***ArtDeliveryDuration*** SHALL be specified. Within a purchase order, only the ***ArtDeliveryDate*** is allowed.
ArtDeliveryType New in JDF 1.1 Modified in JDF 1.2	NMTOKEN	Type of artwork supplied. **Values include:** DigitalFile – Digital data irrespective of the delivery mechanism. The union of "DigitalMedia" and "DigitalNetwork". New in JDF 1.2 DigitalMedia – Digital data on media (e.g., a CD). DigitalNetwork – Digital data via network. ExposedPlate – Pre-exposed press plates, usually used for a rerun. ImposedFilm – Film of the imposed surfaces. LooseFilm – Film of individual pages or sections. OriginalPhysicalArt – Analog artwork (e.g., reflective or transparencies). Proof – Physical proof delivered with digital scan or separated film asset. Tool – Tools needed for processing the job (e.g., a die for die cutting or embossing stamp). None – No artwork exists, and it will be created later.
ArtHandling ? New in JDF 1.1	EnumerationSpan	Describes what SHALL happen to the artwork after usage. The address for the "Return" and "Pickup" values SHALL be specified by ***Contact***[contains (@*ContactTypes*, "ArtReturn")]/***Address***. **Default value is from:** ***ArtDeliveryIntent***/***ArtHandling***. **Allowed values are:** ReturnWithProof – The artwork is delivered back to the customer together with the proof if there is any. ReturnWithProduct – The artwork is delivered back to the customer together with the final product. Return – The artwork is delivered back independently directly after usage. Pickup – The customer picks up the artwork. Destroy – The printer destroys the artwork. PrinterOwns – The artwork belongs to the printer. Store – The printer has to store the artwork for future purposes.
DeliveryCharge ? New in JDF 1.1 Modified in JDF 1.3	EnumerationSpan	Specifies who pays for a delivery being made by a third party. **Default value is from:** ***ArtDeliveryIntent***/***DeliveryCharge***. **Allowed values are from:** ***DeliveryIntent***/***DeliveryCharge***.
HasBleeds = "false"	boolean	If "true", the file has bleeds.
IsTrapped = "false"	boolean	If "true", the file has been trapped.
Method ? Modified in JDF 1.5	NameSpan	Specifies a delivery method. It MAY be a generic item from the list defined in @*Method* in ***ArtDeliveryIntent***. **Values include those from:** ***Drop***/@*Method* **Modification note:** Starting in **JDF** 1.5, values have changed.
PageList ?	IntegerRangeList	Set of pages of the output ***Component*** that are filled by this ***ArtDelivery***. This maps the pages in the ***ArtDelivery*** to the pages in the product that is produced. For example if ***PageList*** = "3 ~ 5", page 0 of the ***ArtDelivery*** (e.g., ***RunList***) is page 3 in the product, page 1 is page 4, etc. If not specified, the @*PageList* SHALL include all pages in reader order. The indices specified in @*PageList* reference the ***PageData*** elements defined in ***PageList***.

Table 7.17: ArtDelivery Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
PreflightOutput ? New in JDF 1.1	URL	Pointer to the output information created by the preflight tool if @*PreflightStatus* is either "WithoutErrors" or "WithErrors".
PreflightStatus ? New in JDF 1.1	enumeration	Information about a **Preflight** process **Default value is from:** ***ArtDeliveryIntent***/@*PreflightStatus*. **Allowed values are from:** ***ArtDeliveryIntent*/@*PreflightStatus*.**
ReturnMethod ? New in JDF 1.1	NameSpan	Specifies a delivery method for returning the artwork if ***ArtHandling***/@***Actual*** = "Return". **Default value is from:** ***ArtDeliveryIntent*/*ReturnMethod***. **Values include those from:** ***ArtDeliveryIntent*/*ReturnMethod*.**
ServiceLevel ? New in JDF 1.2	StringSpan	The service level of the specific carrier. **Values include those from:** ***Drop*/*ServiceLevel***
Transfer ? New in JDF 1.1	Enumera-tionSpan	Describes the responsibility of the transfer. **Default value is from:** ***ArtDeliveryIntent*/*Transfer***. **Allowed values are from:** ***ArtDeliveryIntent*/*Transfer*.**
Company ? Deprecated in JDF 1.1	refelement	Address and further information about the art delivery. This SHALL NOT be specified unless the printer is expected to pick up the art delivery at this address. In **JDF** 1.1 and beyond, ***Company*** is a subelement of ***Contact***.
Component ? Deprecated in JDF 1.1	refelement	Description of a physical component (e.g., physical artwork). If neither ***Component***, ***ExposedMedia***, nor ***RunList*** are specified, no details of the ***ArtDelivery*** except the @*ArtDeliveryType* and @*Amount* are known.
Contact * New in JDF 1.1	refelement	Address and further information about the art transfer. **Default value is from:** ***ArtDeliveryIntent*/*Contact***.
DigitalMedia ? New in JDF 1.2	refelement	Description of any digital media (e.g., CD or tape with artwork that will be delivered). If neither ***ExposedMedia***, ***RunList***, ***DigitalMedia***, nor ***Tool*** are specified, no details of the ***ArtDelivery*** except the @*ArtDeliveryType* and @*Amount* are known.
ExposedMedia ? Modified in JDF 1.2	refelement	Description of exposed media (e.g., film, plate or proof). If neither ***ExposedMedia***, ***RunList***, ***DigitalMedia***, nor ***Tool*** are specified, no details of the ***ArtDelivery***, except the @*ArtDeliveryType* and @*Amount*, are known.
RunList ? Modified in JDF 1.2	refelement	Link to digital artwork that is accessible via a set of URLs that are defined in the ***RunList*/*LayoutElement*/*FileSpec*/@*URL***. If neither ***DigitalMedia***, ***ExposedMedia***, ***RunList***, nor ***Tool*** are specified, no details of the ***ArtDelivery*** except the @*ArtDeliveryType* and @*Amount* are known.
ScanParams ?	refelement	Description of a ***ScanParams*** that defines scanning details for the exposed media defined by ***ExposedMedia***.
Tool ? New in JDF 1.1 Modified in JDF 1.2	refelement	Details of the ***Tool*** if @*ArtDeliveryType* = "Tool". If neither ***ExposedMedia***, ***RunList***, ***DigitalMedia***, nor ***Tool*** are specified, no details of the ***ArtDelivery*** except the @*ArtDeliveryType* and @*Amount* are known.

7.3 BindingIntent

This resource specifies the binding intent for a **JDF** job using information that identifies the desired type of binding and which sides SHALL be bound. The input products that are used as a cover SHALL have a @*ProductType* of "Cover", "FrontCover" or "BackCover". The input products that are used as a hard cover jacket SHALL have a @*ProductType* of "Jacket". The input components that are used as a end sheets for hardcover or soft cover binding SHALL have a @*ProductType* of "EndSheet". All other products are bound in the order of their appearance in the ***ResourceLinkPool*** of the **JDF** node that contains the ***BindingIntent***.

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***BlockPreparationParams***, ***CaseMakingParams***, ***CasingInParams***, ***ChannelBindingParams***, ***CoilBindingParams***, ***CoverApplicationParams***, ***EndSheetGluingParams***, ***GlueApplication***, ***GluingPar-***

ams, *GlueLine*, *InsertingParams*, *JacketingParams*, *PlasticCombBindingParams*, *RingBindingParams*, *StitchingParams*, *SpinePreparationParams*, *SpineTapingParams*, *StitchingParams*, *StripBindingParams*, *ThreadSealingParams*, *ThreadSewingParams*, *WireCombBindingParams*

Example Partition: "Option"

Table 7.18: BindingIntent Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BackCoverColor ? New in JDF 1.1	EnumerationSpan	Defines the color of the back cover material of the binding. **Default value is from:** @*CoverColor* **Allowed values are from:** ▸ NamedColor
BackCoverColorDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If *BackCoverColorDetails* is supplied, *BackCoverColor* SHOULD also be supplied.
BindingColor ?	EnumerationSpan	Defines the color of the spine material of the binding. **Allowed values are from:** ▸ NamedColor
BindingColorDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If *BindingColorDetails* is supplied, *BindingColor* SHOULD also be supplied.
BindingLength ? Deprecated in JDF 1.6	EnumerationSpan	Indicates which side SHALL be bound when no content. Thus, no orientation is available, but a quote for binding is needed. **Allowed values are:** Long Short
BindingOrder = "Gathering" New in JDF 1.1 Modified in JDF 1.4	enumeration	Specifies whether the child *Component* resources are to be collected or gathered if multiple child *Component* resources are combined. **Allowed values are:** None – The products referenced by child product are NOT bound together. Typically used for flatwork jobs. New in JDF 1.4 Collecting – The products referenced by child product are collected on a spine and placed within one another. The first *Component* is on the outside. Gathering – The child *Component* resources are gathered on a pile and placed on top of one another. The first child product is on the top. List – More complex ordering of child *Component* resources is specified using the *BindList* in this intent resource for this product.
BindingSide ?	EnumerationSpan	@*BindingSide* indicates which side of the product SHALL be bound. Each of these values SHALL identify the binding edge. @*BindingSide* is defined in the coordinate system of the product. @*BindingSide* SHALL NOT be provided if @*BindingOrder* = "None". **Constraint:** If both @*BindingSide* and @*BindingLength* are specified, @*BindingSide* has precedence **Default value is from:** @*BindingLength*, unless a non-empty *BindList* was specified. **Allowed values are from:** ▸ Edge.
BindingType ? Modified in JDF 1.2	EnumerationSpan	Describes the desired binding for the job. **Allowed values are from:** ▸ Table 7.19 BindingType Attribute Values.
CoverColor ?	EnumerationSpan	Defines the color of the cover material of the binding. **Allowed values are from:** ▸ NamedColor
CoverColorDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If *CoverColorDetails* is supplied, *CoverColor* SHOULD also be supplied.
AdhesiveNote ? New in JDF 1.6	element	Details of *AdhesiveNote* binding.
AdhesiveBinding ? Deprecated in JDF 1.1	element	Details of *AdhesiveBinding*. Replaced with *SoftCoverBinding* in **JDF** 1.1.

Table 7.18: BindingIntent Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
BindList ? New in JDF 1.1	element	Details of binding of individual child ***Component*** resources.
BookCase ? Deprecated in JDF 1.1	element	Details of the book case. Used in combination with ***AdhesiveBinding***, ***ThreadSewing*** or ***ThreadSealing***. Replaced with ***HardCoverBinding*** in **JDF** 1.1.
ChannelBinding ?	element	Details of ***ChannelBinding***.
CoilBinding ?	element	Details of ***CoilBinding***.
EdgeGluing ? New in JDF 1.1	element	Details of ***EdgeGluing***.
HardCoverBinding ? New in JDF 1.1	element	Details of ***HardCoverBinding***.
PlasticCombBinding ?	element	Details of ***PlasticCombBinding***.
RingBinding ?	element	Details of ***RingBinding***.
SaddleStitching ?	element	Details of ***SaddleStitching***.
SideSewing ?	element	Details of ***SideSewing***.
SideStitching ?	element	Details of ***SideStitching***.
SoftCoverBinding ? New in JDF 1.1	element	Details of ***SoftCoverBinding***.
StripBinding ? New in JDF 1.1	element	Details of ***StripBinding***.
Tabs ?	element	Details of ***Tabs***.
Tape ? New in JDF 1.1	element	Details of ***Tape*** binding.
ThreadSealing ?	element	Details of ***ThreadSealing***.
ThreadSewing ?	element	Details of ***ThreadSewing***.
VeloBinding ? Deprecated in JDF 1.1	element	Details of ***VeloBinding***. Renamed to ***StripBinding*** in **JDF** 1.1.
WireCombBinding ?	element	Details of ***WireCombBinding***.

7.3.1 BindingType Attribute Values

*Table 7.19: **BindingType** Attribute Values*

VALUE	DESCRIPTION
AdhesiveNote New in JDF 1.6	Binding with removable adhesive on the back side of a product. Typically used for small brightly colored paper designed to be stuck prominently to an object or surface and easily removed when necessary.
Adhesive Deprecated in JDF 1.1	This type of binding can be handled with the **AdhesiveBinding** process. It includes perfect binding. Deprecated in **JDF** 1.1 and replaced with "SoftCover" or "HardCover".
ChannelBinding	This type of binding can be handled with the **ChannelBinding** process.
CoilBinding	This type of binding can be handled with the **CoilBinding** process.
CornerStitch New in JDF 1.2	Stitch in the corner that is at the clockwise end binding edge. For example, to stitch in the top left corner, set ***BindingSide***/@*Actual* = "Left". This type of binding can be handled with the **Stitching** process.
EdgeGluing	Gluing gathered sheets at one edge of the pile. This type of binding can be handled with the **Gluing** process. Products of this type are also referred to as padded.
HardCover	This type of binding defines a hard-cover bound book.
None	This type of binding defines a stack of pages with no additional binding.
PlasticComb	This type of binding can be handled with the **PlasticCombBinding** process.
Ring	This type of binding can be handled with the **RingBinding** process.
SaddleStitch	This type of binding can be handled with the **Stitching** process.
Sewn Deprecated in JDF 1.4	This type of binding can be handled with the **ThreadSewing** process.
SideSewn Deprecated in JDF 1.4	This type of binding can be handled with the **ThreadSewing** process.
SideStitch	This type of binding can be handled with the **Stitching** process.
SoftCover	This type of binding defines a soft cover bound book. It includes perfect binding.
StripBind	This type of binding can be handled with the **StripBinding** process.
Tape	This type of binding is an inexpensive version of the "SoftCover".
ThreadSealing Deprecated in JDF 1.4	This type of binding can be handled with the **ThreadSealing** process.
ThreadSewing Deprecated in JDF 1.4	This type of binding can be handled with the **ThreadSealing** process.
WireComb	This type of binding can be handled with the **WireCombBinding** process.

7.3.2 AdhesiveNote

New in JDF 1.6

Details of adhesive note binding.

Table 7.20: AdhesiveNote Element

NAME	DATA TYPE	DESCRIPTION
GlueLine ?	element	***GlueLine*** provides details of the shape of the glue application and type of glue used

7.3.3 BindList

New in JDF 1.1

BindList is used to describe complex bindings where more than one child is bound into a cover (e.g., in promotional products).

Table 7.21: BindList Element

NAME	DATA TYPE	DESCRIPTION
BindItem *	element	Individual bind item description.

7.3.4 BindItem

New in JDF 1.1

A child ***BindItem*** is bound to a parent item. The position of the spine of the child ***BindItem*** is defined by @*ChildFolio* and the position of the child ***BindItem*** in the parent is defined by @*ParentFolio*.

Table 7.22: BindItem Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BindingType ?	EnumerationSpan	Describes the desired binding for the individual ***BindItem***. **Default value is from:** ***BindingIntent***. See ▸ BindingType Attribute Values. **Allowed values are from:** ***BindingIntent*****. See** ▸ BindingType Attribute Values**.**
ChildFolio ?	XYPair	Definition of the fold between two pages in the ***BindItem*** component that is bound to the cover. The two numbers (as integers) in the @*ChildFolio* attribute are the page numbers of the two outer pages of the child ***Component*** which touch the cover or another parent ***Component***. The pages are counted in the order as described in ***LayoutIntent***/@*FolioCount* of the child product. Defaults to the spine of the child.
ParentFolio	XYPair	Definition of the fold between two pages in the cover ***Component*** that receive the ***BindItem***. The two numbers (as integers) in the @*ParentFolio* attribute are the page numbers in the cover ***Component*** which touch the child ***Component***. The pages are counted in the order as described in ***LayoutIntent***/@*FolioCount* of the cover product.
Transformation ?	matrix	Rotation and offset between the ***Component*** to be inserted and the parent ***Component***. For details on transformations, see ▸ Section 2.5.2 Coordinates and Transformations.
WrapPages ?	IntegerRangeList	List of pages of the cover that wrap around a ***BindItem*** after all folds are applied. It is sufficient to specify the pages of the "Front" partition of the cover (e.g. cover pages 1 and 4). **Note:** This attribute SHALL NOT be specified if the position of the cover can be derived from the folding information.
ChannelBinding ?	element	Details of ***ChannelBinding***.
CoilBinding ?	element	Details of ***CoilBinding***.
EdgeGluing ?	element	Details of ***EdgeGluing***.
HardCoverBinding ?	element	Details of ***HardCoverBinding***.
PlasticCombBinding ?	element	Details of ***PlasticCombBinding***.
RingBinding ?	element	Details of ***RingBinding***.
SaddleStitching ?	element	Details of ***SaddleStitching***.
SideSewing ?	element	Details of ***SideSewing***.
SideStitching ?	element	Details of ***SideStitching***.
SoftCoverBinding ?	element	Details of ***SoftCoverBinding***.
StripBinding ?	element	Details of ***StripBinding***.
Tabs ?	element	Details of ***Tabs***.

Table 7.22: BindItem Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Tape ?	element	Details of ***Tape*** binding.
ThreadSealing ?	element	Details of ***ThreadSealing***.
ThreadSewing ?	element	Details of ***ThreadSewing***.
WireCombBinding ?	element	Details of ***WireCombBinding***.

7.3.5 AdhesiveBinding

Deprecated in JDF 1.1

The table defining the deprecated ***AdhesiveBinding*** subelement has been moved to ▸ Section N.6.1 BindingIntent Deprecated Subelements.

7.3.6 BookCase

Deprecated in JDF 1.1

The table defining the deprecated ***BookCase*** subelement has been moved to ▸ Section N.6.1 BindingIntent Deprecated Subelements.

7.3.7 ChannelBinding

Table 7.23: ChannelBinding Element

NAME	DATA TYPE	DESCRIPTION
ChannelBrand ? New in JDF 1.3	StringSpan	Strings providing available brand names for the ***ChannelBinding***.
Cover ?	OptionSpan	If "true", the clamp used in **ChannelBinding** includes a preassembled cover.
Thickness ?	NumberSpan	Specifies thickness of board which is wrapped as front and back covers of a case bound book, in points.

7.3.8 CoilBinding

Table 7.24: CoilBinding Element

NAME	DATA TYPE	DESCRIPTION
CoilBrand ? New in JDF 1.3	StringSpan	Strings providing available brand names for the coil.
CoilMaterial ?	Enumera-tionSpan	The coil materials available for **CoilBinding**. **Allowed values are from:** ▸ BinderMaterial.
HoleList ? New in JDF 1.2	refelement	Details of the holes for coil binding.
HoleType ? New in JDF 1.1	***HoleType*** ? New in JDF 1.1	Predefined hole pattern that matches the binder. For details of the hole types, see the values. **Allowed values are from:** ▸ Appendix L Hole Pattern Catalog**.**

7.3.9 EdgeGluing

New in JDF 1.1

Table 7.25: EdgeGluing Element

NAME	DATA TYPE	DESCRIPTION
EdgeGlue ?	EnumerationSpan	Glue type used to glue the edge of the gathered sheets. **Allowed values are from:** ▸ Glue.

7.3.10 HardCoverBinding

New in JDF 1.1

Table 7.26: HardCoverBinding Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BlockThreadSewing ?	OptionSpan	Specified if the block is thread sewn.
CoverStyle ? New in JDF 1.3	NameSpan	Defines the style of the cover board. **Values include:** Simple – Single layer cover board, see ▸ Figure 7-1: Structure of a normal hardcover book. Padded – Padded cover board, see ▸ Figure 7-2: Structure of a padded hardcover book.
EndSheets ?	OptionSpan	Specified if end sheets are applied. Additional details of the ***EndSheets*** MAY be specified by supplying an input ***Component*** with @*ProcessUsage* with @*ProductType* = "EndSheet".
HeadBands ?	OptionSpan	The following case binding choice specifies the use of headbands on a case bound book. If "true", headbands are inserted both top and bottom.
HeadBandColor ?	EnumerationSpanenumeration	Defines the color of the headband. **Allowed values are from:** ▸ NamedColor.
HeadBandColorDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If ***HeadBandColorDetails*** is supplied, ***HeadBandColor*** SHOULD also be supplied.
Jacket ?	EnumerationSpan	Specifies whether a hard cover jacket is needed and how it is attached. Details of the jacket MAY be described in the ***Component*** with @*ProcessUsage* = "Jacket". **Allowed values are:** None – No jacket is needed. Loose – The jacket is loosely wrapped. Glue – The jacket is glued to the spine
JacketFoldingWidth ? New in JDF 1.3	NumberSpan	Dimension of the jacket folds. See ***JacketingParams*** for details.
JapanBind ?	OptionSpan	Bind the book block at the open edge, so that the folds are visible on the outside. If not specified, explicitly, this option is never selected.
SpineBrushing ?	OptionSpan	Brushing option for **SpinePreparation**.
SpineFiberRoughing ?	OptionSpan	Fiber roughing option for **SpinePreparation**.
SpineGlue ?	EnumerationSpan	Glue type used to glue the book block to the cover. **Allowed values are from:** ▸ Glue.
SpineLevelling ?	OptionSpan	Leveling option for **SpinePreparation**.
SpineMilling ?	OptionSpan	Milling option for **SpinePreparation**.

Table 7.26: HardCoverBinding Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
SpineNotching ?	OptionSpan	Notching option for **SpinePreparation**.
SpineSanding ?	OptionSpan	Sanding option for **SpinePreparation**.
SpineShredding ?	OptionSpan	Shredding option for **SpinePreparation**.
StripMaterial ?	Enumera-tionSpan	Spine taping strip material. **Allowed values are from:** ▸ StripMaterial.
Thickness ?	NumberSpan	Specifies thickness of board which is wrapped as front and back covers of a case bound book, in points.
TightBacking ?	Enumera-tionSpan	Definition of the geometry of the back of the book block. **Allowed values are from:** ▸ TightBacking.
RegisterRibbon *	refelement	Number, materials, colors and details of register ribbons.

Figure 7-1: Structure of a normal hardcover book

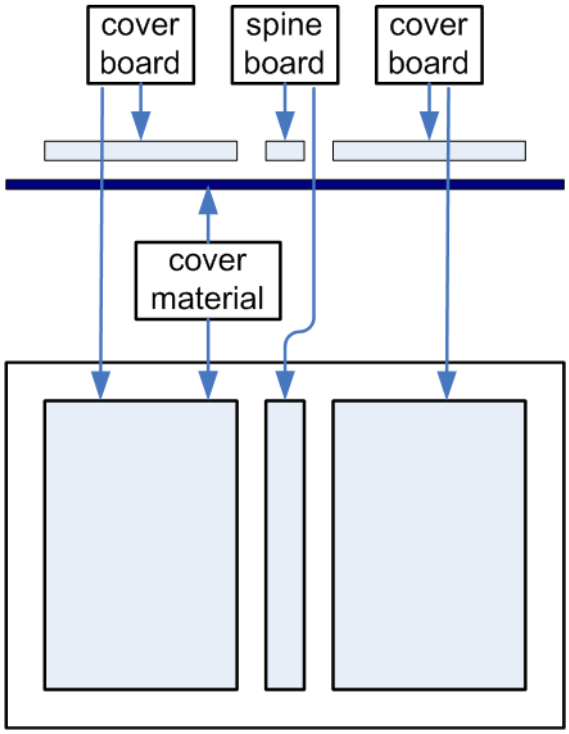
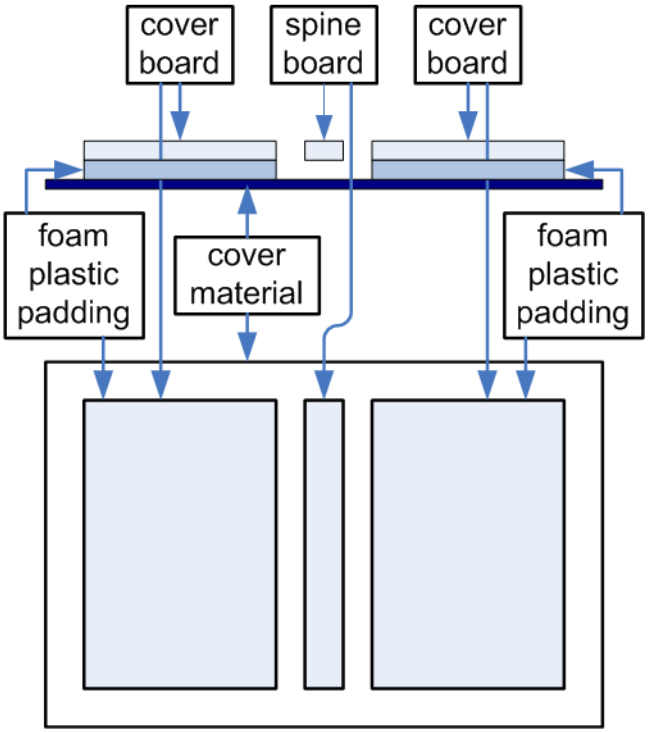


Figure 7-2: Structure of a padded hardcover book



7.3.11 PlasticCombBinding

Table 7.27: PlasticCombBinding Element

NAME	DATA TYPE	DESCRIPTION
CombBrand ? New in JDF 1.3	StringSpan	Strings providing available brand names for the plastic comb.
PlasticCombType ? Modified in JDF 1.1	NameSpan	The distance between the "teeth" in **PlasticCombBinding** and the distance between the holes of the prepunched leaves SHALL be the same. The following values from the hole type catalog in ▸ Appendix L Hole Pattern Catalog exist: **Values include:** P12m-rect-02 – Distance = 12 mm; Holes = 7 mm x 3 mm P16_9i-rect-0t – Distance = 14.28 mm; Holes = 8 mm x 3 mm Euro – Distance = 12 mm; Holes = 7 mm x 3 mm Deprecated in JDF 1.1 USA1 – Distance = 14.28 mm; Holes = 8 mm x 3 mm. Deprecated in JDF 1.1
HoleList ? New in JDF 1.2	element	Details of the holes for the plastic comb. Note that @*Shape* is always rectangular by design of the plastic combs.
HoleType ? New in JDF 1.1	***HoleType*** ? New in JDF 1.1	Predefined hole pattern that matches the binder. For details of the hole types, see the values. **Allowed values are from:** ▸ Appendix L Hole Pattern Catalog**.**

7.3.12 RingBinding

Table 7.28: RingBinding Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BinderBrand ? New in JDF 1.3	StringSpan	Strings providing available brand names for ***RingBinding***.

Table 7.28: RingBinding Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
BinderMaterial ?	NameSpan	The following describe **RingBinding** binder materials used. **Values include:** Cardboard – Cardboard with no covering. ClothCovered – Cardboard with cloth covering. Plastic – Binder cover fabricated from solid plastic sheet material (e.g., PVC sheet). VinylCovered – Cardboard with colored vinyl covering.
HoleType ? New in JDF 1.1	***HoleType*** ? New in JDF 1.1	Predefined hole pattern for the ring system. Multiple hole patterns are not allowed (e.g., 3-hole ring binding and 4-hole ring binding holes on one piece of media). For details of the hole types, see the values. **Allowed values are from:** ▸ Appendix L Hole Pattern Catalog**.**
RingDiameter ?	NumberSpan	Size of the rings in points. The value used in production SHALL be suitable for specified values of ***HoleType***. Note that in ring shapes other than round, this size is specified by industry-standard method.
RingMechanic ?	OptionSpan	The ring binder used includes a lever for opening and closing.
RingShape ?	NameSpan	**RingBinding** shapes. **Values include:** Round Oval D-shape SlantD
RingSystem ? Deprecated in JDF 1.1	NameSpan	**Values include:** 2HoleEuro 3HoleUS 4HoleEuro **Deprecation note:** Starting with **JDF** 1.1, use ***HoleType***.
RivetsExposed ?	OptionSpan	The following **RingBinding** choice describes mounting of the ring mechanism in binder case. If "true", the heads of the rivets are visible on the exterior of the binder. If "false", the binder covering material covers the rivet heads.
ViewBinder ?	NameSpan	The values are **RingBinding** clear vinyl outer wrap types and are used on top of a colored base wrap. **Values include:** Embedded – Printed material is embedded by sealing between the colored and clear vinyl layers during binder manufacturing. Pocket – Binder is designed so that printed material can be inserted between the color and clear vinyl layers after binder manufacturing.

7.3.13 SaddleStitching

Table 7.29: SaddleStitching Element

NAME	DATA TYPE	DESCRIPTION
StitchNumber ? New in JDF 1.1	IntegerSpan-integer	Number of stitches used for saddle stitching.

7.3.14 SideSewing

This is a placeholder that might be filled with private or future data.

Table 7.30: SideSewing Element

NAME	DATA TYPE	DESCRIPTION

7.3.15 SideStitching

Table 7.31: SideStitching Element

NAME	DATA TYPE	DESCRIPTION
StitchNumber ? New in JDF 1.2	IntegerSpan	Number of stitches used for side stitching.

7.3.16 SoftCoverBinding

New in JDF 1.1

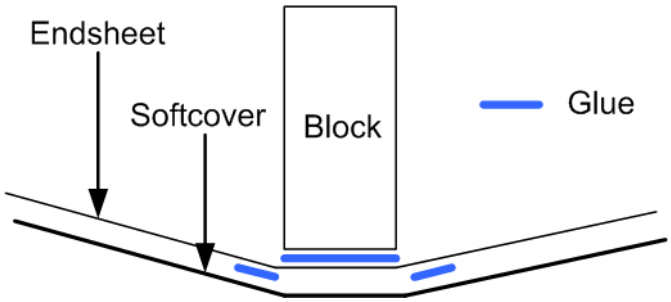
Table 7.32: SoftCoverBinding Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BlockThreadSewing ?	OptionSpan	Specifies whether the block is also thread sewn.
EndSheets ? New in JDF 1.3	OptionSpan	Specified if end sheets are applied. Additional details of the ***EndSheets*** MAY be specified by supplying an input ***Component*** with @*ProcessUsage* = "EndSheet".
FoldingWidth ? New in JDF 1.3	NumberSpan	Definition of the dimension of the folding width of the front cover fold. See ***JacketingParams*** for details.
FoldingWidthBack ? New in JDF 1.3	NumberSpan	Definition of the dimension of the folding width of the back cover fold. If not specified, ***FoldingWidthBack*** defaults to ***FoldingWidth***.
GlueProcedure ?	EnumerationSpan	Glue procedure used to glue the book block to the cover. **Allowed values are:** Spine SideOnly – Glued at the side or endsheets but not at the spine. "SideOnly" books are also referred to as "layflat" if ***EndSheets*** are also specified. See ▸ Figure 7-3: Structure of a book with GlueProcedure = "SideOnly" (Layflat). SingleSide – Swiss brochure. SideSpine – Both side gluing and spine gluing.
Scoring ?	EnumerationSpan	Scoring option for ***SoftCoverBinding***. Values are based on viewing the cover in its flat, prebound state. **Allowed values are:** TwiceScored QuadScored None
SpineBrushing ?	OptionSpan	Brushing option for **SpinePreparation**.
SpineFiberRoughing ?	OptionSpan	FiberRoughing option for **SpinePreparation**.
SpineGlue ?	EnumerationSpan	Glue type used to glue the book block to the cover. **Allowed values are from:** ▸ Glue.
SpineLevelling ?	OptionSpan	Leveling option for **SpinePreparation**.
SpineMilling ?	OptionSpan	Milling option for **SpinePreparation**.
SpineNotching ?	OptionSpan	Notching option for **SpinePreparation**.

Table 7.32: SoftCoverBinding Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
SpineSanding ?	OptionSpan	Sanding option for **SpinePreparation**.
SpineShredding ?	OptionSpan	Shredding option for **SpinePreparation**.

Figure 7-3: Structure of a book with GlueProcedure = "SideOnly" (Layflat)



7.3.17 StripBinding

New in JDF 1.1

Table 7.33: StripBinding Element

NAME	DATA TYPE	DESCRIPTION
HoleList ? New in JDF 1.2	refelement	Note that @*Shape* is always round by design of the strip poles.
HoleType ? New in JDF 1.1	***HoleType*** ? New in JDF 1.1	Predefined hole pattern that matches the binder. For details of the hole types, see the values. **Allowed values are from:** ▸ Appendix L Hole Pattern Catalog**.**

7.3.18 Tabs

Specifies tabs in a bound document.

Table 7.34: Tabs Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
TabBanks = "1" Deprecated in JDF 1.4	integer	Number of rows of tabs on the face of the book. **Deprecation note:** Starting with **JDF** 1.4, @*TabBanks* should be calculated from @*TabCount* and @*TabsPerBank*.
TabBrand ? New in JDF 1.3	StringSpan	Strings providing available brand names for the ***Tabs***.
TabCount ? New in JDF 1.4	integer	Number of tabs across all banks. If @*TabsPerSet* is not an even multiple of @*TabsPerBank*, the last bank in each set is partially filled.
TabsPerBank ?	integer	Number of equal-sized tabs in a single bank if all positions were filled. Note that banks can have tabs only in some of the possible positions
TabExtensionDistance ?	NumberSpan	Distance tab extends beyond the body of the book block, in points.
TabExtensionMylar ?	OptionSpan	If "true", the tab extension will be mylar reinforced.
TabBindMylar ?	OptionSpan	If "true", the tab bind edge will be mylar reinforced.

Table 7.34: Tabs Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
TabBodyCopy ?	OptionSpan	If "true", color will be applied not only on tab extension, but also on tab body. Note that lack of body copy allows all tabs within a bank to be printed on a single sheet.
TabMylarColor ?	Enumera-tionSpan	Specifies the color of the mylar used to reinforce the tab extension. This is conditional on ***TabExtensionMylar*** being "true". **Allowed values are from:** ▸ NamedColor
TabMylarColorDe-tails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If ***TabMylar-ColorDetails*** is supplied, ***TabMylarColor*** SHOULD also be supplied.

7.3.19 Tape

New in JDF 1.1

Table 7.35: Tape Element

NAME	DATA TYPE	DESCRIPTION
TapeColor ? Deprecated in JDF 1.4	Enumera-tionSpan	Defines the color of the tape material of the binding. **Allowed values are from:** ▸ Table A.22 NamedColor Enumeration Values**.** **Deprecation note:** Starting with **JDF** 1.4, use ***BindingIntent***/@***BindingColor***.

7.3.20 ThreadSealing

This is a placeholder that might be filled with private or future data.

Table 7.36: ThreadSealing Element

NAME	DATA TYPE	DESCRIPTION

7.3.21 ThreadSewing

Table 7.37: ThreadSewing Element

NAME	DATA TYPE	DESCRIPTION
Sealing ?	OptionSpan	If "true", thermo-sealing is needed in **ThreadSewing**.

7.3.22 WireCombBinding

Table 7.38: WireCombBinding Element

NAME	DATA TYPE	DESCRIPTION
WireCombBrand ? New in JDF 1.3	StringSpan	Strings providing available brand names for the *WireCombBinding*.
WireCombMaterial ?	Enumera-tionSpan	The material used for forming the **WireCombBinding**. **Allowed values are:** Steel-Silver ColorCoatedSteel
WireCombShape ? Modified in JDF 1.6	NameSpan	The shape of the wire comb. **Values include:** Single – Each "tooth" is made with one wire. SingleCalendar – Each "tooth" is made with one wire and an extension for hanging the bound product is provided in the center. New in JDF 1.6 Twin – The shape of each "tooth" is made with a double wire. TwinCalendar – The shape of each "tooth" is made with a double wire and an extension for hanging the bound product is provided in the center. New in JDF 1.6
HoleList ? New in JDF 1.2	refelement	Details of the holes for the wire comb.
HoleType ? New in JDF 1.1	*HoleType* ? New in JDF 1.1	Predefined hole pattern that matches the binder. For details of the hole types, see the values. **Allowed values are from:** ▸ Appendix L Hole Pattern Catalog**.**

7.4 ColorIntent

This resource specifies the type of ink to be used. Typically, the parameters consist of a manufacturer name and additional identifying information. The resource also specifies any coatings and colors to be used, including the process color model and any spot colors.

In addition to the printed images, ColorIntent also provides details of protective or gloss enhancing coatings. Customers may either specify the performance characteristic they desire in the coating or specify a coating type. Common examples are water-resistance, and rub-resistance. Both characteristics may be required at the same time. An example is in the wine industry, where the white wine label has to survive transport rubbing, followed by water and rubbing ice cubes in a bucket upon serving.

Resource Properties

Resource Class: Intent

Process Resource Pairing: *Color*, *ColorantControl*, *ColorCorrectionParams*, *ColorPool*, *ColorSpaceConversionParams*, *Ink*, *VarnishingParams*

Example Partition: "Option", "PageNumber", "Side"

Table 7.39: ColorIntent Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Coatings ? Modified in JDF 1.5	StringSpan	Material usually applied to a full surface on press as a protective or gloss-enhancing layer over ink. **Values include those from:** ▸ Section C.6 Media Coating Names

Table 7.39: ColorIntent Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ColorICCStandard ? New in JDF 1.2	StringSpan	***ColorICCStandard*** can be used to identify a specific standard printing condition, by reference to characterization data registered with the ICC (http://www.color.org/drsection1.html). This printing condition reference corresponds to the output intent characterization referencing capability in PDF/X. The syntax will be reference name as shown in the examples below. Reference name is the standard reference string name used in both **JDF** and PDF/X, defined for each printing condition in the characterization registry on the ICC website. **Values include:** FOGRA11 – Registered by FOGRA pertaining to offset commercial and specialty printing according to ▸ [ISO12647-2:2004], positive plates, paper type 1 (gloss-coated, above 70 g/m^2) and paper type 2 (matte-coated, above 70 g/m^2), screen frequency 60/cm. Appropriate for black-backing measurement. FOGRA15 – Registered by FOGRA pertaining to offset commercial and specialty printing according to ▸ [ISO12647-2:2004], positive plates, paper type 1 (gloss-coated, above 70 g/m^2) and paper type 2 (matte-coated, above 70 g/m^2), screen frequency 60/cm. Appropriate for self-backing measurement. CGATS TR001 – pertaining to printing conditions that conform to ANSI CGATS.6, which addresses publication printing in the US as defined by SWOP. **Note:** If both of ***ColorICCStandard*** or are specified, the union of the two is specified. If both of ***ColorICCStandard*** and ***ColorStandard*** are specified, then ***ColorICCStandard*** defines the ICC specific details, whereas ***ColorStandard*** defines the generic color standard.
ColorStandard Attribute Values ? Modified in JDF 1.2	NameSpan	The color process (i.e., printing condition) requested for the job. ***ColorStandard Attribute Values*** does not imply values for ***ColorsUsed***. For instance, if ***ColorStandard Attribute Values*** is "CMYK", ***ColorsUsed*** SHALL still contain the four process colors "Cyan", "Magenta", "Yellow" and "Black". If both of ***ColorICCStandard*** and ***ColorStandard Attribute Values*** are specified, then ***ColorICCStandard*** defines the ICC specific details, whereas ***ColorStandard Attribute Values*** defines the generic color standard. **Values include those from:** ▸ Table 7.40 ColorStandard Attribute Values.
Coverage ?	NumberSpan	Cumulative colorant coverage percentage. For example, a full sheet of 100% deep black in CMYK has ***Coverage***/@*Actual* = "400". Typical coverages based on one color plane are: Light – 1-9% Medium – 10-35% Heavy – 36+%
InkManufacturer ? Deprecated in JDF 1.2	NameSpan	Name of the manufacturer of the ink requested (e.g., "ACMEInk", "CIP4_Ink_Company", etc.).
NumColors ? New in JDF 1.5	integer	@*NumColors* specifies the number of colors (Inks) used for a product. A value of 0 implies no printing. A value of 1 implies black. A value of 4 implies CMYK. Spot colors SHALL be specified in @*ColorsUsed*. If both @*NumColors* and @*ColorsUsed* are specified, the sum of both is requested (e.g., @*NumColors* = "4" and @*ColorsUsed* = "Spot1" defines a CMYK product with one additional spot color).
ColorPool ? New in JDF 1.1	refelement	Additional details about the colors used. The ***ColorPool*** resource MAY include some or all details about both ***ColorsUsed*** separation spot colors, spot colors contained in job files that will be printed using process color equivalents and the ***ColorStandard Attribute Values*** process colors.
ColorsUsed ?	element	Array of colorant separation names that are requested. If not specified, the values are implied from ***ColorStandard Attribute Values***. If specified, ***ColorsUsed*** SHALL contain a list of all separation names used by the job. **Note:** If additional information about the colors and colorants is needed, it can be specified in the referenced ***ColorPool*** resource.

7.4.1 ColorStandard Attribute Values

Table 7.40: ColorStandard Attribute Values

VALUE	DESCRIPTION
CMYK	Generic four color process.
FIRST	Flexographic Image Reproduction Specifications & Tolerances.
GRACOL	General Requirements for Applications in Commercial Offset Lithography
Hexachrome	6 colors "CMYK" + "Orange" and "Green".
HIFI	7 colors "CMYK" + "Red", "Green" and "Blue".
ISO12647 Deprecated in JDF 1.2	▸ [ISO12647-2:2004] offset standard.
JapanColor2001	Japan Color 2001 standard ▸ [japancolor].
Monochrome	Generic single color printing condition (e.g., black and white or one single spot color).
None Deprecated in JDF 1.2	No marks. Used to define one-sided printing. **Deprecation note:** Starting with **JDF** 1.2, use ***LayoutIntent***/@*Sides* instead.
SNAP	Specifications for Newsprint Advertising Production
SWOP	Specifications for Web Offset Publications. Registered by ANSI with the ICC as *ICC:CGATS TR001* pertaining to printing conditions that conform to ANSI CGATS.6 which is based on publication printing in the US as defined by SWOP, Inc.

7.4.2 ColorsUsed

Table 7.41: ColorsUsed Element

NAME	DATA TYPE	DESCRIPTION
SeparationSpec * Modified in JDF 1.2	element	These can be process colors, generic spot colors or named spot colors. In addition, partial (spot) coating is specified by adding a ***SeparationSpec*** with anything from ***Coatings*** as ***SeparationSpec***/@*Name*: Aqueous Bronzing DullVarnish GlossVarnish SatinVarnish Silicone Spot – Generic spot color of which the details are unknown. Spot MAY be specified multiple times in one ***ColorsUsed*** element. New in JDF 1.2 UV Varnish – Generic varnish including "DullVarnish", "GlossVarnish" and "SatinVarnish". New in JDF 1.3

7.5 DeliveryIntent

Summarizes the options that describe pickup or delivery time and location of the ***PhysicalResource***s of a job. It also defines the number of copies that are requested for a specific job or delivery. This includes delivery of both final products and of proofs. ***DeliveryIntent*** MAY also be used to describe the delivery of intermediate products such as partial products in a subcontracting description

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***Address***, ***DeliveryParams***

Example Partition: "Option"

Table 7.42: DeliveryIntent Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Accepted ? Deprecated in JDF 1.3	boolean	The quote that is specified by this ***DeliveryIntent*** has been accepted. **Deprecation note:** Starting with **JDF** 1.3, contract negotiation information has been removed and will be handled by the business wrapper around **JDF** (e.g., PrintTalk).
AdditionalAmount = "1" New in JDF 1.2 Deprecated in JDF 1.3	integer	Number of components used to calculate the value of the @*AdditionalPrice* attribute in the ***Pricing***. This value applies to the number of additional items in one ***DropIntent***/***DropItemIntent*** and not to the total additional number of items. In **JDF** 1.3 and beyond, pricing information has been removed and will be handled by the business wrapper around **JDF** (e.g., PrintTalk).
BuyerAccount ?	string	Account ID of the buyer with the delivery service.
DeliveryCharge ? New in JDF 1.1 Modified in JDF 1.2	Enumeration-Span	Specifies who pays for a delivery being made by a third party. **Allowed values are:** Printer – The "Printer" is defined as the person who creates the resource that is delivered. This includes all suppliers (e.g., binders, prepress suppliers, etc.). Buyer – The customer specified in ***CustomerInfo***. Other – The ***Contact***[@*ContactTypes* = "DeliveryCharge"]. New in JDF 1.2
Earliest ?	TimeSpan	Specifies the earliest time after which the transfer SHALL be made. Within an RFQ or a Quote, at most one of ***Earliest*** or ***EarliestDuration*** SHALL be specified.
EarliestDuration ?	DurationSpan	Specifies the earliest time by which the transfer SHALL be made relative to the date of the purchase order. Within an RFQ or a Quote, at most one of ***Earliest*** or ***EarliestDuration*** SHALL be specified. Within a purchase order, ***EarliestDuration*** SHALL NOT be specified.
Method ? Modified in JDF 1.5	NameSpan	Specifies a delivery method, which can be a generic method. **Values include those from:** ***Drop***/@*Method* **Modification note:** Starting in **JDF** 1.5, values have changed.
Overage ?	NumberSpan	Percentage value that defines the acceptable upwards variation of @*Amount*. Defaults to the trade custom defaults as defined by PIA, BVD, etc.
Ownership = "Origin"	enumeration	Point of transfer of ownership: **Allowed values are:** Origin – Ownership of goods is transferred upon leaving point of origin. Destination – Ownership is transferred upon receipt at destination.
Pickup ? Deprecated in JDF 1.1	boolean	Specifies whether the delivery brings or picks up the merchandise. If @*Pickup* = "false", the drop is delivered to the address specified in ***Company***. If @*Pickup* = "true", the ***DeliveryIntent*** describes an input to the job (e.g., a CD for inserting, a preprinted cover, etc.). In this case ***Company*** describes the location where the merchandise is picked up.
Required ?	TimeSpan	Specifies the time by which the transfer SHALL be made. Within an RFQ or a Quote, exactly one of ***Required*** or ***RequiredDuration*** SHALL be specified.
RequiredDuration ?	DurationSpan	Specifies the time by which the transfer SHALL be made relative to the date of the purchase order. Within an RFQ or a Quote, exactly one of ***Required*** or ***RequiredDuration*** SHALL be specified. Within a purchase order, ***RequiredDuration*** SHALL NOT be specified.
ReturnMethod ? New in JDF 1.1	NameSpan	Specifies a delivery method for returning the surplus material and SHALL NOT be specified unless ***SurplusHandling*** = "Return". **Allowed values are from:** ***Method***
ServiceLevel ? New in JDF 1.2	StringSpan	The service level of the specific carrier. **Values include those from:** ***Drop***/***ServiceLevel***

Table 7.42: DeliveryIntent Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
SurplusHandling ? New in JDF 1.1	Enumera-tionSpan	Describes what SHALL happen with unused or redundant parts of the transfer specified with ***Transfer*** = "BuyerToPrinterDeliver" or "BuyerToPrinterPickup" after the job. The return delivery or pickup address is specified by ***Contact***[contains (@*ContactTypes*, "SurplusReturn")]. **Allowed values are:** ReturnWithProduct – The surplus material is delivered back to the customer together with the final product. Return – The surplus material is delivered back independently directly after usage. Pickup – The customer picks up the surplus material. Destroy – The printer destroys the surplus material. PrinterOwns – The surplus material belongs to the printer. Store – The printer has to store the surplus material for future purposes.
Transfer ? New in JDF 1.1	Enumera-tionSpan	Describes the direction and responsibility of the transfer. **Allowed values are from:** ***Drop***/***Transfer***.
Underage ?	NumberSpan	Percentage value that defines the acceptable downwards variation of @*Amount*. Defaults to the trade custom defaults as defined by PIA, BVD, etc. The value SHALL be nonnegative.
Company ? Deprecated in JDF 1.1	refelement	Address and further information of the addressee. In **JDF** 1.1 and beyond, ***Company*** is referenced from ***Contact***.
Contact * New in JDF 1.1	refelement	Address and further information of the ***Contact*** responsible for the transfer. The actual delivery address is specified as the ***Contact***[contains (@*ContactTypes*, "Delivery")]/***Address***. The actual pickup address is specified as the ***Contact***[contains (@*ContactTypes*, "Pickup")]/***Address***. At most one ***Contact*** [contains (@*ContactTypes*, X)] SHALL be specified for X equal to "Delivery", "Pickup" or "Billing",
DropIntent +	element	Includes all locations where the product will be delivered. Note that multiple ***DropIntent*** elements specify multiple deliveries and not options for delivery.
FileSpec (DeliveryContents) ? New in JDF 1.6	element	Reference to a document to that SHALL be printed and packaged together with the delivered items of this ***Drop***.
FileSpec (MailingList) ?	refelement	A ***FileSpec*** resource pointing to a mailing list. The format of the referenced mailing list is implementation dependent.
Pricing ? Deprecated in JDF 1.3	element	***Pricing*** elements that define the pricing of the complete ***DeliveryIntent*** including any ***DropIntent*** or ***DropItemIntent*** elements that MAY contain further ***Pricing*** elements. In **JDF** 1.3 and beyond, pricing information has been removed and will be handled by the business wrapper around **JDF** (e.g., PrintTalk).

This element contains information about the intended individual drop of a delivery. Attributes that are specified in a ***DropIntent*** element overwrite those that are specified in their parent ***DeliveryIntent*** element. If OPTIONAL values are not specified, they default to the values specified in the ***DeliveryIntent***.

7.5.1 DropIntent

Table 7.43: DropIntent Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AdditionalAmount ? New in JDF 1.2 Deprecated in JDF 1.3	integer	Number of components used to calculate the value of the @*AdditionalPrice* attribute in the ***Pricing***. This value applies to the number of additional items in one ***DropIntent***/***DropItemIntent*** and not to the total additional number of items. If not specified, defaults to the value of ***DeliveryIntent***/@*AdditionalAmount*. In **JDF** 1.3 and beyond, pricing information has been removed and will be handled by the business wrapper around **JDF** (e.g., PrintTalk).
BuyerAccount ? New in JDF 1.2	string	Account ID of the buyer with the delivery service. **Default value is from:** ***DeliveryIntent***/@*BuyerAccount*
DropID ? New in JDF 1.5	string	***DropIntent*** elements with the same @*DropID* are part of the same drop. This attribute is provided to allow items from multiple individual **JDF** jobs to be delivered in one drop.
Earliest ?	TimeSpan	Specifies the earliest time after which the transfer SHALL be made. Within an RFQ or a Quote, at most one of ***Earliest*** or ***EarliestDuration*** SHALL be specified.
EarliestDuration ?	DurationSpan	Specifies the earliest time by which the transfer SHALL be made relative to the date of the purchase order. Within an RFQ or a Quote, at most one of ***Earliest*** or ***EarliestDuration*** SHALL be specified. Within a purchase order, ***EarliestDuration*** SHALL NOT be specified.
Method ? Modified in JDF 1.5	NameSpan	Specifies a delivery method. **Values include those from:** ***Drop***/@*Method*. **Modification note:** Starting in **JDF** 1.5, values have changed.
Pickup ? Deprecated in JDF 1.1	boolean	If "true", the merchandise is picked up. If @*Pickup* = "false", the ***DropIntent*** is delivered to the address specified in ***Company***. If @*Pickup* = "true", the ***DropIntent*** describes an input to the job (e.g., a CD for inserting, a preprinted cover, etc.). In this case, ***Company*** describes the location where the merchandise is picked up.
Required ?	TimeSpan	Specifies the time by which the delivery SHALL be made. Within an RFQ or a Quote, at most one of ***Required*** or ***RequiredDuration*** SHALL be specified.
RequiredDuration ?	DurationSpan	Specifies the time by which the delivery SHALL be made relative to the date of the purchase order. Within an RFQ or a Quote, at most one of ***Required*** or ***RequiredDuration*** SHALL be specified. Within a purchase order, ***RequiredDuration*** SHALL NOT be specified.
ReturnMethod ? New in JDF 1.1	NameSpan	Specifies a delivery method for returning the surplus material, and SHALL NOT be specified unless ***SurplusHandling*** = "Return". **Default value is from:** ***DeliveryIntent***/***ReturnMethod***. **Values include those from:** ***DeliveryIntent***/***ReturnMethod***
ServiceLevel ? New in JDF 1.2	StringSpan	The service level of the specific carrier. **Values include those from:** ***Drop***/***ServiceLevel***
SurplusHandling ? New in JDF 1.1	EnumerationSpan	Describes what SHALL happen with unused or redundant parts of the transfer. **Default value is from:** ***DeliveryIntentSurplusHandling***. **Allowed values are from:** ***DeliveryIntent***/***SurplusHandling***.
Transfer ? New in JDF 1.1	EnumerationSpan	Describes the direction and responsibility of the transfer. **Allowed values are from:** ***Drop***/@*Transfer*.
Company ? Deprecated in JDF 1.1	refelement	Address and further information of the addressee. In **JDF** 1.1 and beyond ***Company*** is a subelement of ***Contact***.

Table 7.43: DropIntent Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Contact * New in JDF 1.1	refelement	Address and further information of the ***Contact*** responsible for the transfer. The actual delivery address is specified as the ***Contact***[contains (@*ContactTypes*, "Delivery")]/***Address***. The actual pickup address is specified as the ***Contact***[contains (@*ContactTypes*, "Pickup")]/***Address***. At most one ***Contact***[contains (@*ContactTypes*, X)]/ SHALL be specified for X equal to "Delivery", "Pickup" or "Billing". Defaults to the ***DeliveryIntent***/***Contact***
DropItemIntent +	element	A ***DropIntent*** MAY consist of multiple products, which are represented by their respective ***PhysicalResource***s. Each ***DropItemIntent*** element describes a number of individual resources that is part of this ***DropIntent***.
FileSpec (DeliveryContents) ? New in JDF 1.6	element	Reference to a document to that SHALL be printed and packaged together with the delivered items of this ***Drop***.
Pricing ? Deprecated in JDF 1.3	element	***Pricing*** element that defines the pricing of the ***DropIntent***. In **JDF** 1.3 and beyond, pricing information has been removed and will be handled by the business wrapper around **JDF** (e.g., PrintTalk).

7.5.2 DropItemIntent

Table 7.44: DropItemIntent Element

NAME	DATA TYPE	DESCRIPTION
AdditionalAmount ? Modified in JDF 1.2 Deprecated in JDF 1.3	integer	Number of components used to calculate the value of the @*AdditionalPrice* attribute in the ***Pricing***. If not specified, defaults to the value of ***DropIntent***/@*AdditionalAmount*. In **JDF** 1.3 and beyond, pricing information has been removed and will be handled by the business wrapper around **JDF** (e.g., PrintTalk).
Amount ?	integer	Specifies the final number of resources delivered. If not specified, defaults to the total amount of the resource that is specified by ***PhysicalResource***s or 1 if this ***DropItemIntent*** specifies a proof. Note that ***DropItemIntent***/@*Amount* corresponds semantically to ***ResourceLink***/@*ActualAmount* and ***DropItem***/@*ActualAmount*.
DropID ? New in JDF 1.5	string	***DropItemIntent*** elements with the same @*DropID* are part of the same drop. This attribute is provided to allow items from multiple individual **JDF** jobs to be delivered in one drop.
OrderedAmount ?	integer	Specifies the original number of resources ordered. If not specified, defaults to the value of @*Amount*. Note that ***DropItemIntent***/@*OrderedAmount* corresponds semantically to ***ResourceLink***/@*Amount* and ***DropItem***/@*Amount*.
Proof ? New in JDF 1.1	string	This ***DropItemIntent*** refers to a proof that is specified in a ***ProofItem*** of the ***ProofingIntent*** of this product intent node. **Constraint:** ***ProofingIntent***/***ProofItem***/@*ProofName* SHALL match @*Proof*. Exactly one of***PhysicalResource*** or @*Proof* SHALL be specified.
Unit ?	string	Unit of measurement for the @*Amount* specified in the ***PhysicalResource***s. **Default value is:** value of @*Unit* defined in the resource described by the ***PhysicalResource***
PhysicalResource ? Modified in JDF 1.1	refelement	Description of the individual item that is delivered. **Constraint:** exactly one of ***PhysicalResource*** or @*Proof* SHALL be specified. **Note:** ***PhysicalResource*** represents a resource that SHALL be an instance of a ***PhysicalResource*** (e.g., ***Component***).
Pricing ? Deprecation in JDF 1.3	element	***Pricing*** element that defines the pricing of the ***DropItemIntent***. **Deprecation note:** Starting with **JDF** 1.3, pricing information has been removed and will be handled by the business wrapper around **JDF** (e.g., PrintTalk).

7.5.3 Pricing

Deprecated in JDF 1.3

The table defining the deprecated ***Pricing*** subelement has been moved to ▸ Section N.6.2 DeliveryIntent Deprecated Subelements.

7.5.4 Payment

Deprecated in JDF 1.3

The table defining the deprecated ***Payment*** subelement has been moved to ▸ Section N.6.2 DeliveryIntent Deprecated Subelements.

7.5.5 CreditCard

Deprecated in JDF 1.3

The table defining the deprecated ***CreditCard*** subelement has been moved to ▸ Section N.6.2 DeliveryIntent Deprecated Subelements.

7.6 EmbossingIntent

New in JDF 1.1

This resource specifies the embossing and/or foil stamping intent for a **JDF** job using information that identifies whether the product is embossed or stamped, and if desired, the complexity of the affected area.

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***EmbossingParams***

Example Partition: "Option", "PageNumber", "Side"

Table 7.45: EmbossingIntent Resource

NAME	DATA TYPE	DESCRIPTION
EmbossingItem +	element	Each embossed image is described by one ***EmbossingItem***.

7.6.1 EmbossingItem

Table 7.46: EmbossingItem Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Direction Modified in JDF 1.3	EnumerationSpan	The direction of the image. **Allowed values are from:** ▸ EmbossDirection.
EdgeAngle ?	NumberSpan	The angle of a beveled edge in degrees. Typical values are an angle of: 30, 40, 45, 50 or 60 degrees. If ***EdgeAngle*** is specified, ***EdgeShape*** = "Beveled" SHALL be specified.
EdgeShape ?	EnumerationSpan	The transition between the embossed surface and the surrounding media can be rounded or beveled (angled). **Allowed values are:** Rounded Beveled
EmbossingType Modified in JDF 1.4	StringSpan	The strings defined in ***EmbossingType*** are whitespace separated combinations of the following tokens. **Allowed values are from:** ▸ EmbossType.
FoilColor ?	EnumerationSpan	Defines the color of the foil material which is used for embossing. **Allowed values are from:** ▸ NamedColor.

Table 7.46: EmbossingItem Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
FoilColorDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If ***FoilColorDetails*** is supplied, ***FoilColor*** SHOULD also be supplied. @*FoilColorDetails* SHOULD be used to specifiy specialized foil properties such as holographic or transparent foils. Example combinations of @*FoilColor* and @*FoilColorDetails* include: **Holographic foils:** @*FoilColor* = "Silver" and @*FoilColorDetails* = "Holographic"; **Matte transparent foil:** @*FoilColor* = "white" and @*FoilColorDetails* = "TransparentMatte".
Height ?	NumberSpan	The height of the levels. This value specifies the vertical distance between the highest and lowest point of the stamp, regardless of the value of ***Direction***.
ImageSize ?	XYPairSpan	The size of the bounding box of one single image.
Level ?	Enumera-tionSpan	The level of embossing. **Allowed values are from:** ▸ EmbossLevel.
Location ?	enumeration	Position of the embossing on the product. **Allowed value is :** ▸ Face.
Position ?	XYPairSpan	Position of the lower left corner of the bounding box of the embossed image in the coordinate system of the surface of the ***Component*** that is selected by @*Location*.
ShapeName ? New in JDF 1.6	NameSpan	Identifier of the embossing tool.
Separation ? New in JDF 1.6	string	@*Separation* the name of the separation within the PDL whose color values are used as the embossing values. A value of 0 means that there is no embossing, a value of 1.0 specifies embossing with full depth.
ToolName ? New in JDF 1.6	string	Name of the embossing tool.

7.7 FoldingIntent

This resource specifies the fold intent for a product excluding folds that are implied by binding. See ▸ Section 7.3 BindingIntent for additional details.

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***CreasingParams***, ***CuttingParams***, ***Fold***, ***FoldingParams***, ***PerforatingParams***

Example Partition: "Option"

Table 7.47: FoldingIntent Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
FoldingCatalog	NameSpan	Describes the folding scheme according to the folding catalog in ▸ Figure 8-33: Fold catalog part 1 and ▸ Figure 8-34: Fold catalog part 2. **Value format is:** "Fn-i" where "n" is the number of finished pages and "i" is either an integer, which identifies a particular fold or the letter "X", which identifies a generic fold (e.g., "F6-2" describes a Z-fold of 6 finished pages, and "F6-X" describes a generic fold with 6 finished pages). **Note:** The folding scheme in this context refers to the folding of the finished product as seen after the cutting, not the folding, of the sheet as seen in production. See ***LayoutIntent***/@*Foliocount* for a discussion of pagination of folded end products.
FoldingDetails ? New in JDF 1.6	string	@*FoldingDetails* is a system dependent descriptor of the folding. @*FoldingDetails* MAY be used to differentiate differing fold dimensions with the same general topology, such as asymmetrical Z-folds. @*FoldingDetails* SHALL NOT be specified if @*FoldCatalog* is not present

Table 7.47: FoldingIntent Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Folds ? Deprecated in JDF 1.1	XYPair	Number of folds in x and in y direction. This attribute specifies the number of folds seen in the sheet after folding, and not the number of fold operations needed to achieve that result. If not specified, it SHALL be inferred from @*FoldingCatalog*. If X and Y are the number of folds in the x and y directions, respectively, the product 2*(X+1)*(Y+1) SHALL always match the *n* of "Fn-i" of @*FoldingCatalog*.
Orientation? New in JDF 1.6	Enumera-tionSpan	@*Orientation* indicates the orientation of the unfolded product with respect to the lay of the fold. A value of Rotate0 SHALL be mapped to the lay of the fold on the lower left of the product prior to folding and the front side of the product oriented in the direc-tion of an upward fold. **Allowed value is from:** ▸ Orientation
Fold * New in JDF 1.1	element	This describes the details of folding operations in the sequence described by the value of @*FoldingCatalog*. ***Fold*** SHALL be specified if non-symmetrical folds are requested.

7.8 HoleMakingIntent

This resource specifies the holemaking intent for a **JDF** job, using information that identifies the type of holemaking op-eration or alternatively, an explicit list of holes. This resource does not specify whether the media will be pre-drilled or the media will be drilled or punched as part of making the product.

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***Hole***, ***HoleLine***, ***HoleMakingParams***, ***Media***

Example Partition: "Option"

Table 7.48: HoleMakingIntent Resource

NAME	DATA TYPE	DESCRIPTION
Extent ? New in JDF 1.2	XYPair	Size (bounding box) of the hole in points when specifying a standard hole pattern in ***HoleType***. If not specified the implied default defined in ▸ Appendix L Hole Pattern Catalog is assumed. Ignored when ***HoleType***/@*Actual* = "Explicit".
HoleReferenceEdge = "Left" New in JDF 1.1	enumeration	The edge of the media relative to where the holes are to be punched. Use with ***HoleType***. **Allowed values are:** Left Right Top Bottom Pattern – Specifies that the reference edge implied by the value of ***HoleType*** in ▸ Appendix L Hole Pattern Catalog is used.
HoleType Modified in JDF 1.1	StringSpan	Predefined hole pattern. Multiple hole patterns are specified as one NMTO-KENS string (e.g., 3-hole ring binding and 4-hole ring binding holes on one piece of media). **Values include:** Explicit – Holes are defined in an array of hole elements. 2HoleEuro – Replace by either R2m-DIN or R2m-ISO Deprecated in JDF 1.0. 3HoleUS – Replace by R3I-US Deprecated in JDF 1.0 4HoleEuro – Replace by R4m-DIN-A4 or R4m-DIN-A5 Deprecated in JDF 1.0. **Allowed values are from:** ▸ Appendix L Hole Pattern Catalog.
HoleList ?	element	Array of all ***Hole*** elements. Used only when ***HoleType***/@*Actual* = "Explicit", oth-erwise this element is not used.

7.9 InsertingIntent

This resource specifies the how to assemble multiple product parts into a final product, using information that identifies page location, position and attachment method. The containing product SHALL be referenced in ***InsertingIntent***/

@*Container*. The receiving component is defined by a @*ProcessUsage* attribute of "Parent". All other input components are mapped to the ***Insert*** elements by their ordering in the ***ResourceLinkPool***.

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***InsertingParams***, ***InsertSheet***

Example Partition: "Option"

Table 7.49: InsertingIntent Resource

NAME	DATA TYPE	DESCRIPTION
Folio	IntegerRangeList	List of potential folios where the insert SHALL be placed. A @*Folio* is defined by its first page in case ***Method***/@*Actual* = "BlowIn" and by the page that the glue is applied in case ***Method***/@*Actual* = "BindIn". In general, a list of folios will only be supplied for ***Method***/@*Actual* = "BlowIn".
GlueType ?	EnumerationSpan	Glue used to fasten the insert. **Allowed values are:** Permanent Removable
Method ?	EnumerationSpan	**Allowed values are:** BindIn – Apply glue to fasten the insert BlowIn – Loose insert.
InsertList	element	List of individual inserts.

7.9.1 InsertList

Table 7.50: InsertList Element

NAME	DATA TYPE	DESCRIPTION
Insert *	element	Individual insert description.

7.9.2 Insert

Table 7.51: Insert Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Folio	IntegerRangeList	List of potential folios where the insert SHALL be placed. A @*Folio* is defined by its first page in case ***Method***/@*Actual* = "BlowIn" and by the page that the glue is applied in case ***Method***/@*Actual* = "BindIn". In general, a list of folios will only be supplied for ***Method***/@*Actual* = "BlowIn". The pages are counted in the order, which is described in @*FolioCount* of the parent ***Component***/***Bundle***.
GlueType ?	EnumerationSpan	Glue used to fasten the insert. **Default value is from**: ***InsertingIntent***/***GlueType***. **Allowed values are:** Removable Permanent
Method ?	EnumerationSpan	Inserting method. **Default value is from**: ***InsertingIntent***/***Method***. **Allowed values are:** BindIn – Apply glue to fasten the insert. BlowIn – Loose insert.
SheetOffset ? Deprecated in JDF 1.1	XYPair	Offset between the ***Component*** to be inserted and finished page identified by folio in the parent ***Component***. In **JDF** 1.2 and beyond, the offset is specified in the offset part of @*Transformation*.
Transformation ?	matrix	Rotation and offset between the ***Component*** to be inserted and the parent ***Component***. If not specified, the identity matrix is applied.

Table 7.51: Insert Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
WrapPages ? New in JDF 1.1	IntegerRangeList	List of pages of the cover that wrap around a ***BindItem*** after all folds are applied. It is sufficient to specify the pages of the "Front" partition of the cover (e.g. cover pages 1 and 4). **Note** that this attribute SHALL NOT be specified if the position of the cover can be derived from the folding information
GlueLine * New in JDF 1.1	element	Array of all ***GlueLine*** elements used to glue in the insert. SHALL NOT be specified in conjunction with *@GlueType*.

7.10 LaminatingIntent

This resource specifies the laminating intent for a **JDF** job using information that identifies whether or not the product is laminated, and if desired, the temperature and thickness of the laminate.

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***LaminatingParams***

Example Partition: "Option"

Table 7.52: LaminatingIntent Resource

NAME	DATA TYPE	DESCRIPTION
Laminated ? Deprecated in JDF 1.1	OptionSpan	If "true", the product is laminated. If no ***LaminatingIntent*** is specified, the product SHALL NOT be laminated.
Surface ?	EnumerationSpan	The surface to be laminated. **Allowed values are:** Back Both Front
Temperature ? Modified in JDF 1.3	EnumerationSpan	Temperature used in the **Laminating** process. **Allowed values are:** Hot Cold
Texture ? New in JDF 1.3	NameSpan	The intended texture of the laminate. **Values include:** Antique – Rougher than vellum surface. Deprecated in JDF 1.6 Calendared – Extra-smooth or polished, uncoated paper. Deprecated in JDF 1.6 Grain Deprecated in JDF 1.6 Linen – Texture of coarse woven cloth. Deprecated in JDF 1.6 Glossy – Glossy laminate. New in JDF 1.6 Matte – Matte laminate. Smooth Deprecated in JDF 1.6 Stipple – Fine pebble finish. Deprecated in JDF 1.6 Vellum – Slightly rough surface. Deprecated in JDF 1.6
Thickness ?	NumberSpan	Thickness of the laminating material. Measured in microns [µm].

7.11 LayoutIntent

Modified in JDF 1.2

This resource records the size of the finished pages for the product component. It does not, however, specify the size of any intermediate results such as press sheets. It also describes how the finished pages of the product component are to be imaged onto the finished media. The size definition of the finished media describes the size of a sheet that is folded to create a product, not the size of a production sheet (e.g., in the press).

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***Layout***, ***LayoutPreparationParams***, ***StrippingParams***

Example Partition: "Option"

Table 7.53: LayoutIntent Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Bleed ? New in JDF 1.5	NumberSpan	Bleed of the artwork in points. The value of 0 means NO bleed. A negative value indicates bleed is needed but the value is unknown.
Dimensions ? New in JDF 1.1	XYPairSpan	Specifies the width (X) and height (Y) in points, respectively, of the trimmed and unfolded (flat) product. For example, ***Dimensions*** for a Z-fold is the unfolded dimensions, while ***FinishedDimensions*** is the folded dimensions if known. Use ***Dimensions*** if ***FinishedDimensions*** is not known. The ***Dimensions*** span element is provided for the rare case that ***FinishedDimensions*** does not unambiguously define the finished product, due to complex folding schemes. If both values are specified, ***FinishedDimensions*** takes precedence
FinishedDimensions ? New in JDF 1.1	ShapeSpan	Specifies the width (X), height (Y) and depth (Z) in points, respectively, of the finished product ***Component*** after all finishing operations, including folding, trimming, etc. If the Z coordinate is 0, it is ignored. Only ***FinishedDimensions*** SHOULD be specified if both ***FinishedDimensions*** and ***Dimensions*** are known
FinishedGrainDirection ? New in JDF 1.2 Deprecated in JDF 1.5	EnumerationSpan	Specifies the media grain direction of the finished page with respect to the binding edge. **Allowed values are:** ParallelToBind – Grain direction is parallel to the binding edge. PerpendicularToBind – Grain direction is perpendicular to the binding edge. **Deprecation note:** Use ***MediaIntent***/***GrainDirection.***
FinishedPageOrientation ? Deprecated in JDF 1.1	enumeration	Indicates the desired orientation of the finished media. **Allowed values are:** Portrait – The short edges of the media are the top and bottom. Landscape – The long edges of the media are the top and bottom. **Note:** In **JDF** 1.1, the finished page orientation is implied by the value of ***Dimensions*** and ***FinishedDimensions*** . If height (X) > width (Y), the product is portrait.
FolioCount = "Booklet" New in JDF 1.1	enumeration	Defines the method used when counting finished pages. **Allowed values are:** Booklet – Each sample of the component consists of two finished pages (e.g., a leaf—the front side and the back side of one sample of the component). Folds as specified by ***FoldingIntent***/@*FoldingCatalog* do not affect pagination. Finished pages are counted in reader order of the pages of the component in the product. Flat – The number of finished pages of one sheet of an individual component is given by the product 2*(X+1)*(Y+1), where x denotes the number of folds in x direction and y denotes the number of folds in y direction. The pages are counted from the upper left of the front side of the top media to the lower right of the back side of the bottom media. "Flat" SHALL be used for non-standard products where the reader order is ambiguous. The page breaks on a sheet are defined by the folds as specified by ***FoldingIntent***/@*FoldingCatalog* (see ▸ Figure 8-33: Fold catalog part 1 and ▸ Figure 8-34: Fold catalog part 2) for the product. All sheets are counted, even if they are not included in the product (e.g., due to a ***ShapeCuttingIntent***).
NumberUp = "1 1" Modified in JDF 1.2	XYPair	Specifies a regular, multi-up grid of page cells into which content pages are mapped. The first value specifies the number of columns of page cells and the second value specifies the number of rows of page cells in the multi-up grid (both numbers are integers). At most one of ***Layout*** or @*NumberUp* SHALL be specified.
Orientation ? New in JDF 1.6	enumeration	@*Orientation* SHALL specify the orientation of the artwork on the surface as defined by @*Sides*. @*Orientation* is used to define products such as back-lit displays, where the orientation of the image with respect to the final product is rotated or mirrored. **Allowed value is from:** ▸ Orientation

Table 7.53: LayoutIntent Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Pages ? New in JDF 1.1 Modified in JDF 1.2	IntegerSpan	Specifies the number of finished pages (surfaces) of the product component, including blank pages. This value SHALL be an even number. For example, the value for @***Pages*** for a two-sided booklet with seven reader pages would be "8", whether the booklet were saddle stitched or glued.
PageVariance ? New in JDF 1.1	IntegerSpan	Specifies the number of non-identical finished pages of the product component (i.e., the number of distinct master pages copied to produce the product). If not specified, the value of pages is used as the default. For example, if there are ten finished pages, in which three are identical, ***PageVariance***/@*Actual* = "8" since it would take eight master copies to produce the product.
RotatePolicy ? New in JDF 1.2	enumeration	Specifies the policy to automatically rotate the image to optimize the fit of the image to the page container. For instance, individual landscape pages in a portrait document MAY automatically be rotated. The page container is one cell on the NUp grid of the ***Media*** defined in ***Dimensions*** or ***FinishedDimensions.*** **Allowed values are:** NoRotate – Do not rotate. RotateOrthogonal – Rotate by 90° in either direction. RotateClockwise – Rotate clockwise by 90°. RotateCounterClockwise – Rotate counter-clockwise by 90°.
Sides ? Modified in JDF 1.2	enumeration	Indicates whether contents are to be printed on one or both sides of the media. **Allowed value is from:** ▸ Sides
SizePolicy ? New in JDF 1.2	EnumerationSpan	Allows printing even if the container size defined in ***Dimensions*** or ***FinishedDimensions*** ***FinishedDimensions*** does not match the requirements of the data. The page container is one cell on the NUp grid of the ***Media*** defined in ***Dimensions*** or ***FinishedDimensions*** **Allowed values are:** ClipToMaxPage – The page contents SHALL be clipped to the size of the container. The printed area is centered in the source image. FitToPage – The page contents SHALL be scaled up or down to fit the container. The aspect ratio is maintained. ReduceToFit – The page contents SHALL be scaled down but not scaled up to fit the container. The aspect ratio is maintained. Tile – The page contents SHALL be split into several tiles, each printed on its own container.
SpreadType ? New in JDF 1.6	enumeration	@*SpreadType* SHALL specify the treatment of individual PDF pages referenced by the DPart node for which SpreadType is specified for imposition purposes. **Allowed values are:** Combine - the content of each page SHALL be combined into a surface that folding etc. SHALL apply to. Examples include foldouts that are delivered as multiple pages or a trifold that is delivered as 6 logical pages. FoldOut - the content of each page SHALL be imaged onto a foldout. Each page SHALL be processed as a surface of the foldout. SinglePage - the content of each page SHALL be imaged in a single cell in imposition. Spread - the content of each page SHALL be imaged as a single surface onto the final product. Examples include wraparound covers. Spread SHOULD NOT be provided for adjacent pages that are not imaged onto the same surface. **Note:** If not specified, @*SpreadType* defaults to SinglePage.
Layout ? New in JDF 1.1	refelement	Specifies the details of a more complex ***Layout***. At most one of ***Layout*** or @*NumberUp* SHALL be specified. Note that the ***Layout*** specified in ***LayoutIntent*** specifies the layout definition of the finished product and not the layout of the production sheets.

7.12 MediaIntent

Modified in JDF 1.2

This resource describes the media to be used for the product component. In some cases, the exact identity of the medium is known, while in other cases, the characteristics are described and a particular stock is matched to those characteristics.

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***Media***

Example Partition: "Option"

Table 7.54: MediaIntent Resource (Sheet 1 of 5)

NAME	DATA TYPE	DESCRIPTION
BackCoatings ?	EnumerationSpan	Identical to ***FrontCoatings***, but applied to the back surface of the media. **Default value is from:** @***FrontCoatings***. **Allowed values are from:** ▸ Coating**.**
Brightness ?	NumberSpan	Reflectance percentage of diffuse blue reflectance as defined by ▸ [ISO2470:1999]. The reflectance is reported per ▸ [ISO2470:1999] as the diffuse blue reflectance factor of the paper or board in percent to the nearest 0.5% reflectance factor.
BuyerSupplied ?	OptionSpan	Indicates whether the customer will supply the media. Note that the ***Media*** resource can be used to specify additional media requirements, particularly when the media is supplied by the customer.
Dimensions ? Deprecated in JDF 1.2	XYPairSpan	Specifies the size of the supplied media in points if ***BuyerSupplied*** evaluates to "true". ***Dimensions*** SHALL be ignored if ***BuyerSupplied*** evaluates to "false". Note that the size of the finished product is always specified in ***LayoutIntent***/***FinishedDimensions***. In **JDF** 1.2 and beyond the specifics of ***BuyerSupplied*** media SHOULD be specified using a ***Media*** resource. The dimensions of the finished product are specified with ***LayoutIntent***/***Dimensions*** or ***LayoutIntent***/***FinishedDimensions***.
Flute ? New in JDF 1.4	NameSpan	Single, capital letter that specifies the flute type of corrugated media. Although the classification of flutes using a letter code “A”, “B”, etc., are used very frequently (e.g., in the specification of the order for a box), there seems to be no agreement on the exact numerical specification of those categories. Slightly varying numbers for flute size and frequency can be found between regions (European versus US) and between vendors. **Values include those from:** ***Media***/@*Flute*
FluteDirection ? New in JDF 1.4	EnumerationSpan	Direction of the flute of corrugated media in the coordinate system of the product. Values are the same as ***Media***/@*FluteDirection* with slightly different description. **Allowed values are:** LongEdge – Along the longer axis as defined by ***LayoutIntent***/***Dimensions***. ShortEdge – Along the shorter axis as defined by ***LayoutIntent***/***Dimensions***. XDirection – Along the X-axis of the ***LayoutIntent*** coordinate system YDirection – Along the Y-axis of the ***LayoutIntent*** coordinate system
FrontCoatings ? Modified in JDF 1.2	EnumerationSpan	What pre-process coating has been applied to the front surface of the media. **Allowed values are from:** ▸ Coating**.**

Table 7.54: MediaIntent Resource (Sheet 2 of 5)

NAME	DATA TYPE	DESCRIPTION
Grade ? Modified in JDF 1.5 Deprecated in JDF 1.6	IntegerSpan	The intended grade of the media on a scale of 1 through 5. ***Grade*** is ignored if ***MediaType***/@*Actual* is not "Paper". ***Grade*** of paper material is defined in accordance with the paper "types" set forth in ▸ [ISO12647-2:2004]. Offset printing paper types are defined with integer values. If a workflow supports @*ISOPaperSubstrate*, and both @*Grade* and @*ISOPaperSubstrate* are present, it SHALL use @*ISOPaperSubstrate*. **Note:** ▸ [ISO12647-2:2004] paper grade @*Grade* values do not align with U.S. GRACOL paper grade @*Grade* values (e.g., ▸ [ISO12647-2:2004] type 1 does not equal U.S. GRACOL grade 1). **Values include:** 1 – Gloss-coated paper 2 – Matt-coated paper 3 – Gloss-coated, web paper 4 – Uncoated, white paper 5 – Uncoated, yellowish paper **Modification note:** Starting with **JDF** 1.5, condition for new @*ISOPaperSubstrate* added.
GrainDirection ? New in JDF 1.2 Modified in JDF 1.5	EnumerationSpan	Direction of the grain in the coordinate system defined by ***LayoutIntent***/***Dimensions*** or ***LayoutIntent***/***FinishedDimensions***. **Allowed values are:** Any – No restrictions apply to grain direction. New in JDF 1.5 ShortEdge – Parallel to the shorter axis of the finished page. Deprecated in JDF 1.5 LongEdge – Parallel to the longer axis of the finished page. Deprecated in JDF 1.5 SameDirection – All ordered items SHALL have the same grain direction relative to the finished product. The printer may choose which one. New in JDF 1.5 XDirection – Along the X-axis of the ***LayoutIntent*** coordinate system New in JDF 1.4 YDirection – Along the Y-axis of the ***LayoutIntent*** coordinate system. New in JDF 1.4 **Deprecation note:** For "ShortEdge", use "YDirection" if the product is landscape, and use "XDirection" if the product is portrait. For "LongEdge", use "XDirection" if the product is landscape, use "YDirection" if the product is portrait.
HoleCount ? Deprecated in JDF 1.1	IntegerSpan	The intended number of holes that are to be punched in the media (either pre- or post-punched). Starting with **JDF** 1.1, use ***HoleType*** which includes the number of holes.
HoleType ? New in JDF 1.1	StringSpan	Predefined hole pattern that specifies the pre-punched holes in the media. Multiple hole patterns are specified as one NMTOKENS string (e.g., 3-hole ring binding and 4-hole ring binding holes on one piece of media). **Values include:** None – no holes **Values include those from:** ▸ Appendix L Hole Pattern Catalog**.**
ISOPaperSubstrate ? New in JDF 1.5	EnumerationSpan	@*ISOPaperSubstrate* supersedes @*Grade* and adds new values to allow for improved papers. If a workflow supports @*ISOPaperSubstrate*, and both @*Grade* and @*ISOPaperSubstrate* are present, it SHALL use @*ISOPaperSubstrate*. @*ISOPaperSubstrate* SHALL specify the type of paper material defined in accordance with the print substrate set forth in ▸ [ISO12647-2:2013]. **Allowed value is from:** ▸ ISOPaperSubstrate **Note:** See ▸ Section E.3 Paper Grade for a mapping to the paper grade values defined in ISO12647-2:2004
MediaColor ?	EnumerationSpan	Color of the media. If more-specific, specialized or site-specific media color names are needed, use ***MediaColorDetails***. **Allowed values are from:** ▸ NamedColor

Table 7.54: MediaIntent Resource (Sheet 3 of 5)

NAME	DATA TYPE	DESCRIPTION
MediaColorDetails ? New in JDF 1.2	StringSpan	A more specific, specialized or site-defined name for the media color. If ***MediaColorDetails*** is supplied, ***MediaColor*** SHOULD also be supplied. Note that there is a one-to-many relationship between entries in ***MediaColor*** and ***MediaColorDetails*** (e.g., ***MediaColorDetails*** values of "Burgundy" and "Ruby" both correspond to a ***MediaColor*** of "DarkRed").
MediaQuality ? New in JDF 1.4	StringSpan	Named quality description of the media. For folding carton quality, multiple named quality description systems are in use (e.g., GC1, SBB, etc.). For an overview see http://www.procarton.com/?section=fact_file_publications. When used in a general product description, Media with the same @*MediaQuality* are identical from the customer point of view. Thus characteristics such as weight, coatings or recycling percentage are identical whereas lot or sheet dimension may vary based on production or warehousing requirements.
MediaSetCount ?	integer	When the input media is grouped in sets, identifies the number of pieces of media in each set. For example, if the @*UserMediaType* is "PreCutTabs", a @*MediaSetCount* of 5 would indicate that each set includes 5 tab sheets.
MediaType ? New in JDF 1.1 Modified in JDF 1.5	NameSpan	Describes the medium being employed. **Allowed values are:** CorrugatedBoard New in JDF 1.3 Disc – CD or DVD disc to be printed on. New in JDF 1.2 Other – Any other media. For this value ***MediaTypeDetails*** SHOULD also be specified Paper SelfAdhesive New in JDF 1.3 Textile New in JDF 1.5 Transparency Vinyl New in JDF 1.5
MediaTypeDetails ? New in JDF 1.3	NameSpan	Describes additional details of the medium described in ***MediaType***. **Values include those from:** ***Media***/@*MediaTypeDetails* **Values include:** Continuous – Continuously connected sheets of an opaque material. Which edge is connected is not specified. Transparency – Separately cut sheets of a transparent material. **Note:** Values from ***Media***/@*MediaTypeDetails* are RECOMMENDED. However, some process related values, such as "DryFilm", SHOULD NOT be used for this attribute.
MediaUnit ? Deprecated in JDF 1.2	EnumerationSpan	Describes the format of the media as it is delivered to the device. **Allowed values are:** Roll Sheet **Deprecation note:** Deprecated because intent attributes and span elements pertain to finished product, not the raw media format. If ***BuyerSupplied*** = "true", then the ***Media*** resource can be used to provide this span element.
Opacity ? Modified in JDF 1.2	EnumerationSpan	The opacity of the media. See ***OpacityLevel*** to specify the degree of opacity for any of these values. **Allowed values are from:** ▸ Opacity.
OpacityLevel ? New in JDF 1.2	NumberSpan	Normalized TAPPI opacity, (Cn), as defined and computed in ▸ [ISO2471:1998]. Refer also to ▸ [TAPPI T519] for calculation examples.
PrePrinted = "false"	boolean	Indicates whether the media is preprinted.
Recycled ? Deprecated in JDF 1.2	OptionSpan	If "true", recycled media is requested. In **JDF** 1.2 and beyond, use ***RecycledPercentage***.

Table 7.54: MediaIntent Resource (Sheet 4 of 5)

NAME	DATA TYPE	DESCRIPTION
RecycledPercentage ? New in JDF 1.2	NumberSpan	The percentage, between 0 and 100, of recycled material that the media is expected to contain.
StockBrand ?	StringSpan	Strings providing available brand names. The customer might know exactly what paper is to be used. Example is "Lustro" or "Warren Lustro" even though the manufacturer name is included.
StockType ?	NameSpan	Strings describing the available stock. *@StockType* defines the base size when calculating North American or Japanese paper weights. See ▸ Section Paper Weight and Grade for details including predefined values. **Values include those from:** ***Media*****/@***StockType*
Texture ?	NameSpan	The intended texture of the media. **Values include those from:** ***Media*****/@***Texture***.**
Thickness ? New in JDF 1.1	NumberSpan	The thickness of the chosen medium. Measured in microns [µm].
UserMediaType ? Deprecated in JDF 1.6	NMTOKEN	A human-readable description of the type of media. The value can be used by an operator to select the correct media to load. The semantics of the values will be site-specific. **Values include:** Continuous – Continuously connected sheets of an opaque material. Which edge is connected is not specified. ContinuousLong – Continuously connected sheets of an opaque material connected along the long edge. ContinuousShort – Continuously connected sheets of an opaque material connected along the short edge. Envelope – Envelopes that can be used for conventional mailing purposes. EnvelopePlain – Envelopes that are not preprinted and have no windows. EnvelopeWindow – Envelopes that have windows for addressing purposes. FullCutTabs – Media with a tab that runs the full length of the medium so that only one tab is visible extending out beyond the edge of non-tabbed media. Labels – Label stock (e.g., a sheet of peel-off labels). Letterhead – Separately cut sheets of an opaque material including a letterhead. MultiLayer – Form medium composed of multiple layers which are preattached to one another (e.g., for use with impact printers). MultiPartForm – Form medium composed of multiple layers not preattached to one another; each sheet MAY be drawn separately from an input source. Photographic – Separately cut sheets of an opaque material to produce photographic quality images. PreCutTabs – Media with tabs that are cut so that more than one tab is visible extending out beyond the edge of non-tabbed media. Stationery – Separately cut sheets of an opaque material. TabStock – Media with tabs (either precut or full-cut). Transparency – Separately cut sheets of a transparent material. **Deprecation note:** Starting with **JDF** 1.6, use *@MediaTypeDetails*.
USWeight ? Deprecated in JDF 1.2	NumberSpan	The intended weight of the media, measured in pounds per ream of basis size. At most one of ***Weight*** or ***USWeight*** SHALL be specified. If known, ***Weight*** SHOULD be specified in grammage (g/m^2). In **JDF** 1.2 and beyond, use ***Weight***.
Weight ?	NumberSpan	The intended weight of the media, measured in grammage (g/m^2) of the media. See ▸ Appendix E Paper Weight and Grade for an explanation of how to calculate the US weight from the grammage for different stock types.
Certification * New in JDF 1.6	element	Each ***Certification*** specifies a minimum requested paper certification level.

Table 7.54: MediaIntent Resource (Sheet 5 of 5)

NAME	DATA TYPE	DESCRIPTION
MediaLayers ? New in JDF 1.4	element	Subelement describing the layer structure of media such as corrugated or self adhesive materials.

7.13 NumberingIntent

Deprecated in JDF 1.5

The table defining the deprecated ***NumberingIntent*** subelement has been moved to ▸ Section N.6.3 NumberingIntent.

7.14 PackingIntent

This resource specifies the packaging intent for a **JDF** job, using information that identifies the type of package, the wrapping used, and the shape of the package. Note that this specifies packing for shipping only, not packing of items into custom boxes, etc. Boxes are convenience packaging and are not envisioned to be protection for shipping. Cartons perform this function. All quantities are specified as finished pieces per wrapped/boxed/carton or palletized package. The model for packaging is that products are wrapped together, wrapped packages are placed in *boxes*, boxes are placed in *cartons*, and cartons are stacked on *pallets*.

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***BoxPackingParams***, ***Bundle***, ***Component***, ***PalletizingParams***, ***Pallet***, ***ShrinkingParams***, ***StackingParams***, ***Strap***, ***StrappingParams***, ***WrappingParams***

Example Partition: "Option"

Table 7.55: PackingIntent Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BoxedQuantity ?	IntegerSpan	How many units of *product* in a box.
BoxShape ?	ShapeSpan	Describes the length, width and height of the box, in points.
CartonMaxWeight ?	NumberSpan	Maximum gross weight of an individual carton, in kilograms.
CartonQuantity ?	IntegerSpan	How many units of *product* in a carton.
CartonShape ?	ShapeSpan	Describes the length, width and height of the carton, in points. For example, 288 544 1012
CartonStrength ?	NumberSpan	Strength of the carton, in kilograms.
FoldingCatalog ?	NameSpan	Describes the folding scheme for folding the product for packaging as specified in the folding catalog in ▸ Figure 8-33: Fold catalog part 1 and ▸ Figure 8-34: Fold catalog part 2. **Value format is:** "Fn-i" where "n" is the number of finished pages and "i" is either an integer, which identifies a particular fold or the letter "X", which identifies a generic fold (e.g., "F6-2" describes a Z-fold of 6 finished pages, and "F6-X" describes a generic fold with 6 finished pages). **Note:** The folding scheme in this context refers to the folding of the finished product for packaging only. The folding has no effect on the page/Folio definition.
PalletCornerBoards ? New in JDF 1.3	NameSpan	Additional protective corner boards for packaging on a pallet: **Values include:** Corners – Corner boards on 8 corners of the pallet. VerticalEdge – Corner boards along the 4 vertical edges.
PalletMaxHeight ?	NumberSpan	Maximum height of a loaded pallet, in points.
PalletMaxWeight ?	NumberSpan	Maximum weight of a loaded pallet, in kilograms.
PalletQuantity ?	IntegerSpan	Number of *product* per pallet
PalletSize ?	XYPairSpan	Describes the length and width of the pallet, in points (e.g., "3500 3500").

Table 7.55: PackingIntent Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
PalletType ?	NameSpan	Type of pallet used. **Values include: those from:** ***Pallet*****/@***PalletType*
PalletWrapping ?	NameSpan	Wrapping of the completed pallet. **Values include:** Banding None – explicitly requests no wrapping. StretchWrap
WrappedQuantity ?	IntegerSpan	Number of units of product per wrapped package.
WrappingMaterial ?	NameSpan	**Values include:** None – explicitly requests no wrapping. PaperBand Polyethylene RubberBand ShrinkWrap

7.15 ProductionIntent

This resource specifies the manufacturing intent and considerations for a **JDF** job using information that identifies the desired result or specified manufacturing path. If specific details of print quality, such as color quality need to be specified, *@ProductionResources* SHOULD reference a ***QualityControlParams*** resource.

Resource Properties

Resource Class: Intent

Process Resource Pairing: All

Example Partition: "Option"

Table 7.56: ProductionIntent Resource

NAME	DATA TYPE	DESCRIPTION
PrintPreference ?	Enumera-tionSpan	Intended result or goal. **Allowed values are:** Balanced – Request for a manufacturing process that balances the requirements for cost, speed and quality. CostEffective – Request for the most cost effective manufacturing process. Fastest – Request for the most time effective manufacturing process. Cost and quality can be sacrificed for a fast turnaround time. HighestQuality – Request for the manufacturing process which will result in the highest quality.
PrintProcess ? Modified in JDF 1.3	NameSpan	Print process requested. **Values include:** Electrophotography Flexography Gravure Inkjet Lithography – Includes offset printing Letterpress Screen Thermography **Modification Note:** Starting with **JDF** 1.3, the data type of ***PrintProcess*** is expanded from EnumerationSpan to ***NameSpan***.
Resource * New in JDF 1.3	refelement	Any production resources that are provided by the customer. Some examples include buyer specified media or ink or specific parameter setups. Note that ***DeliveryIntent*** SHALL be specified for any ***PhysicalResource*** elements that are physically supplied by the customer.

7.16 ProofingIntent

This resource product intent element specifies the prepress proofing intent for a **JDF** job using information that identifies the type, quality, brand name and overlay of the proof. The proofs defined in ***ProofingIntent*** define the proofs that will be provided to the customer and does not specify internal production proofs. The delivery options of proofs MAY be specified in ***DeliveryIntent***.

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***ApprovalParams***, ***ApprovalSuccess***,***ColorantControl***, ***ColorSpaceConversionParams***, ***ExposedMedia***, ***ImageSetterParams***, ***InterpretingParams***, ***Layout***, ***Media***, ***RenderingParams***, ***ScreeningParams***, ***SeparationControlParams***, ***StrippingParams***

Example Partition: "Option"

Table 7.57: ProofingIntent Resource

NAME	DATA TYPE	DESCRIPTION
PreflightItem *	element	***PreflightItem*** defines the preflight rules for a range of pages.
ProofItem * New in JDF 1.1	element	Specifies the details of the proofs that are needed. If no ***ProofItem*** exists in a ***ProofingIntent***, it explicitly specifies that no proofs are desired.

7.16.1 PreflightItem

Table 7.58: PreflightItem Element

NAME	DATA TYPE	DESCRIPTION
PageIndex ?	IntegerRangeList	Index of pages that SHALL be proofed in reader order. If @*PageIndex* is not specified, then all pages SHALL be proofed.
PreflightLevel ?	enumeration	Level of content data checking / preflighting. The details are implementation specific. **Allowed values are:** Basic – Check only for severe errors. Examples include missing fonts, unknown file format, incorrect page size, missing passwords. Extended – Check for additional errors that can degrade output quality and can be resolved by the customer. Examples include: low image resolution, unknown color space details. Premium - Highest available check for additional errors. This level MAY include manual repairs by the printer

7.16.2 ProofItem

All parameters of ***ProofingIntent*** have been moved into ***ProofItem*** in **JDF** 1.1

Table 7.59: ProofItem Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Amount ? Modified in JDF 1.1	IntegerSpan	Specifies the total number of copies of this proof that is needed. If not specified, it defaults to an IntegerSpan with @*Preferred* = "1".
BrandName ? Modified in JDF 1.1	StringSpan	Brand name of the proof (e.g., Iris).
ColorType ? Modified in JDF 1.1	EnumerationSpan	Color quality of the proof. **Allowed values are:** Monochrome – Generic single color printing condition (e.g., black and white or one single spot color). BasicColor – Color does not match precisely. This implies the absence of a color matching system. MatchedColor – Color is matched to the output of the press using a color matching system.

Table 7.59: ProofItem Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Contract = "false" Modified in JDF 1.1	boolean	Requires proof to be a legally binding, accurate representation of the image to be printed (i.e., color quality requirements have been met when the printed piece acceptably matches the proof).
HalfTone ? Modified in JDF 1.1	OptionSpan	Specifies whether the proof SHALL emulate halftone screens.
ImageStrategy ? New in JDF 1.2	Enumera-tionSpan	Identifies which images (OPI or other) will be printed on a proof or displayed as a soft proof. **Allowed values are:** NoImages – No images are imaged on the proof. LowResolution – Low resolution images are imaged on the proof. HighResolution – High resolution production images are imaged on the proof, resulting in proofs that accurately represent the final product.
PageIndex ? New in JDF 1.1	Inte-gerRangeList	Index list of pages that are to be proofed in the ***ArtDeliveryIntent***/***RunList***/***PageList***. If no range is specified then all pages SHALL be proofed.
ProofName ? New in JDF 1.1	string	Name of the ***ProofItem***. This field SHALL be specified if delivery of a proof is specified in ***DeliveryIntent DeliveryParams***.
ProofTarget ? Modified in JDF 1.1	URL	Identifies a remote target for the proof output in a remote proofing environ-ment. This can be either a soft or a hard proofing target. The file to be dis-played or output SHALL be sent to the URL specified in @*ProofTarget*.
ProofType ? Modified in JDF 1.1	Enumera-tionSpan	The kind of proof. **Allowed values are:** Page – Page proof Imposition – Imposition proof None – No proof is needed.
Technology ? Modified in JDF 1.1	NameSpan	Technology used for making the proof. **Values include:** BlueLine DyeSub InkJet Laser PressProof SoftProof
ApprovalParams ? New in JDF 1.2	refelement	List of people (e.g., a customer, printer or manager) who can sign the ***ApprovalSuccess***.
SeparationSpec * New in JDF 1.1	element	Separations that are to be proofed. If not specified, all separations SHALL be proofed.

7.17 PublishingIntent

New in JDF 1.3

PublishingIntent specifies publishing metadata that are of general interest for prepress, press and postpress. The data include details on the general structure of product being published.

Resource Properties

Resource Class: Intent

Process Resource Pairing: —

Example Partition: "Edition"

Table 7.60: PublishingIntent Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Circulation ?	IntegerSpan	Specifies the number of copies to be published.

Table 7.60: PublishingIntent Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ContentDataRefs ? New in JDF 1.4	IDREFS	IDs of ***ContentData*** elements in the referenced ***ContentList***. ***ContentData*** elements provide metadata related to the product to be published. @*ContentDataRefs* SHALL NOT be specified if no ***ContentList*** is specified.
IssueDate	TimeSpan	Publication date of the issue.
IssueName	StringSpan	The name of a the publication.
IssueType Modified in JDF 1.4	NameSpan	Defines the product type of the issue. **Values include:** Magazine – The publication is a magazine Newspaper – The publication is a newspaper Supplement – The publication is a supplement to a magazine or newspaper. New in JDF 1.4.
ContentList ? New in JDF 1.4	refelement	***ContentList*** with additional metadata.

7.18 ScreeningIntent

New in JDF 1.2

This Resource specifies the screening intent parameters desired for a **JDF** job.

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***ScreeningParams***, ***SeparationControlParams***

Example Partition: "Option"

Table 7.61: ScreeningIntent Resource

NAME	DATA TYPE	DESCRIPTION
DotSize ?	NumberSpan	Specifies the dot size of the screen in microns [µm] when FM screening is used, otherwise ***DotSize*** is ignored.
Frequency ?	NumberSpan	Specifies the line frequency of the screen in lines per inch (lpi) when AM screening is used, otherwise ***Frequency*** is ignored.
FrequencySelection ?	EnumerationSpan	Selects the AM or FM frequency range. **Allowed values are:** LowestFrequency – Lowest AM or FM frequency supported. MiddleFrequency – Middle AM or FM frequency supported HighestFrequency – Highest AM or FM frequency supported
ScreeningType ?	EnumerationSpan	General type of screening. **Allowed values are:** AM – Can be line or dot. FM

7.19 ShapeCuttingIntent

ShapeCuttingIntent describes finishing of products with irregular shapes, including diecutting and adding windows to envelopes.

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***CuttingParams***, ***ShapeCuttingParams***

Example Partition: "Option"

Table 7.62: ShapeCuttingIntent Resource

NAME	DATA TYPE	DESCRIPTION
ShapeCut *	element	Array of all ***ShapeCut*** elements. Used when each shape is exactly specified.

7.19.1 ShapeCut

Table 7.63: ShapeCut Element

NAME	DATA TYPE	DESCRIPTION
CutBox ?	rectangle	Specification of a rectangular window. An orthogonal line MAY be defined by specifying a rectangle with identical dimensions.
CutDepth ? New in JDF 1.6	enumeration	**Allowed values are:** Full – The form is completely cut out or perforated. Partial – The form is not completely cut out or perforated. The exact depth MAY be specified in ***ShapeCuttingParams***.
CutOut = "false"	boolean	@*CutOut* specifies whether the inside or outside of the ***ShapeCut*** SHALL be removed. If @*CutOut*="true", the inside of a specified shape SHALL be removed otherwise the outside of a specified shape SHALL be removed. An example of an inside shape is a window, while an example of an outside shape is a shaped greeting card.
CutPath ? Modified in JDF 1.2	PDFPath	Specification of a complex path. This MAY be an open path in the case of a single line.
CutType ? Modified in JDF 1.1	EnumerationSpan	Type of cut or perforation used. **Allowed values are:** Cut – Full cut. Perforate – Interrupted perforation that does not span the entire sheet.
Material ?	StringSpan	Transparent material that fills a shape (e.g., an envelope window) that was cut out when @*CutOut* = "true".
Pages ?	IntegerRangeList	List of finished pages to which this shape SHALL be applied. Only the recto finished page of a leaf SHOULD be specified.
ShapeDepth ? New in JDF 1.1 Deprecated in 1.6	NumberSpan	Depth of the shape cut. Measured in microns [µm]. If not specified, the shape is completely cut. **Deprecation Note:** From version 1.6 use @*CutDepth* instead.
ShapeType Modified in JDF 1.3	EnumerationSpan	Describes any precision cutting other than hole making. **Allowed values are:** Line – The coordinates specified in @*CutBox* specify the end points of a straight line. Path - Any irregular shape. Additional details SHOULD be provided in @*CutPath* or @*ShapeTypeDetails*. Rectangular – The coordinates specified in @*CutBox* specify the lower left and upper right coordinates of a rectangle. Round - Circular or elliptical shape depending on the aspect ratio of @*CutBox*. RoundedRectangle – Rectangle with rounded corners. The coordinates specified in @*CutBox* specify the outer bounds of the rectangle. New in JDF 1.3
ShapeTypeDetails ?	string	A more specific, specialized or site-defined name for the shape of the ***ShapeCut***.
TeethPerDimension ?	NumberSpan	Number of teeth in a given perforation extent in teeth/point. MicroPerforation is defined by specifying a large number of teeth (n > 1000).

7.20 SizeIntent

Deprecated in JDF 1.1

SizeIntent has been deprecated in **JDF** 1.1. All contents have been moved to ***LayoutIntent***.

7.21 VariableIntent

New in JDF 1.6

VariableIntent specifies the variations of the content for printed data with variable content such as lottery tickets or direct mail.

If the product that links to ***VariableIntent*** has sub-products, then the ComponentLink/@*Amount* of the referenced sub-products SHALL specify the estimated amounts of the respective products. ***VariableIntent*** SHALL NOT be specified if ***InsertingIntent*** or ***BindingIntent*** are specified.

Table 7.64: VariableIntent Element

NAME	DATA TYPE	DESCRIPTION
Area ?	NumberSpan	Percentage of the document that can contain variable content
AveragePages ?	IntegerSpan	*AveragePages* SHALL specify the average number of printed pages in each record.
MaxPages ?	IntegerSpan	@*MaxPages* SHALL specify the maximum number of printed pages in each record. @*MaxPages* SHALL NOT be smaller than @*AveragePages*.
MinPages ?	IntegerSpan	@*MinPages* SHALL specify the minimum number of printed pages in each record. @*MinPages* SHALL NOT be larger than @*AveragePages*.
NumberOfCopies ?	IntegerSpan	Average number of copies of each record. This value SHALL equal "1" for fully variable data.
VariableType ?	EnumerationSpan	Type of variable content. **Allowed Values are (in order of rising complexity):** OneLine - A single line of text data is variable. OneLine includes simple numbering applications. AddressField - Multiple lines of text data are variable. IdentificationField - The variable data includes a Barcode or QR-Code. Area - The area as defined in area is fully variable.
VariableQuality ?	EnumerationSpan	VariableQuality specifies the desired quality of the variable data. **Allowed Values are:** Simple - The variable text MAY be recognized as printed by a different technology such as dot matrix or simple inkjet overprints. Imprint - : The variable data SHOULD be similar to the non-variable part but MAY be imprinted. Full - All data SHOULD be printed with the same technology.
ColorsUsed	element	Array of colorant separation names that are required to print the variable part of the documents. The values that are specified in ***ColorsUsed*** SHALL also be specified in ***ColorIntent***/***ColorsUsed***. See ***ColorIntent***/***ColorsUsed*** for additional details

Resources

This chapter provides a list (in alphabetical order) of all resources.

8.1 AdhesiveBindingParams

Deprecated in JDF 1.1

See ▸ Section N.7.1 AdhesiveBindingParams for details of this deprecated resource.

8.2 ApprovalParams

This ***Resource*** provides the details of an **Approval** process.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	***ProofItem***, ***ConventionalPrintingParams***, ***DigitalPrintingParams***
Intent Pairing	***ProofingIntent***
Input of Processes:	**Approval**

Table 8.1: ApprovalParams Resource

NAME	DATA TYPE	DESCRIPTION
MinApprovals = "1" New in JDF 1.2	integer	Minimum number of ***ApprovalPerson***[@ *ApprovalRole* = "Group"] whose associated person SHALL sign the ***ApprovalSuccess*** for the ***ApprovalSuccess*** to be "Available".
ApprovalPerson *	element	List of people (e.g., a customer, printer or manager) who can sign the approval.

8.2.1 ApprovalPerson

ApprovalPerson specifies the details of person who is responsible for signing an approval

Table 8.2: ApprovalPerson Element

NAME	DATA TYPE	DESCRIPTION
ApprovalRole = "Obligated" New in JDF 1.2 Modified in JDF 1.3	enumeration	Role of the ***ApprovalPerson***. **Allowed values are:** Approvinator – The decision of this approver immediately overrides the decisions of the other approvers and ends the approval cycle. The "Approvinator" NEED NOT sign for the approval to become valid. New in JDF 1.3 Group – The approver belongs to a group of which @*MinApprovals* members SHALL sign. Informative – The approver is informed of the **Approval** process, but the approval is still valid, even without his approval. Obligated – The approver SHALL sign the approval.
ApprovalRoleDetails ? New in JDF 1.3	string	Additional details on the @*ApprovalRole*.
Obligated ? Deprecated in JDF 1.2	boolean	If "true", the person has to sign this approval. In **JDF** 1.2 and beyond, use @*ApprovalRole*.
Contact	refelement	Contact (e.g., a customer, printer or manager) who SHALL sign the approval. There SHALL be a ***Contact***[contains (@*ContactTypes*, "Approver")].

8.3 ApprovalSuccess

The signed ***ApprovalSuccess*** resource provides the signature that indicates that a resource has been approved. This is frequently used to model the success of a soft proof, color proof, printing proof or any other sort of proof.

Resource Properties

Resource Class: Parameter

Example Partition: "DocIndex", "DocRunIndex", "RunIndex", "RunPage", "RunTags", "DocTags", "PageTags", "SetTags", "SetIndex", "SheetName", "Side", "SignatureName", "TileID"

Input of Processes: **Any Process**

Output of Processes: **Approval**, **Verification**

Table 8.3: ApprovalSuccess Resource

NAME	DATA TYPE	DESCRIPTION
ApprovalDetails * New in JDF 1.3	element	Container for details about the decision for each approver.
Contact * New in JDF 1.2 Deprecated in JDF 1.3	refelement	List of contacts that have signed off on this approval. Use ***ApprovalDetails***/***Contact*** in **JDF** 1.3 and above.
FileSpec ? Deprecated in JDF 1.3	refelement	The file that contains the approval signature. If ***FileSpec*** does not exist, ***ApprovalSuccess*** is a logical placeholder. Use ***ApprovalDetails***/***FileSpec*** in **JDF** 1.3 and above.

8.3.1 ApprovalDetails

New in JDF 1.3

Table 8.4: ApprovalDetails Element

NAME	DATA TYPE	DESCRIPTION
ApprovalState	enumeration	Decision made by the approver specified in this ***ApprovalDetails***/***Contact*** . **Allowed values are:** Approved – approver approved the resource. ApprovedWithComment – approver approved the resource but still had some comments. Rejected – approver rejected the resource.
ApprovalStateDetails ?	string	Additional details on the decision made by the approver are specified in this ***ApprovalDetails***/***Contact*** . This value provides additional machine readable details of @*ApprovalState*. Hand written comments and notes MAY be specified in ***ApprovalDetails***/***Comment*** or ***ApprovalDetails***/@*CommentURL*.
Contact ?	refelement	***Contact*** that signed off on this approval.
FileSpec ?	refelement	The file that contains the approval signature. If ***FileSpec*** does not exist, ***ApprovalSuccess*** is a logical placeholder.

8.4 Assembly

New in JDF 1.2

Assembly describes how the sections of one or multiple jobs or job parts are bound together.

Resource Properties

Resource Class: Parameter

Input of Processes: **Collecting**, **Gathering**, **SheetOptimizing**, **Stripping**, **WebInlineFinishing**

Table 8.5: Assembly Resource

NAME	DATA TYPE	DESCRIPTION
AssemblyID ? Deprecated in JDF 1.3	string	Identification of the ***Assembly*** if **Stripping** produces multiple ***Assembly*** elements.
AssemblyIDs ? New in JDF 1.3	NMTOKENS	Identification of the ***Assembly*** elements if **Stripping** describes an imposition scheme for multiple ***Assembly*** elements. @*AssemblyIDs* MAY contain multiple NMTOKENS, when the ***Assembly*** resource specifies an intermediate product that contains multiple final assemblies.
BindingSide = "Left"	enumeration	Indicates which side SHALL be bound. @*BindingSide* is ignored when @*Order* = "None". **Allowed value is from:** ▸ Edge**.**
JobID ?	string	Identification of the original Job the ***Assembly*** belongs to. If not specified, it defaults to the value specified or implied in the **JDF** node.
JogSide = "Top" New in JDF 1.3	enumeration	@*JogSide* specifies the side on which the ***AssemblySection*** elements will be aligned. **Allowed values are:** Left Right Top Bottom None
Order = "Gathering"	enumeration	Ordering of the individual ***AssemblySection***s. Order specifies the topology of the final ***Assembly***. **Allowed values are:** Collecting – The sections are placed within one another. The first section is on the outside. An example is a saddle-stitched brochure. See ▸ Section 6.5.10 Collecting Gathering – The sections are placed on top of one another. The first section is on the top. An example is a perfect bound magazine. See ▸ Section 6.5.20 Gathering. None – The sections are not bound. Typically used for flatwork Jobs. List – More complex ordering of the sections. If multiple child ***AssemblySection*** elements are provided, these are gathered on top of one another. The first ***AssemblySection*** is on the top. If nested ***AssemblySection*** elements are provided, these are collected into each other. The first ***AssemblySection*** is on the outside.
PhysicalSection ? New in JDF 1.3	IntegerList	Specifies the physical structure of a newspaper. The structure is based on a broadsheet production. For instance, @*PhysicalSection* = "8 6 8 6" represents a 4 book production with 8 pages in the first physical section, 6 in the second one and so on.
AssemblySection *	element	Individual ***AssemblySection*** elements which are gathered. ***AssemblySection*** elements SHALL NOT be specified unless @*Order* = "List".
PageAssignedList * New in JDF 1.3	element	Defines the page sequence for of an ***Assembly***. One ***PageAssignedList*** element corresponds to one or more consecutive Reader Pages. The order of the ***PageAssignedList*** elements specifies the reader order of the assigned pages within the ***Assembly***. ***PageAssignedList*** SHALL NOT be specified if @*Order* = "List".
PageList ? New in JDF 1.3	refelement	Reference to the ***PageList*** that describes the pages used in this ***Assembly***.

8.4.1 AssemblySection

Table 8.6: AssemblySection Element

NAME	DATA TYPE	DESCRIPTION
AssemblyID ? Deprecated in JDF 1.3	string	Identification of the ***AssemblySection*** if **Stripping** produces a multi-section ***Assembly***. If not specified, it defaults to the value specified or implied in the parent ***Assembly*** or ***AssemblySection***.
AssemblyIDs ? New in JDF 1.3	NMTOKENS	Identification of the ***AssemblySection*** elements if **Stripping** describes an imposition scheme for a multi-section ***Assembly***. If not specified, it defaults to the value specified or implied in the parent ***Assembly*** or ***AssemblySection***. In general ***AssemblySection***/@*AssemblyIDs* will contain only a single NMTOKEN value. @*AssemblyIDs* MAY contain multiple NMTOKENS, when the ***AssemblySection*** specifies an intermediate product that contains multiple final products.
JobID ?	string	Identification of the original Job the ***AssemblySection*** belongs to. If not specified, it defaults to the value specified or implied in the parent ***Assembly*** or ***AssemblySection***.
Order = "Gathering" Deprecated in JDF 1.4	enumeration	**Ordering of the child *AssemblySection* elements.** **Allowed values are:** Collecting – The child ***AssemblySection*** elements are placed within one another. The first section is on the outside. Gathering – The child ***AssemblySection*** elements are placed on top of one another. The first section is on the top. **Deprecation note:** Starting with **JDF** 1.4, Sibling ***AssemblySection*** elements are gathered whereas child ***AssemblySection*** elements are collected. Thus the Relationship of the ***AssemblySection*** elements directly reflects the structure of the ***Assembly***.
AssemblySection *	element	Additional child ***AssemblySection*** elements which SHALL be gathered. The resulting set of ***AssemblySection*** elements SHALL be collected inside this ***AssemblySection***.
PageAssignedList * New in JDF 1.	element	Defines the page sequence for of an ***AssemblySection***. One ***PageAssignedList*** element corresponds to one or more consecutive Reader Pages. The order of the ***PageAssignedList*** elements specifies the reader order of the assigned pages within the ***AssemblySection***. ***PageAssignedList*** SHALL NOT be specified if child ***AssemblySection*** elements are present in this ***AssemblySection***.

8.4.2 PageAssignedList

New in JDF 1.3

PageAssignedList specifies the metadata related to assigned pages.

Table 8.7: PageAssignedList Element

NAME	DATA TYPE	DESCRIPTION
BroadsheetNumber ?	integer	Specifies a broadsheet position within a single Web product. Several ***PageAssignedList*** elements MAY show the same value for this attribute (e.g., in a 'tabloid-' or 'magazine production' on a newspaper press).
LogicalPrinterSection ?	string	Specifies a logical grouping of page-placement positions from the press managers point of view (see @*PagePlacementName* for details). A logical section NEED NOT correspond to a physical section.
PageListIndex	IntegerRangeList	List of the indices of the ***PageData*** elements of the ***Assembly***/***PageList*** specified in this ***AssemblySection***.
PagePlacementName ?	string	Specifies the name of a position in a Web product where a Reader Page is placed on a Web Press. In contrast to ***PageList***/***PageData***/@*PageLabel*, @*PagePlacementName* specifies an identifier for a single page on a Web-product level. Therefore, different @*PagePlacementName* values might be assigned to one single ***PageList***/***PageData*** element.

Example 8.1: Perfect Bound (Gathering)

New in JDF 1.4

Cover wrapped around a perfect bound (gathering) body.

```
<Assembly BindingSide="Left" Class="Parameter" ID="ASM000" Order="List"
       Status="Available">
    <AssemblySection AssemblyIDs="Ass_Cover" >
       <AssemblySection AssemblyIDs="Ass_Body1"/>
       <AssemblySection AssemblyIDs="Ass_Body2"/>
       <AssemblySection AssemblyIDs="Ass_Insert"/>
       <AssemblySection AssemblyIDs="Ass_Body3"/>
       <AssemblySection AssemblyIDs="Ass_Body4"/>
    </AssemblySection>
</Assembly>
```

Example 8.2: Saddle-Stitched Brochure (Collecting)

New in JDF 1.4

```
<Assembly BindingSide="Left" Class="Parameter" ID="ASM000" Order="List"
       Status="Available">
    <AssemblySection AssemblyIDs="Ass_Cover" >
       <AssemblySection AssemblyIDs="Ass_Body1" >
          <AssemblySection AssemblyIDs="Ass_Body2" >
             <AssemblySection AssemblyIDs="Ass_Body3" >
                <AssemblySection AssemblyIDs="Ass_Body4" >
                </AssemblySection>
             </AssemblySection>
          </AssemblySection>
       </AssemblySection>
    </AssemblySection>
</Assembly>
```

8.5 AssetListCreationParams

New in JDF 1.2

This resource provides controls for the **AssetListCreation** process.

Resource Properties

Resource Class: Parameter

Input of Processes: **AssetListCreation**

Table 8.8: AssetListCreationParams Resource

NAME	DATA TYPE	DESCRIPTION
AssetTypes ?	regExp	Specifies what type of assets are to be listed. The regular expression represents the *@MimeType* of the assets to be listed. The default behavior SHALL list everything. In case an asset requires a plug-in or extension in order to be opened in an application, this plug-in or extension SHOULD be listed as an asset.
ListPolicy = "All"	enumeration	Policy that defines which assets SHALL be added to the output ***RunList***. **Allowed values are:** All – List all referenced assets, including those that are unavailable. Available – List all referenced assets, excluding those that are unavailable.
FileSpec (SearchPath) *	refelement	An ordered list of search paths that indicates where to search for referenced assets if they are not located in the same directory as the input asset. If no ***FileSpec*** is specified, the search path is the directory in which the input asset resides and SHALL NOT be searched recursively.

8.6 BendingParams

New in JDF 1.3

BendingParams describes the parameter set for a plate bending and punching device. A plate is bent and/or punched to fit the press cylinder.

Resource Properties

Resource Class: Parameter

Input of Processes: **Bending**

Table 8.9: BendingParams Resource

NAME	DATA TYPE	DESCRIPTION
Bend = "true"	boolean	If "true", indicates that the device SHALL bend.
Punch = "true"	boolean	If "true", indicates that the device SHALL create registration punch holes.
PunchType ?	string	Name of the registration punch scheme (e.g., Bacher).

8.7 BinderySignature

New in JDF 1.2

The ***BinderySignature*** is conceptually a folding dummy. It represents multiple pieces of paper, which are folded together in the folder. It is a reusable, size-independent object.

One ***BinderySignature*** (when used with stripping) consists of one or more Strip Cells, which are created either explicitly (via ***SignatureCell*** elements) or implicitly (via the *@FoldCatalog* attribute or the ***Fold*** elements). ***StrippingParams*** describes some attributes for Strip Cells (using ***StripCellParams***). The Strip Cells by them selves belong to a ***BinderySignature***.

Each ***BinderySignature*** consumes a number of Pages from the ***PageList***. If no ***SignatureCell*** elements are specified, each ***BinderySignature*** consumes the number of pages as calculated from *@NumberUp* (X*Y*2) or *@FoldCatalog* (The integer value after the *F* (e.g., "F16-x" consumes 16 pages). If ***SignatureCell*** elements are specified, the number of pages consumed is the sum of the number of pages for all unique ***SignatureCell***/*@SectionIndex*. The number of pages for each ***SignatureCell***/*@SectionIndex* is one more than the maximum value of any ***SignatureCell***/*@FrontPages* or ***SignatureCell***/*@BackPages* for that ***SignatureCell***/*@SectionIndex* (it is one more because ***SignatureCell***/*@FrontPages* and ***SignatureCell***/*@BackPages* begin at zero)

Resource Properties

Resource Class: Parameter

Resource referenced by: ***StrippingParams***

Example Partition: "WebName"

Input of Processes: —

Table 8.10: BinderySignature Resource (Sheet 1 of 4)

NAME	DATA TYPE	DESCRIPTION
AlignmentReferenceWeb ? New in JDF 1.4	NMTOKEN	The Partition *@WebName* value of the reference web that *@WebCellAlignment* refers to.
BinderySignatureType = "Fold" New in JDF 1.3	enumeration	*@BinderySignatureType* specifies the type of ***BinderySignature***. **Allowed values are:** Fold – a folding dummy (as defined in **JDF** 1.2). Grid – a grid based layout. Die – a layout defined by an existing die.
BindingEdge = "Left"	enumeration	Specifies the binding edge of this ***BinderySignature***. *@BindingEdge* defines the Spine side the folded ***BinderySignature***. The opposite side defines the Face side. **Allowed values are:** Left Right Top Bottom None – The Spine is assumed to be at the left side of the ***SignatureCell*** and the Face is assumed to be at the right side of the ***SignatureCell***

Table 8.10: BinderySignature Resource (Sheet 2 of 4)

NAME	DATA TYPE	DESCRIPTION
BindingOrientation ? New in JDF 1.3	enumeration	After folding a ***BinderySignature***, the default reference corner is the lower left corner of the ***BinderySignature***. The side coinciding with the last fold is the @*BindingEdge*, the other side of the reference corner the @*JogEdge*. @*BindingOrientation* is the named orientation describing the transformation of the default reference corner to the new reference corner defined by @*BindingEdge* and @*JogEdge*. For ***BinderySignature*** elements defined by @*FoldCatalog* or ***Fold*** elements, the default value of @*BindingOrientation* = "Rotate0" if the folded ***BinderySignature*** has a closed head, otherwise @*BindingOrientation* = "Flip0". For ***BinderySignature*** elements defined by ***SignatureCell*** elements, the default value @*BindingOrientation* = "Rotate0". For details, see ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component.**Value is from:** ▸ Orientation**.**
BleedBottom ? New in JDF 1.5	double	Value for the bleed at the bottom side of the ***BinderySignature***. **Note:** See ▸ Section 8.7.1 On the use of Bleed.
BleedLeft ? New in JDF 1.5	double	Value for the bleed at the left side of the ***BinderySignature***. **Note:** See ▸ Section 8.7.1 On the use of Bleed.
BleedRight ? New in JDF 1.5	double	Value for the bleed at the right side of the ***BinderySignature***. **Note:** See ▸ Section 8.7.1 On the use of Bleed.
BleedTop ? New in JDF 1.5	double	Value for the bleed at the top side of the ***BinderySignature***. **Note:** See ▸ Section 8.7.1 On the use of Bleed.
Bottling ? New in JDF 1.6	enumeration	@*Bottling* SHALL specify the method to use for compensating the bottle angle, which is the slight rotation of a page needed to compensate for the rotation fault introduced when making cross-folds. **Allowed values are:** All - Compensate all cross-folds Last - Compensate only the bottle angle caused by the final fold None - Do not compensate
FoldCatalog ?	string	@*FoldCatalog* describes the type of fold according to the folding catalog in ▸ Figure 8-33: Fold catalog part 1 and ▸ Figure 8-34: Fold catalog part 2. **Value format is:** "Fn-i" where "n" is the number of finished pages and "i" is either an integer, which identifies a particular fold or the letter "X", which identifies a generic fold (e.g., "F6-2" describes a Z-fold of 6 finished pages, and "F6-X" describes a generic fold with 6 finished pages). **Constraint:** At least one of ***SignatureCell***, @FoldCatalog or ***Fold*** SHALL be specified. @FoldCatalog SHALL NOT be specified unless @BinderySignatureType = "Fold".
FoldLay ? New in JDF 1.4	enumeration	Specification of the orientation applied to the substrate of all stacked webs before applying folding (only specified at root ***BinderySignature*** node, and would default to Rotate0). Allowed value is from: ▸ Orientation
JogEdge = "Top" New in JDF 1.3	enumeration	Specifies the @*JogEdge* of the folded ***BinderySignature***. The @*JogEdge* defines the Head side of the folded ***BinderySignature***. The opposite side defines the Foot side. **Allowed values are:** Left Right Top Bottom None – The Head side is the top of the ***SignatureCell***, the Foot side is the bottom of the ***SignatureCell***.

Table 8.10: BinderySignature Resource (Sheet 3 of 4)

NAME	DATA TYPE	DESCRIPTION
NumberUp = "1 1" Modified in JDF 1.3	XYPair	Specifies a regular, multi-up grid of ***SignatureCell*** elements into which content pages are mapped. The first value specifies the number of columns of ***SignatureCell*** elements, and the second value specifies the number of rows of ***SignatureCell*** elements in the multi-up grid (both numbers are integers). When the ***BinderySignature*** is Partitioned (e.g., by @*WebName*), @*NumberUp* MAY be different from leaf to leaf. When the ***BinderySignature*** is Partitioned (e.g., by @*WebName*), @*NumberUp* MAY be different from leaf to leaf.
OutsideGutter ? New in JDF 1.3	boolean	If @*BinderySignatureType* is "Grid", this boolean defines whether the outside margins of strip cells have to be taken into account (e.g., if @*OutsideGutter* is "false", the spine of the strip cells at the left border of the grid is considered to be 0).
StaggerColumns ? New in JDF 1.3	DoubleList	A list of doubles describing the staggering for subsequent columns. The number of entries in the list describes the periodicity of the staggering. Each value gives a factor of the strip cell height ((y value of @*TrimSize*) + @*TrimHead* + @*TrimFoot*) by which to shift the corresponding column (can be negative) (e.g., @*StaggerColumns* = "0.0 0.333 0.666" specifies to shift each) • 3*n column up by 0% • 3*n+1 column up by 33.3% of the strip cell height • 3*n+2 column up by 66.6% of the strip cell height This element SHALL NOT be present unless @*BinderySignatureType* = "Grid". At most one of @*StaggerColumns* or @*StaggerRows* SHALL be specified.
StaggerContinuous ? New in JDF 1.3	boolean	Indicates if the ***BinderySignature*** has to be considered as a continuous repetition for staggering. This attribute SHALL NOT be present unless exactly one of @*StaggerRows* or @*StaggerColumns* is specified. Consider a grid with *m* columns and *n* rows with @*StaggerContinuous* = "true". If @*StaggerColumns* is specified, the ***BinderySignature*** SHALL be considered continuous with a height *H* equal to *n* multiplied by the strip cell height. If @*StaggerColumns* has a value of *y* for a certain column, that column is shifted up (assuming $y > 0$) by an amount equal to *y* multiplied by the strip cell height (in the same way as described for @*StaggerColumns*). All content (even partial cells) that falls above *H* (the top of ***BinderySignature***) is shifted to the bottom such that the top of the shifted content is just below the original bottom cell in the column. For example, if *y* is 0.666, then the top 66.6% of the top cell is shifted to be just below the original bottom cell. Analogous for @*StaggerRows*.
StaggerRows ? New in JDF 1.3	DoubleList	A list of doubles describing the staggering for subsequent rows. The number of entries in the list describes the periodicity of the staggering. Each value gives a factor of the strip cell width ((x value of @*TrimSize*) + @*TrimFace* + @*Spine*) by which to shift the corresponding row (can be negative) (e.g., "0.0 0.333 0.666" specifies to shift each) • 3*n row right by 0% • 3*n+1 row right by 33.3% of the strip cell width • 3*n+2 row right by 66.6% of the strip cell width This element SHALL NOT be present unless @*BinderySignatureType* = "Grid". At most one of @*StaggerColumns* or @*StaggerRows* SHALL be specified.
TrimBottom ? New in JDF 1.5	double	Value for cutoff at the bottom side of the ***BinderySignature***. **Note:** See ▸ Section 8.7.2 On the use of Trim.
TrimLeft ? New in JDF 1.5	double	Value for the cutoff at the left side of the ***BinderySignature***. **Note:** See ▸ Section 8.7.2 On the use of Trim.
TrimRight ? New in JDF 1.5	double	Value for the cutoff at the right side of the ***BinderySignature***. **Note:** See ▸ Section 8.7.2 On the use of Trim.
TrimTop ? New in JDF 1.5	double	Value for the cutoff at the top side of the ***BinderySignature***. **Note:** See ▸ Section 8.7.2 On the use of Trim.

Table 8.10: BinderySignature Resource (Sheet 4 of 4)

NAME	DATA TYPE	DESCRIPTION
WebCellAlignment ? New in JDF 1.4	XYPair	Zero based ***SignatureCell*** index (coordinate) that the bottom left ***SignatureCell*** in this web is aligned with in the full web (only specified at the *@WebName* Partition, and would default to "0 0"). See ▸ Figure 8-2: WebCellAlignment, Example 1, ▸ Figure 8-3: WebCellAlignment Example 2 and ▸ Figure 8-4: WebCellAlignment Example 3. Also, the "stacking" of the webs is implied by the order of the webs within the ***BinderySignature***. The back side of a *@WebName* Partition of a ***BinderySignature*** will be touching the front side of the *@WebName* partition of the ***BinderySignature*** that follows it in the **JDF** file.
DieLayout ? New in JDF 1.3	refelement	The layout as defined by a pre-existing die. ***DieLayout*** SHALL be present when *@BinderySignatureType* = "Die".
Fold *	element	Describes the folding operations in the sequence in which they are to be carried out. When both ***Fold*** and *@FoldCatalog* are specified, *@FoldCatalog* defines the topology of the folding scheme, and the specifics of each individual fold are described by the ***Fold*** elements. The ***Fold*** elements have precedence. ***Fold*** SHALL NOT be specified if ***SignatureCell*** elements are present. ***Fold*** SHALL NOT be specified unless *@BinderySignatureType* = "Fold".
SignatureCell *	element	Describes the ***SignatureCell*** elements used in this ***BinderySignature***. ***SignatureCell*** elements are ordered in X-Y direction starting at the lower left-hand corner of the ***BinderySignature***. When both ***SignatureCell*** and *@FoldCatalog* are specified, *@FoldCatalog* defines the topology of the folding scheme, and the specifics of each individual signature cell are described by the ***SignatureCell*** elements. The ***SignatureCell*** elements SHALL have precedence. ***SignatureCell*** SHALL NOT be specified if ***Fold*** elements are present.

8.7.1 On the use of Bleed

New in JDF 1.5

If any Strip Cell belonging to the ***BinderySignature*** has any bleed value > 0, where a bleed value is ***StripCellParams***/*@BleedFace*, ***StripCellParams***/*@BleedSpine*, ***StripCellParams***/*@BleedHead* or ***StripCellParams***/*@BleedFoot*, then none of the ***BinderySignature*** /*@BleedLeft*, ***BinderySignature*** /*@BleedRight*, ***BinderySignature*** /*@BleedTop* and ***BinderySignature*** /*@BleedBottom* SHALL be applied.

If any Strip Cell belonging to the ***BinderySignature*** has a ***StripCellParams***/margin value > 0 (where margin value is: *@Spine*, *@TrimFace*, *@TrimFoot*, *@TrimHead*, *@TrimSize*, *@BackOverfold*, *@FrontOverfold*, *@CutWidthFoot*, *@CutWidthHead* or *@MillingDepth*), then none of ***BinderySignature***/*@BleedLeft*, ***BinderySignature*** /*@BleedRight*, ***BinderySignature*** /*@BleedTop* and ***BinderySignature*** /*@BleedBottom* SHALL be applied.

8.7.2 On the use of Trim

New in JDF 1.5

The attributes *@TrimBottom*, *@TrimLeft*, *@TrimRight* and *@TrimTop* are added around the rectangle that is composed of the Strip Cells belonging to the ***BinderySignature***. The Strip Cell includes the margins specified by ***StripCellParams***. ***Layout***/***Position***/*@Orientation* is applied to the ***BinderySignature***/*@TrimLeft*, ***BinderySignature*** /*@TrimRight*, ***BinderySignature*** /*@TrimTop* and ***BinderySignature*** /*@TrimBottom* too.

Figure 8-1: BinderySignature Trims

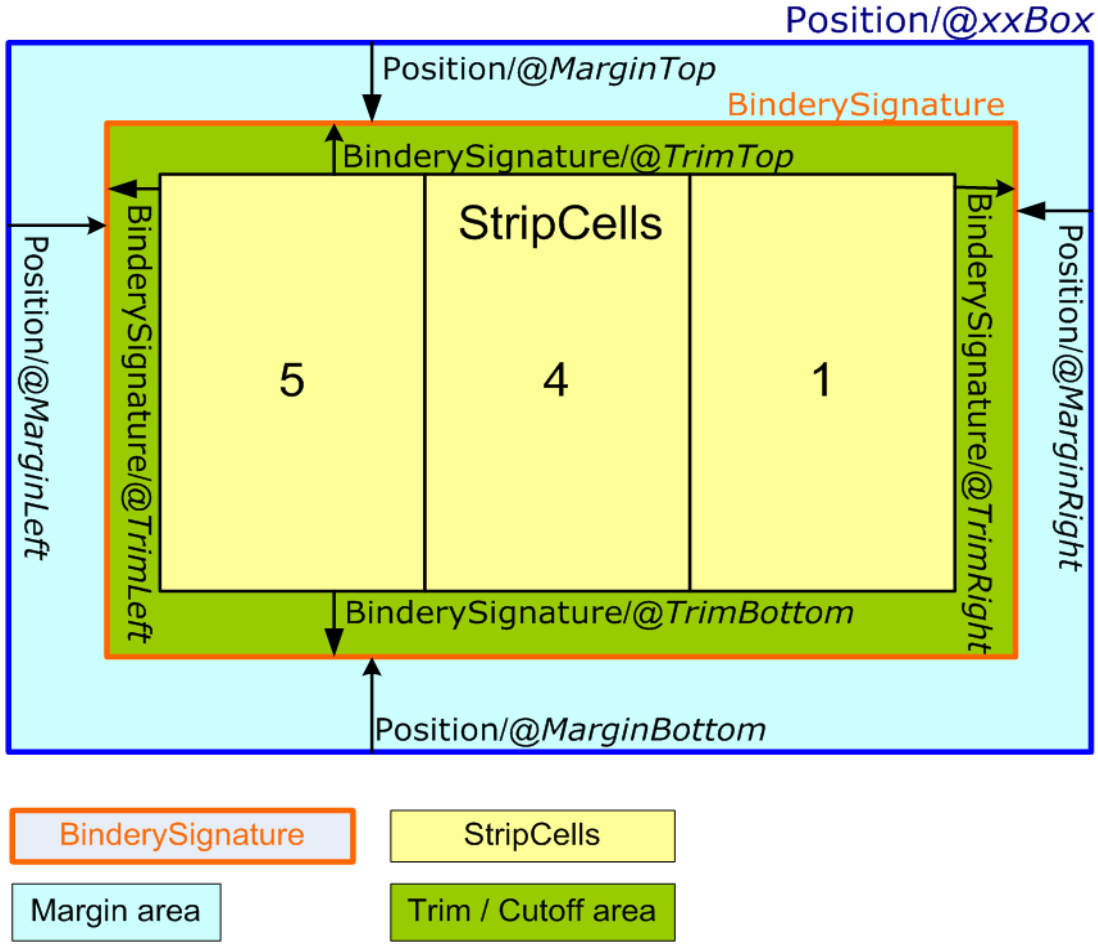


Figure 8-2: WebCellAlignment, Example 1

```
<BinderySignature PartIDKeys="WebName" FoldCatalog="F16-6" AlignmentReferenceWeb="Web1">
    <BinderySignature NumberUp="4 2" WebCellAlignment="0 0" WebName="Web1"/>
    <BinderySignature NumberUp="4 2" WebCellAlignment="0 0" WebName="Web2"/>
</BinderySignature>
```

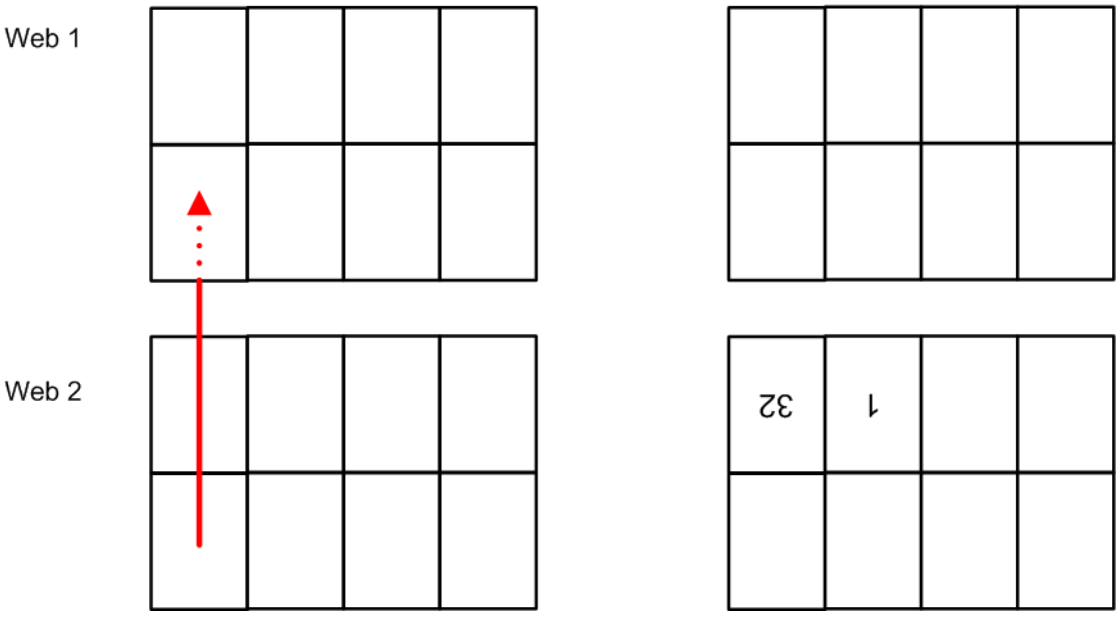


Figure 8-3: WebCellAlignment Example 2

```
<BinderySignature PartIDKeys="WebName" FoldCatalog="F16-6" AlignmentReferenceWeb="Web1">
    <BinderySignature NumberUp="2 2" WebCellAlignment="1 0" WebName="Web1"/>
    <BinderySignature NumberUp="4 2" WebCellAlignment="0 0" WebName="Web2"/>
</BinderySignature>
```

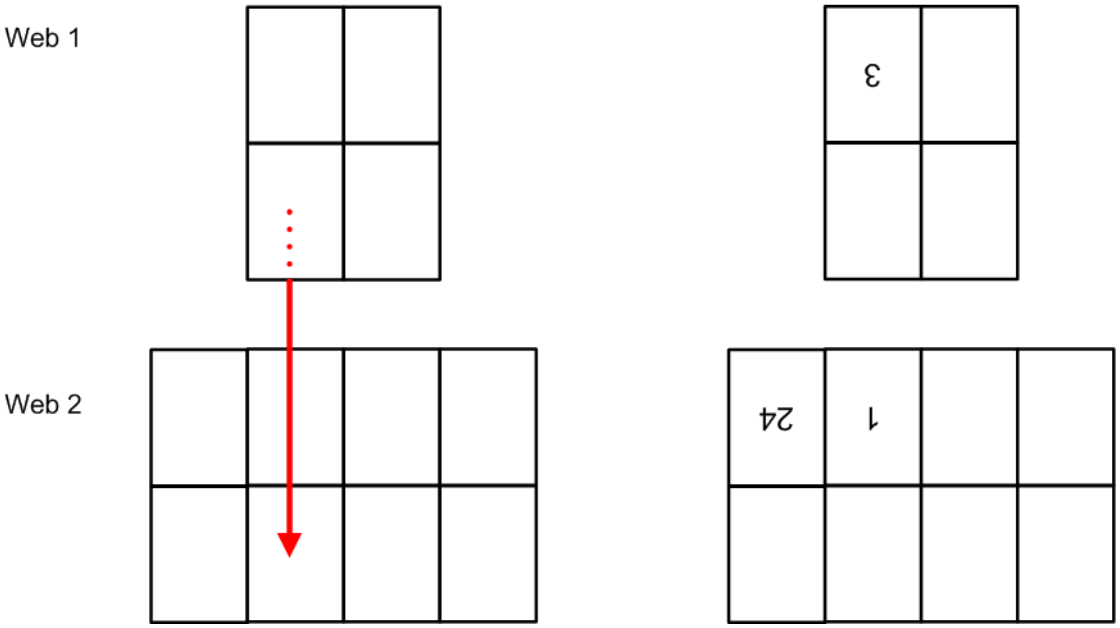
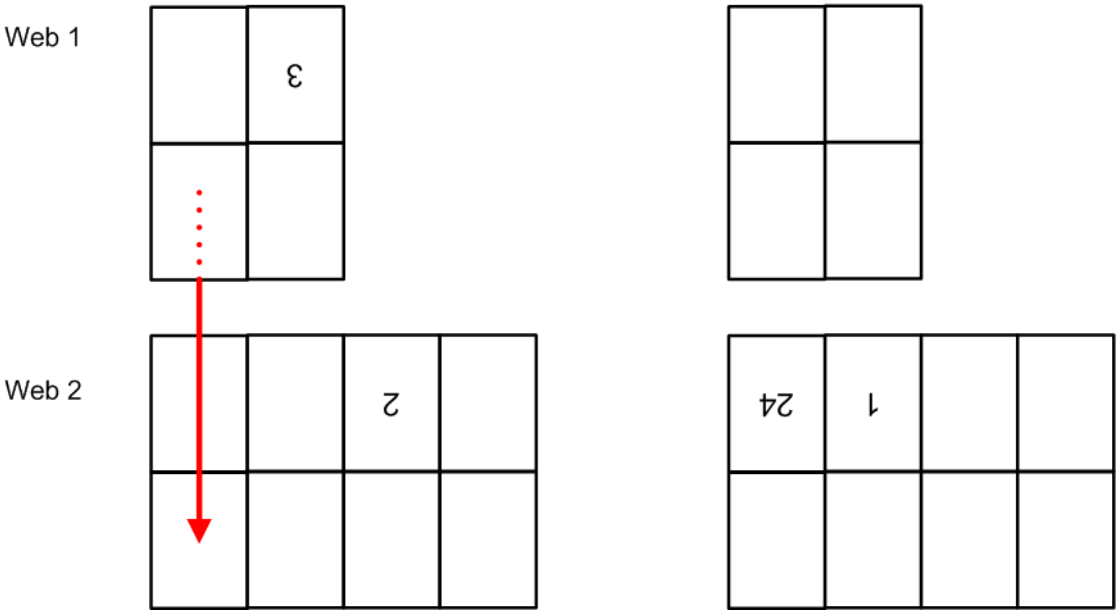


Figure 8-4: WebCellAlignment Example 3

```
<BinderySignature PartIDKeys="WebName" FoldCatalog="F16-6" AlignmentReferenceWeb="Web1">
    <BinderySignature NumberUp="2 2" WebCellAlignment="0 0" WebName="Web1"/>
    <BinderySignature NumberUp="4 2" WebCellAlignment="0 0" WebName="Web2"/>
</BinderySignature>
```



Example 8.3: Pseudo Code to Generate Page Count from SignatureCell Elements

New in JDF 1.4

```
maxSectionIndexSeen = 0
maxSectionPages = [0]
for sc in BinderySignature/SignatureCell
    si = sc@SectionIndex
    if ( si > maxSectionIndexSeen)
        for index from maxSectionIndexSeen to si - 1:
            maxSectionPages.append(0)
        maxSectionIndexSeen = si
    for page in sc@FrontPages
        maxSectionPages[si] = max(maxSectionPages[si],page)
    for page in sc@BackPages
        maxSectionPages[si] = max(maxSectionPages[si],page)
totalPages = 0
for sectionIndex from 0 to maxSectionIndexSeen
    totalPages += 1 + maxSectionPages[sectionIndex]
return totalPages
```

8.7.3 SignatureCell

SignatureCell elements describe a set of individual page cells in a ***BinderySignature*** .
Note: "Page number" in the table below refers to finished pages from the ***PageList*** numbered from 0 to n, as opposed to Folio pages, which are the numbers that appear in print with the content of the document; the difference being that pages without Folio numbering are counted. As the ***BinderySignature*** is a reusable object, the page numbers refer to finished pages numbered from 0 to n as if this ***BinderySignature*** were the only section of the ***Assembly***. The consuming device needs to calculate the final product page number using the ***Assembly*** and ***StrippingParams***/@*SectionList*. The ***BinderySignature*** cells SHALL NOT contain final page numbers unless ***Assembly***/@*Order* = "None"

Table 8.11: SignatureCell Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BackFacePages ? Deprecated in JDF 1.4	IntegerList	Page numbers for the back pages forming a foldout. **Deprecation note:** Starting with **JDF** 1.4, use @*FaceCells* to describe foldouts.
BackPages ?	IntegerList	Page numbers of the back pages of a ***SignatureCell***. The number of entries in @*FrontPages* and @*BackPages* SHALL be identical. The entries with an identical index in @*FrontPages* and @*BackPages* are back-to-back in the layout. If not specified, the layout is one-sided.
BackSpread ? New in JDF 1.5	IntegerList	Index of ***SignatureCell*** elements that are combined into a spread on the back side.
BottleAngle ? Deprecated in JDF 1.6	double	Indicates the bottle angle, which is the slight rotation of the ***SignatureCell*** needed to compensate for the rotation fault introduced when making cross-folds.
BottleAxis ? Deprecated in JDF 1.6	enumeration	Indicates the point around which the cell is bottled. **Allowed values are:** FaceFoot FaceHead SpineFoot SpineHead
FaceCells ? New in JDF 1.4	IntegerList	List of indices of ***SignatureCell*** elements that form a foldout together with this ***SignatureCell***. The ***SignatureCell*** that contains @*FaceCells* is the parent of the foldout, typically the Page that is attached to the spine. Details of each foldout Page are described by a ***SignatureCell*** element.
FrontFacePages ? Deprecated in JDF 1.4	IntegerList	Page numbers for the front pages forming a foldout. **Deprecation note:** Starting with **JDF** 1.4, use @*FaceCells* to describe foldouts.
FrontPages ?	IntegerList	Page numbers of the front pages of a ***SignatureCell***. Multiple page cells with the same properties except for the pages to which they are assigned MAY be summarized as one ***SignatureCell*** with multiple entries in @*FrontPages*.

Table 8.11: SignatureCell Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
FrontSpread ? New in JDF 1.5	IntegerList	Index of ***SignatureCell*** elements that are combined into a spread on the front side.
Orientation = "Up" Modified in JDF 1.3	enumeration	Indicates the orientation of the ***SignatureCell***. **Allowed values are:** Down – 180° rotation. Left – 90° counter-clockwise rotation. New in JDF 1.3 Right – 270° counter-clockwise rotation New in JDF 1.3 Up – 0° rotation.
SectionIndex = "0"	integer	Unique logical index of the page section that are to fill this ***SignatureCell***. This is an indirect logical index. The actual section index is defined in ***StrippingParams***/@*SectionList*.
StationName ? New in JDF 1.3	string	The name of the 1-up station in the die layout. **Constraint:** if ***BinderySignature***/@*BinderySignatureType* = "Die", this element SHOULD be specified. **Constraint:** if ***BinderySignature***/@*BinderySignatureType* = "Die" and ***BinderySignature***/***DieLayout*** contains more than 1 ***Station***, this attribute SHALL be specified.

Example 8.4: StrippingParams: Foldout Using FaceCells

New in JDF 1.4

```
<!--Stripping Foldout example corresponding to spec example O-24 - with new
   attribute FaceCells-->
<StrippingParams Class="Parameter" ID="r000005"
      PartIDKeys="CellIndex" Status="Available">
   <BinderySignatureRef rRef="r000006"/>
   <StrippingParams CellIndex="0">
      <!--stripcell for the folded out foldout(front page=4)-->
      <StripCellParams TrimSize="200 400"/>
   </StrippingParams>
   <StrippingParams CellIndex="1">
      <!--stripcell for the inner page of the foldout foldout(front page=5)-->
      <StripCellParams TrimSize="300 400"/>
   </StrippingParams>
   <StrippingParams CellIndex="2">
      <!--stripcell for the inner page of the foldout foldout(front page=0)-->
      <StripCellParams TrimSize="320 400"/>
   </StrippingParams>
</StrippingParams>
<BinderySignature Class="Parameter" ID="r000006" Status="Available">
   <!--this is the foldout foldout cell-->
   <SignatureCell BackPages="3" FrontPages="4"/>
   <!--this cell is the inner page of the foldout, i.e. the page that is
      attached to the spine The new attribute FaceCells refers to the cell(s)
      that describe the foldout; in this case the cell to the left. The front
      and back pages of the foldout are listed in the respective cell(s)
   -->
   <SignatureCell BackPages="2" FaceCells="0" FrontPages="5"/>
   <!--this is the cell that has no foldout-->
   <SignatureCell BackPages="1" FrontPages="0"/>
</BinderySignature>
```

8.8 BlockPreparationParams

New in JDF 1.1

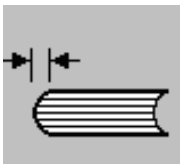
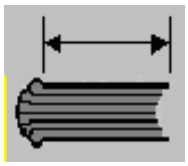
This resource describes the settings of a **BlockPreparation** process. For the tightbacking there are four different kinds of book forms as shown in ▸ Figure 8-5: Tightbacking for Block Preparation.

Figure 8-5: Tightbacking for Block Preparation

Kinds of Book Forms	Flat	Round	Flat and Backed	Rounded and Backed
@*TightBacking* =	"Flat"	"Round"	"FlatBacked"	"RoundBacked"

For the rounding and for the backing there are two additional measurement as shown in ▸ Figure 8-6: Rounding and Backing for Block Preparation.

Figure 8-6: Rounding and Backing for Block Preparation

Measurement	Rounding Way	Backing Way
Attribute	@*Rounding*="m"	@*Backing*="n"
		

Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **BlockPreparation**

Table 8.12: BlockPreparationParams Resource

NAME	DATA TYPE	DESCRIPTION
Backing ?	double	Backing distance in points.
Rounding ?	double	Rounding distance in points.
TightBacking ?	enumeration	Definition of the geometry of the back of the book block. **Allowed value is from:** ▸ TightBacking.
RegisterRibbon *	refelement	Description of the register ribbons that are included within the book block.

8.9 BoxFoldingParams

New in JDF 1.3

This resource defines the parameters for folding and gluing blanks to folded flat boxes in a box folder-gluer device.

Resource Properties

Resource Class: Parameter

Input of Processes: **BoxFolding**

Table 8.13: BoxFoldingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BlankDimensionsX ?	DoubleList	X position of folds for an unfolded box beginning from the origin of the coordinate system (left side) increasing from minimum to maximum (expressed in points). See ▸ Figure 8-8: BoxFoldingType Attribute for values of Type00, Type01 and Type02 through ▸ Figure 8-11: BoxFoldingType Attribute for values of Type15 and Type20. The first value of @*BlankDimensionsX* is the position of the fold marked by X0 in a diagram (e.g., ▸ Figure 8-8: BoxFoldingType Attribute for values of Type00, Type01 and Type02). The second value of @*BlankDimensionsX* is the position of the fold marked by X1, and so on. @*BlankDimensionsX* SHALL NOT be specified unless @*BoxFoldingType* is also specified.

Table 8.13: BoxFoldingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
BlankDimensionsY ?	DoubleList	Y position of folds for of an unfolded box beginning from the origin of the coordinate system (bottom side) increasing from minimum to maximum (expressed in points). See ▸ Figure 8-8: BoxFoldingType Attribute for values of Type00, Type01 and Type02 through ▸ Figure 8-11: BoxFoldingType Attribute for values of Type15 and Type20. The first value of @*BlankDimensionsY* is the position of the fold marked by Y0 in a diagram (e.g., ▸ Figure 8-8: BoxFoldingType Attribute for values of Type00, Type01 and Type02). The second value of @*BlankDimensionsY* is the position of the fold marked by Y2, and so on. @*BlankDimensionsY* SHALL NOT be specified unless @*BoxFoldingType* is also present.
BoxFoldingType ?	enumeration	Basic predefined folding types. See the drawings referenced from each defined value below. Each drawing is shown from the print side with the lid at the top. Each type is described with a sequence of ***BoxFoldAction*** elements. The most common sequences (folding types) are predefined, All other are 'special' and SHALL be described in detail. **Allowed values are:** Type00 – Special type for boxes that are not pre-defined. See ▸ Figure 8-8: BoxFoldingType Attribute for values of Type00, Type01 and Type02. Type01 – see ▸ Figure 8-8: BoxFoldingType Attribute for values of Type00, Type01 and Type02. Type02 – see ▸ Figure 8-8: BoxFoldingType Attribute for values of Type00, Type01 and Type02. Type03 – see ▸ Figure 8-9: BoxFoldingType Attribute for values of Type03, Type04 and Type10. Type04 – see ▸ Figure 8-9: BoxFoldingType Attribute for values of Type03, Type04 and Type10. Type10 – see ▸ Figure 8-9: BoxFoldingType Attribute for values of Type03, Type04 and Type10. Type11 – see ▸ Figure 8-10: BoxFoldingType Attribute for values of Type 11, Type12 and Type13. Type12 – see ▸ Figure 8-10: BoxFoldingType Attribute for values of Type 11, Type12 and Type13. Type13 – see ▸ Figure 8-10: BoxFoldingType Attribute for values of Type 11, Type12 and Type13. Type15 – see ▸ Figure 8-11: BoxFoldingType Attribute for values of Type15 and Type20. Type20 – see ▸ Figure 8-11: BoxFoldingType Attribute for values of Type15 and Type20.
BoxApplication * Deprecated in JDF 1.4	element	Application work step in a box folder-gluer. The sequence of ***BoxFoldAction***, ***BoxApplication*** and ***GlueLine*** elements defines the sequence of work steps. The first element is applied first. Application SHOULD be described with a combined **Inserting** process. **Deprecation note:** Starting with **JDF** 1.4, a combined process that includes the **BoxFolding** and **Inserting** processes replaces ***BoxApplication***.
BoxFoldAction *	element	Individual work step in a box folder-gluer. The sequence of ***BoxFoldAction***, ***BoxApplication*** and ***GlueLine*** elements defines the sequence of work steps. The first element is applied first.
GlueLine *	element	Specification of a glue line. The ***GlueLine*** is applied to the blank in the coordinate system of the folder gluer at the state after all prior ***BoxFoldAction*** and ***BoxApplication*** elements have been applied. The sequence of ***BoxFoldAction***, ***BoxApplication*** and ***GlueLine*** elements defines the sequence of work steps. The first element is applied first.

8.9.1 BoxApplication

Deprecated in JDF 1.4

The table defining the deprecated ***BoxApplication*** Subelement has been moved to ▸ Section N.7.2.1 BoxApplication.

8.9.2 BoxFoldAction

BoxFoldAction describes an action in the folder-gluer that is perpendicular or diagonal to the movement path of the blank.

Table 8.14: BoxFoldAction Element

NAME	DATA TYPE	DESCRIPTION
FoldIndex	XYPair	Identification of the upper right corner of the flap or fold that is affected by this ***BoxFoldAction***. The first value of the XYPair refers to an indexed fold in @*BlankDimensionsX*; the second value of the XYPair refers to an indexed fold in @*BlankDimensionsY*. If either X or Y spans multiple flaps, it SHALL be set to -1.
Action	enumeration	@*Action* describes an individual action in the folder gluer. See ▸ Figure 8-7: Folding examples for some values of BoxFoldAction/@Action. Individual Action in the folder gluer. **Allowed values are from:** ▸ Table 8.15 Action Attribute Values**.**
GlueLine *	element	Specification of a glue lines needed to glue the ***Component*** described in this ***BoxApplication***. The ***GlueLine***s are applied to the ***Component*** in the coordinate system of the ***BoxApplication***/***Component***. The ***GlueLine***s applied to the blank are specified in ***BoxFoldingParams***/***GlueLine***.

8.9.2.1 Action Attribute Values

Table 8.15: Action Attribute Values

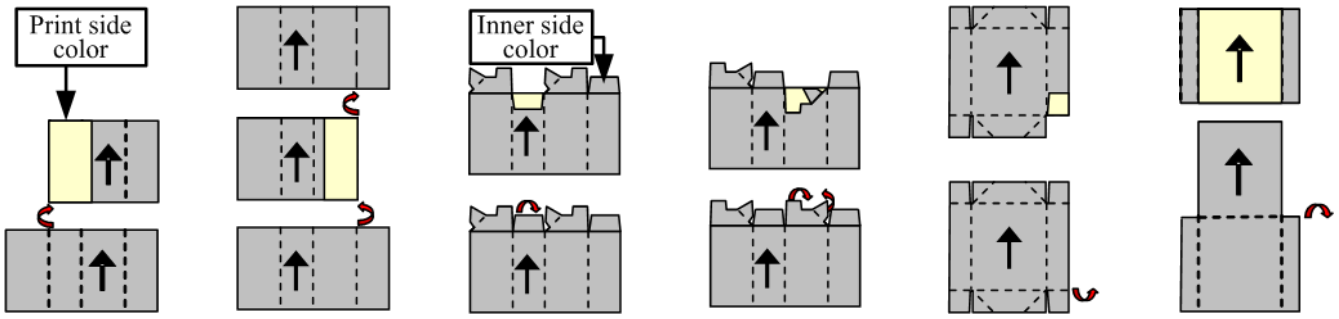
VALUE	DESCRIPTION
LongFoldLeftToRight	
LongFoldRightToLeft	
LongPreFoldLeftToRight	
LongPreFoldRightToLeft	
FrontFoldComplete	
FrontFoldDiagonal	
FrontFoldCompleteDiagonal	
BackFoldComplete	
BackFoldDiagonal	
BackFoldCompleteDiagonal	
ReverseFold	A "ReverseFold" is topologically equivalent to "FrontFoldDiagonal" but uses different equipment with other restrictions on Media weight and size and is therefore specified individually.
Milling	
Rotate90	90° counter-clockwise rotation
Rotate180	180° rotation
Rotate270	90° clockwise rotation

Example 8.5: BoxFoldingParams/BoxFoldAction

For instance, processing a Type01 blank (▸ Figure 8-8: BoxFoldingType Attribute for values of Type00, Type01 and Type02) has the following actions:

```
<BoxFoldingParams Class="Parameter" ID="BFP000" Status="Available">
    <BoxFoldAction FoldIndex="0 -1" Action="LongPreFoldLeftToRight"/>
    <BoxFoldAction FoldIndex="2 -1" Action="LongPreFoldRightToLeft"/>
    <BoxFoldAction FoldIndex="1 -1" Action="LongFoldLeftToRight"/>
    <BoxFoldAction FoldIndex="3 -1" Action="LongFoldRightToLeft"/>
</BoxFoldingParams>
```

Figure 8-7: Folding examples for some values of BoxFoldAction/@Action



LongFoldLeft-ToRight | LongPreFoldRight-ToLeft | FrontFoldComplete | FrontFoldCom-pleteDiagonal | BackFoldComplete | ReverseFold

Dimensions and Actions for below Figures:

- Shown from print side, lid at the top, Arrow is transport direction in folder-gluer.
- In the folder-gluer the blank box is fed with the print side down.
- From this point of view all folds are made toward the -z axis.
- For front and back folds, pay attention to transport direction

Figure 8-8: BoxFoldingType Attribute for values of Type00, Type01 and Type02

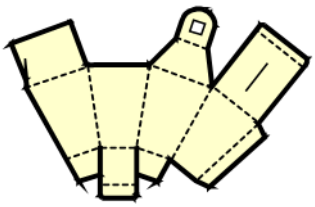
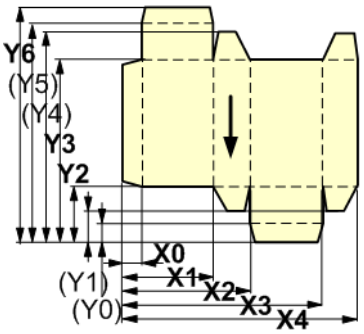
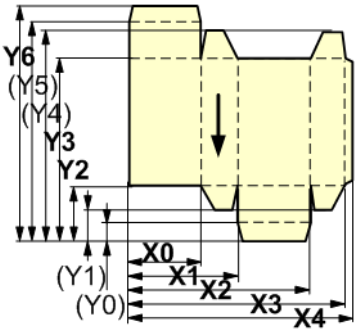
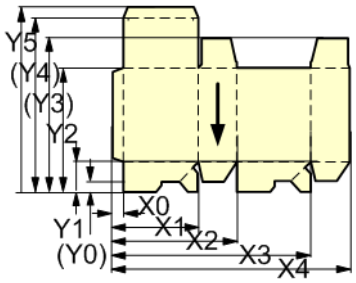
Type00	Type01	Type02
		
Individual shape with customer defined dimensions and actions	X0 LongPreFoldLeftToRight X2 LongPreFoldRightToLeft X1 LongFoldLeftToRight X3 LongFoldRightToLeft	X3 LongPreFoldLeftToRight X1 LongPreFoldRightToLeft X0 LongFoldLeftToRight X2 LongFoldRightToLeft

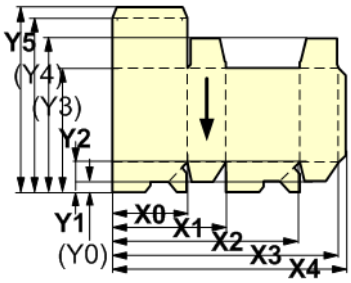
Figure 8-9: BoxFoldingType Attribute for values of Type03, Type04 and Type10

Type03	Type04	Type10

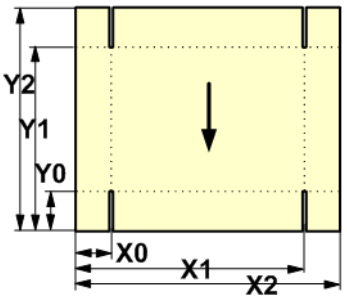
Figure 8-9: BoxFoldingType Attribute for values of Type03, Type04 and Type10



X0 LongPreFoldLeftToRight
X2 LongPreFoldRightToLeft
X2/Y1: FrontFoldComplete
X4/Y1: FrontFoldComplete
X1/Y1: FrontFoldCompleteDiagonal
X3/Y1: FrontFoldCompleteDiagonal
X1 LongFoldLeftToRight
X3 LongFoldRightToLeft



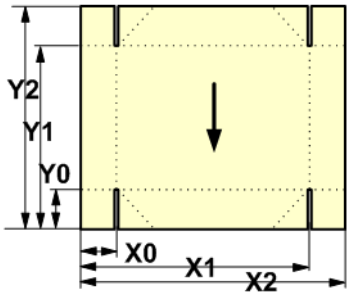
X3 LongPreFoldLeftToRight
X1 LongPreFoldRightToLeft
X1/Y1: FrontFoldComplete
X3/Y1: FrontFoldComplete
X0/Y1: FrontFoldCompleteDiagonal
X2/Y1: FrontFoldCompleteDiagonal
X0 LongFoldLeftToRight
X2 LongFoldRightToLeft



X0 LongPreFoldLeftToRight
X1 LongPreFoldRightToLeft

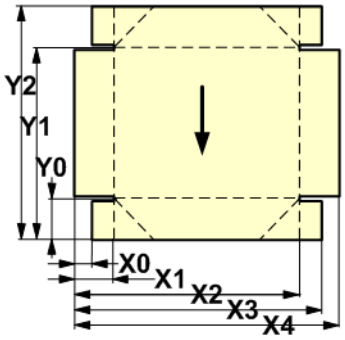
Figure 8-10: BoxFoldingType Attribute for values of Type 11, Type12 and Type13

Type11



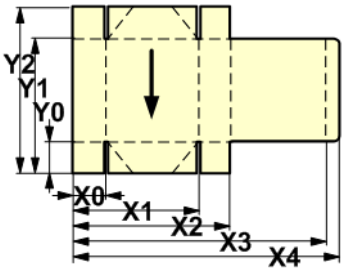
X0/Y0: FrontFoldComplete
X2/Y0: FrontFoldComplete
X0/Y2: BackFoldComplete
X2/Y2: BackFoldComplete
X1/Y0: FrontFoldCompleteDiagonal
X1/Y2: BackFoldCompleteDiagonal
X0 LongFoldLeftToRight
X2 LongFoldRightToLeft

Type12



X1/Y0:
FrontFoldCompleteDiagonal
X1/Y2:
BackFoldCompleteDiagonal
X0 LongFoldLeftToRight
X2 LongFoldRightToLeft

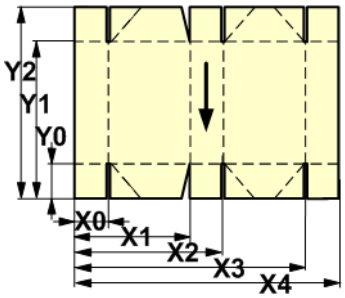
Type13



X0/Y0: FrontFoldComplete
X2/Y0: FrontFoldComplete
X0/Y2: BackFoldComplete
X2/Y2: BackFoldComplete
X1/Y0: FrontFoldCompleteDiagonal
X1/Y2: BackFoldCompleteDiagonal
X0 LongFoldLeftToRight
X2 LongFoldRightToLeft

Figure 8-11: BoxFoldingType Attribute for values of Type15 and Type20

Type15



Type20

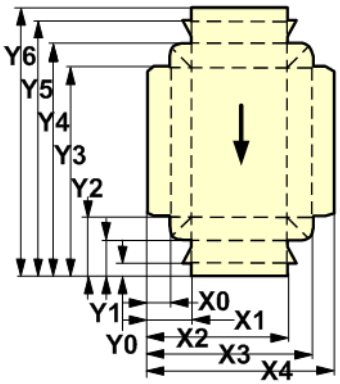


Figure 8-11: BoxFoldingType Attribute for values of Type15 and Type20

X0/Y0: FrontFoldComplete
X2/Y0 FrontFoldComplete
X4/Y0 FrontFoldComplete
X0/Y2 BackFoldComplete
X2/Y2 BackFoldComplete
X4/Y2 BackFoldComplete
X1/Y0 FrontFoldCompleteDiagonal

(continued from previous column)
X3/Y0 FrontFoldCompleteDiagonal
X1/Y2 BackFoldCompleteDiagonal
X3/Y2 BackFoldCompleteDiagonal
X0 LongFoldLeftToRight
X3 LongFoldRightToLeft
X2 LongFoldRightToLeft

(continued from previous column)
X0 LongFoldLeftToRight
X3 LongFoldRightToLeft

8.10 BoxPackingParams

New in JDF 1.1

This resource defines the parameters for packing a box of components. Details of the box used for **BoxPacking** can be found in the ***Component*** (Box) resource that is also an input of the **BoxPacking** process.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***PackingIntent***

Input of Processes: **BoxPacking**

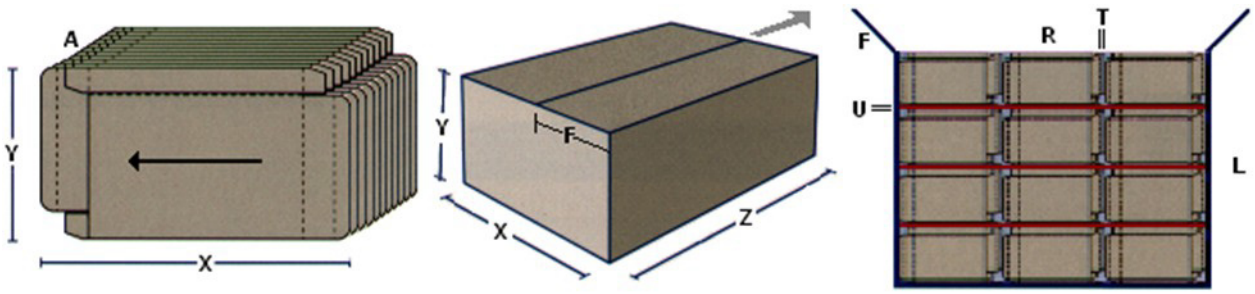
Table 8.16: BoxPackingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BoxType New in JDF 1.6	enumeration	@*BoxType* specifies the general category of the package to be packed. **Allowed values are:** Box – Boxes are convenience packaging and are not envisioned to be protection for shipping. Carton – Cartons envisioned to be protection for shipping. Envelope – Envelopes are packages that are envisioned for shipping. Tube – Tubes are cylinder shaped cartons that are envisioned for shipping.
BoxTypeDetails ? New in JDF 1.6	string	Additional details of @*BoxType*. @*BoxType* MAY be a site specific identifier. **Values include:** Neutral Carton Branded Carton Easter Bunny Box
ComponentsPerRow ? New in JDF 1.3	integer	***Component***s per row in the shipping box, as illustrated by A in ▸ Figure 8-12: Box packing. If the ***Component***s represent ***Bundle***s, the number of ***Bundle***s is specified.
Columns ? New in JDF 1.4	integer	Columns per shipping box. Columns are in the 3rd Dimension in ▸ Figure 8-12: Box packing, and are thus not illustrated.
ComponentOrientation ? New in JDF 1.4	enumeration	Defines the coordinate pair that is facing the bottom of the box, defining the horizontal plane. **Allowed values are:** XY – Axis X and Y XZ – Axis X and Z YZ – Axis Y and Z
Copies ? New in JDF 1.4	integer	Number of copies in the box. @*Copies* SHALL NOT be specified if @*MaxWeight* is present.
FillMaterial ?	NMTOKEN	Material to fill boxes that are not completely filled, as illustrated by F in ▸ Figure 8-12: Box packing. **Values include:** Any – Explicit request for system specified filling. BlisterPack None – Explicit request for no filling. Paper Styrofoam

Table 8.16: BoxPackingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Layers ? New in JDF 1.3	integer	Layers per shipping box, as illustrated by L in ▸ Figure 8-12: Box packing.
MaxWeight ? New in JDF 1.4	double	Maximum weight of a packed box in grams. @*MaxWeight* SHALL NOT be specified if @*Copies* is present.
Pattern ?	string	Name of the box packing pattern. Used to store a predefined pattern that defines the layers and positioning of individual component in the box or carton.
Rows ? New in JDF 1.3	integer	Rows per shipping box, as illustrated by R in ▸ Figure 8-12: Box packing.
Ties ? New in JDF 1.3	IntegerList	Number of tie Sheets at each row. The first value is outside the first row, the next value between the first and second row and so forth. If more rows than values are specified, counting restarts at the 0 position. If fewer layers than values are specified, all tie Sheets that are not adjacent to a row are ignored.
UnderLays ? New in JDF 1.3	IntegerList	Number of underlay Sheets at each layer, as illustrated by U in ▸ Figure 8-12: Box packing. The first value is underneath the bottom layer, the next value above the first layer and so forth. If more layers than values are specified, counting restarts at the 0 position. If less layers than values are specified, all underlay Sheets that are not adjacent to a layer are ignored.

Figure 8-12: Box packing



8.11 BufferParams

New in JDF 1.1

This resource provides controls for **Buffer** process.

Resource Properties

Resource Class: Parameter

Input of Processes: **Buffer**

Table 8.17: BufferParams Resource

NAME	DATA TYPE	DESCRIPTION
MinimumWait ?	duration	Minimum amount of time that an individual resource SHALL be buffered.

8.12 Bundle

New in JDF 1.1

Bundles are used to describe various kinds of sets of ***Bundle***s. Note that ***Bundle*** resources can MAY be created by many press or postpress processes and not only **Bundling**.

Resource Properties

Resource Class: Quantity

Resource referenced by: ***Bundle***, ***PalletizingParams***

Input of Processes:

Table 8.18: Bundle Resource

NAME	DATA TYPE	DESCRIPTION
BundleType = "Stack" Modified in JDF 1.5	enumeration	**Allowed values are:** BoundSet – Stack of components that are bound together. Box Carton CollectedStack – Components collected on a saddle, result of **Collecting** process CompensatedStack – Loose stack of compensated components Pallet Roll – Rolled components on a print Roll. Sheet – Multiple individual items printed onto one Sheet. SheetStream – Stream of individual sheets that are continuously moved from one device to another (e.g., in an inline digital finishing device). New in JDF 1.5 Stack – Loose stack of equally stacked components. StrappedStack – Strapped stack of equally stacked components. StrappedCompensatedStack – Strapped stack of compensated components. WrappedBundle
FolioCount ?	integer	Total amount of individual finished pages that this bundle contains. If not specified, it SHALL be calculated from the individual ***BundleItem*** elements.
ReaderPageCount ?	integer	Total amount of individual Reader Pages that this bundle contains. If not specified, it SHALL be calculated from the individual ***BundleItem*** elements.
SheetCount ? New in JDF 1.5	integer	Total number of physical sheets that this ***Bundle*** contains.
TotalAmount ?	integer	Total amount of individual products that this bundle contains. If the bundle contains one or more ***Bundle***[xecontains (@*ComponentType*, "FinalProduct")], @*TotalAmount* refers to the number of final products. Note that this is neither always the next level of BundleItem nor the lowest level of BundleItem. For instance, the next level MAY be the boxes in a carton, whereas the lowest level MAY be the Sheets comprising the brochure. The correct number in this example would be the number of brochures. If not specified, it SHALL be calculated from the individual ***BundleItem*** elements.
BundleItem *	element	References to the individual items that form this ***Bundle***.

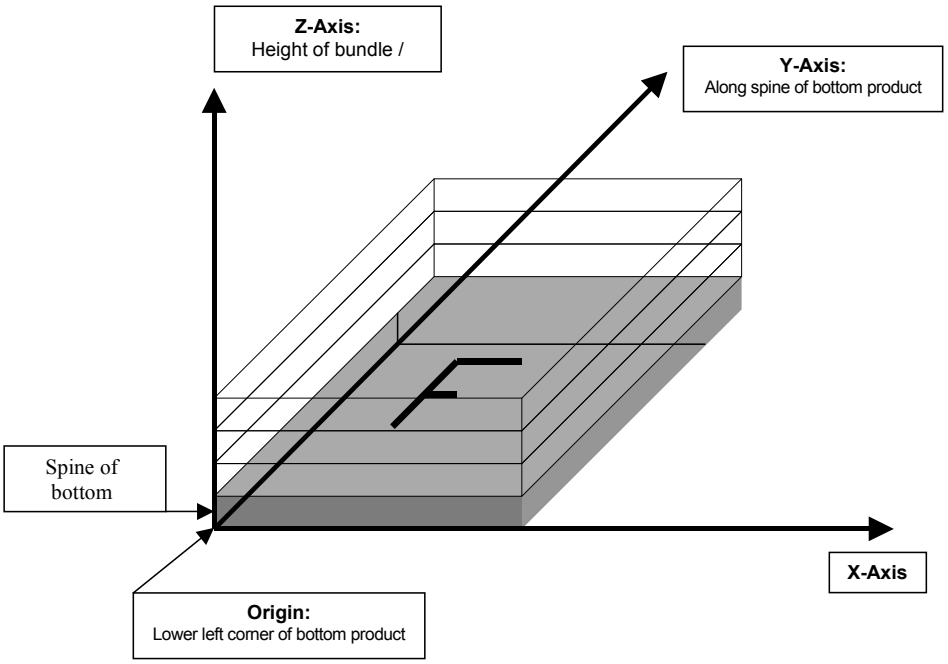
8.12.1 BundleItem

A ***Bundle*** is described as a set of ***BundleItem*** elements. Since ***BundleItem*** elements reference ***Bundle*** resources which themselves can reference further ***Bundle*** resources, the structure is recursive.

Table 8.19: BundleItem Element

NAME	DATA TYPE	DESCRIPTION
Amount	integer	Number of this type of item.
ItemName ? New in JDF 1.2	NMTOKEN	Name of the bundle item. Used for referencing individual ***BundleItem*** elements in a ***Bundle***.
Orientation ?	enumeration	Named orientation of the ***Bundle*** respective to the ***Bundle*** coordinate system. For details, see ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component. At most one of @*Orientation* or @*Transformation* SHALL be specified. **Allowed value is from:** ▸ Orientation.
Transformation ?	matrix	Orientation of the ***Bundle*** respective to the ***Bundle*** coordinate system. At most one of @*Orientation* or @*Transformation* SHALL be specified.
Bundle	refelement	Reference to a ***Bundle*** that is part of this ***Bundle***.

Figure 8-13: Packaging Process Coordinate System



Example 8.6: Bundle: Boxing and Palletizing

The following example code shows a **JDF** that describes boxing and palletizing for 4200 books. The appropriate ***Bundle*** elements have orange tags and magenta Attributes. The resources have not yet been completely filled in.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Bundle" Status="Waiting"
Type="ProcessGroup" JobPartID="ID20" Version="1.4">
    <!-- The BoxPacking Process consumes the thing to pack and the boxes-->
    <!-- The BoxPacking Process creates packed boxes -->
    <JDF ID="n0235" Status="Waiting" Type="BoxPacking" JobPartID="ID21" >
        <ResourceLinkPool>
            <ComponentLink ProcessUsage="Box" Usage="Input" rRef="BoxID"/>
            <BoxPackingParamsLink Usage="Input" rRef="BoxParamsID"/>
            <ComponentLink Usage="Input" rRef="ComponentID"/>
            <ComponentLink Usage="Output" rRef="PackedBoxID"/>
        </ResourceLinkPool>
        <!-- The BoxPacking Process has the following local resources -->
        <ResourcePool>
            <BoxPackingParams Class="Parameter" ID="BoxParamsID"
Status="Available"/>
            <Component Amount="100" Class="Quantity" ID="BoxID"
Status="Available" ComponentType="Sheet"/>
        </ResourcePool>
    </JDF>
    <ResourcePool>
        <!-- This Component describes a Box with 42 Books -->
        <Component Amount="100" Class="Quantity" ID="PackedBoxID"
Status="Unavailable" ComponentType="Sheet" >
            <Bundle BundleType="Box" TotalAmount="42">
                <BundleItem Amount="42">
                    <ComponentRef rRef="ComponentID"/>
                </BundleItem>
            </Bundle>
        </Component>
        <Component Amount="4200" Class="Quantity" ID="ComponentID"
Status="Available" ComponentType="Sheet" />
        <!-- This Component describes the contents of the pallet: 100
Boxes w. 42 Books -->
        <Component Amount="10" Class="Quantity" ID="palletContentsID"
Status="Unavailable" ComponentType="Sheet" >
            <Bundle BundleType="Pallet" TotalAmount="420">
                <BundleItem Amount="10">
                    <ComponentRef rRef="PackedBoxID"/>
                </BundleItem>
            </Bundle>
        </Component>
    </ResourcePool>
    <JDF ID="n0239" Status="Waiting" Type="Palletizing" JobPartID="ID22">
        <ResourceLinkPool>
            <ComponentLink Usage="Input" rRef="PackedBoxID"/>
            <PalletLink Usage="Input" rRef="palletID"/>
            <PalletizingParamsLink Usage="Input" rRef="palletParamsID"/>
            <ComponentLink Usage="Output" rRef="palletContentsID"/>
        </ResourceLinkPool>
        <ResourcePool>
            <Pallet Amount="10" Class="Consumable" ID="palletID"
Status="Available" PalletType="Euro800x600"/>
            <PalletizingParams Class="Parameter" ID="palletParamsID"
Status="Available" />
        </ResourcePool>
    </JDF>
</JDF>
```

8.13 BundlingParams

New in JDF 1.2

BundlingParams describes the details of a **Bundling** process.

Resource Properties

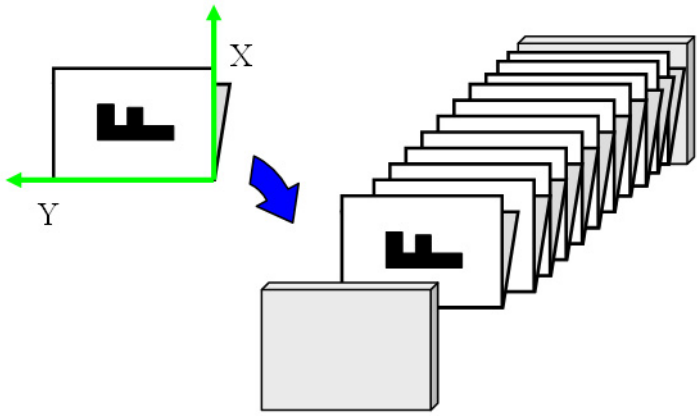
Resource Class: Parameter

Input of Processes: Bundling

Table 8.20: BundlingParams Resource

NAME	DATA TYPE	DESCRIPTION
Copies ?	integer	Number of copies within a bundle. @*Copies* SHALL NOT be specified if @*Length* is present.
Length ?	double	Length of a bundle. @*Length* SHALL NOT be specified if @*Copies* is present.

Figure 8-14: BundlingParams Coordinate System



8.14 ByteMap

This resource specifies the structure of bytemaps produced by various processes within a **JDF** system. A ***ByteMap*** represents a raster of image data. This data MAY have multiple bits per pixel, MAY represent a varying set of color planes, and MAY be interleaved. A bitmap is a special case of a ***ByteMap*** in which each pixel is represented by a single bit per color.

Personalized printing requires that certain regions of a given page be dynamically replaced. The OPTIONAL mask associated with each band of data allows for omitting certain pixels from the base image represented by the ***ByteMap*** so that they can be replaced.

Resource Properties

Resource Class: Parameter

Resource references: ***RunList***

Table 8.21: ByteMap Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BandOrdering ?	enumeration	Identifies the precedence given when ordering the produced bands. @*BandOrdering* is REQUIRED for non-interleaved data and SHALL be ignored for interleaved data if specified. **Allowed values are:** BandMajor – The position of the bands on the page is prioritized over the color. ColorMajor – All bands of a single color are played in order before progressing to the next plane. This is only possible with non-interleaved data.
ElementType ? New in JDF 1.4	enumeration	**Allowed values are from:** ▸ Table 8.161 ElementType Attribute Values**.** **Note:** Values are the same as ***LayoutElement***@*ElementType*.
FrameHeight ? Modified in JDF 1.4	integer	Height of the overall image that MAY be broken into multiple bands. **Modification note:** Starting with **JDF** 1.4, @*FrameHeight* is optional.

Table 8.21: ByteMap Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
FrameWidth ? Modified in JDF 1.4	integer	Width of overall image that MAY be broken into multiple columns. **Modification note:** Starting with **JDF** 1.4, @*FrameWidth* is optional.
Halftoned ? Modified in JDF 1.4	boolean	Indicates whether or not the data has been halftoned. **Modification note:** Starting with **JDF** 1.4, @*Halftoned* is optional.
Interleaved ? Modified in JDF 1.4	boolean	If "true", the data are interleaved or chunky. Otherwise the data are non-interleaved or planar. **Modification note:** Starting with **JDF** 1.4, @*Interleaved* is optional.
PixelSkip ?	integer	Number of bits to skip between pixels of interleaved data.
Resolution ? Modified in JDF 1.4	XYPair	Output resolution. **Modification note:** Starting with **JDF** 1.4, @*Resolution* is optional.
Band * Modified in JDF 1.4	element	Array of bands containing raster data. **Modification note:** Starting with **JDF** 1.4, ***Band*** is optional.
ColorPool ? New in JDF 1.2	refelement	Details of the colors represented in this ***ByteMap***.
FileSpec (RasterFileLocation) ?	refelement	A ***FileSpec*** resource pointing to a location where the raster is stored or is be stored shortly.
PixelColorant * Modified in JDF 1.4	element	Ordered list containing information about which colorants are represented and how many bits per pixel are used. **Modification note:** Starting with **JDF** 1.4, ***PixelColorant*** is optional.

8.14.1 Band

Table 8.22: Band Element

NAME	DATA TYPE	DESCRIPTION
Data ? Modified in JDF 1.4	URL	Actual bytes of data. **Modification note:** Starting with **JDF** 1.4, @*Data* is optional.
Height ? Modified in JDF 1.4	integer	Height in pixels of the band. **Modification note:** Starting with **JDF** 1.4, @*Height* is optional.
Mask ?	URL	1-bit mask of raster data indicating which bits of the band data to use. The mask dimensions and resolution SHALL be equivalent to the contents of the band itself.
WasMarked ? Modified in JDF 1.4	boolean	Indicates whether any rendering marks were made in this band. Thil attribute allows a band to be skipped if no marks were made in the band. **Modification note:** Starting with **JDF** 1.4, @*WasMarked* is optional.
Width ? Modified in JDF 1.4	integer	Width in pixels of the band **Modification note:** Starting with **JDF** 1.4, @*Width* is optional.

8.14.2 PixelColorant

Table 8.23: PixelColorant Element

NAME	DATA TYPE	DESCRIPTION
ColorantName	string	Name of colorant.
PixelDepth	integer	Number of bits per pixel for each colorant.

8.15 CaseMakingParams

New in JDF 1.1

This resource describes the settings of a **CaseMaking** process.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **CaseMaking**

Figure 8-15: CaseMakingParams

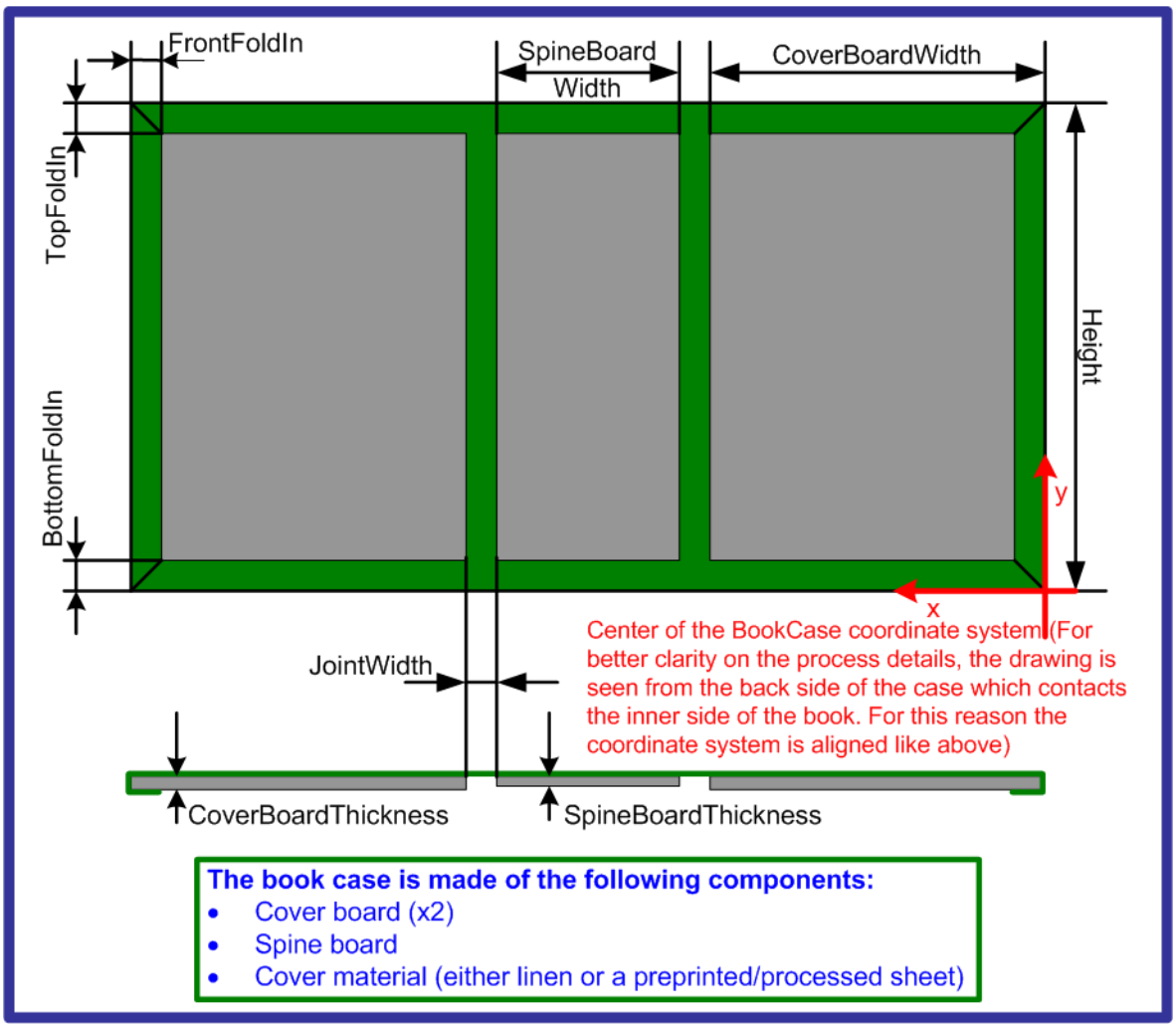


Table 8.24: CaseMakingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BottomFoldIn ?	double	Defines the width of the part of the cover material on the lower edge inside of the case. If not specified, defaults to @*TopFoldIn*.
CornerType ?	NMTOKEN	Method of wrapping the corners of the cover material around the corners of the board. **Values include:** LibraryCorner – The American Library Corner style.
CoverWidth ?	double	Width of the cover cardboard in points.
FrontFoldIn ?	double	Defines the width of the part of the cover material on the front edges inside of the case.
Height ?	double	Height of the book case, in points.

Table 8.24: CaseMakingParams Resource (Sheet 2 of 2)

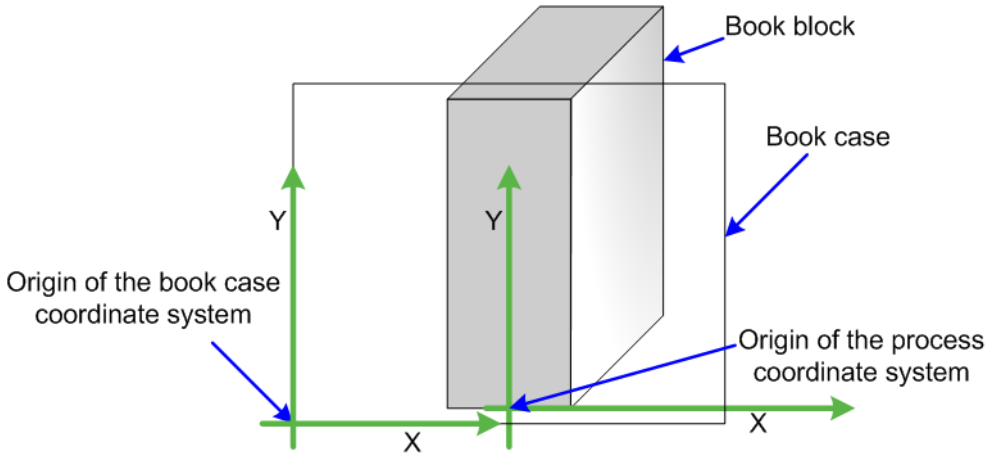
NAME	DATA TYPE	DESCRIPTION
JointWidth ?	double	Width of the joint as seen when laying the cardboard on the cover material, in points.
SpineWidth ?	double	Width of the spine cardboard, in points.
TopFoldIn ?	double	Defines the width of the cover material on the top edge inside of the case.
GlueLine ?	element	Details of the glue. Because the glue is applied to the whole back side of the cover material, ***GlueLine***/@*AreaGlue* SHALL be set to "true".

8.16 CasingInParams

New in JDF 1.1

This resource describes the settings of a **CasingIn** process. The geometry is always centered See ▸ Figure 8-16: Parameters and coordinate system for CasingIn.

Figure 8-16: Parameters and coordinate system for CasingIn



Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **CasingIn**

Table 8.25: CasingInParams Resource

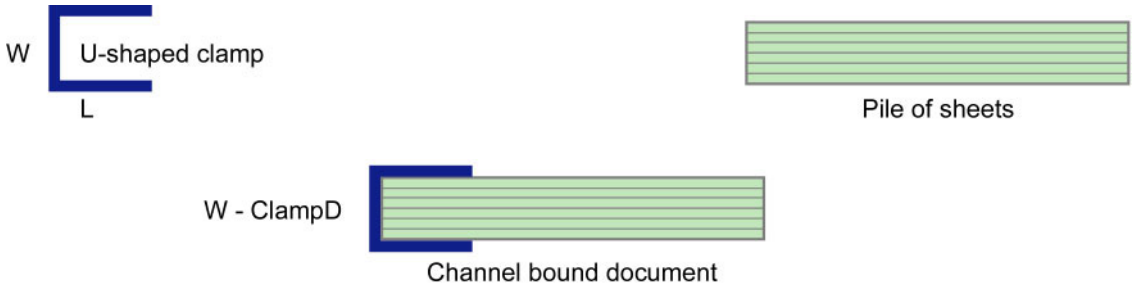
NAME	DATA TYPE	DESCRIPTION
CaseRadius ?	double	Inner radius of the case spine rounding. If not specified, no rounding of the case spine is performed.
CoverBoardWidth ? New in JDF 1.5	double	Width of the CoverBoard. Note that Height and total Case Dimensions are specified in the ***Component***(Case) of the **CasingIn** process. For details of @*CoverBoardWidth*, see also ▸ Figure 8-15: CaseMakingParams.
SpineBoardWidth ? New in JDF 1.5	double	Width of the SpineBoard. Note that Height and total Case Dimensions are specified in the ***Component***(Case) of the **CasingIn** process. For details of @*SpineBoardWidth*, see also ▸ Figure 8-15: CaseMakingParams.
GlueApplication * New in JDF 1.4	refelement	Properties of the glue to attach the case.
GlueLine + Deprecated in JDF 1.4	element	Properties of the glue used. **Deprecation note:** Starting with **JDF** 1.4, use ***GlueApplication***.

8.17 ChannelBindingParams

This resource describes the details of the **ChannelBinding** process.

Figure 8-17: Parameters used for channel binding depicts the **ChannelBinding** Process.

Figure 8-17: Parameters used for channel binding



The symbols W, L and ClampD of Figure 8-17: Parameters used for channel binding are described by the attributes @*ClampD* and @*ClampSize* of the table below.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **ChannelBinding**

Table 8.26: ChannelBindingParams Resource

NAME	DATA TYPE	DESCRIPTION
Brand ?	string	The name of the clamp (or preassembled cover with clamp) manufacturer and the name of the specific item.
ClampColor ?	NamedColor	Determines the color of the clamp/cover. If @*ClampSystem* = "true", then the color of the cover is also meant.
ClampColorDetails ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If @*ClampColorDetails* is supplied, @*ClampColor* SHOULD also be supplied.
ClampD ?	double	The distance of the clamp that was "pressed away" (see Figure 8-17: Parameters used for channel binding Parameters used for channel binding).
ClampSize ?	shape	The shape size of the clamp. The first number of the shape data type corresponds to the clamp width W (see Figure 8-17: Parameters used for channel binding) which is determined by the final height of the block of Sheets to be bound. The second number corresponds to the length L (see Figure 8-17: Parameters used for channel binding). The third corresponds to the spine length (not visible in Figure 8-17: Parameters used for channel binding). The spine length is perpendicular on the paper plane.
ClampSystem = "false"	boolean	If "true" the clamp is inside of a preassembled cover.

8.18 CoilBindingParams

This resource describes the details of the **CoilBinding** process.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **CoilBinding**

Table 8.27: CoilBindingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Brand ?	string	The name of the coil manufacturer and the name of the specific item.

Table 8.27: CoilBindingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Color ?	NamedColor	Determines the color of the coil.
ColorDetails ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If @*ColorDetails* is supplied, @*Color* SHOULD also be supplied.
Diameter ?	double	The coil diameter to be produced is determined by the height of the block of Sheets to be bound.
Material ?	enumeration	The material used for forming the coil binding. **Allowed values are:** LaqueredSteel NylonCoatedSteel PVC TinnedSteel ZincsSteel
Shift ? Deprecated in JDF 1.2	double	Amount of vertical shift that occurs as a result of the coil action while opening the document. It is determined by the distance between the holes. In **JDF** 1.2 and beyond, use the value implied by ***HoleMakingParams***/@*HoleType*.
Thickness ?	double	The thickness of the coil.
Tucked = "false"	boolean	If "true", the ends of the coils are “tucked in”.
HoleMakingParams ? New in JDF 1.2	refelement	Details of the holes in **CoilBinding**.

8.19 CollectingParams

The **Collecting** process needs no special attributes. However, this resource is provided as a container for extensions of the **Collecting** process.

Resource Properties

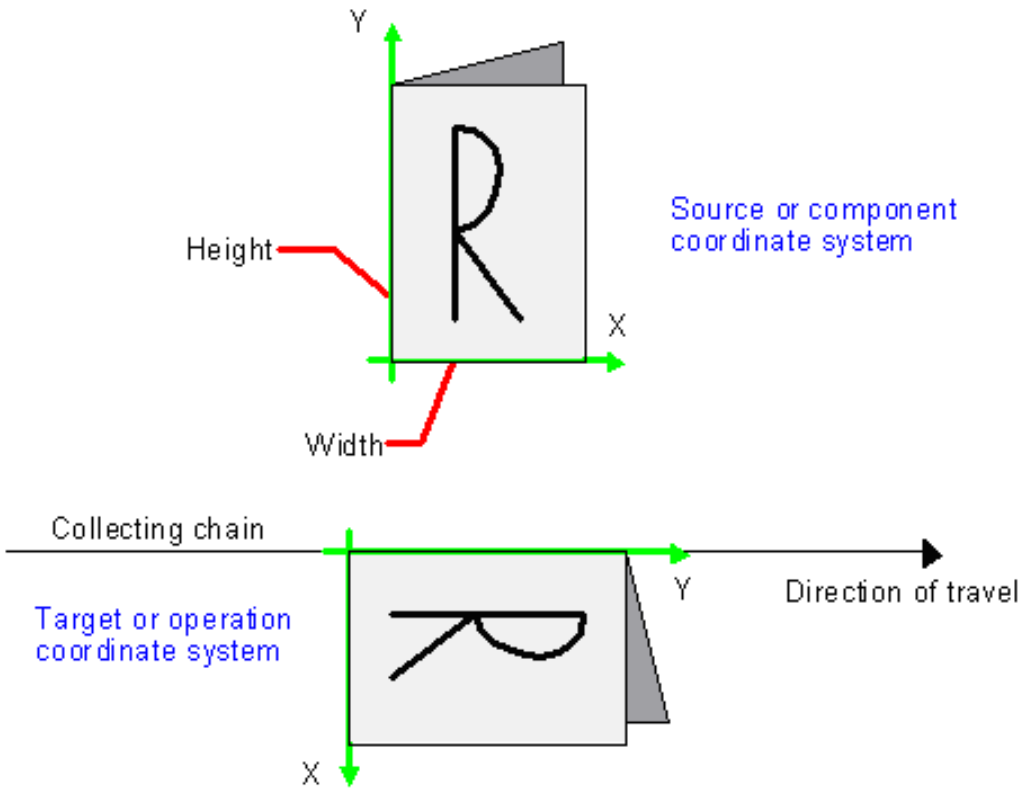
Resource Class: Parameter

Input of Processes: **Collecting**

Table 8.28: CollectingParams Resource

NAME	DATA TYPE	DESCRIPTION

Figure 8-18: Coordinate systems used for collecting



8.20 Color

Color describes the details of spot color inks, process color inks and any other coating, for instance varnish or gloss coating. Spot colors are named colors that can either be separated or converted to process colors. It is important to know the neutral density of the colorant for trapping and, in many cases, the @*Lab* values for representing them on screen. If you know the @*Lab* value, you can calculate the neutral density. When representing colors on screen, a conversion to process colors SHALL be defined. This conversion is a simple linear interpolation between the @*CMYK* value of the 100% spot color and its tint.

A color is represented by a ***Color*** element. It has a REQUIRED @*Name* attribute, which represents the name of either a spot color or a process color. When ***ColorantAlias*** has been used in ***ElementColorParams*** and/or in ***ColorantControl*** to clean up string names of spot colors, the resolved, not the uncorrected duplicate, ***ColorantAlias***/@*ReplacementColorantName* spot color name SHALL match ***Color***/@*Name*. The four names that are reserved for representing process CMYK color names are "Cyan", "Magenta", "Yellow" and "Black". Every colorant MAY have a @*Lab* and/or @*CMYK* color value. If both are specified and a system is capable of interpreting both values, the @*Lab* value overrides the @*CMYK* definition, unless the target device is compatible with CMYK (i.e., ***ColorantControl***/@*ProcessColorModel* = "DeviceCMYK"). In this case the CMYK value has precedence.

The @*Lab* value represents the *L*, *a*, *b* readings of the ink on certain media. This means that spot inks printed on three different kinds of stocks have different @*Lab* values. Pantone books, for example, provide @*Lab* values for three kinds of paper: "Coated" (not necessarily glossy), "Matte" and "Uncoated". Thus a color of ink SHOULD identify the media for which the color is specified. CMYK colors are used to approximate spot colors when they are not separated. This conversion can be done by a color management system, or there can be fixed CMYK representation defined by color books such as Pantone.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***ColorPool***, ***LayoutPreparationParams***/***PageCell***

Intent Pairing: ***ColorIntent***

Table 8.29: Color Resource (Sheet 1 of 4)

NAME	DATA TYPE	DESCRIPTION
ActualColorName ? New in JDF 1.3	string	Actual name of the color in the PDL. @*ActualColorName* SHOULD be used to identify the color.

Table 8.29: Color Resource (Sheet 2 of 4)

NAME	DATA TYPE	DESCRIPTION
CMYK ? Modified in JDF 1.2	CMYKColor	CMYK value of the 100% tint value of the colorant. @*CMYK* SHOULD be specified if known and @*ColorType*!="Transparent" and @*ColorType*!="DieLine".
ColorBook ? Modified in JDF 1.2	string	Definition of the color identification book name that is used to represent this color. The color book name SHALL match the name defined by the color book vendor. **Values include:** CIP4 ColorBook Uncoated Grade 5 PANTONE C – an example PANTONE C – an example PlaceHolder – "PlaceHolder" is a special token that indicates that the ***Color***/@*Name* is not a real color but a place holder like 'Spot1' that SHALL be resolved when the content arrives. New in JDF 1.3 **Modification note:** Starting with **JDF** 1.2, the data type changes from NMTOKEN to string.
ColorBookEntry ? Modified in JDF 1.2	string	Definition of the ***Color*** within the standard specified by @*ColorBook*. This entry SHALL exactly match the color book entry as defined by the @*ColorBook* specified vendor, including capitalization and media type extension. When using ICC Profiles, this maps to the NCL2 value of a namedColorType tag of an ICC color profile. This entry is used to map from the **JDF** ***Color*** to an ICC namedColorType tag.
ColorBookPrefix ?	string	Definition of the name prefix of the color book entry within a named ICC profile. This entry is used to map from the **JDF** ***Color*** to an ICC namedColorType tag.
ColorBookSuffix ?	string	Definition of the name suffix of the color book entry within a named ICC profile. This entry is used to map from the **JDF** ***Color*** to an ICC namedColorType tag.
ColorDetails ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If @*ColorDetails* is supplied, @*ColorName* SHOULD also be supplied.
ColorName ? New in JDF 1.1	enumeration	Mapping to a color name. **Allowed value is from:** ▸ NamedColor**.**
ColorType ? Modified in JDF 1.2	enumeration	A name that characterizes the colorant. **Allowed values are:** DieLine – Marks made with colorants of this type are ignored for trapping. **Trapping** processes need not generate a color plane for this colorant. "DieLine" can be used for auxiliary process separations. "DieLine" marks will generally appear on proof output but will not be marked on final output (e.g., plates). Note that the ***ColorantControl*** resource SHALL be correctly set up for the RIP and that @*ColorType* = "DieLine" does not implicitly remove the "DieLine" separation from final output. Normal – Marks made with colorants of this type, marks covered by colorants of this type, and marks on top of colorants of this type are trapped. Opaque – Marks covered by colorants of this type are ignored for trapping. "Opaque" can be used for metallic inks. OpaqueIgnore – Marks made with colorants of this type and marks covered by colorants of this type are ignored for trapping. "OpaqueIgnore" can be used for metallic inks. Primer – Colors with @*ColorType*="Primer" are used as background filler and SHALL be ignored when trapping. Transparent – Marks made with colorants of this type are to be ignored for trapping. **Trapping** processes are not to generate a color plane for this colorant. This value SHOULD be used for varnish.
ColorTypeDetails ? New in JDF 1.5	string	Additional information about the color type. If @*ColorType* = "DieLine", this attribute SHOULD specify the type of die line (e.g., DDES-numbers, For details, see ▸ Table 8.31 Diecutting Data (DDES3) for a list of DDES3 die line types.

Table 8.29: Color Resource (Sheet 3 of 4)

NAME	DATA TYPE	DESCRIPTION
Density ? New in JDF 1.2	double	Density value of colorant (100% tint). Whereas @*NeutralDensity* describes measurements of inks on substrate with wide-band filter functions, @*Density* is derived from measurements of inks on substrate with special small-band filter functions according to ANSI and DIN.
Gray ? New in JDF 1.4	double	Gray value of the 100% tint value of the colorant. @*Gray* SHALL be specified using a subtractive color model: 0.0 means 100% coverage with colorant, while 1.0 means no coverage.
Lab ?	LabColor	L, a, b value of the 100% tint value of the colorant.
MappingSelection ? New in JDF 1.2 Modified in JDF 1.5	enumeration	This value specifies the mapping method to be used for this color. @*MappingSelection* can be specifically used to indicate how a combination of process colorant values will be obtained for any spot color when the separation spot colorant itself is not to be used. **Allowed values is from:** ▸ MappingSelection. **Modification note:** Starting with **JDF** 1.5, the schema default has been removed and the default SHOULD be obtained from ***ColorantControl***/@*MappingSelection*.
MediaType ? Modified in JDF 1.2	string	Specifies the media type. **Values include:** Coated – Pertains to gloss coated. Matte – Pertains to matte or dull coated. Uncoated
Name	string	Name of the colorant. This is the value that SHALL match the @*Name* attribute of a ***SeparationSpec*** that references this color (e.g., in ***ColorantControl***/***DeviceNSpace***/***SeparationSpec***/@*Name* or ***ColorantControl***/***ColorantParams***/***SeparationSpec***/@*Name*). This @*Name* attribute MAY also be referenced from the @*Name* attribute in the ***Ink*** resource. Name MAY also be referenced from ***ColorantAlias***/@*ReplacementColorantName*. Only one ***Color*** with any given @*Name* SHALL be specified in a ***ColorPool***.
NeutralDensity ?	double	A number in the range of 0.001 to 10 that represents the neutral density of the colorant, defined as 10*log(1/Y). Y is the tristimulus value in CIEXYZ coordinates, normalized to 1.0.
PrintingTechnology ? New in JDF 1.5	NMTOKEN	Printing technology of the press, press module or printer. For digital printing, describes the printing technology that the media or coatings on the media are intended for or optimized for. **Values include:** DyeSublimation – for digital printing. Electrostatic – for digital printing. Flexo – for conventional printing. Gravure – for conventional printing. InkJet – for digital printing. Laser – for digital printing. Offset – for digital and conventional printing. Screen – for conventional printing. Thermal – for digital printing. **Creation Note:** Starting in **JDF** 1.5, @*PrintingTechnology* moved from ***ConventionalPrintingParams*** and additional values were copied from ***Media***,
RawName ? New in JDF 1.2	hexBinary	Representation of the original 8-bit byte stream of the ***Color*** @*Name*. Used to transport the original byte representation of a ***Color*** @*Name* when moving **JDF** tickets between computers with different locales. Only one ***Color*** with any given @*RawName* SHALL be specified in a ***ColorPool***.
sRGB ?	sRGBColor	sRGB value of the 100% tint value of the colorant.

Table 8.29: Color Resource (Sheet 4 of 4)

NAME	DATA TYPE	DESCRIPTION
UsePDLAlternateCS ? Deprecated in JDF 1.2	boolean	If "true", the alternate color space definition defined in the PDL SHALL be used for color space transformations when available. If "false", the alternate color space definitions defined in @*sRGB*, @*CMYK* or ***DeviceNColor*** of this ***Color*** SHALL be used depending on the value of ***ColorantControl***/@*ProcessColorModel*. In **JDF** 1.2 and beyond, use @*MappingSelection*.
ColorMeasurementConditions ? New in JDF 1.1	refelement	Detailed description of the measurement conditions for color measurements.
DeviceNColor *	element	Each ***DeviceNColor*** element defines the colorant in the DeviceN color space that is defined by ***DeviceNColor***/@*Name*.
FileSpec (ColorProfile) ?	refelement	A ***FileSpec*** resource pointing to an ICC named color profile that describes further details of the color. This ICC profile is intended as a source profile for the named color whose equivalent CMYK value is given in the @*CMYK* attribute.
FileSpec (TargetProfile) ?	refelement	A ***FileSpec*** resource pointing to an ICC profile that defines the target output device in case the object that uses the ***Color*** has been color space converted to a device color space. ***FileSpec*** (TargetProfile) applies to the alternate color defined by the value of @*MappingSelection*.
PrintConditionColor * New in JDF 1.2	element	Description of the printing condition specific color properties of a colorant (i.e., how is the printed color result specific to media, screening, etc.).
TransferCurve * Modified in JDF 1.1	refelement	A list of color transfer functions that is used to convert a tint value to one of the alternative color spaces. The transfer functions that are not specified here default to a linear transfer: "0 0 1 1".

8.20.1 DeviceNColor

Table 8.30: DeviceNColor Element

NAME	DATA TYPE	DESCRIPTION
ColorList	DoubleList	Value of the 100% tint value of the colorant in the ordered DeviceN space. The list SHALL have @*N* elements. A value of 0 SHALL specify no ink and a value of 1 SHALL specify full ink. The mapping of indices to colors is specified in the ***DeviceNSpace*** element of the ***ColorantControl*** resource.
N	integer	Number of colors that define the color space.
Name	string	Color space name (e.g., HexaChrome or HiFi). @*Name* SHALL match ***ColorantControl***/***DeviceNSpace***/@*Name*.

8.20.2 Diecutting Data (DDES3)

The following list of line types is taken from Annex A of ANSI® IT8.6-2002 Graphic technology — Prepress digital data exchange — Diecutting data ▸ [DDES3]. The list is included in the **JDF** specification with permission of IT8.6.

Table 8.31: Diecutting Data (DDES3) (Sheet 1 of 2)

DDES 3 LINE TYPE NUMBER	DDES3 LINE TYPE	DESCRIPTION
12	Non-varnish / UV area	Contour indicating a varnish free area
15	Printing / UV Blanket Edge	Contour enclosing a spot varnish area. Spot varnish will be applied with a varnish blanket.

Table 8.31: Diecutting Data (DDES3) (Sheet 2 of 2)

DDES 3 LINE TYPE NUMBER	DDES3 LINE TYPE	DESCRIPTION
16	Zipper / Tear Strip / Tear Edge - reference lines for cutting edge	Cutting contours indicating a tear strip.
17	Wave / Scallop - reference lines for cutting edge	Cutting contours indicating a wave /scallop.
18	Punches - reference lines for center / cutting edge	Contours indicating the shape and center of a punch
100	Miscellaneous ruled lines for dies	
101	Knife / Cutting rule	Contour indicating how the printed artwork will be cut from the printed sheet e.g. with a guillotine cutter or die cutting device.
102	Crease / Scoring rule	Contour indicating where the substrate will be creased to guide subsequent folding.
103	Perforation (Alternating cutting and spaces)	Contour indicating where the substrate will be perforated.
104	Cutscore / Halfcut (Partial depth cutting rule)	Contour indicating where the substrate will be cut partially i.e. not entirely through the material. Cutting is done from the front side.
105	Cut-Crease rule (Alternating cutting and creasing rule)	Contour indicating alternating cutting and creasing
106	Cutscore-Crease (Alternating partial depth cutting and creasing rule)	Contour indicating alternating half-cutting and creasing
107	Reverse cutscore / halfcut (for anvil in die)	Contour indicating where the substrate will be cut partially i.e. not entirely through the material. Cutting is done from the back side.
108	Emboss / Deboss crease profile	Contour enclosing an area where embossing will be applied.

8.20.3 PrintConditionColor

New in JDF 1.2

The ***Color*** element describes the specific properties of a colorant (named in ***Color***/@*Name*) when applied in a given printing condition (i.e., media surface, media opacity, media color and screening/RIP (e.g., halftone) technology). It is used to overwrite the generic values of ***Color***, which are supplied as the default. See the descriptions in color for details of the individual attributes and elements.

Table 8.32: PrintConditionColor Element

NAME	DATA TYPE	DESCRIPTION
CMYK ?	CMYKColor	@*CMYK* of the ***Color***. **Default value is from:** parent ***Color***/@*CMYK*
ColorBook ?	string	@*ColorBook* of the ***Color***. **Default value is from:** parent ***Color***/@*ColorBook*
ColorBookEntry ?	string	@*ColorBookEntry* of the ***Color***. **Default value is from:** parent ***Color***/@*ColorBookEntry*
ColorBookPrefix ?	string	@*ColorBookPrefix* of the ***Color***. **Default value is from:** parent ***Color***/@*ColorBookPrefix*
ColorBookSuffix ?	string	@*ColorBookSuffix* of the ***Color***. **Default value is from:** parent ***Color***/@*ColorBookSuffix*
Density ?	double	@*Density* of the ***Color***. **Default value is from:** parent ***Color***/@*Density*
Lab ?	LabColor	@*Lab* of the ***Color***. **Default value is from:** parent ***Color***/@*Lab*
MappingSelection ? New in JDF 1.2	enumeration	This value specified the mapping method to be used for this color. **Default value is from:** parent ***Color***/@*MappingSelection*. **Allowed value is from:** ▸ MappingSelection.
MediaSide = "Both"	enumeration	***Media*** front and back surfaces can be different, affecting color results. If the ***Media***/@*FrontCoatings*, ***Media***/@*BackCoatings* or ***Media***/@*Gloss* attributes indicate differences in surface then @*MediaSide* can be used to specify the side of the media to which the ***Color*** attributes pertain. **Allowed values are:** Front Back Both
NeutralDensity ?	double	@*NeutralDensity* of the ***Color***. **Default value is from:** parent ***Color***/@*NeutralDensity*
PrintConditionName ?	NMTOKEN	@*PrintConditionName* specifies a particular screening condition and printing condition that this ***Color*** element applies to.
sRGB ?	sRGBColor	@*sRGB* of the ***Color***. If not specified, defaults to the parent ***Color***/@*sRGB*.
DeviceNColor *	element	***DeviceNColor*** of the ***Color***. If not specified, defaults to the parent ***Color***/***DeviceNColor***.
FileSpec	refelement	***FileSpec*** (TargetProfile) of the ***Color***. If not specified, defaults to the parent ***Color***/***FileSpec*** (TargetProfile)
Media *	refelement	Specifies one or more ***Media*** that this ***Color*** applies to. When PrintCondition-Color is present, the parent attribute, ***Color***/@*MediaType*, is ignored. If ***Media*** is not specified, ***Color*** applies to print processes with a matching @*PrintConditionName*.
TransferCurve *	refelement	***TransferCurve*** of the ***Color***. If not specified, defaults to the parent ***Color***/***TransferCurve***.

Example 8.7: Color

This is an example of the structure for ***Color***. The transfer curves in this example are defined for process CMYK and sRGB, independently.

```
        <Color Class="Parameter" ID="C000" Status="Available" CMYK="0.2 0.3 0.4 0.5"
Density="3.14" Lab="20. 30. 40." MediaType="Coated"
Name="PANTONE Deep Blue" sRGB="0.6 0.7 0.9">
            <TransferCurve Curve="0 0 .5 .4 1 1" Separation="Cyan"/>
            <TransferCurve Curve="0 0 .5 .6 1 1" Separation="Magenta"/>
            <TransferCurve Curve="0 0 1 1" Separation="Yellow"/>
            <TransferCurve Curve="0 0 1 1" Separation="Black"/>
            <TransferCurve Curve="0 0 1 1" Separation="sRed"/>
            <TransferCurve Curve="0 0 1 1" Separation="sGreen"/>
            <TransferCurve Curve="0 0 1 1" Separation="sBlue"/>
        </Color>
```

Example 8.8: ColorantControl: Content-Ignorant MIS

New in JDF 1.4

```
<ColorantControl Class="Parameter" ID="r000004" ProcessColorModel="DeviceCMYK"
        Status="Available">
    <!--Note that all Strings in ColorantParams etc. use Color/@Name,
        NOT Color/@ActualColorName-->
    <ColorantParams>
        <SeparationSpec Name="Spot1"/>
        <SeparationSpec Name="BlackText"/>
    </ColorantParams>
</ColorantControl>
```

Example 8.9: ColorantControl: Synchronized with Input

New in JDF 1.4

Example of initial (previous) ***ColorantControl*** after synchronizing with input. This example specifies the replacement color name with a new *@ActualColorName* attribute in the ***Color*** element. This approach has the disadvantage of needing a new attribute. However, it has the following advantages:

- no ambiguity in case of multiple names (***ColorantAlias*** is used only as a pure aliasing mechanism)
- The name is localized in the ***ColorPool***, which should be more central and not differ (e.g., between proofing and final imaging).

- it is “easier” to implement

```
        <!--ColorantControl after prepress has correctly set ActualColorName based
            on pdl content-->
        <ColorantControl Class="Parameter" ID="r000004" ProcessColorModel="DeviceCMYK"
Status="Available">
            <!--Note that all Strings in ColorantParams etc. use Color/@Name,
                NOT Color/@ActualColorName-->
            <ColorantParams>
                <SeparationSpec Name="Spot1"/>
                <SeparationSpec Name="BlackText"/>
            </ColorantParams>
            <ColorPoolRef rRef="r000005"/>
        </ColorantControl>
        <ColorPool Class="Parameter" ID="r000005" Status="Available">
            <!--Color that maps the predefined separation Black
                ActualColorName is the new attribute that replaces
                ExposedMedia/@DescriptiveName as the "Main" PDL color-->
            <Color ActualColorName="Schwarz" CMYK="0 0 0 1" Class="Parameter"
                Name="Black"/>
            <Color ActualColorName="Gelb" CMYK="0 0 1 0" Class="Parameter"
                Name="Yellow"/>
            <!--ActualColorName defaults to Name if not specified-->
            <Color CMYK="1 0 0 0" Class="Parameter" Name="Cyan"/>
            <Color Class="Parameter" Name="Magenta"/>
            <Color ActualColorName="Acme Aqua" CMYK="0.7 0.2 0.03 0.1"
                Class="Parameter" Name="Spot1"/>
            <Color ActualColorName="VersionsText" CMYK="0 0 0 1" Class="Parameter"
                Name="BlackText"/>
        </ColorPool>
```

Example 8.10: ColorantControl: Synchronized with Input with Alias

New in JDF 1.4

Example of initial ***ColorantControl*** after synchronizing with input that contains an alias.

```
        <ColorantControl Class="Parameter" ID="r000004" ProcessColorModel="DeviceCMYK"
                Status="Available">
            <!--ColorantControl after prepress has correctly set ActualColorName based
                on pdl content-->
            <!--Note that all Strings in ColorantParams etc. use Color/@Name,
                NOT Color/@ActualColorName-->
            <ColorantParams>
                <SeparationSpec Name="Spot1"/>
                <SeparationSpec Name="BlackText"/>
            </ColorantParams>
            <ColorPoolRef rRef="r000005"/>
            <!--ColorantAlias that maps the additional representations
(               noir, schwarz) to the predefined separation Black-->
            <ColorantAlias Class="Parameter" RawNames="6E6F6972 73636877E4727A"
                    ReplacementColorantName="Black">
                <SeparationSpec Name="noir"/>
                <SeparationSpec Name="schwarz"/>
            </ColorantAlias>
        </ColorantControl>
        <ColorPool Class="Parameter" ID="r000005" Status="Available">
            <!-- ColorPool is same as previous example -->
        </ColorPool>
```

8.21 ColorantControl

ColorantControl is a resource used to control the use of color when processing PDL pages. The attributes and elements of the ***ColorantControl*** resource describe how color information embedded in PDL pages SHALL be translated into device colorant information.

Colorants are referenced in ***ColorantControl*** by name only. Additional details about individual colorants can be found in the ***Color*** element of the ***ColorPool*** resource. ***ColorantControl*** uses the subset of colors specified in

@*ColorantConvertProcess*. The ***ColorantControl*** resources control which device colorants will be used as well as how document colors will be converted into device color spaces and how conflicting color information are to be resolved. Separation control is specified by the process being present. For example:

ColorantControl can be used as follows to define the specific colorants of a targeted output ***DeviceNSpace*** when the ***DeviceNSpace*** process colors are the only colorants used on the Job:

- ***ColorantControl***/***ColorPool***/@*ColorantSetName* matches ***ColorantControl***/***DeviceNSpace***/@*Name*, and
- a ***ColorantControl***/***ColorPool***/***Color*** resource (with correct @*Name* of colorant and other defining attributes) exists for each colorant of the ***DeviceNSpace*** as given in:
  - ***ColorantControl***/***DeviceNSpace***/***SeparationSpec***/@*Name*.

ColorantControl can be used as follows to define the specific colorants of a targeted output when both CMYK process colors and separate spot colorants are used for the final production printing, but a local printer equivalent of the spot color is used for proofing:

- ***ColorPool***/@*ColorantSetName* is and expanded name set including ***Color*** resources for the CMYK process primaries and the @*ReplacementColorantName* spot colorant, and
- Then for that spot color...
  - ***ColorPool***/***Color***/@*Name*
  - ***ColorPool***/***Color***/@*MappingSelection* attribute Value = "UseLocalPrinterValues", (used by a **ColorSpaceConversion** process only in the proofing instance).
- For proof printing:
  - ***ColorantControl***/@*ColorantParams* does not list that spot colorant.
- For production printing:
  - ***ColorantControl***/@*ColorantParams* and ***ColorantControl***/@*ColorantOrder* both include that spot colorant.

Resource Properties

Resource Class:	Parameter
Intent Pairing:	***ColorIntent***
Example Partition:	"DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetName", "Side", "SignatureName"
Input of Processes	**ColorCorrection**, **ColorSpaceConversion**, **ConventionalPrinting**, **DigitalPrinting**, **ImageSetting**, **Interpreting**, **PreviewGeneration**, **Separation**, **Stripping**, **Trapping**
Output of Processes:	**ColorSpaceConversion**

Table 8.33: ColorantControl Resource (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
ForceSeparations = "false"	boolean	If "true", forces all colorants to be output as individual separations, regardless of any values defined in ***ColorantControl*** (i.e., all separations in a document are assumed to be valid and are output individually). A value of "false" specifies to respect the parameters specified in ***ColorantControl*** and elsewhere in the **JDF**.
InternalColorModel ? New in JDF 1.5	enumeration	Internal color model that SHALL be used by a device that supports enhanced color models. **Allowed values are:** Basic – Use the basic color model selected by this ***ColorantControl***. Enhanced – Use the enhanced color model that is implied by this ***ColorantControl*** (e.g., Use "LightCyan", "LightMagenta" in addition to CMYK). Explicit – Use the elements of the enhanced color model that are explicitly listed in ***ColorantOrder***.
MappingSelection ? New in JDF 1.5	enumeration	This value specifies the default mapping method to be used for all separations. Note that @*MappingSelection* MAY be overridden by ***Color***/@*MappingSelection*. @*MappingSelection* can be specifically used to indicate how a combination of process colorant values SHALL be obtained for any spot color when the separation spot colorant itself is not to be used. **Allowed value is from:** ▸ MappingSelection.

Table 8.33: ColorantControl Resource (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
ProcessColorModel ? Modified in JDF 1.4	NMTOKEN	Specifies the model to be used for rendering the colorants defined in color spaces into process colorants. **Values include:** DeviceCMY – Process colors SHALL be Cyan Magenta and Yellow. DeviceCMYK – Process colors SHALL be Cyan Magenta Yellow and Black. DeviceGray – Process color SHALL be Black. DeviceN – The specific DeviceN color space to operate on is defined in the ***DeviceNSpace*** resource. If this value is specified then ***DeviceNSpace*** SHALL also be present. DeviceRGB – Process colors SHALL be Red Green and Blue. None – No colorants other than those specified in ***ColorantParams*** SHALL be output. New in JDF 1.4
ColorantAlias *	element	Identify one or more named colorants that are to be replaced with a specified named colorant. The identified colorant remappings in this ***ColorantAlias*** MAY be consolidated for processing from the information received in the ***LayoutElement***/***ElementColorParams***/***ColorantAlias*** resources with the Job content. Multiple ***ColorantAlias*** elements with identical values of ***ColorantAlias***/@*ReplacementColorantName* SHALL NOT be specified in the same ***ColorantControl*** resource context.
ColorantConvertProcess ? New in JDF 1.4	element	List of colors that SHALL be converted to process colors. Defaults to all colors that are neither listed in ***ColorantParams*** nor implied by @*ProcessColorModel*. Application can issue a warning for all PDL colors that are not in (***ColorantParams*** + ***ColorantConvertProcess*** + implied by @*ProcessColorModel*) lists.
ColorantOrder ?	element	The ordering of named colorants to be processed, for example in the RIP. All of the colorants named SHALL either occur in the ***ColorantParams*** list or be implied by the @*ProcessColorModel*. If present, then only the colorants specified by ***ColorantOrder*** SHALL be output. Colorants listed in the ***ColorantParams***, or implied by the @*ProcessColorModel*, but not listed in ***ColorantOrder***, SHALL NOT be output. They SHALL still be processed for side effects in the colorants that are listed such as knockouts or trapping. If not present, then all colorants specified in ***ColorantParams*** and implied by @*ProcessColorModel* are output. The explicit or implied value of ***ColorantOrder*** MAY be modified by an implied Partition of the ***ColorantControlLink***. If one or more ***ColorantControlLink*** /***Part***/@*Separation* are specified, ***ColorantOrder*** is reduced to the list. It is an error to specify values of ***ColorantControlLink***/***Part***/@*Separation* that are not explicitly stated or implied by ***ColorantOrder***.
ColorantParams ?	element	A set of named colorants. This list defines all the colorants that are expected to be available on the device where the process will be executed. Named colors found in the PDL that are not listed in ***ColorantParams*** will be implemented through their @*ProcessColorModel* equivalents. (See ***ElementColorParams*** and **ColorSpaceConversion** process.) The colorants implied by the value of @*ProcessColorModel* are assumed and SHALL NOT be specified in this list. The spot colors defined in ***ColorIntent***/***ColorsUsed*** will in general be mapped to ***ColorantParams*** for each spot color to be used as part of any Product Intent to process conversion.

Table 8.33: ColorantControl Resource (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
ColorPool ?	refelement	Pool of ***Color*** elements that define the specifics of the colors implied by @*ProcessColorModel* and named in ***ColorantControl***. ***ColorantControl*** uses a subset of the total ***ColorPool***. The subset that ***ColorantControl*** uses from ***ColorPool*** is the subset of @*ProcessColorModel* colors (possibly all), and the subset of spot colors (possibly all) designated to be processed in this instance using specific separation colorants. ***ColorPool*** in total includes spot colors in the Job for which a **JDF** process color equivalent mapping is required. Those colors are used by **ColorSpaceConversion** when ***ColorPool***/***Color***/@*MappingSelection* = "UseProcessColorValues". In that case, the process color equivalent for the spot color is taken from the available information in the ***Color*** resource for that spot color.
ColorSpaceSubstitute *	element	Each Subelement identifies a colorant that SHALL be replaced by another colorant.
DeviceColorantOrder ?	element	The ordering of named colorants (e.g., order of laying them down) to be output on the device, such as press modules. Note that this SHALL be synchronized with the device output ICC profile. All of the named colorants SHALL occur in ***ColorantOrder*** if it is present. If ***ColorantOrder*** is not present, then all of the named colorants SHALL occur in the ***ColorantParams*** list, or be implied by the @*ProcessColorModel*. If the ***DeviceColorantOrder*** element is not specified, the order for laying down colorants defaults to ***ColorantOrder***.
DeviceNSpace * Modified in JDF 1.5	element	***DeviceNSpace***defines the colorants that make up a DeviceN color space. The ***DeviceNSpace*** SHALL be present if the @*ProcessColorModel* value is "DeviceN". **Modification note:** Starting with **JDF** 1.5, the data type changes from refelement to element.

8.21.1 ColorantConvertProcess

New in JDF 1.4.

Table 8.34: ColorantConvertProcess Element

NAME	DATA TYPE	DESCRIPTION
SeparationSpec *	element	The names of the colorants that define the respective lists.

8.21.2 ColorantOrder

Table 8.35: ColorantOrder Element

NAME	DATA TYPE	DESCRIPTION
SeparationSpec *	element	The names of the colorants that define the respective lists.

8.21.3 ColorantParams

Table 8.36: ColorantParams Element

NAME	DATA TYPE	DESCRIPTION
SeparationSpec *	element	The names of the colorants that define the respective lists.

8.21.4 DeviceColorantOrder

Table 8.37: DeviceColorantOrder Element

NAME	DATA TYPE	DESCRIPTION
SeparationSpec *	element	The names of the colorants that define the respective lists.

8.21.5 ColorSpaceSubstitute

Table 8.38: ColorSpaceSubstitute Element

NAME	DATA TYPE	DESCRIPTION
PDLResourceAlias	refelement	A reference to a color space description that replaces the color space defined by the colorants described by the ***SeparationSpec*** element(s).
SeparationSpec + Modified in JDF 1.2	element	A list of names that defines the colorants to be replaced. This could be a single name in the case of a @*Separation* color space, or more than one name in the case of a DeviceN color space.

The following table describes which separations are output for various values of @*ProcessColorModel*, ***ColorantOrder***, ***ColorantControlLink***, ***ColorantParams*** and ***DeviceColorantOrder***. Note that all separations that are neither specified in ***ColorantParams*** nor implied by @*ProcessColorModel* are mapped to the colors implied by @*ProcessColorModel* prior to any color selection defined by ***ColorantOrder***.

Table 8.39: Sample output for different values of ProcessColorModel, ColorantParams, ColorantOrder, ColorantControlLink and DeviceColorantOrder Elements. (Sheet 1 of 2)

PROCESSCOLORMODEL	COLORANTPARAMS	COLORANTORDER	COLORANTCONTROLLINK /PART/@SEPARATION	COLORANTS NOT SHOWN IN THE OUTPUT	SEPARATIONS THAT ARE OUTPUT AND ORDERED FOR PRESS USING DEVICECOLORANTORDER
DeviceCMYK	Not Present	Cyan Magenta	—	Yellow Black	Cyan Magenta (If ***DeviceColorantOrder*** is not present then lay down order will be Cyan first, Magenta last.)
DeviceCMYK	Spot1 Spot2	Cyan Magenta Yellow Black Spot2	—	Spot1	Cyan Magenta Yellow Black Spot2
DeviceCMYK	Spot1 Spot2	Cyan Magenta Yellow Black Spot2	Cyan Magenta	Spot1 Spot2 Yellow Black	Cyan Magenta
DeviceGray	Spot1 Spot2	Black Spot2	—	Spot1	Black Spot2

Table 8.39: Sample output for different values of ProcessColorModel, ColorantParams, ColorantOrder, ColorantControlLink and DeviceColorantOrder Elements. (Sheet 2 of 2)

PROCESSCOLORMODEL	COLORANTPARAMS	COLORANTORDER	COLORANTCONTROLLINK /PART/@SEPARATION	COLORANTS NOT SHOWN IN THE OUTPUT	SEPARATIONS THAT ARE OUTPUT AND ORDERED FOR PRESS USING DEVICECOLORANTORDER
DeviceN (with example N = 2 colorants as identified in ***DeviceNSpace***)	Spot1 Spot2	Spot2 DeviceN 1 DeviceN 2	—	Spot1	DeviceN 1 DeviceN 2 Spot2 The reordering is accomplished using ***DeviceColorantOrder***.

8.22 ColorCorrectionParams

This resource provides the information needed for an operator to correct colors on some PDL pages or content elements such as image, graphics or formatted text.

The preferred color adjustment method allows for multi-dimensional adjustments through the use of either an ICC Abstract profile or an ICC DeviceLink profile. The adjustments are not universally colorimetrically calibrated. However, when either of the ICC profile adjustment methods are used, these standard ICC profile formats can be interpreted and applied using generally recognized ICC profile processing techniques. Use of the ICC Abstract profile adjustment will cause the adjustment to be applied in ICC Profile Connection Space, after each source profile is applied, in sequence before final target color conversion. Use of the ICC DeviceLink profile adjustment will cause the adjustment to be applied in final target device space, after the final target color conversion.

In addition to color adjustment using an ICC profile, the @*AdjustXXX* attributes each provide a direct color adjustment applied to the interpretation of the PDL data at an implementation dependent point in the processing after each source profile is applied (if source-to-destination color conversion is needed). The L*a*b* values range from -100 to +100 to indicate the minimum and maximum of the range that the system supports. A "0" value means no adjustment. The color adjustment attributes differ from the Tone Reproduction Curve (TRC) attributes that can be applied later in the processing path in two key ways. First, the @*AdjustXXX* use, even when included in the Job, will vary as a function of Job content. Second, the data values associated with the @*AdjustXXX* attributes are arbitrary, and their interpretation will be printer dependent. For details about these attributes, see ▸ Appendix D Color Adjustment.

Note: These color adjustments are not available in any ***Intent Resource*** (e.g., ***ColorIntent***). In order to request such adjustment in a Product Intent Job ticket supplied to a print provider, attach to a Product Intent node an incomplete **ColorCorrection** process with a ***ColorCorrectionParams*** resource specifying the requested@*AdjustXXX* attributes.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***ColorIntent***

Example Partition: "DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetName", "Side", "SignatureName"

Input of Processes: **ColorCorrection**

Table 8.40: ColorCorrectionParams Resource

NAME	DATA TYPE	DESCRIPTION
ColorManagementSystem ?	string	Identifies the preferred ICC color-management system to use when performing color transformations. When specified, this attribute overrides any default selection of a color management system by an application and overrides the "CMM Type" value (bytes 4-7 of an ICC Profile Header) in any of the Job related ICC profiles. This string attribute Value identifies the manufacturer of the preferred CMM and SHALL match one of the registered four-character ICC CMM Type values. **Values include those from:** *ICC Manufacturer's Signature Registry* **at** http://www.color.org. **Example values:** "ADBE" for the Adobe CMM and "KODA" for the Kodak CMM.
ColorCorrectionOp *	element	List of ***ColorCorrectionOp*** Subelements. ***ColorCorrectionOp*** SHOULD contain the complete set of parameters for a given color correction operation. Otherwise the results are implementation dependent.
FileSpec ?	refelement	A ***FileSpec*** resource pointing to an ICC profile that describes the characterization of the final output target device.
FileSpec (WorkingColorSpace) ? Deprecated in JDF 1.1	refelement	A ***FileSpec*** resource pointing to an ICC profile that describes the assumed characterization of "CMYK", "RGB" and "Gray" color spaces.

8.22.1

If present, the following attributes SHALL be applied at a point where an abstract profile would be applied following any abstract profiles used in the order: @*AdjustLightness*, @*AdjustContrast*, @*AdjustSaturation*, @*AdjustHue*.

8.23 ColorPool

The ***ColorPool*** resource contains a pool of all ***Color*** elements referred to in the Job. In general, it will be referenced as a ***ResourceRef*** from within resources that require access to color information.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***ColorIntent***, ***NumberingIntent***, ***ByteMap***, ***ColorantControl***, ***FormatConversionParams***, ***LayoutElement***, ***PageList***, ***ShapeDef***

Intent Pairing: ***ColorIntent***

Table 8.41: ColorPool Resource

NAME	DATA TYPE	DESCRIPTION
ColorantSetName ?	string	A string used to identify the named colorant parameter set. This string will be used to identify a set of color definitions (typically associated with a particular class of Job or a particular press). **Note:** This value will typically be identical to ***ColorIntent***/@*ICCColorStandard* or ***ColorIntent***/@*ColorStandard*.
Color *	element	Individual named color.

8.24 ColorSpaceConversionParams

This set of parameters defines the rules for a **ColorSpaceConversion** process, the elements of which define the set of operations to be performed. Information inside the ***ColorSpaceConversionOp*** elements defines the operation and identifies the color spaces and types of objects to operate on. Other attributes define the color management system to use, as well as the working color space and the final target device.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***ColorIntent***, ***ProofingIntent***

Example Partition: "DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetName", "Side", "SignatureName"

Input of Processes: **ColorSpaceConversion**

Table 8.42: ColorSpaceConversionParams Resource

NAME	DATA TYPE	DESCRIPTION
ColorManagementSystem ?	string	Identifies the preferred ICC color management system to use when performing color transformations. When specified, this attribute overrides any default selection of a color management system by an application and overrides the "CMM Type" value (bytes 4-7 of an ICC Profile Header) in any of the Job related ICC profiles. This string attribute Value identifies the manufacturer of the preferred CMM and SHALL match one of the registered four-character ICC CMM Type values. **Values include those from:** *ICC Manufacturer's Signature Registry* **at** http://www.color.org**.** **Example values:** "ADBE" for the Adobe CMM and "KODA" for the Kodak CMM.
ConvertDevIndepColors ? Deprecated in JDF 1.1	boolean	When "true", incoming device-independent colors are processed to the selected device space. If the chosen operation is "Untag" and the characterization data are in the form of an ICC profile, then the profile is removed. Otherwise, these colors are left untouched. The functionality of @*ConvertDevIndepColors* is superseded by including one or more ***ColorSpaceConversionOp*** with @*SourceCS* = "DevIndep" in **JDF** 1.1.
ICCProfileUsage = "UsePDL" New in JDF 1.2	enumeration	@*ICCProfileUsage* specifies where to obtain either the destination profile or device Link transform that SHALL be applied. **Note:** Use of a final target device profile provides a profiled destination to be used when converting a source object through PCS (Profiled Connection Space) to that profiled destination, and a device Link transform specifies a conversion directly of the source object from the source space directly to the destination. **Note:** PDF/X workflows assume that @*ICCProfileUsage*="UsePDL" **Allowed values are:** UsePDL – If present, the embedded target profile SHALL be used. UseSupplied – The embedded target profile SHALL NOT be used.
ColorSpaceConversionOp *	element	List of ***ColorSpaceConversionOp*** elements, each of which identifies a type of object, defines the source color space for that type of object, and specifies the behavior of the conversion operation for that type of object. The XML order of ***ColorSpaceConversionOp*** elements is significant, and when multiple elements apply to the same object, they are applied in that XML order. A ***ColorSpaceConversionOp*** can modify the characteristics of an object such that its selection criteria is also modified. Thus, if two ***ColorSpaceConversionOp*** elements select the same set of objects, and the first element changes the object in such a way that the object would no longer be selected by the second element, then the second ***ColorSpaceConversionOp*** SHALL NOT be applied to that object. ***ColorSpaceConversionOp*** SHOULD contain the complete set of parameters for a given color space conversion operation. Otherwise the results are implementation dependent. A ***ColorSpaceConversionOp*** process included as part of a raster image processing combined process shall include an implied convert operation as its last operation (causing all other unconverted color spaces to be converted according to the raster image processor's PDL).
FileSpec (FinalTargetDevice) ?	refelement	A ***FileSpec*** resource pointing to an ICC profile that describes the characterization of the final output target device.
FileSpec (WorkingColorSpace) ? Deprecated in JDF 1.1	refelement	A ***FileSpec*** resource pointing to an ICC profile that describes the assumed characterization of "CMYK", "RGB" and "Gray" color spaces.

8.25 Component

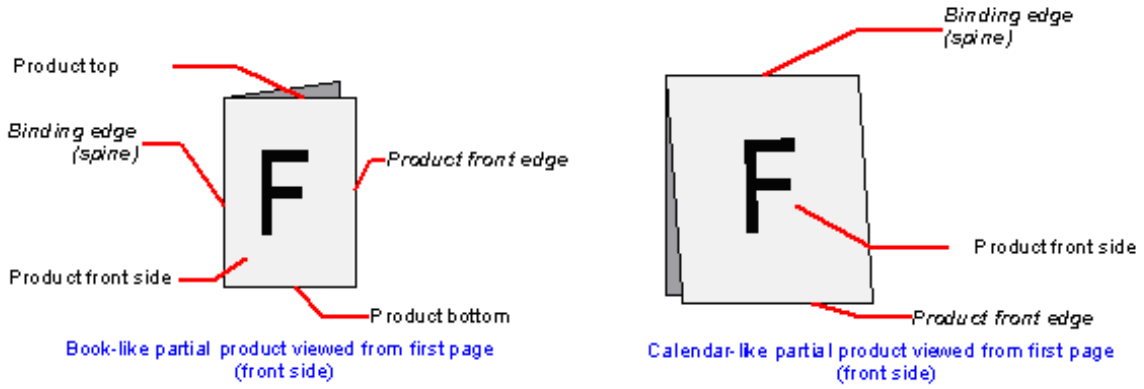
Component is used to describe the various versions of semi-finished goods in the press and postpress area, such as a pile of folded Sheets that have been collected and are then be joined and trimmed. Nearly every postpress process has a ***Component*** resource as an input as well as an output. Typically the first components in the process chain are some printed sheets or ribbons, while the last ***Component*** is a book or a brochure.

Glossary – Component

The descriptions of ***Component*** specific attributes use some terms whose meaning depends on the culture in which they are used. For example, different cultures mean different things when they refer to the "front" side of a magazine. Other terms (e.g., binding) are defined by the production process and, therefore, do not depend on the culture.

Whenever possible, this specification endeavors to use culturally independent terms. In cases where this is not possible, Western style (left-to-right writing) is assumed. Please note that these terms might have a different meaning in other cultures (i.e., those writing from right to left).

Figure 8-19: Component – terms and definitions



The table below describes the terms used to define the components.

Table 8.43: Glossary – Component

TERM	DEFINITION
Binding edge	The edge on which the (partial) product is glued or stitched. This edge is also often called *working edge* or *spine*.
Product front edge	The side, where you open the (partial) product. This edge is opposite to the binding edge.
Registered edge	A side on which a collection of Sheets or partial products is aligned during a production step. All production steps require two registered edges, which SHALL NOT be opposite to each other. The two registered edges define the coordinate system used within the production step. When there is a binding edge, this is one of the registered edges.

Resource Properties

Resource Class: Quantity

Resource referenced by: ***Bundle***/***BundleItem***, ***DigitalPrintingParams***, ***FeedingParams***/***Feeder***, ***FeedingParams***/***CollatingItem***

Example Partition: "Condition", "RibbonName", "SheetName", "SignatureName", "WebName"

Input of Processes: Any Product Intent node (▸ Section 7.0.1 Product Intent Descriptions), **ConventionalPrinting**, **DigitalPrinting**, **Varnishing**, **BlockPreparation**, **BoxFolding**, **BoxPacking**, **Bundling**, **CaseMaking**, **CasingIn**, **ChannelBinding**, **CoilBinding**, ***Collecting***, **CoverApplication**, **Creasing**, **Cutting**, **Embossing**, **EndSheetGluing**, **Feeding**, **Folding**, **Gathering**, **Gluing**, **HeadBandApplication**, **HoleMaking**, **Inserting**, **Jacketing**, **Labeling**, **Laminating**, **Numbering**, **Palletizing**, **Perforating**, **PlasticCombBinding**, **PrintRolling**, **RingBinding**, **ShapeCutting**, **Shrinking**, **SpinePreparation**, **SpineTaping**, **Stacking**, **StaticBlocking**, **Stitching**, **Strapping**, **StripBinding**, **ThreadSealing**, **ThreadSewing**, ***Trimming***, **WebInlineFinishing**, **Winding**, **WireCombBinding**, **Wrapping**

Output of Processes: Any Product Intent node (▸ Section 7.0.1 Product Intent Descriptions), **ConventionalPrinting**, **DigitalPrinting**, **Varnishing**, **BlockPreparation**, **BoxFolding**, **BoxPacking**, **Bundling**, **CaseMak-**

ing, CasingIn, ChannelBinding, CoilBinding, *Collecting*, CoverApplication, Creasing, Cutting, Embossing, EndSheetGluing, Feeding, Folding, Gathering, Gluing, HeadBandApplication, HoleMaking, Inserting, Jacketing, Labeling, Laminating, Numbering, Palletizing, Perforating, PlasticCombBinding, PrintRolling, RingBinding, ShapeCutting, Shrinking, SpinePreparation, SpineTaping, Stacking, StaticBlocking, Stitching, Strapping, StripBinding, ThreadSealing, ThreadSewing, *Trimming*, WebInlineFinishing, Winding, WireCombBinding, Wrapping

Table 8.44: Component Resource (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
AssemblyIDs ? New in JDF 1.3	NMTOKENS	@*AssemblyIDs* of the ***Assembly***, ***AssemblySection*** or ***StrippingParams*** (@*BinderySignatureName*) which this ***Component*** carries.
Automation ? New in JDF 1.5	enumeration	Identifies dynamic and static components. When a ***Component*** is referenced from a binding process, @*Automation* modifies the scope of the ***Component*** that SHALL be bound. If @*Automation*="Static", the individual ***Component*** elements that SHALL be bound are one instance of the referenced ***Component***. If @*Automation* ="Dynamic", the individual ***Component*** elements that SHALL be bound are identified by ***Component*** of the referenced partition. This may either be marked by the availability of all child partitions of the referenced partition or by the number of surfaces matching the value of @*SurfaceCount* specified in the ***IdentificationField*** or Pipe **JMF** Messages, respectively. The structure of @*PartIDKey* generation for automated imposition is defined in detail in: ▸ Section 6.3.18.3 Execution Model for Automated Imposition. This structure SHALL be retained in the ***Component*** description. **Allowed value is from:** ▸ Automation. **Note:** If @*Automation*="Dynamic" and @*PipeID* is also present, details are specified in **JMF** pipe messages. See ▸ Section 4.3.3.1 Dynamic Pipes. If an ***IdentificationField***/***MetadataMap*** element is present, the details are controlled by the barcode that is represented by ***IdentificationField***/***MetadataMap***.
CartonTopFlaps ? New in JDF 1.3	XYPair	Size (F1,F2) (See ▸ Figure 8-12: Box packing) of the two top flaps of a carton for shipping. SHALL NOT be specified unless @*ProductType* = "Carton".
Columns ? New in JDF 1.5	integer	Number of columns of images that are placed on a finished roll, such as by the **Winding** process. This value is typically used to describe rolls with multiple columns of printed labels.
ComponentType Modified in JDF 1.3	enumerations	Specifies the category of the component. **Allowed values are:** Block – Folded or stacked product (e.g., book block). Other – The ***Component*** describes a sample that has not been produced in this Job. Examples are perfume samples, CDs or toys that are inserted into a printed product. New in JDF 1.3 Ribbon – The ***Component*** is a ribbon on a Web Press. Sheet – Single layer (Sheet) of paper. Web – The ***Component*** is a web on a Web Press. FinalProduct – The ***Component*** is the final product that was ordered by the customer. PartialProduct – The ***Component*** is an intermediate product that will be input to a following process. Proof – The ***Component*** is a proof (e.g., a press proof or output from a digital press). Note that in **JDF** 1.2, proof was defined in the 1st list of categories, above. Modified in JDF 1.3 **Constraint:** Further details of the component are specified in @*ProductType*. At most one of "FinalProduct", "PartialProduct" or "Proof" SHALL be specified in addition to one of the first five enumerations specified as values.

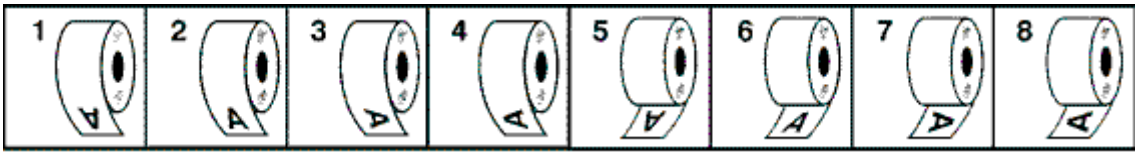
Table 8.44: Component Resource (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
Dimensions ?	shape	The dimensions of the component. These dimensions MAY differ from the original size of the original product. For example, the dimensions of a folded Sheet MAY be unequal to the dimensions of the Sheet before it was folded. The dimension is always the bounding box around the ***Component***. If not specified, a portrait orientation (Y > X) is assumed **Note:** It is crucial for enabling postpress to specify @*Dimensions* unless they really are unknown.
IsWaste = "false" Deprecated in JDF 1.4	boolean	If "true", the ***Component*** is waste from a previous process that can be used to set up a Machine. **Deprecation note:** Starting with **JDF** 1.4, use Partitioning with @*Condition* instead of @*IsWaste*.
MaxHeat ?	double	Maximum temperature the ***Component*** can resist (in degrees centigrade). The default setting SHALL impose no restriction in terms of heat (e.g., fusers in electrophotographic process or shrink wrapping).
Overfold ? New in JDF 1.1	double	Expansion of the overfold of a ***Component***. This attribute is needed for the **Inserting** or other postpress processes.
OverfoldSide ? New in JDF 1.1	enumeration	Specifies the longer side of a folded component. **Allowed value is from:** ▸ Side
PageListIndex ? New in JDF 1.3	IntegerRangeList	List of the indices of the ***PageData*** elements of the ***PageList*** specified in this ***Component***.
ProductType ? Modified in JDF 1.5	NMTOKEN	Type of product that this component specifies. **Values include those from:** ▸ Table 8.45 ProductType Attribute Values**.**
ProductTypeDetails ? New in JDF 1.3	string	Additional details of the product: If @*ProductType* = "BlankBox" or @*ProductType* = "FlatBox", @*ProductTypeDetails* specifies a box type (e.g., ▸ [ECMA], ▸ [FEFCO] or company internal box type standard). **Values include:** NewspaperNormal – Standard newspaper. NewspaperMixed – multiple ***Component*** resources of a newspaper are produced in parallel. NewspaperCombi – ***Component*** resources are collected to one ***Component*** in an inline production chain after press.
ReaderPageCount ? New in JDF 1.1	integer	Total amount of individual Reader Pages that this ***Component*** contains. Count of -1 means "unknown." If not specified, the value is unknown.
SheetPart ?	rectangle	Only used if contains (@*ComponentType*, "Block") and ***Layout*** is present. Position of the block on the ***Layout*** in @*SurfaceContentsBox* coordinates used in this ***Component***.
SourceRibbon ? Deprecated in JDF 1.3	string	SHALL NOT be specified unless contains (@*ComponentType*, "Ribbon"). @*RibbonName* of the ribbon used in this ***Component***. **Deprecation note:** Starting with **JDF** 1.3, use a direct reference to the ***Layout*** Partition that represents the ribbon.
SourceSheet ? Deprecated in JDF 1.3	string	SHALL NOT be specified unless contains (@*ComponentType*, "Sheet") or contains (@*ComponentType*, "Block"). Matches the ***Layout***/***Signature***/***Sheet***/@*Name* used in this ***Component***. **Deprecation note:** Starting with **JDF** 1.3, use a direct reference to the ***Layout*** partition that represents the Sheet.
SourceWeb ? Deprecated in JDF 1.3	string	SHALL NOT be specified unless contains (@*ComponentType*, "Ribbon"). @*WebName* of the ribbon used in this ***Component***. **Deprecation note:** Starting with **JDF** 1.3, use a direct reference to the ***Layout*** Partition that represents the Web.

Table 8.44: Component Resource (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
SpineThickness ? New in JDF 1.4	double	Thickness
SurfaceCount ? New in JDF 1.1	integer	Total amount of individual surfaces that this ***Component*** contains. **Note:** A sheet always has two Surfaces regardless of the number of images or reader pages. In case of homogeneous ***Component*** elements, @*SurfaceCount* refers to surfaces with a size of ***Component***/@*Dimensions*
Transformation ? Deprecated in JDF 1.1	matrix	Matrix describing the transformation of the orientation of a ***Component*** for the process using this resource as input. This is needed to convert the coordinate system of the ***Component*** to the coordinate system of the process. When this attribute is not present, the identity matrix (1 0 0 1 0 0) is assumed. In version 1.1 and beyond, use ***ResourceLink***/@*Transformation* or ***ResourceLink***/@*Orientation*.
WindingResult ? New in JDF 1.5	integer	Orientation of the finished product on the roll. For an image, see ▸ Figure 8-20: Orientation of the Finished Product on the Roll. The integers in the figure correspond to values specified by the labeling trade association, refer to ▸ [FINAT]. **Note:** The orientation and number of windings in a ***Winding*** process are modified based on the value of @*WindingResult*.
Assembly ? New in JDF 1.3	refelement	Specifies the assembly of the ***Component***. In case of a newspaper or web press, the output ***Component*** MAY already be built up of several "booklets". @*AssemblyIDs* additionally specifies which ***AssemblySection*** elements of the ***Assembly*** belong to this ***Component***.
Bundle ? New in JDF 1.1	refelement	Description of a ***Bundle*** of ***Component*** resources if the ***Component*** represents multiple individual items. If no ***Bundle*** is present, the ***Component*** represents an individual item. Note that it is essential to keep a reference of the child ***Component*** resources that comprise a ***Component***, as this information is useful to postpress processes.
Disjointing ?	element	A stack of components can be processed using physical separators. This is useful in operations such as feeding.
Layout ? New in JDF 1.2	refelement	Specifies the original ***Layout*** of the source Sheet of the ***Component*** if it contains (@*ComponentType*, "Sheet") or contains (@*ComponentType*, "Block"). The original sheet is the ***Layout*** partition element where @*SourceSheet* matches the ***Layout***/@*SheetName* used in this ***Component***.
Media ? New in JDF 1.4	refelement	***Media*** for the component. The coordinate system of ***Media*** coincides with the coordinate system of the component.
PageList ? New in JDF 1.3	refelement	Specification of page metadata for pages described by this ***Component***.
Sheet ? Deprecated in JDF 1.2	refelement	The ***Sheet*** resource that describes the details of this ***Component*** if it contains (@*ComponentType*, "Sheet") or contains (@*ComponentType*, "Block"). Replaced with ***Layout*** in **JDF** 1.2 and beyond. The Sheet in the referenced ***Layout*** is accessed by matching @*SourceSheet* with ***Layout***/***Signature***/***Sheet***/@*Name*.

Figure 8-20: Orientation of the Finished Product on the Roll



8.25.1 ProductType Attribute Values

Table 8.45: ProductType Attribute Values

VALUE	DESCRIPTION
BackCover	
BlankBox New in JDF 1.3	Cut, Unfolded box, input for folder-gluer
BlankSheet New in JDF 1.4	A sheet with connected blanks after a die cutting
BlankWeb New in JDF 1.4	A web with connected blanks after a die cutting.
Body New in JDF 1.2	Generic content inside of a cover.
Book	
BookBlock	
BookCase	
Box	Convenience packaging that is not envisioned to be protection for shipping.
Brochure	
BusinessCard	
Carton	Protection packaging for shipping.
Cover	
EndSheet New in JDF 1.5	Endsheet for hard cover books.
FlatBox New in JDF 1.3	A folded and glued blank (not opened). Output from a box folder-gluer.
FlatWork New in JDF 1.5	Non-bound, non-folded Products or Products that only have packaging folds.
FrontCover	
Insert New in JDF 1.2	
Jacket	Hard cover case jacket.
Label	
Newspaper New in JDF 1.3	A newspaper-product
Notebook New in JDF 1.6	A book or block with a set of identical or similar pages, e.g. a writing tablet, where all page fronts have identical content, and all page backs have identical content.
Pallet New in JDF 1.3	Loaded pallet of boxes, cartons or ***Component*** resources
Poster	
Stack New in JDF 1.4	Stacked ***Component***s.
Unknown Deprecated in JDF 1.2	

8.26 Contact

Element describing a contact to a person or address.The @*ProductID* attribute SHALL be unique within the company.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***Abstract PhysicalResource***, ***ArtDeliveryIntent***, ***ArtDeliveryIntent/ArtDelivery***, ***DeliveryIntent***, ***DeliveryIntent/DropIntent***, ***ApprovalParams/ApprovalPerson***, ***ApprovalSuccess/ApprovalDetails***, ***ContentList /ContentData/ContentMetadata***, ***CustomerInfo***, ***DeliveryParams***, ***DeliveryParams/Drop***, ***DigitalDeliveryParams***

Table 8.46: Contact Resource

NAME	DATA TYPE	DESCRIPTION
ContactTypeDetails ? New in JDF 1.2	string	Details of the contact's role or roles. For instance, if contains (@*ContactTypes*, "Delivery") this could be a description for which delivery location this ***Contact*** is responsible.
ContactTypes Modified in JDF 1.4	NMTOKENS	Classification of the contact. **Values include:** Accounting – Address of where to send to the bill. Administrator – Person to contact for queries concerning the execution of the Job. Agency – The contact is an employee of an Agency. New in JDF 1.4 Approver – Person who approves this Job. New in JDF 1.2 ArtDelivery – Delivery contact for artwork of this Job. ArtReturn – Return delivery or pickup address for artwork of this Job. Author – New in JDF 1.4 Billing – Contact information that refers to a payment method (e.g., credit card). Customer – The end customer. Delivery – The delivery address for all products of this Job. DeliveryCharge – The ***Contact*** is charged for delivery of this Job. Designer – New in JDF 1.4 Editor – New in JDF 1.4 Illustrator – New in JDF 1.4 Owner – The owner of a resource. Photographer – New in JDF 1.4 Pickup – The pickup address for all products of this Job. Sender – The source address of the delivery. New in JDF 1.2 SenderAlias – The sender address that shall be printed on a delivery to an end customer. New in JDF 1.6 **Note:** This allows a contractor to hide any subcontractor. Supplier – Address of a supplier of needed goods. SurplusReturn – Return delivery or pickup address for surplus products of this Job. TelephoneSanitizer – New in JDF 1.4
UserID ? New in JDF 1.5	string	User ID of user, as specified when logging into the operating system or into the submitting application.
Address ?	element	Element describing the address.
ComChannel * Modified in JDF 1.2	element	Communication channels to of the contact. These elements define communication channels that MAY be assigned to multiple Persons, for instance the communication channel of a reception area.
Company ? New in JDF 1.1	refelement	Company that this ***Contact*** is associated with.
Person ?	element	Name of the contact person.

8.26.1 Company

Company defines the organization name and organizational units (ORG) of the organizational properties defined in ▸ [vCard]. @*ProductID* SHALL be globally unique across all companies.

Table 8.47: Company Element

NAME	DATA TYPE	DESCRIPTION
OrganizationName	string	Name of the organization or company (vCard: ORG:orgnam (e.g., ABC, Inc.)).
Contact * Deprecated in JDF 1.1	refelement	A contact of the company. In **JDF** 1.1 and beyond, contacts reference multiple companies.
OrganizationalUnit *	text element	Describes the organizational unit (vCard: ORG:orgunit. For example, if two elements are present: 1. "North American Division" and 2. "Marketing").

8.27 ContactCopyParams

New in JDF 1.1

Element describing the parameters of **ContactCopying**.

Resource Properties

Resource Class: Parameter

Input of Processes: **ContactCopying**

Table 8.48: ContactCopyParams Resource

NAME	DATA TYPE	DESCRIPTION
ContactScreen = "false"	boolean	If @*ContactScreen* = "true" then a halftone screen on film SHALL be used to produce halftones.
Cycle ?	integer	Number of exposure light units to be used. The amount depends on the subject to be exposed.
Diffusion ?	enumeration	The diffusion foil setting. **Allowed values are:** On Off
PolarityChange = "true"	boolean	If @*PolarityChange* = "true" then the copy SHALL change polarity with respect to the original image.
RepeatStep = "1 1"	XYPair	Number (as integers) of copies in each direction for a Step/Repeat camera.
Vacuum ?	double	Amount of vacuum pressure to be used, measured in bars.
ScreeningParams ?	refelement	Properties of the halftone screen on film. Ignored if @*ContactScreen* = "false".

8.28 ContentList

New in JDF 1.3

ContentList provides a list of ***ContentData*** elements. ***ContentData*** elements are independent of pages. Thus multiple ***ContentData*** elements MAY reside on one page and a ***ContentData*** element MAY span multiple pages.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***PublishingIntent***, ***LayoutElement***, ***PageList***,

Table 8.49: ContentList Resource

NAME	DATA TYPE	DESCRIPTION
ContentData + Modified in JDF 1.4	element	Details of the individual content element. A ***ContentData*** element is referred to by its ID (i.e., the value of ***ContentData***/@*ID*). **Modification note:** Before **JDF** 1.4, a ***ContentData*** element was referred to by its index in ***ContentList*** with the warning that ***ContentData*** elements not be removed or inserted in a position other than the end of the list.

8.28.1 ContentData

ContentData defines the additional metadata of individual elementsof a page. If the ***ContentList*** is Partitioned, the index refers to ***ContentData*** elements in the respective leaves of the Partitioned ***ContentList***. The index restarts at 0 with each Partitioned leaf.

Table 8.50: ContentData Element

NAME	DATA TYPE	DESCRIPTION
CatalogDetails ? Deprecated in JDF 1.6	string	Additional details of a resource in a catalog environment.
CatalogID ? Deprecated in JDF 1.6	string	Identification of the resource (e.g., in a catalog environment).
ContentRefs ? New in JDF 1.4	IDREFS	List of ***ContentData***/@*ID* values that specify the ***ContentData*** elements children of this ***ContentData*** element. For instance, a book may refer to individual chapters. The reference ***ContentData*** object SHALL reside in the same ***ContentList*** as this ***ContentData***.
ContentType ?	NMTOKEN	Type of content. **Values include those from:** ▸ Table 8.51 ContentType Attribute Values**.**
HasBleeds ?	boolean	If "true", the file has bleeds.
ID ? New in JDF 1.4	ID	For reference by @*ContentRefs*.
IsBlank ?	boolean	If "true", the has no content marks and is blank.
IsTrapped ?	boolean	If "true", the file has been trapped.
JobID ?	string	ID of the Job that this ***ContentData*** belongs to.
ProductID ?	string	An ID of the ***ContentData*** as defined in the MIS system.
ContentMetadata ? New in JDF 1.4	element	Container for document related metadata such as ISBN, Author etc.
ElementColorParams ?	refelement	Color details of the ***ContentData*** element.
ImageCompressionParams ?	refelement	Specification of the image compression properties.
ScreeningParams ?	refelement	Specification of the screening properties of the .
SeparationSpec *	element	List of separation names defined in the ***ContentList***.

8.28.1.1 ContentType Attribute Values

*Table 8.51: **ContentType** Attribute Values*

VALUE	DESCRIPTION
Ad	The content represents a single ad
Article	The content represents a single article. Including headers, text bodies, photos etc.
Barcode	A barcode.
ClassifiedAd	Specifies a classified ad.
ClassifiedsPageElement	Specifies a grouping page element dealing with content of classified ads
Composed	Combination of elements that define an element that is not bound to a document page.
Editorial	Defines this element to contain editorial matter (e.g., text, photos etc.).
EditorialPageElement	Specifies a grouping page element dealing with content of the editorial department
Graphic	Line art.
IdentificationField	A general identification field excluding bar codes.
Image	Bitmap image.
Page	Representation of one document page.
PageHeader	For instance a newspaper title shown on the front page or on each single page. Usually, these headers contain information like page number, editorial desk and the date.
ROPAd	Specifies this element as an ROP ad. An ROP ad is an ad which is placed by the planner. Generally speaking, in a newspaper environment these include color ads, ads with placement requests in the editorial section and large ads.
Surface	Representation of an imposed surface.
Text	Formatted or unformatted text.

8.28.2 ContentMetadata

New in JDF 1.4

ContentMetadata is a container for metadata pertaining to this ***ContentData*** element. Additional metadata fields may be created using ***GeneralID***.

Table 8.52: ContentMetadata Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ISBN ? New in JDF 1.6	string	An International Standard Book Number, that allows for both 10 and 13 digit values. (see ▸ [ISO2108:2005]) **Note:** This replaces @*ISBN10* and @*ISBN13*
ISBN10 ? Deprecated in JDF 1.6	string	A 10 digit ISBN (see ▸ [ISO2108:2005]) **Deprecation Note:** This has been replaced by @*ISBN* which allows for both 10 and 13 digit values.
ISBN13 ? Deprecated in JDF 1.6	string	A 13 digit ISBN (see ▸ [ISO2108:2005]) **Deprecation Note:** This has been replaced by @*ISBN* which allows for both 10 and 13 digit values.
Title ?	string	The title of the content
Comment ?	element	If required, an abstract should be specified in ***Comment***[@*Name* = "Abstract"].
Contact *	refelement	The person who is responsible for this content.

Table 8.52: ContentMetadata Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Employee *	refelement	If required, the author should be specified in an ***Employee***[contains(@*Roles*, "Author")].
Part ?	element	If present, conserves the values of the specified Partition Keys related to the content being processed. It is illegal to set Partition Key values where that key is used to explicitly Partition the referencing resource, or is implied by that resource. **Note:** This allows partition keys and values to be conserved in a ***RunList*** (Surface) or ***Component***.

Example 8.11: ContentList

New in JDF 1.4

```
<RunList Class="Parameter" ID="r071030_02242378_000004"
    Status="Available">
    <PageListRef rRef="PageList"/>
</RunList>
<PageList Class="Parameter" ID="PageList" Status="Available">
    <ContentListRef rRef="ContentList"/>
</PageList>
<ContentList Class="Parameter" ID="ContentList" Status="Available">
    <ContentData>
        <ContentMetadata ISBN10="0123456789" Title="book thing">
            <Comment ID="c071030_022423109_000005" Name="Abstract">
                Abstract of the book in english
            </Comment>
            <Contact ContactTypes="Editor">
                <Person DescriptiveName="authorName" FamilyName="authorName"/>
            </Contact>
        </ContentMetadata>
    </ContentData>
    <ContentData>
        <ContentMetadata Title="Chapter 1">
            <Contact ContactTypes="Customer">
                <Person DescriptiveName="authorName1" FamilyName="authorName1"/>
            </Contact>
        </ContentMetadata>
    </ContentData>
    <ContentData>
        <ContentMetadata Title="Chapter 2">
            <Contact ContactTypes="Customer">
                <Person DescriptiveName="authorName2" FamilyName="authorName2"/>
            </Contact>
        </ContentMetadata>
    </ContentData>
    <ContentData>
        <ContentMetadata Title="Chapter 3">
            <Contact ContactTypes="Customer" >
                <Person DescriptiveName="authorName3" FamilyName="authorName3"/>
            </Contact>
        </ContentMetadata>
    </ContentData>
</ContentList>
```

Example 8.12: ContentList: Extended with ISBN, Author, etc.

New in JDF 1.4

Example of ***ContentList*** with ISBN, Author, etc.

```
<!-- Information about the input (file, author) -->
<RunList ID="NodeIDRunList" Status="Available" Class="Parameter" >
    <LayoutElementRef rRef="NodeIDLE" />
    <PageList>
        <ContentList>
            <ContentData>
                <!-- String for title -->
                <new:DocumentInfo Title="This is the title of the book"
                        ISBN="0123456789" xmlns:new="new_schema_URI">
                    <!-- Multi-lines string for Abstract -->
                    <new:DocumentAbstract>
                        This is the abstract of the book
                        It has several lines...
                    </new:DocumentAbstract>
                    <!-- List of authors. Using a PersonRef allows reusing the same
                        Person element -->
                    <new:Author Subject="Preface">
                        <PersonRef rRef="AuthorID1" />
                    </new:Author>
                    <new:Author Subject="Content">
                        <new:PersonRef rRef="AuthorID2" />
                        <new:PersonRef rRef="AuthorID3" />
                    </new:Author>
                </new:DocumentInfo>
            </ContentData>
        </ContentList>
    </PageList>
</RunList>
<LayoutElement ID="NodeIDLE" Status="Available" Class="Parameter" >
    <FileSpec URL="file:////hotfolder/files/Document2747.pdf"
        MimeType="application/pdf" UserFileName="JDF1.3.pdf" />
</LayoutElement>
<!-- Information about the authors -->
<Person ID="AuthorID1" Class="Parameter" Status="Available" FirstName="James"
    FamilyName="Smith" JobTitle="Author" />
<Person ID="AuthorID2" Class="Parameter" Status="Available" FirstName="John"
    FamilyName="Smith" JobTitle="Author" />
<Person ID="AuthorID3" Class="Parameter" Status="Available" FirstName="William"
    FamilyName="Smith" JobTitle="Author" />
<!-- Media: A3 white paper coated on both sides, 70 gr/m2 -->
<Media ID="MediaID" Class="Consumable" Status="Available" Weight="70"
    Dimension="1190 842" MediaType="Paper" MediaColorName="White"
    FrontCoatings="Coated" BackCoatings="Coated" />
<!-- Media: A4 yellow paper for Banner Page -->
<Media ID="MediaID2" Class="Consumable" Status="Available" Weight="70"
    Dimension="595 842" MediaType="Paper" MediaColorName="Yellow" />
<!-- Booklet layout + banner page with ISBN and Authors printed on it -->
<LayoutPreparationParams ID="NodeIDLPP" Class="Parameter" Status="Available"
    Sides="TwoSidedFlipY" NumberUp="2 1" BindingEdge="Left"
    PresentationDirection="FoldCatalog" FoldCatalog="F4-1"
    FinishingOrder="GatherFold" PageDistributionScheme="Saddle">
    <InsertSheet SheetType="JobSheet" SheetUsage="Header" IsWaste="true"
        SheetFormat="Standard" >
        <Layout>
            <MediaRef rRef="MediaID2" />
            <MarkObject CTM="1 0 0 1 0 0" >
                <JobField ShowList="new:ISBN new:Authors" />
            </MarkObject>
        </Layout>
    </InsertSheet>
</LayoutPreparationParams>
```

8.29 ConventionalPrintingParams

This resource defines the device specific setup of the **ConventionalPrinting** process.

Resource Properties

Resource Class: Parameter

Example Partition: "BlockName", "FountainNumber", "PartVersion", "RibbonName", "Separation", "SheetName", "Side", "SignatureName", "WebName", "WebProduct"

Input of Processes: **ConventionalPrinting**

Table 8.53: ConventionalPrintingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
DirectProof = "false"	boolean	If "true", the proof is directly produced and subsequently an approval might be given by a person (e.g., the customer, foreman or floor manager) shortly after the first final-quality printed Sheet is printed. The approval is needed for the actual print run, and not for setup. If the **ConventionalPrinting** process is waiting for approval of a direct proof, the @*Status* of the **JDF** node is switched to "Suspended" with the @*StatusDetails* = "WaitForApproval".
Drying ?	enumeration	The way in which ink is dried after a print run. **Allowed value is from:** ▸ Drying.
FirstSurface ? Modified in JDF 1.2	enumeration	Printing order of the surfaces on the Sheet. **Allowed values are:** Either – Deprecated in JDF 1.2 **Deprecation note:** Starting with **JDF** 1.2, omit @*FirstSurface* to specify "Either". Front Back
FountainSolution ?	enumeration	State of the fountain solution module in the printing units. **Allowed values are:** On Off
MediaLocation ?	string	Identifies the location of the ***Media***. The value identifies a physical location on the press (e.g., unwinder 1, unwinder 2 and unwinder 3). If the media resource is Partitioned by @*Location* (see also ▸ Section 3.10.6.4 Locations of PhysicalResources) there SHOULD be a match between one @*Location* Partition Key and this @*MediaLocation* value. **Values include those from:** ▸ Table C.10 Input Tray and Output Bin Names**.** **Note: The specified values are for printer locations.**
ModuleAvailableIndex ? New in JDF 1.1 Deprecated in JDF 1.4	IntegerRangeList	Zero-based list of print modules that are available for printing. In some cases modules are not available because the print module is replaced with in-line tooling (e.g., a perforating unit). If not specified, all modules are used for printing. The list is based on all modules of the printer and is not influenced by the value of @*ModuleIndex*. **Deprecation note:** Starting with **JDF** 1.4, the skipping of press modules is now handled by specifying ***ColorantControl***/***DeviceColorantOrder***/***SeparationSpec*** with no @*Name*
ModuleDrying ?	enumeration	The way in which ink is dried in individual modules. **Allowed value is from:** ▸ Drying.
ModuleIndex ?	IntegerRangeList	Zero-based, ordered list of print modules that are to be used. @*ModuleIndex* does not influence the ink sequence. It is used only to skip individual modules. The list is based on all modules of the printer and is not influenced by the value of @*ModuleAvailableIndex*. **Note:** Starting with **JDF** 1.4, the skipping of press modules SHOULD additionally be specified by supplying ***ColorantControl***/***DeviceColorantOrder***/***SeparationSpec*** with no @*Name*.
NonPrintableMarginBottom ? New in JDF 1.3	double	The width in points of the bottom margin measured inward from the edge of the ***Media*** with respect to the idealized process coordinate system of the **ConventionalPrinting** process. The ***Media*** origin is unaffected by @*NonPrintableMarginBottom*.

Table 8.53: ConventionalPrintingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
NonPrintableMarginLeft ? New in JDF 1.3	double	Same as @*NonPrintableMarginBottom* except for the left margin.
NonPrintableMarginRight ? New in JDF 1.3	double	Same as @*NonPrintableMarginBottom* except for the right margin.
NonPrintableMarginTop ? New in JDF 1.3	double	Same as @*NonPrintableMarginBottom* except for the top margin.
PerfectingModule ? New in JDF 1.1	integer	Index of the perfecting module if @*WorkStyle* = "Perfecting" and multiple perfecting modules are installed.
Powder ?	double	Quantity of powder in%.
PrintingTechnology ? New in JDF 1.4 Deprecated in JDF 1.5	enumeration	Printing technology of the press or press module. **Allowed values are:** Flexo Gravure Offset Screen **Deprecation note:** Starting with **JDF** 1.5, use ***Color***/@*PrintingTechnology*.
PrintingType Modified in JDF 1.3	enumeration	Type of printing Machine. **Allowed values are:** ContinuousFed – connected Sheets including fan fold. New in JDF 1.2 SheetFed – Separate cut Sheets. WebFed – Paper supplied to press on rolls. Deprecated in JDF 1.3 WebMultiple – Web Printing with multiple plates per cylinder. Generally used with Newspaper Web Printing. New in JDF 1.3 WebSingle – Web Printing with only one plate per cylinder. Generally used in commercial and publication workflows. New in JDF 1.3
SheetLay ?	enumeration	Lay of input media. Reference edge of where paper is placed in a feeder. Allowed value is from: ▸ SheetLay.
Speed ? Modified in JDF 1.3	double	Maximum print speed in Sheets/hour (Sheet fed) or revolutions/hour (WebFed). Defaults to device specific full speed.
WorkStyle ?	enumeration	The direction in which to turn the press sheet.**Allowed value is from:** ▸ WorkStyle.
ApprovalParams ? New in JDF 1.2	refelement	Details of the direct **Approval** process, when @*DirectProof* = "true".
Ink * Modified in JDF 1.2 Deprecated in JDF 1.4	refelement	Details of varnishing. Defines the varnish to be used for coatings on printed sides. Coatings are applied after printing all the colors. Other coating sequences SHALL use the Partition mechanism of this resource. Selective varnishing in print modules has to use a separate separation for the respective varnish. Varnish is specified by ***Ink***/@*Family* = "Varnish". If both ***Ink*** and ***ExposedMedia*** (Plate) are specified for a given separation, spot varnishing is specified. If only ***Ink*** and not ***ExposedMedia*** (Plate) is specified, overall varnishing is specified. In **JDF** 1.2 and beyond, ***Ink*** MAY occur in multiples in order to specify multiple layers of varnish. **Note:** The color inks are direct Input resources of the process and SHALL NOT be specified here. **Deprecation note:** Starting with **JDF** 1.4, use **Varnishing**.

8.30 CoverApplicationParams

New in JDF 1.1

CoverApplicationParams define the parameters for applying a cover to a book block.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **CoverApplication**

Table 8.54: CoverApplicationParams Resource

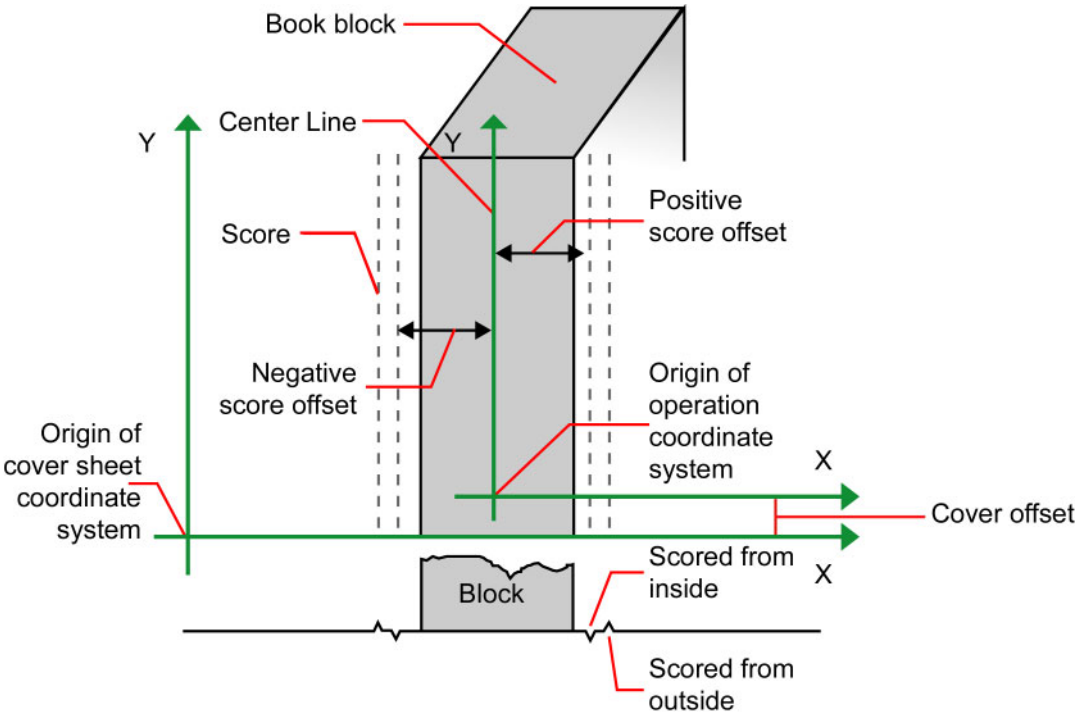
NAME	DATA TYPE	DESCRIPTION
CoverOffset ? Deprecated in JDF 1.2	XYPair	Position of the cover in relation to the book block given in the cover-Sheet coordinate system. In **JDF** 1.2 and beyond, @*CoverOffset* is implied by the transformation matrix of the ***ResourceLink***/@*Transformation* of the cover's ***ComponentLink***.
GlueApplication *	refelement	Describes where and how to apply glue to the book block.
Score *	element	Describes where and how to score the cover. The sequence of ***Score*** elements specifies the sequence in which the tool is applied.

8.30.1 Score

Table 8.55: Score Element

NAME	DATA TYPE	DESCRIPTION
Offset	double	Position of scoring given in the operation coordinate system.
Side = "FromInside"	enumeration	Specifies the side from which the scoring tool works. **Allowed values are:** FromInside FromOutside

Figure 8-21: Parameters and coordinate system for cover application



8.31 CreasingParams

New in JDF 1.1

CreasingParams define the parameters for creasing or grooving a Sheet.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***FoldingIntent***

Example Partition: "BlockName", "RibbonName", "SheetName", "SignatureName", "WebName"

Input of Processes: **Creasing**

Table 8.56: CreasingParams Resource

NAME	DATA TYPE	DESCRIPTION
Crease *	element	Defines one or more crease lines.

8.32 CustomerInfo

Modified in JDF 1.3

The ***CustomerInfo*** resource contains information about the customer who orders the Job. ***CustomerInfo*** has been moved from a direct element of **JDF** to a resource in **JDF** 1.3.

Before **JDF** 1.3, ***CustomerInfo*** was a Subelement of a **JDF** node, and "inherited" down to child nodes. Starting with **JDF** 1.3, ***CustomerInfo*** became a resource that SHALL be linked like any other resource; there is no "inheritance". Any node MAY link to the same ***CustomerInfo*** resource as its parent. A normative ***CustomerInfo*** is specified by a linked resource. An informative ***CustomerInfo*** MAY be retrieved by searching for ***CustomerInfo*** of parent nodes or ancestor elements

Creating Better Job Tracking & Reporting

Customer information within JDF can provide a bridge between your CRM systems and production. How could JDF be used to automate the process of reporting to customers on the status of their Jobs?

Resource Properties

Resource Class: Parameter

Resource referenced by: ***Ancestor***

Input of Processes: **Any Process**

Table 8.57: CustomerInfo Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BillingCode ?	string	A code to bill charges incurred while executing the node.
CustomerID ?	string	Customer identification used by the application that created the Job. This is usually the internal customer number of the MIS system that created the Job.
CustomerJobName ?	string	The name that the customer uses to refer to the Job.
CustomerOrderID ?	string	The internal order number in the system of the customer. This number is usually provided when the order is placed and then referenced on the order confirmation or the bill.
CustomerProjectID ? New in JDF 1.2	string	The internal project id in the system of the customer. This number MAY be provided when the order is placed and then referenced on the order confirmation or the bill.
rRefs ? Deprecated in JDF 1.2	IDREFS	Array of IDs of any elements that are specified as ***ResourceRef*** elements. In version 1.1 it was the IDREF of a ***ContactRef***. In **JDF** 1.2 and beyond, it is up to the implementation to maintain references.
Company ? Deprecated in JDF 1.1	refelement	Resource element describing the business or organization of the contact. In **JDF** 1.1 and beyond, ***Company*** affiliation of ***Contact*** elements is specified in ***Contact***.
Contact * New in JDF 1.1	refelement	Resource element describing contacts associated with the customer. There SHOULD be one ***Contact***[contains (@*ContactTypes*, "Customer")]. Such a ***Contact*** specifies the primary customer's name, address etc.

Table 8.57: CustomerInfo Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
CustomerMessage * New in JDF 1.2 Deprecated in JDF 1.5	element	Element that describes messages to the customer. **Deprecation note:** Use ***Comment*** instead.

8.32.1 CustomerMessage

New in JDF 1.2

Deprecated in JDF 1.5

See ▸ Section N.7.3 CustomerMessage for details of this deprecated Parameter element.

8.33 CutBlock

Defines a cut block on a Sheet. It is possible to define a block that contains a matrix of elements of equal size. In this scenario, the intermediate cut dimension is calculated from the information about element size, block size and the number of elements in both directions. Each cut block has its own coordinate system, which is defined by the *@BlockTrf* attribute.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***CuttingParams***

Table 8.58: CutBlock Resource

NAME	DATA TYPE	DESCRIPTION
AssemblyIDs ? New in JDF 1.3	NMTOKENS	The *@AssemblyIDs* of the ***Assembly***, ***AssemblySection*** or ***StrippingParams***[*@BinderySignatureName*] which are contained in this ***CutBlock***.
BlockElementSize ?	XYPair	Element dimension in X and Y direction. The default value is equivalent to the XYPair value in *@BlockSize*.
BlockElementType ?	enumeration	Element type. **Allowed values are:** CutElement – Cutting element. PunchElement – Punching element.
BlockName	NMTOKEN	Name of the block. Used for reference by the ***CutMark*** resource. Note that ***CutBlock*** resources are not Partitioned although they are nested. The semantics of nested ***CutBlock*** elements are different.
BlockSize	XYPair	Size of the block.
BlockSubdivision = "1 1"	XYPair	Number (as integers) of elements in X and Y direction.
BlockTrf = "1 0 0 1 0 0"	matrix	Block transformation matrix. Defines the position and orientation of the block relative to the ***Component*** coordinate system.
BlockType	enumeration	Block type. **Allowed values are:** CutBlock – Block to be cut. SaveBlock – Protected block, cut only via outer contour. TempBlock – Auxiliary block that is not taken into account during cutting. MarkBlock – Contains no elements, only marks.
CutWidth ? New in JDF 1.4	double	Width in points of u-shaped knife, saw blade, etc.
Assembly ? New in JDF 1.3	refelement	***Assembly*** that is referred to by *@AssemblyIDs* or contains the ***AssemblySection*** that is referred to by *@AssemblyIDs*.

8.34 CutMark

This resource, along with ***CutBlock***, provides the means to position cut marks on the Sheet. After printing, these marks can be used to adapt the theoretical block positions (as specified in ***CutBlock***) to the real position of the corresponding blocks on the printed Sheet.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***Layout***/***MarkObject***

Table 8.59: CutMark Resource

NAME	DATA TYPE	DESCRIPTION
Blocks ? Modified in JDF 1.1	NMTOKENS	Values of the @*BlockName* Partition attributes of the blocks defined by the ***CutMark*** resource.
MarkType	enumeration	Cut mark type. **Allowed values are:** CrossCutMark TopVerticalCutMark BottomVerticalCutMark LeftHorizontalCutMark RightHorizontalCutMark LowerLeftCutMark UpperLeftCutMark LowerRightCutMark UpperRightCutMark
Position	XYPair	Position of the logical center of the cut mark in the coordinates of the ***MarkObject*** that contains this mark. Note that the logical center of the cut mark does not always directly specify the center of the visible cut mark symbol.

Table 8.60: Cut mark types as specified by CutMark/@MarkType (Sheet 1 of 2)

SYMBOL	MARKTYPE VALUE	POSITION OF SYMBOL
	"CrossCut-Mark"	Centered at logical position
	"TopVertical-CutMark"	Slightly above logical position
	"BottomVerti-calCutMark"	Slightly below logical position
	"LeftHorizon-talCutMark"	Slightly to the left of logical position
	"RighHorizon-talCutMark"	Slightly to the right of logical position
	"LowerLeft-CutMark"	Corner at logical position

Table 8.60: Cut mark types as specified by CutMark/@MarkType (Sheet 2 of 2)

SYMBOL	MARKTYPE VALUE	POSITION OF SYMBOL
	"UpperLeft-CutMark"	Corner at logical position
	"LowerRight-CutMark"	Corner at logical position
	"UpperRight-CutMark"	Corner at logical position

8.35 CuttingParams

New in JDF 1.1

This resource describes the parameters of a **Cutting** process that uses nested ***CutBlock*** elements as input.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***FoldingIntent***, ***ShapeCuttingIntent***

Example Partition: "BlockName", "RibbonName", "SheetName", "SignatureName", "WebName"

Input of Processes: **Cutting**

Table 8.61: CuttingParams Resource

NAME	DATA TYPE	DESCRIPTION
NUpSeparation ? New in JDF 1.4	XYPair	Defines the number of ***CutBlock*** elements in x and y direction. For example, a 2-up book sawed apart would have @*NUpSeparation* = "2 1".
SheetLay ? New in JDF 1.5	enumeration	Lay of input ***Component***. **Allowed value is from:** ▸ SheetLay. **Note:** @*SheetLay* does not modify the coordinate references of the **Cutting** process.
Cut *	element	***Cut*** elements describe an individual cut. ***Cut*** elements SHALL NOT be specified if ***CutBlock*** elements are specified.
CutBlock *	refelement	One or several ***CutBlock*** elements can be used to find the **Cutting** sequence.The ***CutBlock*** elements SHALL NOT be written if ***Cut*** elements are specified.
CutMark * Deprecated in JDF 1.3	refelement	***CutMark*** resources can be used to adapt the theoretical cut positions to the real positions of the corresponding blocks on the ***Component*** to be cut. Replaced by ***Component***/***Layout*** in **JDF** 1.3 and above.
FileSpec (CIP3) ? New in JDF 1.5	refelement	Reference to a CIP3 file that contains cutting instructions in the ▸ [CIP3 - PPF] format.

8.36 CylinderLayout

New in JDF 1.3

Describes the mapping of plates to cylinders on a newspaper-Web Press. This information might be important for pre-press systems. For instance, if a system wants to indicate the cylinder position as human readable text onto the plate.

Resource Properties

Resource Class: Parameter

Example Partition: "PlateLayout", "Separation", "WebProduct"

Output of Processes: *CylinderLayoutPreparation*

Table 8.62: CylinderLayout Resource

NAME	DATA TYPE	DESCRIPTION
DeviceID ?	NMTOKEN	Specifies the ***Device*** that this ***CylinderLayout*** belongs to.
CylinderPosition +	element	Specifies the position of a plate on a cylinder of a newspaper-Web Press.
Layout ?	refelement	References the ***Layout*** that describes the plates to be mounted.

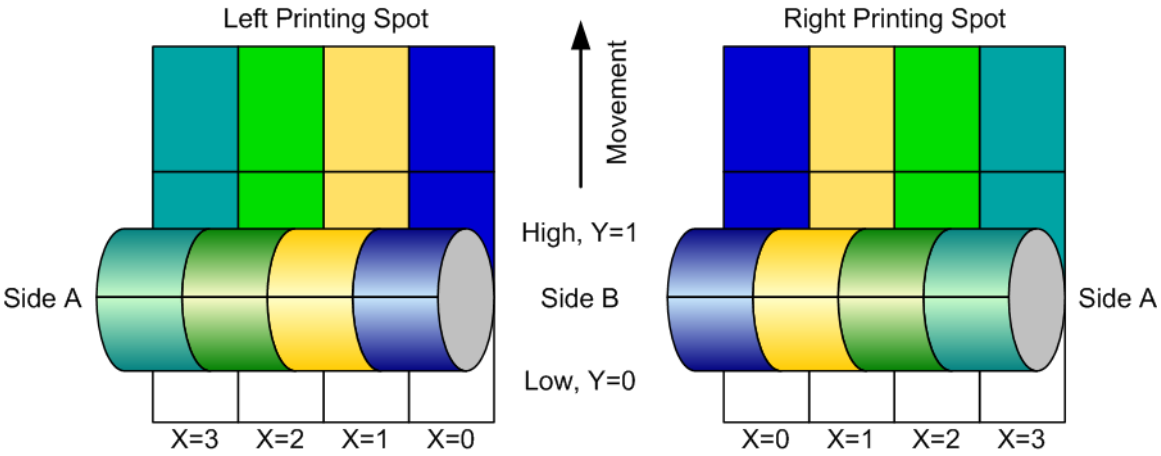
8.36.1 CylinderPosition

Table 8.63: CylinderPosition Element

NAME	DATA TYPE	DESCRIPTION
DeviceModuleIndex	integer	Defines a ***Module*** with *@ModuleType* = "PrintModule" within the ***Device*** specified by ***CylinderLayout***/*@DeviceID*. In a newspaper-Web Press, "PrintModule" corresponds to a single cylinder.
PlatePosition	XYPair-RangeList	Specifies where to mount this plate onto the cylinder. See figure below for details.
PlateType = "Exposed"	enumeration	Specifies whether the plate contains content data or represents a dummy plate. Additionally, it indicates where in the workflow it will be produced. **Allowed values are:** Dummy – Indicates that the plate is a dummy plate. It SHALL be bent by a **Bending** process. But it is unlikely to be exposed by an **ImageSetting** process. Exposed – Indicates that the plate contains content data and SHALL be exposed by an **ImageSetting** process.
PlateUsage = "Original"	enumeration	Specifies, whether a plate has to be produced for a specific Web run or not. **Allowed values are:** Original – indicates that the plate SHALL be produced specifically for this run. Reuse – indicates that a plate of a previous run will be re-used (same plate position on the Web Press). For instance, a dummy on a specific ***CylinderPosition*** can be used in multiple Web runs.

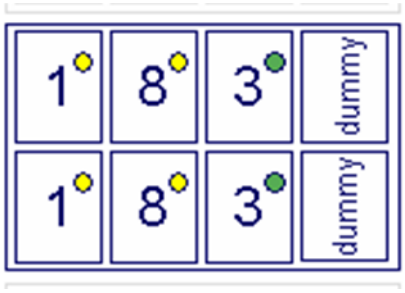
In ▸ Figure 8-22: Definition of the PlatePosition Attribute on a newspaper-Web Press, the direction of the view is from the plate cylinder towards the paper. If this direction is vectored as the direction of the former module, this is a left-printing spot. Otherwise it is a right-printing spot. If a 'left-printing spot' is considered, 'Side A' is to the left and 'Side B' to the right. And vice versa for a 'right-printing spot'. The plate position in X-dimension starts numbering at Side B. Thus, for the innermost Side B position X = "0". For the outmost Side A position X = "1" for single-width presses. On double-width presses X = "3" for the outmost Side A position. On triple-width presses X = "5" for the outmost Side A position.
Note: The "Back" and "Front" side have the same X position on corresponding segments of a Web.

Figure 8-22: Definition of the PlatePosition Attribute on a newspaper-Web Press



The sketch in ▸ Figure 8-23: Example of a single physical section of eight pages shows a single cylinder of a newspaper-Web Press for a broadSheet production. The numbers indicate Reader Page numbers. The colored dots indicate color separations. Dummy means no content-bearing plates are mounted on this cylinder position. Instead, so called dummy forms are mounted.

Figure 8-23: Example of a single physical section of eight pages



Example 8.13: CylinderLayout

The following ***CylinderLayout*** is an example representation of the cylinder layout as shown in the sketch.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Bundle" Status="Waiting"
       Type="CylinderLayoutPreparation" JobPartID="ID20" Version="1.4">
    <ResourcePool>
       <CylinderLayoutPreparationParams ID="CL002" Class="Parameter"
              Status="Available" >
          <ProductionPath/>
       </CylinderLayoutPreparationParams>
       <RunList ID="R-002" Class="Parameter" Status="Available" />
       <Device ID="DEV-001" Manufacturer="MAN" ModelName="GEOMAN" Status="Available"
              Class="Implementation" DeviceID="DEV-001">
          <Module ModuleIndex="0" ModuleType="Folder" ModelName="Folder 1">
             <Module ModuleIndex="1" ModuleType="PrintUnit"
                    DescriptiveName="PU-1">
                <Module ModuleIndex="2" SubModuleIndex="0"
                       ModuleType="PrintModule" DescriptiveName="PM-1"/>
                <Module ModuleIndex="3" SubModuleIndex="1"
                       ModuleType="PrintModule" DescriptiveName="PM-2"/>
                <Module ModuleIndex="4" SubModuleIndex="2"
                       ModuleType="PrintModule" DescriptiveName="PM-3"/>
                <Module ModuleIndex="5" SubModuleIndex="3"
                       ModuleType="PrintModule" DescriptiveName="PM-4"/>
                <Module ModuleIndex="6" SubModuleIndex="4"
                       ModuleType="PrintModule" DescriptiveName="PM-5"/>
                <Module ModuleIndex="7" SubModuleIndex="5"
                       ModuleType="PrintModule" DescriptiveName="PM-6"/>
                <Module ModuleIndex="8" SubModuleIndex="6"
                       ModuleType="PrintModule" DescriptiveName="PM-7"/>
                <Module ModuleIndex="9" SubModuleIndex="7"
                       ModuleType="PrintModule" DescriptiveName="PM-8"/>
             </Module>
          </Module>
       </Device>
```

```
        <Layout ID="L-001" Class="Parameter" Status="Available"/>
        <CylinderLayout ID="CL-001" Class="Parameter" Status="Available"
               PartIDKeys="WebSetup PlateLayout Separation"
               DeviceID="DEV-001">
            <LayoutRef rRef="L-001"/>
            <CylinderLayout WebSetup="Run-1">
                <CylinderLayout PlateLayout="PL-001">
                    <CylinderLayout Separation="Yellow">
                        <CylinderPosition DeviceModuleIndex="2" PlatePosition="0 0"
                               PlateType="Exposed" PlateUsage="Original"/>
                        <!-- page 1 -->
                        <CylinderPosition DeviceModuleIndex="2" PlatePosition="0 1"
                               PlateType="Exposed" PlateUsage="Original"/>
                        <!-- page 1 -->
                    </CylinderLayout>
                </CylinderLayout>
                <CylinderLayout PlateLayout="PL-002">
                    <CylinderLayout Separation="Yellow">
                        <CylinderPosition DeviceModuleIndex="2" PlatePosition="1 0"
                               PlateType="Exposed" PlateUsage="Original"/>
                        <!-- page 8 -->
                        <CylinderPosition DeviceModuleIndex="2" PlatePosition="1 1"
                               PlateType="Exposed" PlateUsage="Original"/>
                        <!-- page 8 -->
                    </CylinderLayout>
                </CylinderLayout>
                <CylinderLayout PlateLayout="PL-003">
                    <CylinderLayout Separation="HKS57">
                        <CylinderPosition DeviceModuleIndex="2" PlatePosition="2 0"
                               PlateType="Exposed" PlateUsage="Reuse"/>
                        <!-- page 3 -->
                        <CylinderPosition DeviceModuleIndex="2" PlatePosition="2 1"
                               PlateType="Exposed" PlateUsage="Reuse"/>
                        <!-- page 3 -->
                    </CylinderLayout>
                </CylinderLayout>
                <CylinderPosition DeviceModuleIndex="2" PlatePosition="3 0"
                       PlateType="Dummy" PlateUsage="Reuse"/>
                <CylinderPosition DeviceModuleIndex="2" PlatePosition="3 1"
                       PlateType="Dummy" PlateUsage="Reuse"/>
            </CylinderLayout>
        </CylinderLayout>
    </ResourcePool>
    <ResourceLinkPool>
        <DeviceLink  Usage="Input" rRef="DEV-001"/>
        <LayoutLink  Usage="Input" rRef="L-001"/>
        <RunListLink Usage="Input" rRef="R-002"/>
        <CylinderLayoutPreparationParamsLink Usage="Input" rRef="CL002"/>
        <CylinderLayoutLink Usage="Output" rRef="CL-001"/>
    </ResourceLinkPool>
</JDF>
```

Example 8.14: CylinderLayout: Double-Spread-Page Plate

In case of a double-spread-page plate (or double-truck-page plate) the ***CylinderPosition*** MAY be set as:

```
<CylinderLayout ID="CL-001" Class="Parameter" Status="Available"
       PartIDKeys="WebSetup PlateLayout Separation"
       DeviceID="DEV-001">
    <!-- ... -->
    <!-- PlatePosition (XYPairRangeList)-->
    <CylinderPosition DeviceModuleIndex="2" PlatePosition="0 0 ~ 1 0"
           PlateType="Exposed" PlateUsage="Original"/>
    <!-- ... -->
</CylinderLayout>
```

8.37 CylinderLayoutPreparationParams

New in JDF 1.3

This resource specifies the parameters of the **CylinderLayoutPreparation** process.

Resource Properties

Resource Class: Parameter

Example Partition: "WebName", "WebProduct"

Input of Processes: **CylinderLayoutPreparation**

Table 8.64: CylinderLayoutPreparationParams Resource

NAME	DATA TYPE	DESCRIPTION
ProductionPath	refelement	***ProductionPath*** describes the individual paper path through the different modules of a Web Press.

8.38 DBMergeParams

Deprecated in JDF 1.5

See ▸ Section N.7.4 DBMergeParams for details of this deprecated element.

8.39 DBRules

Deprecated in JDF 1.5

See ▸ Section N.7.5 DBRules for details of this deprecated element.

8.40 DBSchema

Deprecated in JDF 1.5

See ▸ Section N.7.6 DBSchema for details of this deprecated element.

8.41 DBSelection

Deprecated in JDF 1.5

See ▸ Section N.7.7 DBSelection for details of this deprecated element.

8.42 DeliveryParams

DeliveryParams provides information needed by a **Delivery** process. A **Delivery** process consists of sending one or more products to one or more delivery destinations. Delivery is also used to specify the scheduled transfer of digital assets.

In order to instruct a digital delivery device to compress or encode the files one can use the input and output ***RunList*** with ***FileSpec***/@*Compression* attribute, even if no URL is specified. See ▸ Section O.7 DigitalDelivery Examples for a set of examples.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***ArtDeliveryIntent***, ***DeliveryIntent***

Input of Processes: **Delivery**

Table 8.65: DeliveryParams Resource (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
DeliveryID ?	string	Deliveries with the same @*DeliveryID* are part of the same physical delivery. This attribute allows items from multiple individual **JDF** jobs to be delivered in one delivery.
Earliest ?	dateTime	@*Earliest* SHALL specify the earliest date and time after which the delivery is intended to be made.
EarliestDuration ?	duration	@*EarliestDuration* SHALL specify the earliest duration by which the delivery SHALL be made relative to the date and time that the order is ready for production.

Table 8.65: DeliveryParams Resource (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
Method ? Modified in JDF 1.5	string	Specifies a delivery method or Brand (e.g., "ExpressMail" or "InterofficeMail"). Note that it is strongly RECOMMENDED to use an NMTOKEN compatible string without blank spaces in this attribute. **Values include** BestWay – The sender decides how to deliver. CompanyTruck Courier CourierNoSignature – a delivery service that does not require receipt stamps at recipient's mailbox and/or mail room. This new value covers the commonly used Japanese 'Mail bin' delivery service.New in JDF 1.5 Email ExpressMail InstantMessaging New in JDF 1.5 InterofficeMail Local – The files are already in place and a **DigitalDelivery** process is not needed. New in JDF 1.5 NetworkCopy – This includes LAN and VPN. New in JDF 1.5 Storage – The product is stored by the supplier. OvernightService Deprecated in JDF 1.5 WebServer – Upload / download from HTTP / FTP server. New in JDF 1.5 **Modification note:** Starting in **JDF** 1.5, values have changed.
Pickup ? Deprecated in JDF 1.2	boolean	If "true", the merchandise is picked up. If "false", the merchandise is delivered. Replaced with @*Transfer* in **JDF** 1.2.
Required ?	dateTime	@*Required* SHALL specify the date and time by which the delivery is intended to be made.
RequiredDuration ?	duration	@*RequiredDuration* SHALL specify the time duration by which the delivery SHALL be made relative to the date and time that the order is ready for production.
ServiceLevel ? New in JDF 1.2	string	The service level of the specific carrier. **Values include:** Next Day 2nd Day Air Ground.
Transfer ? New in JDF 1.2	EnumerationSpan	Describes the direction and responsibility of the transfer. **Note:** If these values are for ***DeliveryIntent***/@*Transfer* or ***DropIntent***/@***Transfer***, then treat each occurrence of ***DeliveryParams*** below as ***DeliveryIntent***. **Allowed values are:** BuyerToPrinterDeliver – The ***DeliveryParams*** describes an input to the job (e.g., a CD for inserting, a preprinted cover, etc.). In this case, the buyer delivers the merchandise to the printer. The printer SHALL specify in the quote a special ***Contact***[contains(./***Part***/@*ContactTypes* = "Delivery")]. The ***Contact*** specifies where the buyer SHALL send the merchandise. BuyerToPrinterPickup – The ***DeliveryParams*** describes an input to the job (e.g., a CD for inserting, a preprinted cover, etc.). In this case, the printer picks up the merchandise. ***Contact***[contains(./***Part***/@*ContactTypes* = "Pickup")] specifies where the printer has to pick up the merchandise. PrinterToBuyerDeliver – The ***DeliveryParams*** describes an output of the job. In this case, the printer delivers the merchandise to the buyer. The ***Contact***[contains(./***Part***/@*ContactTypes* = "Delivery")] specifies where the printer SHALL send the merchandise. PrinterToBuyerPickup – The ***DeliveryParams*** describes an output of the job. In this case, the buyer picks up the merchandise. The printer SHALL specify in the quote a special ***Contact***[contains(./***Part***/@*ContactTypes* = "Pickup")]. The ***Contact*** specifies where the buyer SHALL pick up the merchandise.
Company ? Deprecated in JDF 1.1	refelement	Address and further information of the addressee. In **JDF** 1.1 and beyond, use ***Contact***/***Company***

Table 8.65: DeliveryParams Resource (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
Contact * New in JDF 1.1	refelement	Address and further information of the ***Contact*** responsible for this delivery.
Drop +	element	All locations where the product will be delivered.
FileSpec (DeliveryContents) ? New in JDF 1.6	element	A ***FileSpec*** resource that references a document that SHALL be printed and packaged together with the delivered items.
FileSpec (MailingList) ? New in JDF 1.5	refelement	A ***FileSpec*** resource pointing to a mailing list. The format of the referenced mailing list is implementation dependent.

8.42.1 Drop

Table 8.66: Drop Element

NAME	DATA TYPE	DESCRIPTION
DropID ? New in JDF 1.5	string	***Drop*** elements with the same @*DropID* are part of the same drop. This attribute is provided to allow items from multiple individual **JDF** jobs to be delivered in one drop.
Earliest ?	dateTime	Specified the earliest time after which the delivery SHALL be made.
Method ? Modified in JDF 1.5	string	Specifies a delivery method (e.g., "ExpressMail" or "InterofficeMail"). Note that it is strongly RECOMMENDED to use an NMTOKEN compatible string in this attribute, without blanks. **Values include those from:** ***DeliveryParams***/@*Method* **Modification note:** Starting in **JDF** 1.5, values have changed.
Pickup ? Deprecated in JDF 1.2	boolean	If "true", the merchandise is picked up. If "false", the merchandise is delivered. Default = ***DeliveryParams***/@*Pickup*. Replaced with @*Transfer* in **JDF** 1.2.
Required ?	dateTime	Specifies the time by which the delivery is intended to be made. **Default value is from:** ***DeliveryParams***/@*Required*
ServiceLevel ? New in JDF 1.2	string	The service level of the specific carrier. **Values include those from:** ***DeliveryParams***/@*ServiceLevel*.
TrackingID ? New in JDF 1.2	string	The string that can help in tracking the delivery. The value of the @*TrackingID* attribute will depend on the carrier chosen to ship the products.
Transfer ? New in JDF 1.2	enumeration	Describes the direction and responsibility of the transfer. **Allowed values are from:** ***DeliveryParams***/@*Transfer*.
Company ? Deprecated in JDF 1.1	refelement	Address and further information of the addressee. Defaults to the value of ***Company*** specified in the root ***DeliveryParams*** resource.
Contact * New in JDF 1.1	refelement	Address and further information of the contacts responsible for this delivery. Default = ***DeliveryParams***/***Contact***.
DropItem +	element	A ***Drop*** MAY consist of multiple products, which are represented by their respective ***PhysicalResource*** elements. Each ***DropItem*** describes an individual resource that is part of this ***Drop***.
FileSpec (DeliveryContents) ? New in JDF 1.6	element	Reference to a document that SHALL be printed and packaged together with the delivered items.

8.42.2 DropItem

Table 8.67: DropItem Element

NAME	DATA TYPE	DESCRIPTION
ActualAmount ? New in JDF 1.3	integer	Actual amount of items delivered in this drop. Note that this logs the information after the fact in a way that is similar to an ***Audit***. @*ActualAmount* was placed here because it is very difficult to map the ***DeliveryParams*** structure of individual ***Drop*** and ***DropItem*** elements to ***ResourceLink*** and ***Audit*** elements.
ActualTotalAmount ? New in JDF 1.3	integer	Actual @*TotalAmount* of items delivered in this drop. Note that this logs the information after the fact in a way that is similar to an ***Audit***. @*ActualTotalAmount* was placed here because it is very difficult to map the ***DeliveryParams*** structure of individual ***Drop*** and ***DropItem*** elements to ***ResourceLink*** and ***Audit*** elements.
Amount ?	integer	Specifies the number of ***PhysicalResource*** ordered. If @*Amount* is not specified, defaults to the total amount of the resource that is referenced by ***PhysicalResource***.
TotalAmount ? New in JDF 1.3	integer	Total amount of individual items delivered in this drop. The @*TotalAmount* and @*Amount* differ if the ***PhysicalResource*** is a ***Bundle*** of multiple resources. The @*Amount* specifies the number of ***Bundle***s (e.g., boxes, pallets etc.). Whereas @*TotalAmount* specifies the number of final products (e.g., books, magazines etc.).
TotalDimensions ? New in JDF 1.3	shape	Total dimensions in points of all individual items including packaging delivered in this drop.
TotalVolume ? New in JDF 1.3	double	Total volume in liters of all individual items including packaging delivered in this drop.
TotalWeight ? New in JDF 1.3	double	Total weight in gram of all individual items including packaging delivered in this drop.
TrackingID ? New in JDF 1.2	string	The string that can help in tracking the delivery. The value of the @*TrackingID* attribute will depend on the carrier chosen to ship the products. Defaults to ***Drop***/@*TrackingID*.
Unit ?	string	Unit of measurement for the @*Amount* of the resource that is referenced by ***PhysicalResource***. **Default value is from:** ***PhysicalResource***/@*Unit*
Resource ? Modified in JDF 1.2	refelement	Description of the individual item to be delivered. It can be any kind of ***Resource***.

8.43 DevelopingParams

New in JDF 1.1

DevelopingParams specifies information about the chemical and physical properties of the developing and fixing process for film and plates. Includes details of preheating, postbaking and postexposure.

- Preheating is necessary for negative working plates. It hardens the exposed areas of the plate to make it durable for the following developing process. The stability and uniformity of the preheat temperature influence the evenness of tints and the run length of the plate on press.
- Postbaking is an optional process of heating that is applied to most polymer plates to enhance the run length of the plate. A factor 5 to 10 can be gained compared to plates that are not postbaked.
- Postexposure is an optional exposure process for photopolymer plates to enhance the run length of the plate. A factor of 5 to 10 can be gained compared with plates that are not postexposed.

Note: Postbaking and postexposure are mutually exclusive.

Resource Properties

Resource Class: Parameter

Input of Processes: ContactCopying, ImageSetting

Table 8.68: DevelopingParams Resource

NAME	DATA TYPE	DESCRIPTION
PreHeatTemp ?	double	Temperature of the preheating process in °C.
PreHeatTime ?	duration	Duration of the preheating process.
PostBakeTemp ?	double	Temperature of the postbaking process in °C.
PostBakeTime ?	duration	Duration of the postbaking process. @*PostBakeTime* SHALL NOT be specified if @*PostExposeTime* is present.
PostExposeTime ?	duration	Duration of the postexposing process. @*PostExposeTime* SHALL NOT be specified if @*PostBakeTime* is present.

8.44 Device

Information about a specific device. This can include information about the devices capabilities. For more information, see ▸ Section 3.8.5.3 ImplementationResource and ▸ Section 10.1 Capability and Constraint Definitions. ***Device*** describes the physical properties of the main device that executes a **JDF** process See ▸ Chapter 6 Processes. Examples are a press or a finishing machine. See ***Tool*** for a description of auxilliary devices such as fork lifts.

Resource Properties

Resource Class: Implementation

Resource referenced by: ***PhaseTime***, ***DeviceFilter***, ***IDInfo***, ***DeviceInfo*** , ***Queue***, ***QueueFilter***, ***DieLayout***, ***DieLayoutProductionParams***/***ConvertingConfig***, ***InkZoneCalculationParams***, ***RollStand***, ***StrippingParams***

Input of Processes: **Any Process**

Table 8.69: Device Resource (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
DeviceClass ? New in JDF 1.5	NMTOKENS	Indicates the class of device. Multiple NMTOKENS SHALL be used to describe integrated devices with multiple classes. See ▸ Appendix C DeviceClass and ModuleType Supported Strings for a list of predefined @*DeviceClass* values.
DeviceFamily ? Deprecated in JDF 1.1	string	Manufacturer family type ID. The @*DeviceFamily* is replaced by the appropriate @*ModelXXX* attributes in this list.
DeviceID ?	string	Name of the device. The @*DeviceID* SHALL be unique within the workflow. @*DeviceID* SHALL be the same over time for a specific device instance (i.e., SHALL survive reboots). If the device sends **JMF** Messages, this value SHALL also used for ***JMF***/@*SenderID*. For UPNP Devices, this SHALL match UPNP:UDN. See ▸ [UPNP]. @*DeviceID* need not be specified when ***Device*** is used as a filter to specify a set of Devices.
DeviceType ?	string	Manufacturer type ID, including a revision stamp. Type of the device. Used for grouping and filtering of Devices
Directory ? New in JDF 1.1	URL	Defines a directory where the URLs that are associated with this ***Device*** can be located. If @*Directory* is specified, it SHALL be an Absolute URI ▸ [RFC3986] that implicitly also specifies a Base URI which is used to resolve any relative URL of ***Device***. See ▸ Appendix J Resolving Directory URL References and ▸ [FileURL].
FriendlyName ? New in JDF 1.1 Deprecated in JDF 1.4	string	Short user-friendly title. **Deprecation note:** Starting with **JDF** 1.4, use @*DescriptiveName*.
ICSVersions ? New in JDF 1.3	NMTOKENS	CIP4 Interoperability Conformance Specification (ICS) Versions that this ***Device*** complies with. **Values include those from: *JDF*/@*ICSVersions* (** ▸ Table 3.4 JDF**).**

Table 8.69: Device Resource (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
JDFErrorURL ? New in JDF 1.2	URL	URL where, by default, the device will post **JDF** output Job tickets that are aborted or in error and when ***NodeInfo***/@*TargetRoute* is not specified. If @*JDFErrorURL* is specified in the "file" scheme, it SHALL specify a directory. If not specified, @*JDFErrorURL* defaults to the value of @*JDFOutputURL*.
JDFInputURL ? New in JDF 1.2	URL	URL where, by default, the device can accept **JDF** input Job tickets. If @*JDFInputURL* is specified in the "file" scheme, it SHALL specify a directory. The persistence of **JDF** tickets in this location is implementation dependent. If not specified, the device does not accept **JDF** without a **JMF** ***SubmitQueueEntry***.
JDFOutputURL ? New in JDF 1.2	URL	URL where, by default, the device will post **JDF** output Job tickets that are successfully completed and when ***NodeInfo***/@*TargetRoute* is not specified. If @*JDFOutputURL* is specified in the "file" scheme, it SHALL specify a directory.
JDFVersions ? New in JDF 1.1	*JDFJMF*Versions	Whitespace separated list of supported **JDF** versions that this device supports (e.g., "1.0 1.1" specifies that both the 1.0 and 1.1 versions are supported).
JMFSenderID ? New in JDF 1.1	string	ID of the controller will process **JMF** Messages for the device. This corresponds to the @*SenderID* attribute that is specified for the device in **JMF** Messages. If a device emits its own **JMF** Messages, this value SHALL match the @*DeviceID*.
JMFURL ? New in JDF 1.1	URL	URL of the device port that will accept **JMF** Messages. A controller that manages a device MAY specify its own @*JMFURL* when responding to ***KnownDevices*** Messages. This is how a controller inserts itself as the manager for a device.
KnownLocalizations ? New in JDF 1.2	languages	A list of all language codes supported by the device for localization. If not specified, then the device supports no localizations.
Manufacturer ? New in JDF 1.1	string	Manufacturer name.
ManufacturerURL ? New in JDF 1.1	string	Web site for manufacturer.
ModelDescription ? New in JDF 1.1	string	Long description for end user.
ModelName ? New in JDF 1.1	string	Model name.
ModelNumber ? New in JDF 1.1	string	Model number.
ModelURL ? New in JDF 1.1	string	Web site for model.
PresentationURL ? New in JDF 1.1	string	@*PresentationURL* specifies a URL to a device-provided user interface for configuration, status, etc. For instance, if the device has an embedded Web server, this is a URL to the configuration page hosted on that Web server.
SecureJMFURL? New in JDF 1.3	URL	URL of the device port that will accept **JMF** Messages via the "https" protocol.
SerialNumber ? New in JDF 1.1	string	Serial number of the device.
UPC ? New in JDF 1.1	string	Universal Product Code for the device. A 12-digit, all-numeric code that identifies the consumer package. Managed by the Uniform Code.
CostCenter ?	element	MIS cost center ID.

Table 8.69: Device Resource (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
DeviceCap * New in JDF 1.1	element	Description of the capabilities of the device. The ***DeviceCap*** elements are combined with a logical OR (i.e., if a **JDF** resides within any parameter space defined by a ***DeviceCap***, the device can process the Job). For details see ▸ Section Device Capabilities.
IconList ? New in JDF 1.1	element	List of locations of icons that can be used to represent the device.
Location ? New in JDF 1.4	element	Description of the device location.
Module * New in JDF 1.3	element	Individual modules that are represented by this ***Device***.

8.44.1 IconList

New in JDF 1.1

The ***IconList*** is a list of individual icon descriptions.

Table 8.70: IconList Element

NAME	DATA TYPE	DESCRIPTION
Icon +	element	Individual icon description.

8.44.2 Icon

New in JDF 1.1

An ***Icon*** represents a device in the user interface.

Table 8.71: Icon Element

NAME	DATA TYPE	DESCRIPTION
BitDepth	integer	Bit depth of one color.
IconUsage ?	enumerations	The ***DeviceInfo***/@*DeviceStatus* of the device that this ***Icon*** represents. Any combination of values are allowed. If not specified, the icon is independent of the status of the device. **Default value is:** a list of all values (i.e., no limit on ***Icon*** use). **Allowed values are:** Unknown – No link to the device exists Idle Down Setup Running Cleanup Stopped **Note:** The meaning of the individual enumerations is described in the ***DeviceInfo*** message element. See ▸ Section 5.27 KnownDevices.
Size	XYPair	Height and width of the icon.
FileSpec	element	Details of the file containing the icon data.

8.44.3 Module

New in JDF 1.3

A ***Module*** represents a physical Machine or part of a ***Device***.

Table 8.72: Module Element

NAME	DATA TYPE	DESCRIPTION
DeviceType ?	string	Manufacturer type ID, including a revision stamp.
Manufacturer ?	string	Manufacturer name.
ManufacturerURL ?	string	Web site for manufacturer.
ModelDescription ?	string	Long description for end user.
ModelName ?	string	Model name.
ModelNumber ?	string	Model number.
ModelURL ?	string	Web site for model.
ModuleID ?	string	Name of the Module. This is a unique identifier within the workflow. @*ModuleID* SHALL be the same over time for a specific device instance (i.e., SHALL survive reboots). At least one of @*ModuleID* or @*ModuleIndex* SHALL be specified. If multiple logical Devices share a physical ***Module***, @*ModuleID* SHALL be identical. @*ModuleID* SHOULD be used to specify Machines that comprise a ***Device***.
ModuleIndex ?	integer	Zero-based index of the module within the Machine. This index used to reference an individual Module. At least one of @*ModuleID* or @*ModuleIndex* SHALL be specified. @*ModuleIndex* SHOULD be used to specify identical modules (e.g., print modules in a complex device).
ModuleType ?	NMTOKEN	Type of ***Module***. **Values include those from:** ▸ Section C.2 DeviceClass and ModuleType Supported Strings**.** **Note:** The allowed values depend on the type of device. Each type of device has a separate table of values.
SerialNumber ?	string	Serial number of the device.
SubModuleIndex ?	integer	Zero-based index of the ***Module*** in the unit as specified by the parent ***Module***. SHALL NOT be specified if ***Module*** is a direct child of ***Device***.
Module *	element	Recursive modules that are part of this module.

8.45 DieLayout

New in JDF 1.3

DieLayout represents a die layout described in an external file. This resource is also used as the input for the actual die making process and is also used in **Stripping**. The external file is by preference a ▸ [DDES3] file. The usage of other files like CFF2, DDES2, DXF or proprietary formats is not excluded but MAY have a negative impact on interoperability.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***BinderySignature***, ***ShapeCuttingParams***

Input of Processes: **DieDesign**, **DieMaking**

Output of Processes: **DieDesign**, **DieLayoutProduction**

Table 8.73: DieLayout Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
CutBox ? New in JDF 1.5	rectangle	A rectangle describing the bounding box of all cut lines in the ***DieLayout***. This is sometimes referred to as the knife to knife dimensions of the ***DieLayout***. If the position on the ***Media*** is not known, the lower left SHOULD be set to 0 0.

Table 8.73: DieLayout Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
DieSide ? New in JDF 1.4	enumeration	Determines the die side for which the ***DieLayout*** is made. **Allowed values are:** Up – the ***DieLayout*** is made with the knifes pointing upwards. Down – the ***DieLayout*** is made with the knifes pointing downwards.
MediaSide ? New in JDF 1.4	enumeration	Determines the printing side for which the ***DieLayout*** is made. Front corresponds to the outside of a box, Back corresponds to the inside of a box. **Allowed value is from:** ▸ Side
Rotated ? New in JDF 1.4	boolean	Indicates if some of the structural designs are oriented cross grain/flute in the layout.
Waste ? New in JDF 1.4	double	The percent of the material that is wasted. Inner waste (i.e., cut out windows are not included in the waste).
Device * New in JDF 1.4	refelement	The Devices for which this ***DieLayout*** was made (printing press and die cutter). Typically only the type of device would be used (e.g., the model of the die cutter).
CutLines ? New in JDF 1.5	element	Selects the die line separations from the file referenced by ***FileSpec***. Additional details of the usage of the separations MAY be specified in the respective ***ColorPool***/***Color*** elements.
FileSpec ?	refelement	Reference to an external URL that represents the die. This is typically a CAD design file.
Media ? New in JDF 1.4	refelement	***Media*** for which this ***DieLayout*** was intended. The ***Media*** description defines important design parameters as the type of ***Media***, dimensions, grain direction or flute direction.
RuleLength * New in JDF 1.4	element	Elements describing the length of die rules for the different types of rules. Each ***RuleLength*** element describes the accumulated length of all rules of a certain type.
Station *	element	Description of the stations in a ***DieLayout***. One ***Station*** produces one shape.

8.45.1 RuleLength

New in JDF 1.4

Table 8.74: RuleLength Element

NAME	DATA TYPE	DESCRIPTION
DDESCutType	integer	Type of rule. **Values include:** a number between "0" and "999" corresponding to a line type as defined in DDES.
Length	double	Accumulated length of the rules of this type in the ***DieLayout*** (pt).

8.45.2 Station

Description of the stations in a ***DieLayout***. One station produces one shape type. One ***Station*** element MAY represent multiple identical one-up stations on an N-up ***DieLayout***.

Table 8.75: Station Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AssemblyIDs ? New in JDF 1.3	NMTOKENS	The list of @*AssemblyIDs* of the graphic elements that are processed by this ***Station***. **Note:** @*AssemblyIDs* was added to **JDF** 1.3 Errata.

Table 8.75: Station Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
StationAmount ="1"	integer	The number of stations in the ***DieLayout*** with this @*StationName*.
StationName ?	string	The name of the 1-up design in the ***DieLayout***.
ShapeDef ? New in JDF 1.4	refelement	The ***ShapeDef*** corresponding to this station in the ***DieLayout***.

8.46 DieLayoutProductionParams

New in JDF 1.4

Parameters for the die layout.

Resource Properties

Resource Class: Parameter

Input of Processes: **DieLayoutProduction**

Table 8.76: DieLayoutProductionParams Resource

NAME	DATA TYPE	DESCRIPTION
Estimate ?	boolean	Determines if the process runs in estimate mode or not. When in estimate mode multiple solutions are generated.
Position ?	enumeration	The position of layout on the sheet. **Allowed value is from:** ▸ Anchor.
ConvertingConfig +	element	A ***ConvertingConfig*** element describes a range of sheet sizes that can be taken into account to create a new ***DieLayout***. Typically a ***ConvertingConfig*** will correspond to a single combination of printing press and further finishing equipment such as die cutters.
RepeatDesc +	element	Step and repeat parameters for a ***ShapeDef***.There is either a single ***RepeatDesc*** giving the parameters for all ***ShapeDef*** resources at the input or there is exactly 1 ***RepeatDesc*** per ***ShapeDef*** in the input in which case the sequence of both determines which ***RepeatDesc*** should be used for a ***ShapeDef***.

8.46.1 RepeatDesc

New in JDF 1.4

The ***RepeatDesc*** element describes the layout specs for a ***ShapeDef***.

Table 8.77: RepeatDesc Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AllowedRotate ?	enumeration	Allowed methods to rotate structural designs in with respect to grain/flute. **Allowed values are:** None – No Rotation at all. Grain – 0° or 180° Rotation. MinorGrain – device dependent small rotations that retain the general grain direction (e.g., +/- 10°). CrossGrain – Cross grain rotations (e.g., 90° are acceptable).
GutterX ?	double	Gutter between columns (see also @*GutterX2*)
GutterX2 ?	double	Secondary gutter between columns. When the @*LayoutStyle* = "Reverse2ndColumn", the gutter between columns (2n+1) and (2n+2) is @*GutterX* and between columns (2n+2) and (2n+3) is @*GutterX2*. When @*GutterX2* is not specified @*GutterX2* = @*GutterX*. See ▸ Figure 8-30: RepeatDesc/@GutterX2 and @GutterY2: Secondary Gutters.
GutterY ?	double	Gutter between rows (see also @*GutterY2*).

Table 8.77: RepeatDesc Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
GutterY2 ?	double	Secondary gutter between rows. When the @*LayoutStyle* = "Reverse2ndRow" the gutter between rows (2n+1) and (2n+2) is @*GutterY* and between rows (2n+2) and (2n+3) @*GutterY2*. When @*GutterY2* is not specified @*GutterY2* = @*GutterY*. See ▸ Figure 8-30: RepeatDesc/@GutterX2 and @GutterY2: Secondary Gutters.
LayoutStyle ?	NMTOKENS	The allowed styles for the ***Layout*** **Values include:** StraightNest Reverse2ndRow Reverse2ndRowAligned Reverse2ndColumn Reverse2ndColumnAligned **Note:** For diagrams of the above values, see ▸ Figure 8-24: Basic Shape for RepeatDesc/@LayoutStyle Examples and the following five figures
OrderQuantity ?	integer	The order quantity for the 1-up for which this layout will be optimized. This information needs to be present when a ***Layout*** is being made for more than 1 ***ShapeDef***.
UseBleed ?	boolean	If true, the print bleed defined in the structural design is used to calculate the layout. If false, the outer cut is used.

The following Figure shows the basic shape for subsequent Figures. that relate to ***RepeatDesc***.

Figure 8-24: Basic Shape for RepeatDesc/@LayoutStyle Examples

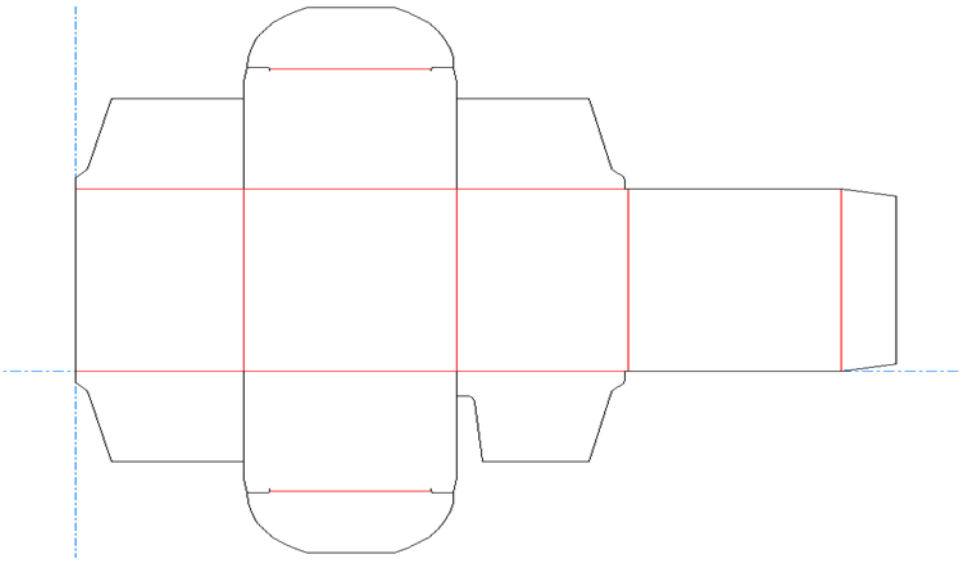
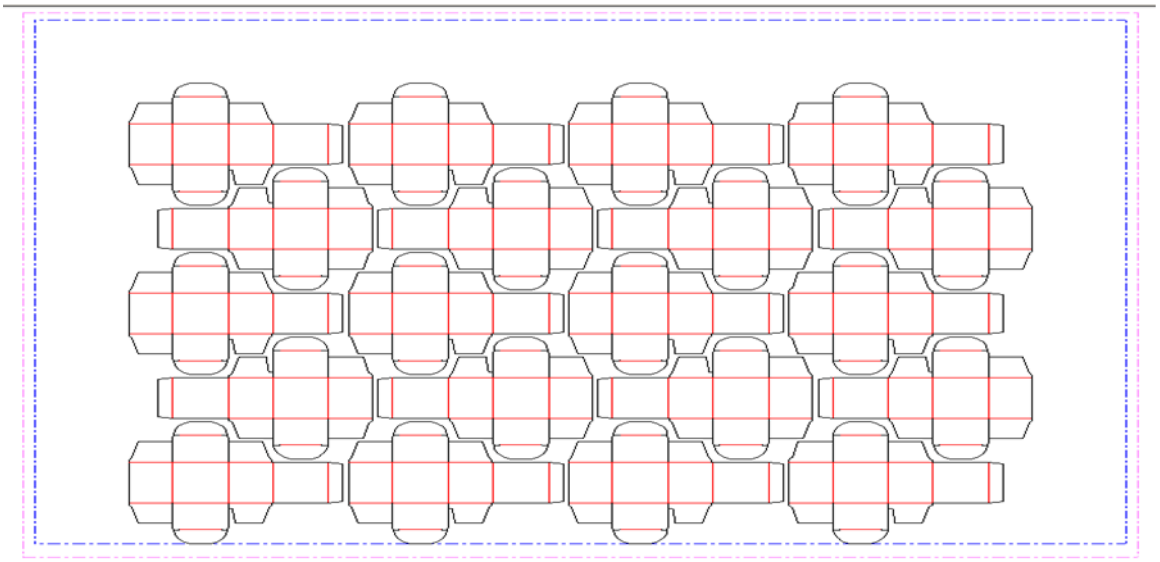
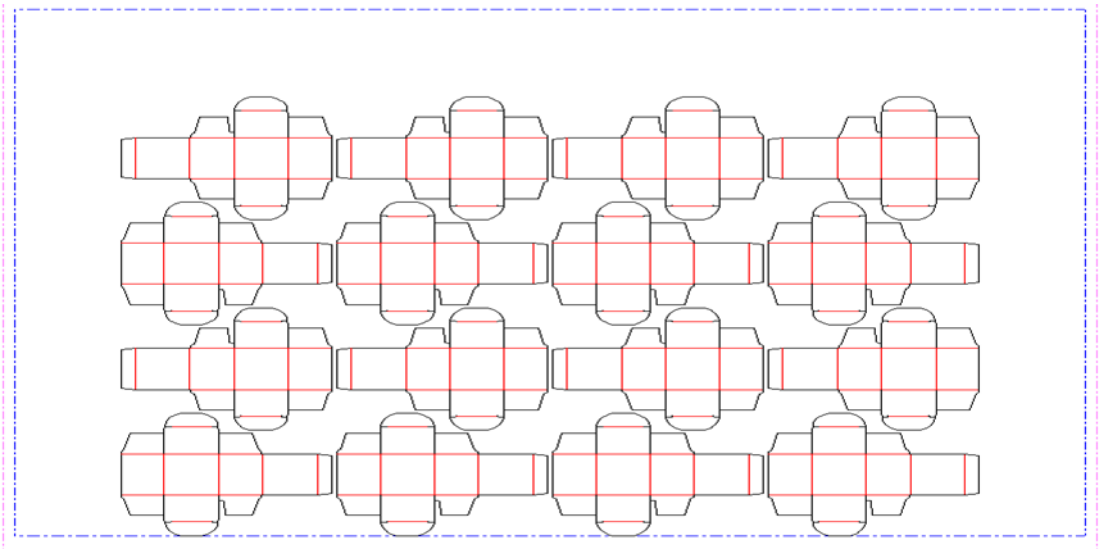


Figure 8-25: RepeatDesc/@LayoutStyle = "StraightNest"



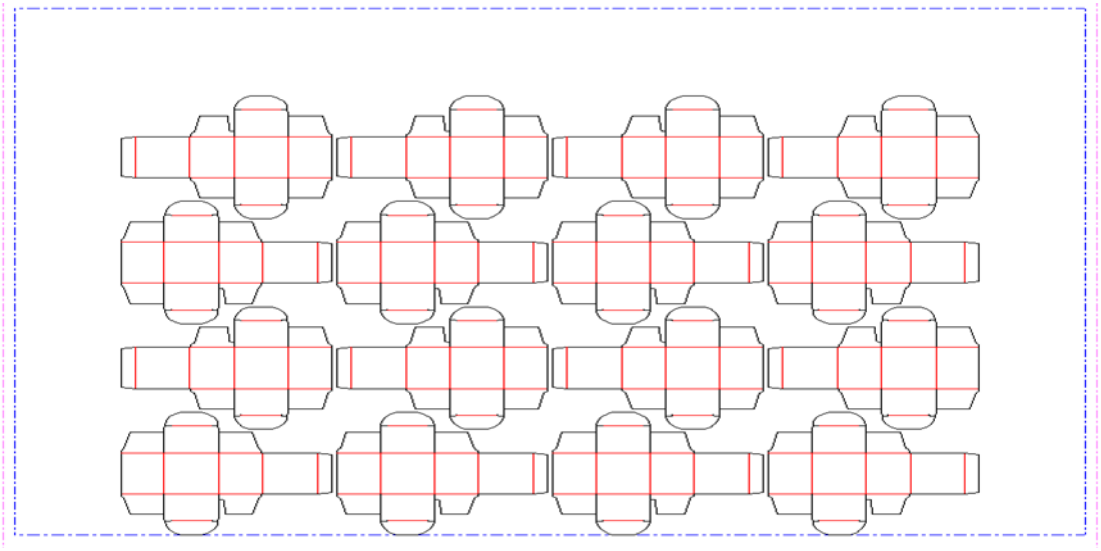
In the following Figure, 1-ups on even rows are rotated 180 degrees. Even rows are shifted horizontally and vertically to obtain optimal nesting.

Figure 8-26: RepeatDesc/@LayoutStyle = "Reverse2ndRow"



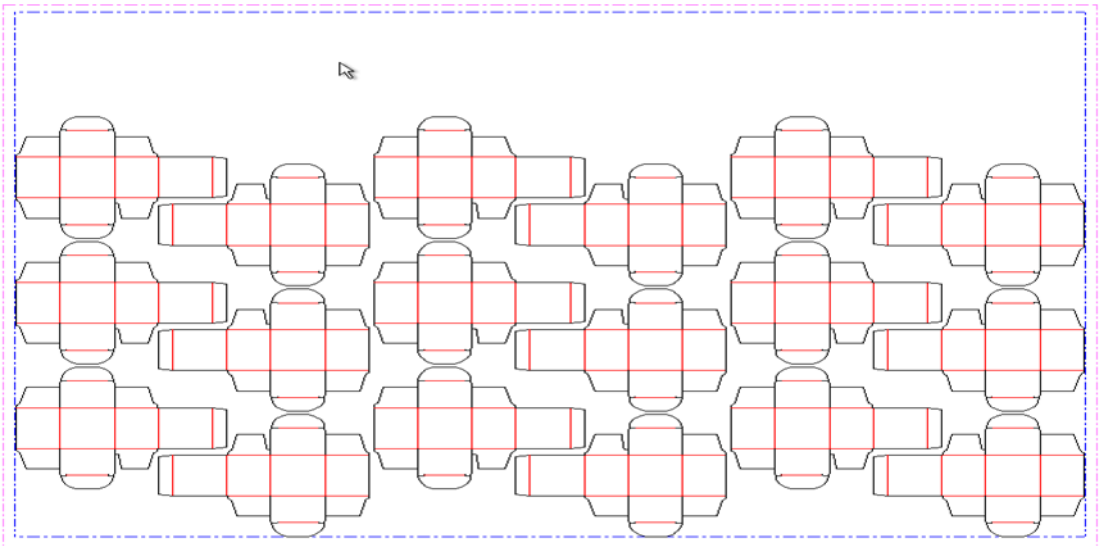
In the following Figure, 1-ups on even rows are rotated 180 degrees. Even rows are shifted vertically to obtain optimal nesting. The even rows are not shifted horizontally. (Left and right edges are aligned between rows)

Figure 8-27: RepeatDesc/@LayoutStyle = "Reverse2ndRowAligned"



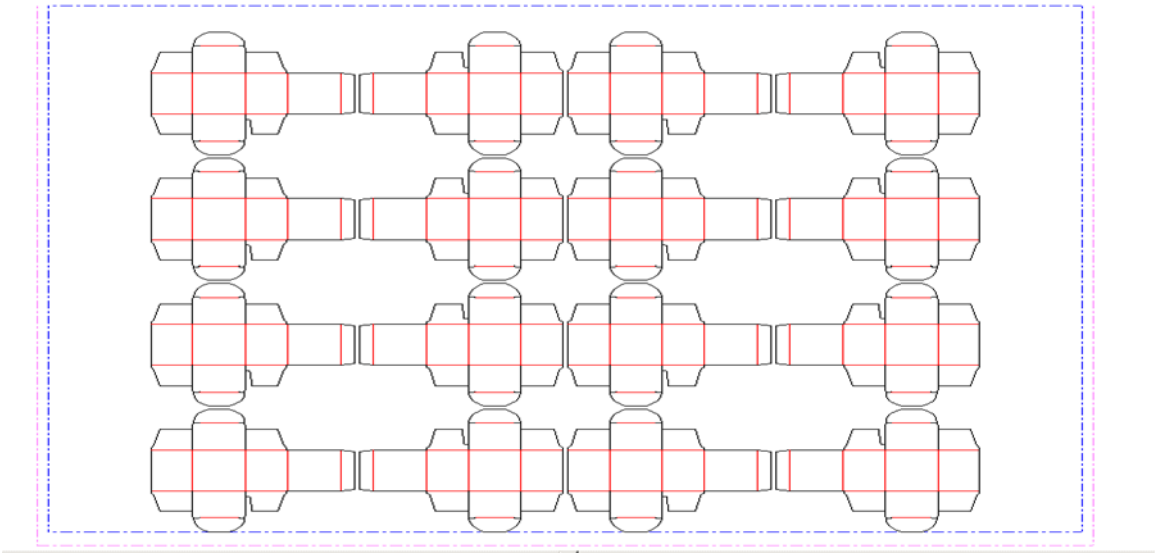
In the following Figure, 1-ups on even columns are rotated 180 degrees. Even columns are shifted vertically and horizontally to obtain optimal nesting.

Figure 8-28: RepeatDesc/@LayoutStyle = "Reverse2ndColumn"



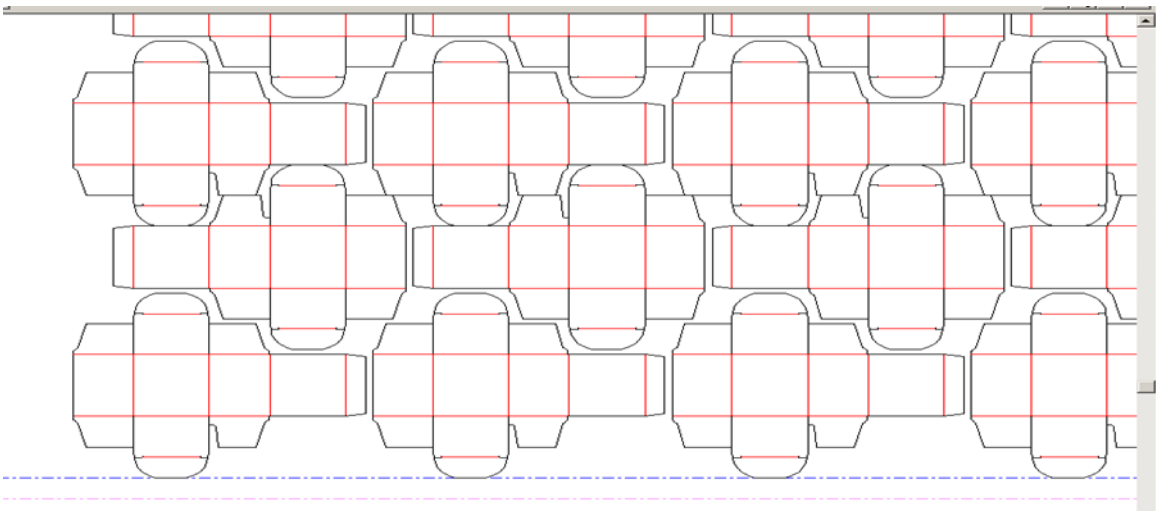
In the following Figure, 1-ups on even columns are rotated 180 degrees. Even columns are shifted horizontally to obtain optimal nesting. No vertical shifting of even columns is done (top and bottom edges are aligned between columns).

Figure 8-29: RepeatDesc/@LayoutStyle = "Reverse2ndColumnAligned"



In the following Figure, @*LayoutStyle* = "Reverse2ndRow", @*GutterY* = "15", @*GutterY2* = "0".

Figure 8-30: RepeatDesc/@GutterX2 and @GutterY2: Secondary Gutters



8.47 DigitalDeliveryParams

New in JDF 1.2

This resource specifies the parameters of the **DigitalDelivery** process.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***ArtDeliveryIntent***

Example Partition: "Location"

Input of Processes: **DigitalDelivery**

Table 8.78: DigitalDeliveryParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
DigitalDeliveryDirection ?	enumeration	Describes which side activates the delivery. **Allowed values are:** Push – The artwork will be sent (the source end is active). Pull – The artwork will be retrieved (the destination end is active).

Table 8.78: DigitalDeliveryParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
DigitalDeliveryProtocol ?	NMTOKEN	Identifies the delivery network protocol. **Values include:** FTP HTTP HTTPS SMTP
Method ? Modified in JDF 1.5	NMTOKEN	Identifies the delivery method. **Values include:** EMail NetworkCopy – This includes LAN and VPN. WebServer – Upload / Download from HTTP / FTP server. InstantMessaging Values include also: any brand name of a network provider. New in JDF 1.5
Contact *	refelement	Source and destination address for the transfer of the artwork. The destination delivery address is specified as the ***Contact***[contains (@*ContactTypes*, "Delivery")]/***ComChannel***. Exactly one such ***Contact*** SHALL be specified per destination. If multiple delivery destinations are specified within one **DigitalDelivery** process, such a ***Contact*** SHALL be Partitioned with the Partition Key "Location". If the output ***RunList*** completely specifies the destination, a ***Contact***[contains (@*ContactTypes*, "Delivery")] SHOULD be omitted. This is generally the case if @*Method* = "NetworkCopy" or "WebServer". A ***Contact***[contains (@*ContactTypes*, "Sender")] specifies the source address.

Compression & Encoding of the transferred files:

In order to instruct a digital delivery device to compress or encode the files one can use the input and output ***RunList*** with ***FileSpec***/@*Compression* attribute, even if no URL is specified. See ▸ Section O.7 DigitalDelivery Examples for a set of examples.

8.48 DigitalMedia

New in JDF 1.2

This resource represents a processed removable digital media-based ***Handling Resource*** such as tape or removable disk.

Resource Properties

Resource Class: Handling

Resource referenced by: ***ArtDeliveryIntent***/***ArtDelivery***

Table 8.79: DigitalMedia Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Capacity ?	integer	Size of the digital media in megabytes.
MediaLabel ?	string	Electronic label of the media.
MediaType	NMTOKEN	The digital media type. **Values include:** CD – Recordable compact disc. DAT – DAT tape backup media. DLT – DLT tape backup media. DVD – DVD disc. Exabyte – Exabyte tape backup media. HardDrive – Removable hard drives from a rack. Jaz – Jaz removable disk drive. Optical – Optical removable disk drive. Excluding CDs and DVDs. Tape – Tape backup media. Use only when the explicit tape type is not listed here. Zip – Zip removable disk drive.

Table 8.79: DigitalMedia Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
MediaTypeDetails ?	string	The digital media type details — could be vendor or model name. For example: "8mm" or "VHS" for tape media.
RunList ?	refelement	Link to the relevant files on the media. The URLs specified in ***RunList***/***LayoutElement***/***FileSpec***/@*URL* SHOULD be relative paths to the media's mount point.

8.49 DigitalPrintingParams

This resource contains attributes and elements used in executing the **DigitalPrinting** process. The @*PrintingType* attribute in this resource defines two types of printing: "SheetFed" and "WebFed". The principal difference between them is the shape of the paper each is equipped to accept. Presses that execute "WebFed" processes use substrates that are continuous and cut after printing is accomplished. Most newspapers are printed on Web Presses. "SheetFed" printing, on the other hand, accepts precut substrates.

8.49.1 Coordinate systems in DigitalPrinting

New in JDF 1.2

▸ Figure 2-11: Press coordinate system used for Web Printing in ▸ Section 2.5 Coordinate Systems in JDF defines the coordinate system for **ConventionalPrinting** and **DigitalPrinting**. Note that the paper feed direction of the idealized process is towards the X-axis which corresponds to bottom edge first.

Resource Properties

Resource Class: Parameter

Example Partition: "BlockName", "DocRunIndex", "DocSheetIndex", "PartVersion", "Run", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetIndex", "Separation", "SheetName", "Side", "SignatureName", "DocIndex"

Input of Processes: **DigitalPrinting**

Table 8.80: DigitalPrintingParams Resource (Sheet 1 of 4)

NAME	DATA TYPE	DESCRIPTION
Collate ? New in JDF 1.1	enumeration	Determines the sequencing of the Sheets in the document and the documents in the Job when multiple copies of a document or a Job are requested as output. Document copies can be requested by specifying ***RunList***/@*DocCopies* and Job copies can be requested by specifying the output ***Component*** @*Amount*. **Allowed values are:** None – Do not collate Sheets in the document or document(s) in the Job. Sheet – Collate the Sheets in each document; do not collate the documents in the Job. The result of "Sheet" and "SheetAndSet" is the same when there is one document in the set. The result of "Sheet" and "SheetSetAndJob" is the same when there is one document in the set and one set in the Job. SheetAndSet – Collate the Sheets in the document and collate the documents in the set. Do not collate the sets in the Job. The result of "SheetAndSet" and "SheetSetAndJob" is the same when there is one set in the Job. SheetSetAndJob – Collate the Sheets in the document and collate the documents in the set and collate the sets in the Job. **Example:** two documents, A and B, each have two Sheets, A1, A2 and B1, B2. The number of document copies requested is one for both documents and the number of Job copies requested is three (***Component***/@*Amount* = 3). The Job contains no Document Set boundaries. If @*Collate* = "None", the Sheet order will be: A1A1A1 A2A2A2 B1B1B1 B2B2B2 If @*Collate* = "Sheet", the Sheet order will be: A1A2 A1A2 A1A2 B1B2 B1B2 B1B2 If @*Collate* = "SheetAndSet" or "SheetSetAndJob", the Sheet order will be: A1A2 B1B2 A1A2 B1B2 A1A2 B1B2

Table 8.80: DigitalPrintingParams Resource (Sheet 2 of 4)

NAME	DATA TYPE	DESCRIPTION
DirectProofAmount = "0" New in JDF 1.2	integer	If greater than zero (>0), a set of proofs is directly produced and subsequently an approval might be given by a person (e.g., the customer, foreman or floor manager) shortly after the first final-quality printed Sheet is printed. Approval is needed for the actual print run, but not for setup. If the **DigitalPrinting** process is waiting for approval of a direct proof, the **JDF** node's @*Status* is switched to "Suspended" with the @*StatusDetails* = "WaitForApproval".
ManualFeed = "false" New in JDF 1.1	boolean	Indicates whether the media will be fed manually.
NonPrintableMarginBottom ? New in JDF 1.2	double	The width in points of the bottom margin measured inward from the edge of the media (before trimming if any) with respect to the idealized process coordinate system of the **DigitalPrinting** process. The **DigitalPrinting** process SHALL put marks up to, but not in, the non-printable margin area. The ***Media***'s origin is unaffected by @*NonPrintableMarginBottom*. These margins are independent of the PDL content.
NonPrintableMarginLeft ? New in JDF 1.2	double	Same as @*NonPrintableMarginBottom* except for the left margin.
NonPrintableMarginRight ? New in JDF 1.2	double	Same as @*NonPrintableMarginBottom* except for the right margin.
NonPrintableMarginTop ? New in JDF 1.2	double	Same as @*NonPrintableMarginBottom* except for the top margin.
OutputBin ? New in JDF 1.1 Modified in JDF 1.5	NMTOKENS	Specifies the bin or bins to which the finished documents SHALL be output. If multiple values are provided, the output bins SHALL be filled in sequence. See @*StackAmount*. **Values include those from:** ▸ Table C.10 Input Tray and Output Bin Names. **Modification note:** Starting with **JDF** 1.5, the data type changes from NMTOKEN to NMTOKENS.
PageDelivery ? New in JDF 1.1	enumeration	Indicates how pages are to be delivered to the output bin or finisher. **Note:** These values refer to the orientation of the entire stack being output from the press, not individual sheets. For example, "SameOrderFaceDown" means that the stack can be picked up and turned over to find the output sheets in the same order as the input ***RunList*** with the first page on top facing up **Allowed values are:** FanFold – The output is alternating face-up, face down. SameOrderFaceUp – Order as defined by the ***RunList***, with the "Front" sides of the media up and the first Sheet on top. SameOrderFaceDown – Order as defined by the ***RunList***, with the "Front" sides of the media down and the first Sheet on the bottom. ReverseOrderFaceUp – Sheet order reversed compared to "SameOrderFaceUp", with the *Front* sides of the media up and the last Sheet on top. ReverseOrderFaceDown – Sheet order reversed compared to "SameOrderFaceDown", with the *Front* sides of the media down and the last Sheet on the bottom.
PrintingType ? Modified in JDF 1.2	enumeration	Type of printing Machine. **Allowed values are:** ContinuousFed – connected Sheets including fan fold. New in JDF 1.2 SheetFed WebFed

Table 8.80: DigitalPrintingParams Resource (Sheet 3 of 4)

NAME	DATA TYPE	DESCRIPTION
PrintPass ? New in JDF 1.5	enumeration	Defines how many passes are required to lay down all separations. **Allowed values are:** OneShot – all separations are laid down in one pass; MultiShot – separations are laid down individually in multiple passes.
PrintQuality ? Deprecated in JDF 1.1	enumeration	Indicates how pages are to be delivered to the output bin or finisher. **Allowed values are:** High – Highest quality available on the printer. Normal – The default quality provided by the printer. Draft – Lowest quality available on the printer **Deprecation note:** Starting with **JDF** 1.1, use ***InterpretingParams***/@*PrintQuality*
SheetLay ?	enumeration	Lay of input media. Reference edge of where paper is placed in feeder. **Allowed value is from:** ▸ SheetLay.
Sides ? New in JDF 1.3 Modified in JDF 1.5	enumeration	Indicates whether the ***ByteMap*** SHALL be imaged on one or both sides of the media. If the ***RunList***(Surface) input to **DigitalPrinting** is Partitioned by @*Side* (either explicitly or implicitly using the ***RunList***/@*SheetSides* attribute), then the input ***RunList*** provides a binding of front and back surfaces to sheets. If @*Sides* = "OneSidedFront" or "OneSidedBack", then that binding is ignored and one surface is imaged per sheet. If the ***RunList*** (Surface) does not provide the binding of surfaces to sides, then the @*Sides* attribute specifies the binding to be applied. When a different value for this attribute is encountered, it SHALL force a new Sheet. However, when the same value for this attribute is restated for consecutive pages, it is the same as if that restatement was not present. **Allowed values are:** OneSidedBack New in JDF 1.5 OneSidedFront OneSidedBackFlipX Deprecated in JDF 1.5 OneSidedBackFlipY Deprecated in JDF 1.5 TwoSided New in JDF 1.5 TwoSidedFlipX Deprecated in JDF 1.5 TwoSidedFlipY Deprecated in JDF 1.5 **Note:** Starting with **JDF** 1.5, the orientation of the front pages relative to back pages SHOULD be completely defined in the explicit or implied imposition ***Layout***.
StackAmount ? New in JDF 1.5	integer	Specifies the maximum sheet count before switching to the next stacker in the list of @*OutputBin* values.
ApprovalParams ? New in JDF 1.2	refelement	Details of the direct approval process, when @*DirectProofAmount* > 0.
Component ? New in JDF 1.1	refelement	Describes the preprocessed media to be used. Different ***Media*** and/or ***Component*** resources MAY be specified in different Partition leaves to enable content-driven input ***Media*** selection. At most one of ***Media*** or ***Component*** SHALL be specified per Partition.
Disjointing ? New in JDF 1.1 Deprecated in JDF 1.6	element	Describes how individual components are separated from one another in the output bin. **Deprecation note:** Starting with **JDF** 1.6, use combined stacking.
Ink ? New in JDF 1.3 Deprecated in JDF 1.4	refelement	If present indicates that overcoating SHALL be applied to the surface(s) of printed Sheets and specifies the ink to be used for overcoating. Overcoating ink SHALL be applied after imaging colorants have been printed. **Note:** For selective image-wise overcoating (e.g., spot varnishing) a separate separation utilizing overcoating ink SHALL be specified. **Deprecation note:** Starting with **JDF** 1.4, use the **Varnishing** process.
Media ? New in JDF 1.1	refelement	Describes the media to be used. Different ***Media*** and/or ***Component*** resources MAY be specified in different Partition leaves to enable content driven input ***Media*** selection. At most one of ***Media*** and ***Component*** SHALL be specified per Partition.

Table 8.80: DigitalPrintingParams Resource (Sheet 4 of 4)

NAME	DATA TYPE	DESCRIPTION
MediaSource ? Deprecated in JDF 1.1	refelement	Describes the media to be used. At most one of ***MediaSource*** and ***Component*** SHALL be specified. Replaced with ***Media*** in **JDF** 1.1.

8.50 DividingParams

Deprecated in JDF 1.1.

Since the **Dividing** process has been replaced by **Cutting**, this resource is no longer needed. See ▸ Section N.7.8 DividingParams for details of this deprecated resource.

8.51 ElementColorParams

New in JDF 1.2

This resource provides the current state of color management related metadata such as the targeted printing condition that a content element has been prepared for.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***ContentList***/***ContentData***, ***LayoutElement***, ***PageList***, ***PageList***/***PageData***

Table 8.81: ElementColorParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AutomatedOverPrintParams ?	element	A resource that provides controls for the automated selection of overprinting of black text or graphics.
ColorManagementSystem ?	NMTOKEN	Identifies the preferred ICC color management system to use when performing color transformations on the particular ***LayoutElement***. When specified, this attribute overrides any default selection of a color management system by an application and overrides the "CMM Type" value (bytes 4-7 of an ICC Profile Header) in any of the Job related ICC profiles. This string attribute Value identifies the manufacturer of the preferred CMM and SHALL match one of the registered four-character ICC CMM Type values. **Values include those from: the ICC Manufacturer's Signature Registry at** http://www.color.org**.** **Example values:** "ACME" for the Acme Corp. CMM.
ICCOutputProfileUsage ?	enumeration	This attribute specifies the usage of the output intent profile or specified printing condition from the PDL. **Allowed values are:** PDLActual – The embedded PDL output printing condition defines the actual output intent profile (e.g., the final press output). PDLReference – The embedded PDL output printing condition defines the reference output intent profile (e.g., the press profile for proofing). IgnorePDL – The embedded ICC output profile is incorrect and SHOULD be ignored.
AutomatedOverPrintParams ?	element	A resource that provides controls for the automated selection of overprinting of black text or graphics.
ColorantAlias *	element	Each resource instance specifies a replacement colorant name string to be used instead of one or more named colorant strings found in the ***Layout*** resource referenced element. Multiple ***ColorantAlias*** elements with identical values of ***ColorantAlias***/@*ReplacementColorantName* SHALL NOT be specified in the same ***ElementColorParams*** resource context.
ColorCorrectionOp * New in JDF 1.5	element	List of ***ColorCorrectionOp*** Subelements, each of which identifies a type of object and specifies the behavior of the color correction for that type of object.

Table 8.81: ElementColorParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ColorSpaceConversionOp ?	element	List of ***ColorSpaceConversionOp*** Subelements, each of which identifies a type of object, defines the source color space for that type of object, and specifies the behavior of the conversion operation for that type of object. If not present, the default conversion behavior is derived from @*ColorStandard*. ***ColorSpaceConversionOp***/@*Operation* is ignored in the context of ***ElementColorParams***.
FileSpec (ActualOutputProfile) ?	refelement	A ***FileSpec*** resource pointing to an ICC profile that describes the characterization of an actual output target device.
FileSpec (ReferenceOutputProfile) ?	refelement	A ***FileSpec*** resource pointing to an ICC profile that describes a reference output print condition behavior that SHALL be simulated as a part of a requested color transformation. This profile corresponds to the output intent contained in a PDF/X file. It SHOULD be a specific implementation of ***ColorIntent***/@*ColorStandard*.

8.52 EmbossingParams

New in JDF 1.1

This resource contains attributes and elements used in executing the **Embossing** process. The **Embossing** can also be used to model a foil stamping process.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***EmbossingIntent***

Example Partition: "BlockName", "RibbonName", "SheetName", "SignatureName", "WebName"

Input of Processes: **Embossing**

Table 8.82: EmbossingParams Resource

NAME	DATA TYPE	DESCRIPTION
ModuleIndex ? New in JDF 1.4	integer	Index of the embossing Module in the Press. See ***ConventionalPrintingParams***. In a combined process, all modules of the device, including press modules, finishing modules and varnishing modules are counted to calculate @*ModuleIndex*.
Emboss *	element	One ***Emboss*** element is specified for each impression.

8.52.1 Emboss

Table 8.83: Emboss Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Direction Modified in JDF 1.3	enumeration	The direction of the image. **Allowed value is from:** ▸ EmbossDirection.
EdgeAngle ?	double	The angle of a beveled edge in degrees. Typical values are an angle of: 30, 40, 45, 50 or 60 degrees. If @*EdgeAngle* is specified, @*EdgeShape* = "Beveled" SHALL be specified.
EdgeShape = "Rounded"	enumeration	The transition between the embossed surface and the surrounding media can be rounded or beveled (angled). **Allowed values are:** Rounded Beveled

Table 8.83: Emboss Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
EmbossingType Modified in JDF 1.3	enumeration	**Allowed value is from:** ▸ EmbossType. BlindEmbossing – Embossed forms that are not inked or foiled. The color of the image is the same as the paper. Braille – 6 dot Braille embossing. **Note:** "Braille" was added to **JDF** 1.3 Errata. New in JDF 1.3 EmbossedFinish – The overall design or pattern impressed in laminated paper when passed between metal rolls engraved with the desired pattern. Produced on a special embossing to create finishes such as linen. FoilEmbossing – Combines embossing with foil stamping in one single impression. FoilStamping – Using a heated die to place a metallic or pigmented image from a coated foil on the paper. RegisteredEmbossing – Creates an embossed image that exactly registers to a printed image.
Face ? New in JDF 1.6	enumeration	Position of the embossing on the product. Allowed value is from: ▸ Face.
Height ?	double	The height of the levels. This value specifies the *vertical* distance between the highest and lowest point of the stamp, regardless of the value of @*Direction*.
ImageSize ?	XYPair	The size of the bounding box of one single image.
Level ?	enumeration	The level of embossing. **Allowed value is from:** ▸ EmbossLevel.
Position ?	XYPair	Position of the lower left corner of the bounding box of the embossed image in the coordinate system of the ***Component***.
IdentificationField ? New in JDF 1.4	refelement	If @*EmbossingType* = "Braille", ***IdentificationField*** describes the content of the Braille element.
Media ? New in JDF 1.4	refelement	If the @*EmbossingType* = "FoilEmbossing" or "FoilStamping", ***Media*** describes the foil.
Tool ? New in JDF 1.4	refelement	The tool used to make the embossing described by this element.

8.53 Employee

Information about a specific device or Machine operator (see ▸ Section 3.8.5.3 ImplementationResource). ***Employee*** is also used to describe the contact person who is responsible for executing a node, as defined in ***NodeInfo***.

Resource Properties

Resource Class: Implementation

Resource referenced by: ***Abstract Audit***, ***Notification***, ***PhaseTime***, ***ModulePhase***, **JMF**, ***Message***, ***DeviceInfo***, ***ModuleStatus***, ***ContentList***/***ContentData***/***ContentMetadata***, ***NodeInfo***

Input of Processes: **Any Process**

Table 8.84: Employee Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
PersonalID ?	string	ID of the relevant MIS employee. The @*PersonalID*attribute SHALL be unique within the site.

Table 8.84: Employee Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Roles ? New in JDF 1.2 Modified in JDF 1.4	NMTOKENS	Defines the list of roles that the employee fills. **Values include:** Apprentice – Employee that is in training (“Auszubildender” / “Auszubildende” in German). Assistant – Assistant operator. Craftsman – Trained employee (“Geselle” / “Facharbeiter” in German). CSR – Customer Service Representative Manager – Manager. Master – Highly trained employee (“Meister” in German). Operator – Operator. ShiftLeader – The leader of the shift. StandBy – Employee who is allocated to a specific task on demand. New in JDF 1.4
Shift ?	string	Defines the shift to which the employee belongs.
CostCenter ?	element	MIS cost center ID.
Person ?	element	Describes the employee. If no ***Person*** resource is specified, the ***Employee*** resource represents any employee who fulfills the selection criteria.

8.54 EndSheetGluingParams

This resource describes the attributes and elements used in executing the **EndSheetGluing** process.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **EndSheetGluing**

Table 8.85: EndSheetGluingParams Resource

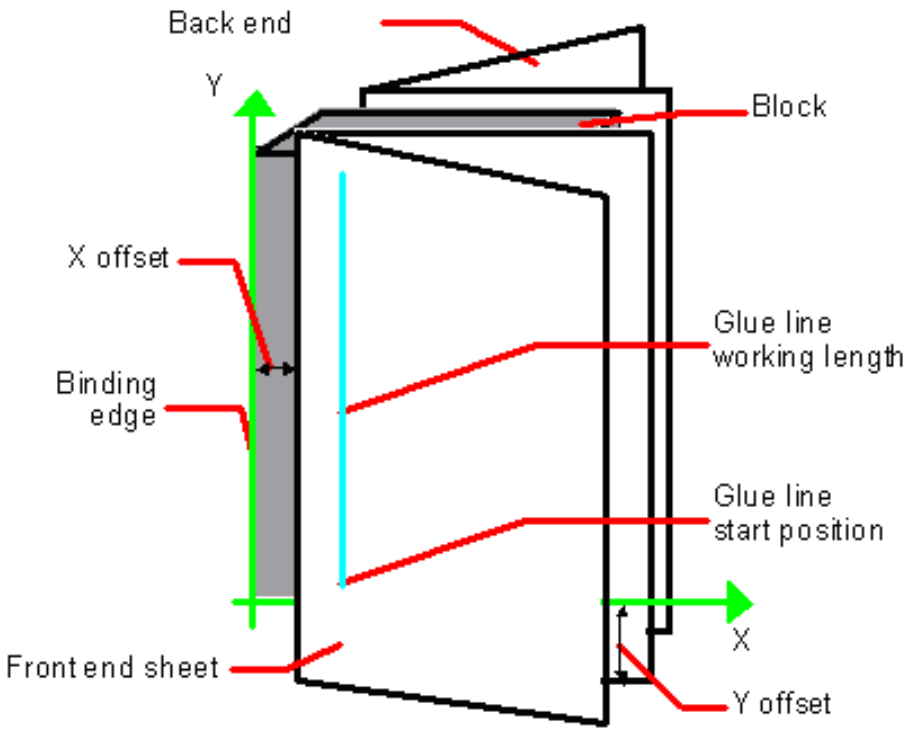
NAME	DATA TYPE	DESCRIPTION
EndSheet (Front) ? Modified in JDF 1.5	element	Information about the front-end Sheet. The @*Side* attribute of this element SHALL be "Front". **Modification note:** Starting with **JDF** 1.5, this element is optional.
EndSheet (Back) ? Modified in JDF 1.5	element	Information about the back-end Sheet. The @*Side* attribute of this element SHALL be "Back". **Modification note:** Starting with **JDF** 1.5, this element is optional.

8.54.1 EndSheet

Table 8.86: EndSheet Element

NAME	DATA TYPE	DESCRIPTION
Offset ? Deprecated in JDF 1.2	XYPair	Offset of end Sheet in X and Y direction. In **JDF** 1.2 and beyond, @*Offset* is implied by the Transformation matrix in ***ResourceLink***/@*Transformation* of the ***EndSheet*** element's ***ComponentLink***.
Side	enumeration	Location of the end Sheet. **Allowed value is from:** ▸ Side
GlueLine	element	Description of the glue line.

Figure 8-31: Parameters and coordinate system used for end-Sheet gluing



The process coordinate system is defined as follows: The Y-axis is aligned with the binding edge of the book block. It increases from the registered edge to the edge opposite to the registered edge. The X-axis is aligned with the registered edge. It increases from the binding edge to the edge opposite the binding edge (i.e., the product front edge).

8.55 ExposedMedia

This resource represents a processed ***Media***-based ***Handling Resource*** such as film, plate or paper proof. It is also used as an Input resource for the **Scanning** process. The @*ProductID* attribute SHALL be unique within the workflow.

Resource Properties

Resource Class: Handling

Resource referenced by: ***ArtDeliveryIntent***/***ArtDelivery***

Example Partition: "DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "Separation", "SheetName", "Side", "SignatureName", "TileID", "WebName"

Input of Processes: **Bending**, **ContactCopying**, **ConventionalPrinting**, **DigitalPrinting**, **ImageSetting**, **PreviewGeneration**, **Scanning**, **Varnishing**

Output of Processes: **Bending**, **ContactCopying**, **ImageSetting**

Table 8.87: ExposedMedia Resource

NAME	DATA TYPE	DESCRIPTION
ColorType ?	enumeration	If this ***ExposedMedia*** represents a proof, @*ColorType* SHALL specify the color of the proof. **Allowed values are:** Color GrayScale Monochrome – Black and white.
PageListIndex ? New in JDF 1.3	IntegerRangeList	List of the indices of the ***PageData*** elements of the ***PageList*** specified in this ***ExposedMedia***.
PlateType ? New in JDF 1.3	enumeration	Specifies whether a plate is exposed or a dummy plate. **Allowed values are:** Exposed – The plate has been imaged. Dummy – Specifies a dummy plate that has not been imaged. Usually, dummy plates are only needed on newspaper/web presses or for **Varnishing**.
Polarity = "true"	boolean	"false" if the media contains a negative image.
ProofName ? New in JDF 1.2	string	When this ***ExposedMedia*** specifies a proof, @*ProofName* is the name of the ***ProofingIntent***/***ProofItem*** that specified this proof in the Product Intent section.
ProofQuality ? Modified in JDF 1.2	enumeration	This attribute is present if the ***ExposedMedia*** resource describes a proof. **Allowed values are:** None – Not a proof or the quality is unknown. Deprecated in JDF 1.2 Halftone – Halftones are emulated. Contone – No halftones, but exact color. Conceptual – Color does not match precisely.
ProofType ? Modified in JDF 1.2	enumeration	**Allowed values are:** None – Not a proof or the type is unknown. Deprecated in JDF 1.2 Page – A page proof. Imposition – An imposition proof.
PunchType ?	string	Name of the registration punch scheme. If not specified, no holes are punched. **Values include:** Bacher Stoesser
Resolution ?	XYPair	Resolution of the output.
FileSpec (OutputProfile) ?	refelement	A ***FileSpec*** resource pointing to an ICC profile that describes the output process for which this media was exposed.
Media	refelement	Describes media specifics such as size and type.
PageList ? New in JDF 1.3	refelement	Specification of page metadata for pages described by this ***ExposedMedia***.
ScreeningParams ?	refelement	Used to describe the screening in case of rasterized media.

8.56 ExternalImpositionTemplate

New in JDF 1.3

ExternalImpositionTemplate specifies a reference to an external imposition template.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***LayoutPreparationParams***, ***StrippingParams***

Table 8.88: ExternalImpositionTemplate Resource

NAME	DATA TYPE	DESCRIPTION
FileSpec	refelement	A reference to a file that contains an external imposition template in a private (non-**JDF**) format.

8.57 FeedingParams

New in JDF 1.2

The parameters for any **JDF** Feeder processing device.

Resource Properties

Resource Class: Parameter

Example Partition: "DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "Separation", "SheetName", "Side", "SignatureName", "TileID", "WebName"

Input of Processes: **Feeding**

Table 8.89: FeedingParams Resource

NAME	DATA TYPE	DESCRIPTION
CollatingItem *	element	Defines the collating sequence of the input ***Component***(s). If a ***CollatingItem*** is not defined, then one ***Component*** in the order of input ***ResourceLink*** list is consumed.
Feeder *	element	Defines the specifics of an individual ***Feeder***. If a ***Component*** or ***Media*** from the Input resource list is not referenced from a ***Feeder*** in this list, a system defined ***Feeder*** will be used.

8.57.1 Feeder

Table 8.90: Feeder Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AlternatePositions ?	IntegerList	Positions of alternate feeders including the feeder specified in @*Position* on a feeding chain. Alternate feeders share the load according to the policy defined in @*FeederSynchronization*. If not specified, it defaults to the value of @*Position*. @*AlternatePositions* SHALL be non-negative.
FeederSynchronization = "Primary"	enumeration	Specifies the synchronization of multiple ***Feeder*** elements with identical ***Component*** elements: **Allowed values are:** Alternate – The feeders specified in @*Position* alternate. Backup – This ***Feeder*** is the backup feeder for the ***Component*** in case of a mis-feed or malfunction. The priority of backup feeders is defined by their position in @*AlternatePositions*. Chain – This feeder is activated as soon as the feeder prior to it in the list is empty. Primary – This ***Feeder*** is the primary feeder for the ***Component***.
FeederType ? Modified in JDF 1.4 Modified in JDF 1.5	NMTOKEN	Specifies the feeder type. **Values include:** AddOn – Add on feeder (e.g., CDs). BookBlock – A feeder for book blocks. New in JDF 1.4 Folding – A folding feeder that folds the input ***Component*** or ***Media***. Gluing – A gluing feeder Roll – Roll feeder for Web processes. These are also known as unwinders. New in JDF 1.5 Sheet – Single Sheet feeder. Signature – Single signature feeder.

Table 8.90: Feeder Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Loading ?	NMTOKEN	Specifies the feeder loading. **Values include:** Bundle – Stream feeder, using the output of the **Bundling** process. FanFold – Automatic loading of FanFold ***Media***. Manual – Manual loading of stacks Online – Loaded by a gripper or conveyor. The "Online" value is also applicable for @*FeederType* = "Roll". PrintRoll – Automatic loading of single products from a print Roll, using the output of the **PrintRolling** process.
Opening = "None"	enumeration	Specifies the opening of Signatures: **Allowed values are:** Back – Overfold on back. Front – Overfold on front. None – Signatures are not opened. Sucker – Sucker opening, no overfold.
Position ?	integer	@*Position* of feeder on a collecting and gathering chain in chain movement direction. @*Position* = "0" is first feeder feeding to the collecting and gathering chain. Only one ***Feeder*** can be specified for any given @*Position*. If @*Position* is negative, it specifies the position counted from the back of the chain (e.g., "-1" = last position, "-2" = next to last position, etc.).
Component ?	refelement	Specifies the ***Component*** that SHALL be loaded into this ***Feeder***. This ***Component*** SHALL be an input of the **Feeding** process. Exactly one of ***Component*** or ***Media*** SHALL be specified.
FeederQualityParams ?	element	Definition of the setup and policy for feeding quality.
Media ?	refelement	Specifies the ***Media*** that SHALL be loaded into this ***Feeder***. This ***Media*** SHALL be an input of the **Feeding** process. Exactly one of ***Component*** or ***Media*** SHALL be specified.

8.57.2 FeederQualityParams

The ***FeederQualityParams*** element defines the setup and policy for feeding quality control. It can be specified individually for each ***Feeder***.

Table 8.91: FeederQualityParams Element

NAME	DATA TYPE	DESCRIPTION
BadFeedQuality ?	enumeration	Defines the operation of the bad feed quality control. **Allowed value is from:** ▸ FeedQuality.
BadFeeds ?	integer	Number of consecutive bad feeds until the device stops.
DoubleFeedQuality ?	enumeration	Defines the operation of the double feed quality control. **Allowed value is from:** ▸ FeedQuality.
DoubleFeeds ?	integer	Number of consecutive double feeds until the device stops.
IncorrectComponentQuality ?	enumeration	Defines the operation of the incorrect components quality control: **Allowed value is from:** ▸ FeedQuality.
IncorrectComponents ?	integer	Number of consecutive incorrect components until the device stops.

8.57.3 CollatingItem

Table 8.92: CollatingItem Element

NAME	DATA TYPE	DESCRIPTION
Amount = "1"	integer	Determines, how many consecutive items shall be consumed.
BundleDepth ?	integer	In case of nested bundles with @*BundleType* = "Stack", this parameter addresses the element to be consumed within the "tree" of such bundles. If the real bundle depth level (@*BundleType* = "Stack") is smaller than the value of @*BundleDepth*, individual stack items (i.e., the smallest available level) shall be consumed. If the input component referenced does not contain bundles, then this parameter is ignored. A @*BundleDepth* value of "0" means the ***Component*** itself. A value of "1" addresses the ***BundleItem*** elements referenced from the ***Component*** (i.e., the ***Component***/***Bundle***/***BundleItem***/***Component***(Ref), and so on).
Orientation ?	enumeration	Named @*Orientation* of the ***CollatingItem*** relative to the input coordinate system. For details see ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component. At most one of @*Orientation* or @*Transformation* SHALL be specified. If neither is specified, no transformation is applied. The transformation specified here is applied in addition to orientation/transformation specified in the respective ***ResourceLink***.**Allowed value is from:** ▸ Orientation**.**
Transformation ?	matrix	Orientation of the ***Component*** respective to the input coordinate system. This @*Transformation* specified here is applied in addition to orientation/transformation specified in the respective ***ResourceLink***. At most one of @*Orientation* and @*Transformation* SHALL be specified. If neither is specified, no transformation is applied.
TransformationContext = "StackItem"	enumeration	This parameter specifies the object, which SHALL be manipulated in orientation/transformation, and it is important to determine the sequence of stack items after flipping. **Allowed values are:** StackItem – Apply individually to the smallest element on the stack which can be manipulated individually (e.g., to a single Sheet in the case of a stack of Sheets). Component – Apply to each single element of a ***CollatingItem*** individually. CollateItem – apply to a ***CollatingItem*** as a whole. **Note:** If @*Amount* = "1", ***Component*** and ***CollatingItem*** are referring to the same object and, therefore, result in the same output.
Component ?	refelement	References one of the input components to the process to be (partially) consumed by the ***CollatingItem*** element. This ***Component*** SHALL be an input of the **Feeding** process. Exactly one of ***Component*** or ***Media*** SHALL be specified.
Media ?	refelement	References one of the input media to the process to be consumed by the ***CollatingItem*** element. This ***Media*** SHALL be an input of the **Feeding** process. Exactly one of ***Component*** or ***Media*** SHALL be specified.

Note: Most real world devices process stack items one by one, and hence will hardly ever support @*TransformationContext* = "CollateItem". This requires some kind of buffer for the stack items belonging to a single collating item plus a flipping mechanism for **PrintRolling** process.

8.58 FileSpec

Specification of a file or a set of files. If a single ***FileSpec*** instance specifies a set of files, it SHALL do so using the @*FileFormat* and @*FileTemplate* attributes to specify a sequence of URLs. Otherwise, each ***FileSpec*** instance specifies a single file. If that single file is inside a container file (e.g., a Zip file or is compressed or encoded as indicated by @*Compression*), the ***FileSpec*** instance SHALL define a ***Container*** Subelement which defines another ***FileSpec*** instance that specifies the container file. In such a case, the attributes of each ***FileSpec*** instance SHALL apply only to the properties of the file at that level.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***ApprovalSuccess***/***ApprovalDetails***, ***AssetListCreationParams***, ***ByteMap***, ***Color***, ***Color***/***FileSpec***, ***ColorCorrectionParams***, ***ColorCorrectionParams***/***ColorCorrectionOp***, ***ColorSpaceConversion-***

Op, *ColorSpaceConversionParams*, *DBMergeParams*, *Device*/*IconList*/*Icon*, *DieLayout*, *ElementColorParams*, *ExposedMedia*, *ExternalImpositionTemplate*, *FileSpec*/*Container*, *FileSpec*/*FileAlias*, *FormatConversionParams*/*TIFFFormatParams*/*TIFFEmbeddedFile*, *ImageReplacementParams*, *LayoutElement*, *LayoutElementProductionParams*, *PDLResourceAlias*, *QualityControlResult*, *ScanParams*, *ShapeDef*, *ShapeDefProductionParams*/*ObjectModel*, *ShapeDefProductionParams*/*ShapeTemplate*

Example Partition: "Separation"

Table 8.93: FileSpec Resource (Sheet 1 of 4)

NAME	DATA TYPE	DESCRIPTION
Application ?	string	Creator application. See @*AppVersion* for the application version number.
AppOS ? Modified in JDF 1.2 Deprecated in JDF 1.6	string	Operating system of the application that created the file. **Values include:** DG_UX HP_UX IRIX Linux Mac Solaris Windows **Note:** Additional values can be used from the IANA Operating System Names ▸ [iana-os] which allows up to 40 uppercase US ASCII alphabetical values as well as "-", "_" and "/" — but only for values not covered by the above values. For example, "OS/2". See ▸ Appendix M AppOS and OSVersion Attributes for combinations of @*AppOS* and @*OSVersion* values.
AppVersion ?	string	Version of the value of the @*Application* attribute. **Examples are:** "8.1" "8.1 (4331)" "9.0.3 SR3437"
CheckSum ? New in JDF 1.1 Modified in JDF 1.1A	hexBinary	Checksum of the file being referenced using the RSA MD5 algorithm. In **JDF** 1.1A, the term RSA MD was completed to RSA MD5. The data type was modified to hexBinary to accommodate the 128 bit output of the MD5 algorithm. The @*CheckSum* SHALL be for the entire file, not just parts of the file.
Compression = "None" Modified in JDF 1.2	NMTOKEN	Indicates the compression or encoding for the entire file. This is not compression used internally within the file. **Values include:** Base64 – A format for encoding arbitrary binary information for transmission by electronic mail. ▸ [RFC3548] BinHex – BinHex encoding converts an 8-bit file into a 7-bit format, similar to Uuencoding ▸ [RFC1741]. Compress – UNIX compression ▸ [RFC1977]. Deflate – The file is compressed using Zip public domain compression format ▸ [RFC1951]. Gzip – GNU Zip compression technology ▸ [RFC1952]. MacBinary – A format that combines the two forks of a Mac file, together with the file information into a single binary data stream, suitable for storage or transferring through non-Mac systems. ▸ [macbinary] None – The file is neither compressed nor encoded. UUEncode – A set of algorithms for converting files into a series of 7-bit ASCII characters that can be transmitted over the Internet. ▸ [uuencode] ZLIB – ZLIB compression ▸ [RFC1950].

Table 8.93: FileSpec Resource (Sheet 2 of 4)

NAME	DATA TYPE	DESCRIPTION
Disposition ? Deprecated in JDF 1.2	enumeration	Indicates what the device SHALL do with the file when the process that uses this resource as an Input resource completes. **Allowed values are:** Unlink – The device SHALL release the file. Delete – The device SHALL attempt to delete the file. Retain – The device SHALL do nothing with the file. **Deprecation note:** Starting with **JDF** 1.2, retention of assets is specified in the ***Disposition*** resource.
DocumentNaturalLang ?	language	The natural language of the document this ***FileSpec*** refers to. If the document contains more than one language, the value is the primary language of the document.
Encoding ? New in JDF 1.4	string	Encoding or code page of the file contents. **Values include those from:** IANA Character Set Names see ▸ [iana-character sets].
FileFormat ?	string	A formatting string used with the @*FileTemplate* attribute to define a sequence of URLs in a batch process, each of which has the same semantics as the @*URL* attribute. **Allowed values are from:** ▸ Appendix H Dynamic String Generation. **Constraint:** if neither @*URL* nor @*UID* is present, both @*FileFormat* and @*FileTemplate* SHALL be present, unless the resource is a pipe. If either @*URL* or @*UID* is specified, then @*FileFormat* and @*FileTemplate* SHALL NOT be specified.
FileSize ? Modified in JDF 1.2	LongInteger	Size of the file in bytes. The data type was changed from integer to LongInteger in **JDF** 1.2.
FileTargetDeviceModel ? New in JDF 1.2	string	Identifies the model of the **JDF** device for which the document was formatted, including manufacturer name, when the file is device-dependent. **Default behavior:** the file is device independent **Value format is from:** IEEE 1284-2000 Device ID string. **Note:** The value of this attribute SHALL exactly match the IEEE 1284-2000 Device ID string, except the length field SHALL NOT be specified. See the Microsoft Universal Plug-and-Play ▸ [UPNP] section *2.2.6 DeviceId parameter* for details. **Example:** It shows only the REQUIRED fields for a PostScript document formatted for a *LaserBeam 9*: `MANUFACTURER:ACME Co.;COMMAND SET:PS;MODEL:LaserBeam 9;` (See ▸ [IEEE1284] clause 7.6).
FileTemplate ?	string	A template, used with @*FileFormat*, to define a sequence of URLs in a batch process, each of which has the same semantics as the @*URL* attribute. **Constraint:** if neither @*URL* nor @*UID* is present, both @*FileFormat* and @*FileTemplate* SHALL be present, unless the resource is a pipe. **Allowed values are from:** ▸ Appendix H Dynamic String Generation.
FileVersion ? New in JDF 1.1	string	Version of the file referenced by this ***FileSpec***.
MimeType ? Modified in JDF 1.2	string	MIME type or file type of the file (or files of identical type when specifying a sequence of file names using the @*FileFormat* and @*FileTemplate* attributes). See @*Compression* for the indication of compression or encoding of the file. See @*MimeTypeVersion* for the format version. If the file format has a MIME Media Type ▸ [iana-mt] registered with IANA, that value SHALL be used. The ▸ [RFC2046] defines that MIME Media Types are case-insensitive. If the file format does not have a MIME Media Type registered with IANA, then the **JDF** spec defines string values, called file types, which SHALL be used. **Values include those from:** ▸ Appendix G MimeTypes.

Table 8.93: FileSpec Resource (Sheet 3 of 4)

NAME	DATA TYPE	DESCRIPTION
MimeTypeVersion ? New in JDF 1.2	string	The level or version of the file format identified by @*MimeType*, whether the value of @*MimeType* is a MIME Media Type or a file type value defined by the **JDF** spec. Example values include: "PDF/1.3", "PDF/1.4" and "PDF/X-1a:2001" for @*MimeType* = "application/pdf" "TIFF-IT/FP:1998", "TIFF-IT/CT:1998" and "TIFF-IT/LW/P1:1998" for @*MimeType* = "TIFF/IT" **Values include those from:** ▸ Appendix G MimeTypes**.**
OSVersion ? Modified in JDF 1.2 Deprecated in JDF 1.6	string	Version of the operating system specified by @*AppOS*. The IANA Registry provides a list. **Values include those from:** ▸ Table M.2 AppOS and OSVersion Examples**.**
OverwritePolicy ? New in JDF 1.2	enumeration	Policy that specifies the policy to follow when a file already exists and the ***FileSpec*** is used as an output resource. **Allowed values are:** Overwrite – Overwrite the old file. RenameNew – Rename the new file. RenameOld – Rename the old file. NewVersion – Create a new file version. Only valid when the ***FileSpec*** references a file on a version aware file system. OperatorIntervention – Present a dialog to an operator. Abort – Abort the process without modifying the old file.
PageOrder ?	enumeration	Indicates the order of pages in the file containing pages. **Allowed values are:** Ascending – The first page in the file is the lowest numbered page. Descending – The first page in the file is the highest numbered page.
Password ? New in JDF 1.3	string	Password or decryption key that is needed to read the file contents. **Note:** Since this password string is not encrypted, it SHOULD only be passed around within a protected environment.
RequestQuality ? New in JDF 1.3	double	@*RequestQuality* specifies a requested quality of the encoded data when reading image data with selected @*MimeType* values which support variable quality. @*RequestQuality* is ignored when the ***FileSpec*** is referenced from an Output resource or the ***FileSpec*** does not reference image data which support variable quality. The value in the range of 0 to 1.0 represents a factor of the maximum quality encoded in the file. If left unspecified, the value defaults to 1.0 meaning all information encoded will be returned. The following details how values are interpreted for the supported @*MimeType* values: "image/jp2", "image/jpx" – The value represents the ratio of the encoding bitrate of the maximum bitrate layer encoded in the file. "image/gif" – The number represents a ratio of the total interleaved layers of the file. **Note:** Only interleaved GIF. "image/tiff" – The number represents the ratio of the total resolution of the complete image. **Note:** Only pyramid TIFF.
ResourceUsage ?	NMTOKEN	If an element uses more than one ***FileSpec*** subelement, this attribute is used to refer from the parent element to a certain child element of this type, for example, see ***FormatConversionParams***. **Values include those from:** ▸ Table 8.94 ResourceUsage Attribute Values**.**
SearchDepth ? New in JDF 1.2	integer	Used when @*ResourceUsage* = "SearchPath" to specify the maximum directory depth that will be recursively searched. 0 specifies this directory only, "INF" specifies an unlimited search.

Table 8.93: FileSpec Resource (Sheet 4 of 4)

NAME	DATA TYPE	DESCRIPTION
UID ? New in JDF 1.1	string	Internal ID of the referenced file. The @*UID* SHALL be unique within the workflow. This attribute is dependent on the type of file that is referenced: **Values include:** PDF – Variable unique identifier in the ID field of the PDF file's trailer. ICC Profile – The Profile ID in bytes 84-99 of the ICC profile header. Others – Format specific. **Constraint:** If neither @*URL* nor @*UID* is present on an input ***FileSpec***, and neither @*FileFormat* nor @*FileTemplate* is present, the referencing resource SHALL be a pipe. If either @*URL* or @*UID* is specified, then @*FileFormat* and @*FileTemplate* SHALL NOT be specified.
URL ?	URL	Location of the file specified as either an Absolute URI or a Relative URI. If neither @*URL* nor @*UID* is present on an input ***FileSpec***, and neither @*FileFormat* nor @*FileTemplate* is present, the referencing resource SHALL be a pipe. If either @*URL* or @*UID* is specified, then @*FileFormat* and @*FileTemplate* SHALL NOT be specified. If @*URL* is not specified in an Output resource, the system-specified location will be assumed, but this value SHALL be updated as soon as the Output resource is available. For example, an instruction for a digital delivery **JDF** device to compress the files MAY specify the output ***RunList*** with the @*Compression* attribute without the @*URL* attribute. See ▸ [RFC3986] and ▸ Appendix J Resolving Directory URL References and ▸ Appendix M FileSpec Use Cases for the syntax and examples. For the "file" URL scheme see also ▸ [RFC1738] and ▸ [FileURL].
UserFileName ?	string	A user-friendly name which can be used to identify the file. MAY be used by an agent to identify a file on a device without knowing the file's internal location.
Container ? New in JDF 1.2	refelement	Specifies the container for this file. When a container ***FileSpec*** is pointed to by ***Container***, that ***FileSpec*** SHALL NOT also specify @*FileFormat* and @*FileTemplate* attributes. The container mechanism MAY be used recursively (e.g., for a Zip file held in a tar file, a Zip file in a Zip file, an encoded Zip file, etc.). See ▸ Appendix J Resolving Directory URL References for details.
Disposition ? New in JDF 1.2	element	Indicates what the device SHOULD do with the file when the process that uses this resource completes. If not specified here or in the parent ***RunList***, the file specified by this ***FileSpec*** SHOULD NOT be deleted by the device. If ***FileSpec***/***Disposition*** is specified, it takes precedence over ***RunList***/***Disposition***.
FileAlias *	element	Defines a set of mappings between file names that can occur in the document and URLs (which can refer to external files or parts of a MIME message).

8.58.1 ResourceUsage Attribute Values

Table 8.94: ResourceUsage Attribute Values (Sheet 1 of 2)

VALUE	DESCRIPTION
AbstractProfile	Used for ***ColorCorrectionOp***/***FileSpec*** and ***ColorSpaceConversionOp***/***FileSpec***
ActualOutputProfile	Used for ***ElementColorParams***/***FileSpec***
ColorProfile	Used in ***Color***/***FileSpec***
CorrectionProfile	Used for ***ScanParams***/***FileSpec***
DeviceLinkProfile	Used for ***ColorCorrectionOp***/***FileSpec*** and ***ColorSpaceConversionOp*** /***FileSpec***.
FinalTargetDevice	Used for ***ColorCorrectionParams***/***FileSpec*** and ***ColorSpaceConversionParams***/***FileSpec***
InputFormat	Used for ***FormatConversionParams***/***FileSpec***

Table 8.94: ResourceUsage Attribute Values (Sheet 2 of 2)

VALUE	DESCRIPTION
OutputFormat	Used for ***FormatConversionParams***/***FileSpec***
OutputProfile	Used for ***ExposedMedia***/***FileSpec***
RasterFileLocation	Used for ***ByteMap***/***FileSpec***
ReferenceOutputProfile	Used for ***ElementColorParams***/***FileSpec***
ScanProfile	Used for ***ScanParams***/***FileSpec***
SearchPath	Used for ***AssetListCreationParams***/***FileSpec*** and ***ImageReplacementParams***/***FileSpec***.
SourceProfile	Used for ***ColorSpaceConversionOp***/***FileSpec***
TargetProfile	Used for ***Color***/***FileSpec***, ***Color***/***PrintConditionColor***/***FileSpec***, ***ScanParams***/***FileSpec***.
WorkingColorSpace	Used for ***ColorCorrectionParams***/***FileSpec*** and ***ColorSpaceConversionParams***/***FileSpec***

8.58.2 Container

New in JDF 1.2

The ***Container*** specifies the containing file for a ***FileSpec*** (e.g., a zip file or tar archive). The ***Container*** elements MAY be specified recursively in their respective child ***FileSpec*** elements.

Table 8.95: Container Element

NAME	DATA TYPE	DESCRIPTION
FileSpec	refelement	Link to another ***FileSpec*** resource that describes the container (e.g., a packaging file, such as Zip, Multipart/Related, tar file or an otherwise compressed or encoded file that contains the file represented by this ***FileSpec*** resource). The link value is only to be used for locating that container ***FileSpec*** resource. See ▸ Appendix J Resolving Directory URL References for details.

8.58.3 FileAlias

Table 8.96: FileAlias Element

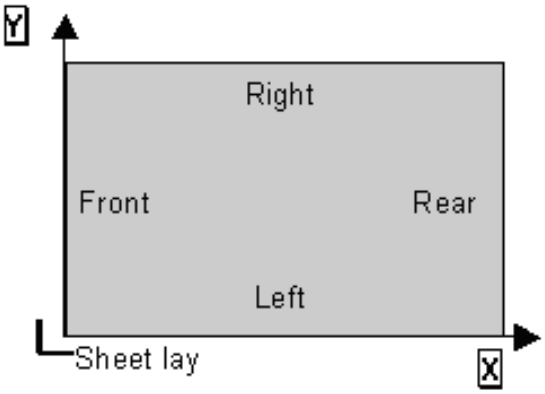
NAME	DATA TYPE	DESCRIPTION
Alias	string	The filename which is expected to occur in the file.
Disposition ? Deprecated in JDF 1.2	enumeration	Indicates what the device SHALL do with the file referenced by this alias when the process that uses this resource as an Input resource completes. **Allowed values are:** Unlink – The device SHALL release the file. Delete – The device SHALL attempt to delete the file. Retain – The device SHALL do nothing with the file. **Deprecation note:** Starting with **JDF** 1.2, use ***FileSpec***/***Disposition***.
MimeType ? Deprecated in JDF 1.2	string	MIME type of the file. **Deprecation note:** Starting with **JDF**/1.2, use ***FileSpec***/@*MimeType*.
RawAlias ? New in JDF 1.2	hexBinary	Representation of the original 8-bit byte stream of the Alias Name. Used to transport the original byte representation of an Alias name when moving **JDF** tickets between computers with different locales.
URL ? Deprecated in JDF 1.2	URL	The URL which identifies the file the alias refers to. In **JDF**/1.2 and beyond, use ***FileSpec***/@*URL*.
FileSpec ? New in JDF 1.2	refelement	For **JDF** version 1.2 and beyond, ***FileSpec*** SHALL be present, and SHALL contain a @*URL* attribute. ***FileSpec*** MAY contain additional properties of the file (e.g., ***Disposition***, @*MimeType*, @*MimeTypeVersion*, etc.).

8.59 FoldingParams

This resource describes the folding parameters, including the sequence of folding steps. It is also possible to execute the predefined steps of the folding catalog. After each folding step of a folding procedure, the origin of the coordinate system SHALL be moved to the lower left corner of the intermediate folding product. For details see ▸ Section 2.5.5 Product Example: Simple Brochure.

The specification of @SheetLay and reference edges (i.e., "Front", "Rear", "Left" and "Right") for the description of an operation (e.g., the positioning of a tool) is done by means of determined names as shown in ▸ Figure 8-32: Names of the reference edges of a Sheet in the FoldingParams Resource below.

Figure 8-32: Names of the reference edges of a Sheet in the FoldingParams Resource



Resource Properties

Resource Class: Parameter

Intent Pairing: ***FoldingIntent***

Example Partition: "BlockName", "RibbonName", "SheetName", "SignatureName", "WebName"

Input of Processes: **Folding**

Table 8.97: FoldingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
DescriptionType ? Deprecated in JDF 1.2	enumeration	How the folding operations are described. **Allowed values are:** FoldProc – Detailed description of each individual fold. FoldCatalog – Selection of fold procedure from *FoldCatalog*. **Deprecation note:** Starting with **JDF** 1.2, the *FoldCatalog* defines the topology of the folding scheme. The specifics of each individual fold can be described using ***Fold*** elements. If both *FoldCatalog* and ***Fold*** are specified, ***Fold*** takes precedence
FoldCatalog ? Modified in JDF 1.4	string	Describes the type of fold according to the folding catalog in ▸ Figure 8-33: Fold catalog part 1 and ▸ Figure 8-34: Fold catalog part 2. In case of any ambiguity, the folding notation takes precedence over the graphic illustration in the aforementioned Figures. **Value format is:** "Fn-i" where "n" is the number of finished pages and "i" is either an integer, which identifies a particular fold or the letter "X", which identifies a generic fold (e.g., "F6-2" describes a Z-fold of 6 finished pages, and "F6-X" describes a generic fold with 6 finished pages). **Modification note:** Starting with **JDF** 1.4, the letter "X" is added for a generic fold.
FoldingDetails ? New in JDF 1.6	NMTOKEN	@*FoldingDetails* is a system dependent descriptor of the folding. @*FoldingDetails* MAY be used to differentiate differing fold dimensions with the same general topology.
FoldSheetIn ? Deprecated in JDF 1.1	XYPair	Input Sheet format. If the specified size does not match the size of the X and Y dimensions of the input ***Component***, all coordinates of the folding procedure are scaled accordingly. The scaling factors in X and Y direction MAY differ. **Implementation Note:** This attribute SHOULD always match the @*Size* attribute of the input ***Component***, which is the default.

Table 8.97: FoldingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
SheetLay = "Left"	enumeration	Lay of input media. **Allowed value is from:** ▸ SheetLay. **Note:** @*SheetLay* does not modify the coordinate references of the **Folding** process.
FileSpec (CIP3) ? New in JDF 1.5	refelement	Reference to a CIP3 file that contains folding instructions in the ▸ [CIP3 - PPF] format.
Fold * New in JDF 1.1	element	Describes the folding operations in the sequence in which they are to be carried out. It is RECOMMENDED to specify a set of subsequent ***Fold*** operations as multiple ***Fold*** elements in one **Folding** procedure, rather than specifying a combined process that combines multiple **Folding** processes. If both @*FoldCatalog* and ***Fold*** elements are specified, the ***Fold*** elements have precedence, and the @*FoldCatalog* specifies only the topology. For instance a cover-fold with a page size ratio of 0.52 to 0.48 would still be defined as an "F4-1".
FoldOperation * Deprecated in JDF 1.1	element	Abstract element that describes the folding operations in the sequence in which they are to be carried out. Replaced by the explicit ***Fold*** element in **JDF** 1.1 and beyond.

Figure 8-33: Fold catalog part 1

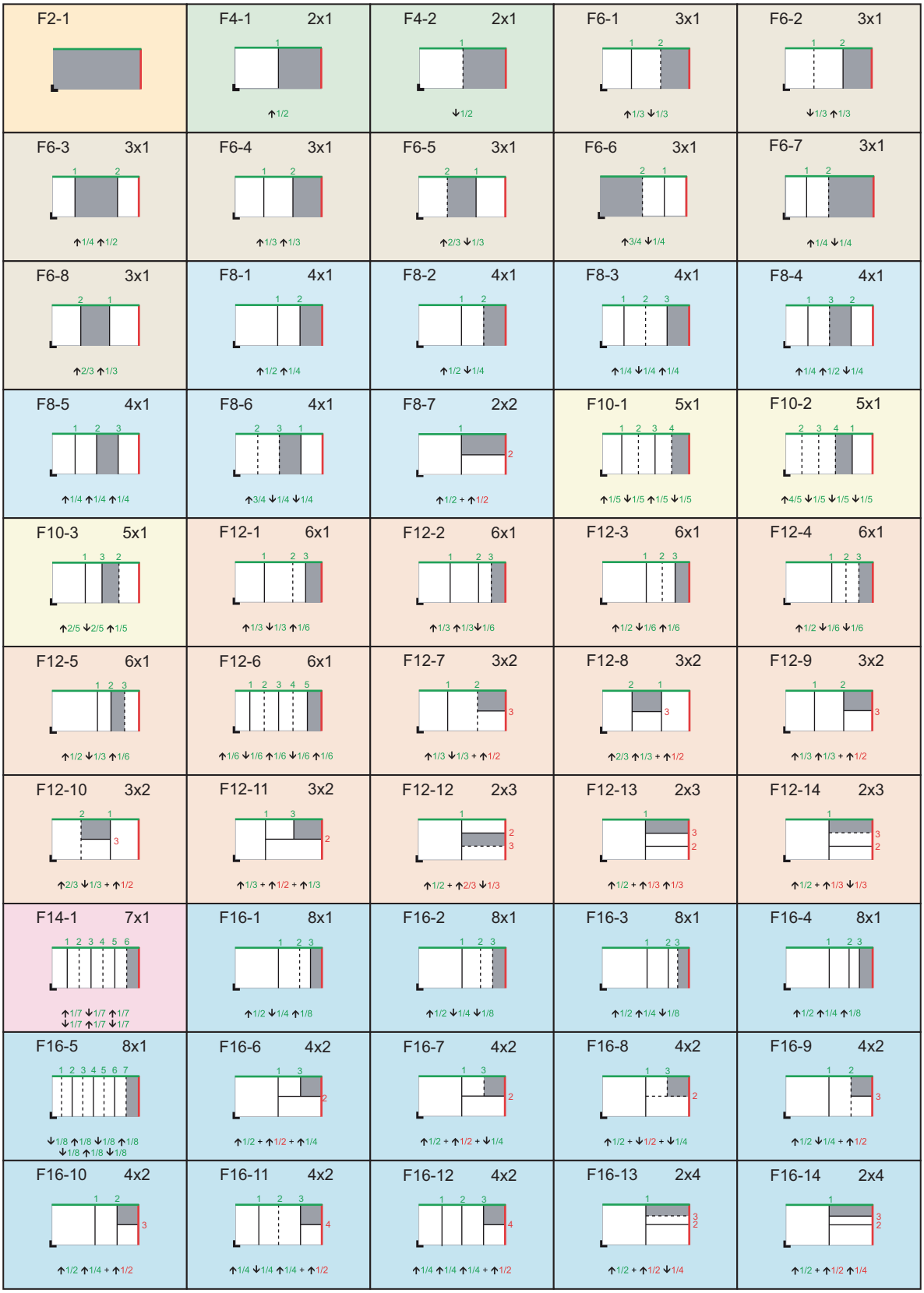
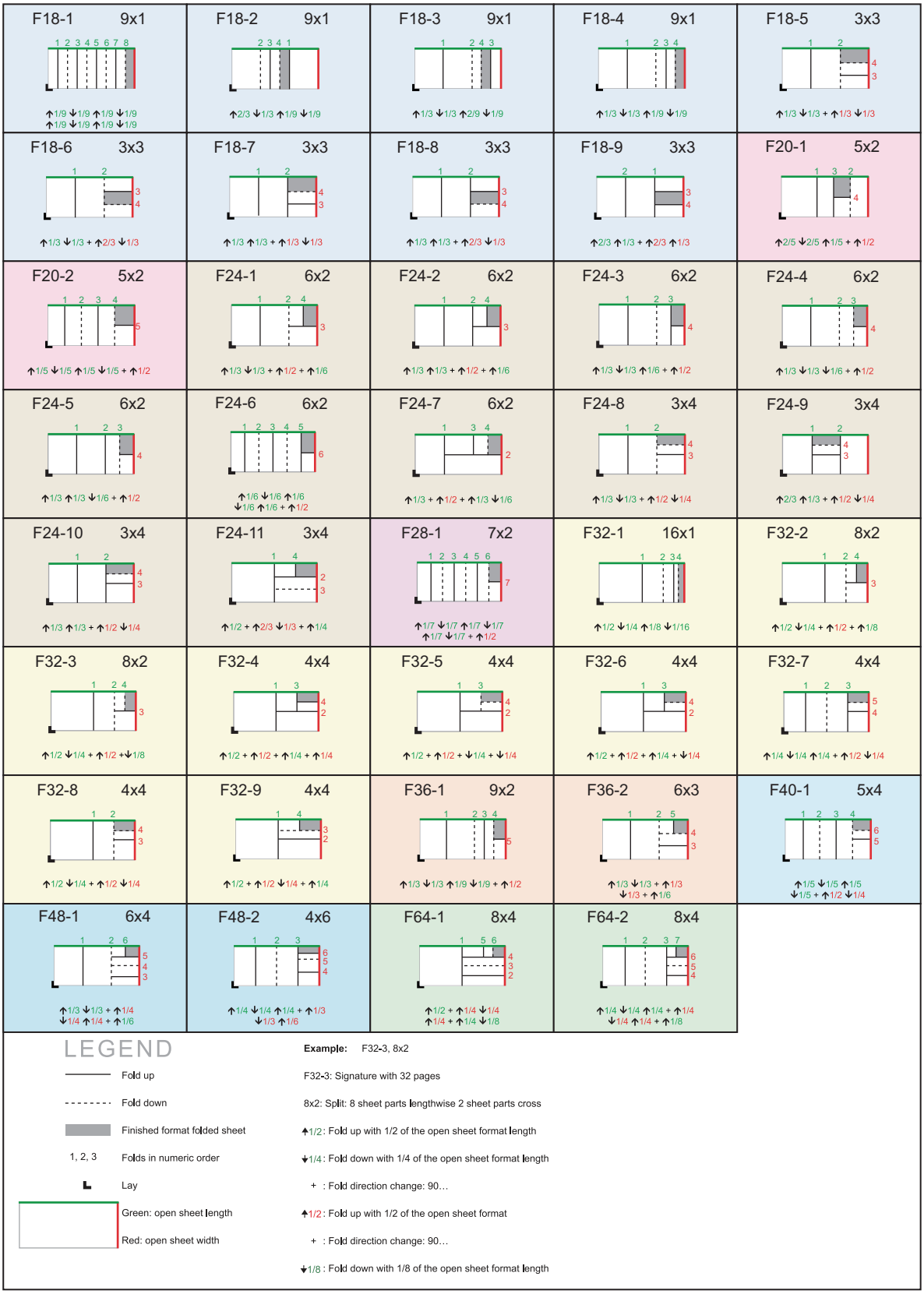


Figure 8-34: Fold catalog part 2



8.60 FontParams

This resource describes how fonts are handled when converting PostScript or other PDL files to PDF.

Resource Properties

Resource Class: Parameter

Example Partition: "DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetName", "Side", "SignatureName"

Input of Processes: **PSToPDFConversion**

Table 8.98: FontParams Resource

NAME	DATA TYPE	DESCRIPTION
AlwaysEmbed ?	NMTOKENS	One or more names of fonts that are always to be embedded in the PDF file. Each name SHALL be the PostScript language name of the font. An entry that occurs in both the @*AlwaysEmbed* and @*NeverEmbed* lists constitutes an error.
CannotEmbedFontPolicy = "Warning"	enumeration	Determines what occurs when a font cannot be embedded. **Allowed values are:** Error – Log an error and abort the process if any font can not be found or embedded. Warning – Warn and continue if any font cannot be found or embedded. OK – Continue without warning or error if any font can not be found or embedded.
EmbedAllFonts = "false"	boolean	If "true", specifies that all fonts, except those in the @*NeverEmbed* list, are to be embedded in the PDF file.
MaxSubsetPct ?	integer	The maximum percentage of glyphs in a font that can be used before the entire font is embedded instead of a subset. This value is only used if @*SubsetFonts* = "true".
NeverEmbed ?	NMTOKENS	One or more names of fonts that are never to be embedded in the PDF file. Each name SHALL be the PostScript language name of the font. An entry that occurs in both the @*AlwaysEmbed* and @*NeverEmbed* lists constitutes an error.
SubsetFonts ?	boolean	If "true", font subsetting is enabled. If "false", it is not. Font subsetting embeds only those glyphs that are used, instead of the entire font. This reduces the size of a PDF file that contains embedded fonts. If font subsetting is enabled, the decision whether to embed the entire font or a subset is determined by number of glyphs in the font that are used and the value of @*MaxSubsetPct*. **Note:** Embedded instances of multiple master fonts are always subsetted, regardless of the setting of @*SubsetFonts*.

8.61 FontPolicy

This resource defines the policies that Devices follow when font errors occur while PDL files are being processed. When fonts are referenced by PDL files but are not provided, Devices SHALL provide one of the following two fallback behaviors:

1 The device provides a standard default font which is substituted whenever a font cannot be found.

2 The device provides an emulation of the missing font.

If neither fallback behavior is requested (i.e., both @*UseDefaultFont* and @*UseFontEmulation* are "false"), then the Job will fail if a referenced font is not provided. The ***FontPolicy*** allows Jobs to specify whether either of these fallback behaviors are to be employed when missing fonts occur.

Resource Properties

Resource Class: Parameter

Example Partition: "DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetName", "Side", "SignatureName"

Input of Processes: **Interpreting**, **Trapping**

Table 8.99: FontPolicy Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
PreferredFont	NMTOKEN	The name of a font to be used as the default font for this job. It is not an error if the device cannot use the specified font as its default font.

Table 8.99: FontPolicy Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
UseDefaultFont	boolean	If "true", the device SHALL resort to a default font if a font cannot be found. This is the normal behavior of the PostScript interpreter, which defaults to Courier when a font cannot be found.
UseFontEmulation	boolean	If "true", the device SHALL emulate a requested font if a font cannot be found.

8.62 FormatConversionParams

New in JDF 1.1

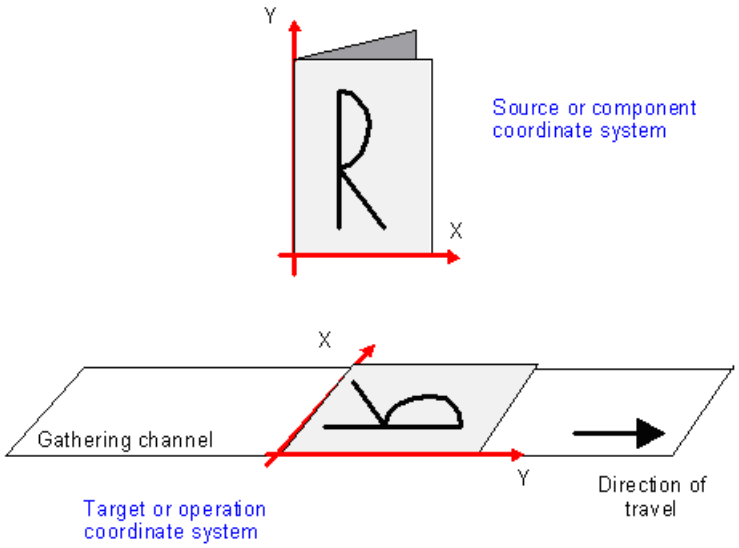
Deprecated in JDF 1.5

See ▸ Section N.7.9 FormatConversionParams for details. For ***TIFFFormatParams***, see . ▸ Section 8.127.1 TIFFFormatParams

8.63 GatheringParams

This resource contains the attributes of the **Gathering** process.

Figure 8-35: Coordinate system used for Gathering



Resource Properties

Resource Class: Parameter

Input of Processes: **Gathering**

Table 8.100: GatheringParams Resource

NAME	DATA TYPE	DESCRIPTION
Disjointing ? Deprecated in JDF 1.6	element	Description of the separation properties between individual components on a gathered pile. The default case is that no physical separation between components is used and this element is omitted. **Deprecation note:** Starting with **JDF** 1.6, use a combined Stacking process.

8.64 GlueApplication

New in JDF 1.1

This resource specifies glue application in hard and soft cover book production.

Resource Properties

Resource Class: Parameter

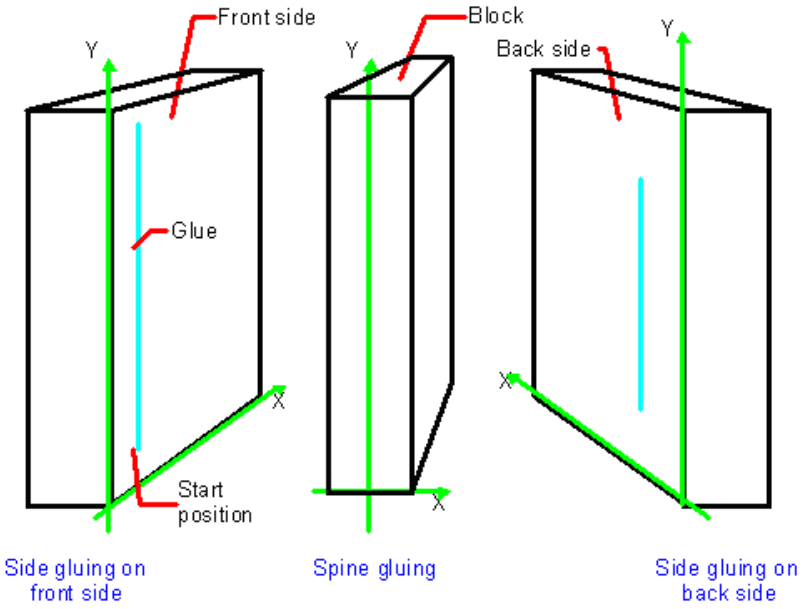
Resource referenced by: ***CasingInParams***, ***CoverApplicationParams***, ***GluingParams***/***Glue***, ***SpineTapingParams***

Intent Pairing: ***BindingIntent***

Table 8.101: GlueApplication Resource

NAME	DATA TYPE	DESCRIPTION
GluingTechnique	enumeration	Type or technique of gluing application. **Allowed values are:** SpineGluing SideGluingFront SideGluingBack
GlueLine	element	Structure of the glue line.

Figure 8-36: Parameters and coordinate system for glue application



8.65 GluingParams

New in JDF 1.1

GluingParams define the parameters applying a generic line of glue to a component.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Example Partition: "WebName", "WebProduct"

Input of Processes: **Gluing**

Table 8.102: GluingParams Resource

NAME	DATA TYPE	DESCRIPTION
GluingProductionID ? New in JDF 1.3	string	Defines a gluing scheme for production.
GlueLine *	element	Definition of one or more ***GlueLine*** line applications.

8.65.1 Glue

The ***Glue*** element describes how to apply a line of glue.

Table 8.103: Glue Element

NAME	DATA TYPE	DESCRIPTION
WorkingDirection ? Modified in JDF 1.5	enumeration	Direction from which the tool is working. **Allowed value is from:** ▸ WorkingDirection. **Modification Note:** Starting in **JDF** 1.5, @*WorkingDirection* is optional.
GlueApplication ? Modified in JDF 1.3	refelement	Describes the glue application. Exactly one of ***GlueApplication*** or ***GlueLine*** SHALL be specified.
GlueLine ? New in JDF 1.3	element	Structure of the GlueLine used for generic gluing. Exactly one of ***GlueApplication*** or ***GlueLine*** SHALL be specified.

8.66 HeadBandApplicationParams

New in JDF 1.1

This resource specifies how to apply headbands in hard cover book production.

Resource Properties

Resource Class: Parameter

Input of Processes: **HeadBandApplication**

Table 8.104: HeadBandApplicationParams Resource

NAME	DATA TYPE	DESCRIPTION
BottomBrand ?	string	Bottom head band brand. If not specified, defaults to the value of @*TopBrand*.
BottomColor ?	NamedColor	Color of the bottom head band. If not specified, defaults to the value of @*TopColor*.
BottomColorDetails ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If @*BottomColorDetails* is supplied, @*BottomColor* SHOULD also be supplied.
BottomLength ?	double	Length of the carrier material of the bottom head band along binding edge. If not specified, both head bands are on one carrier.
StripMaterial ?	enumeration	Strip material. **Allowed value is from:** ▸ StripMaterial.
TopBrand ?	string	Top head band brand.
TopColor ?	NamedColor	Color of the top head band.
TopColorDetails ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If @*TopColorDetails* is supplied, @*TopColor* SHOULD also be supplied.
TopLength ?	double	Length of carrier material of the top head band along binding edge. If not specified, both head bands are on one carrier which has the length of the book block.
Width ?	double	Width of the head bands and carrier.
GlueLine *	element	The carrier can be applied to the book block with glue. The coordinate system for the ***GlueLine*** is defined in the ▸ Section 8.54 EndSheetGluingParams.

8.67 HoleList

This resource is used to describe holes or rows of holes in ***Intent Resource***s or ***Media***. Note that it was an Intent resource Subelement prior to **JDF** 1.2.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***BindingIntent***/***CoilBinding***, ***BindingIntent***/***PlasticCombBinding***, ***BindingIntent***/***StripBinding***, ***BindingIntent***/***WireCombBinding***, ***BindingIntent***/***BindList***/***BindItem***/***CoilBinding***, ***BindingIn-***

tent//*BindItem*/*PlasticCombBinding*, *BindingIntent*/*BindList*/*BindItem*/*StripBinding*, *BindingIntent*/*BindList*/*BindItem*/*WireCombBinding*, *HoleMakingIntent*, *Media*

Table 8.105: HoleList Resource

NAME	DATA TYPE	DESCRIPTION
Hole * Modified in JDF 1.1	element	Description of individual holes. See ▸ Section 9.25 Hole.
HoleLine * New in JDF 1.1	element	Array of all *HoleLine* elements. See ▸ Section 9.26 HoleLine.

8.68 HoleMakingParams

This resource specifies where to make a hole of what shape in components. This information is used by the **HoleMaking** process.

Default behavior for @*HoleCount*: For dealing with the Default case of @*HoleCount* (i.e., when not supplied), intelligent systems will take into consideration Job parameters like the length of the binding edge or distance of holes to the paper edges to calculate the appropriate number of holes. For production of the holes and selection/production of the matching binding element, the "system specified" values need to match 100% between the **HoleMaking** and the binding process for obvious reasons. In practice, if no details are specified for **HoleMaking**, they SHOULD also be absent for binding. In this case, either the operator provides the missing value when setting up the binding device for the Job, or the device itself needs to have some kind of automatic hole detection mechanism.

Resource Properties

Resource Class: Parameter

Resource referenced by: *CoilBindingParams*, *PlasticCombBindingParams*, *RingBindingParams*, *StripBindingParams*, *WireCombBindingParams*

Intent Pairing: *HoleMakingIntent*

Example Partition: "SheetName", "SignatureName"

Input of Processes: **HoleMaking**

Table 8.106: HoleMakingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Center ? Modified in JDF 1.1	XYPair	Position of the center of the hole pattern relative to the *Component* coordinate system if @*HoleType* is not "Explicit". If not specified, it defaults to the value implied by @*HoleType*.
CenterReference = "TrailingEdge" New in JDF 1.1	enumeration	Defines the reference coordinate system for @*Center*. **Allowed values are:** TrailingEdge – Physical coordinate system of the component. RegistrationMark – The center is relative to a registration mark.
Extent ?	XYPair	Size (Bounding Box) of the hole in points if @*HoleType* is not "Explicit". If @*Shape* is "Round", only the first entry of @*Extent* is evaluated and defines the hole diameter. If not specified, it defaults to the value implied by @*HoleType*.
HoleCount ? New in JDF 1.2	IntegerList	For patterns with @*HoleType* whose enumeration values begin with a "P", "W" or "C", this parameter specifies the number of consecutive holes and spaces. The first entry defines the number of holes, the second entry defines the number of spaces, and consecutive entries alternately define holes (h) and spaces (s), for instance: "2 2 2" = "h h s s h h". "0 3 3 3 3" = "s s s h h h s s s h h h". Note: @*HoleCount* is typically applied to patterns with @*HoleType* whose enumeration values begin with a "P", "W" or "C" in ▸ Table L.1 Naming Scheme for Hole Patterns. **Default behavior:** see "**Default behavior for** @*HoleCount*"

Table 8.106: HoleMakingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
HoleReferenceEdge ? New in JDF 1.1 Deprecated in JDF 1.2	enumeration	The edge of the media relative to where the holes are to be punched. Use with @*HoleType*. **Default value:** if @*HoleType* is "Explicit", "Pattern"; otherwise "Left". **Allowed values are:** Left Right Top Bottom Pattern – Specifies that the reference edge implied by the value of @*HoleType* in ▸ Appendix L Hole Pattern Catalog is used. **Deprecation note:** Starting with **JDF** 1.1, use an explicit @*Transformation* or @*Orientation* of the input ***Component***. If either @*Transformation* or @*Orientation* along with @*HoleReferenceEdge* is specified, the result is the matrix product of both transformations. @*Transformation* or @*Orientation* SHALL be applied first.
HoleType New in JDF 1.1	enumerations	Predefined hole pattern. Multiple hole patterns are specified as one NMTOKENS string (e.g., 3-hole ring binding and 4-hole ring binding holes on one piece of media). **Allowed values are:** Explicit – Holes are defined in an array of Hole elements. 2HoleEuro – Replaced by either *R2m-DIN* or *R2m-ISO*. Deprecated in JDF 1.0 3HoleUS – Replaced by *R3I-US*. Deprecated in JDF 1.0 4HoleEuro – Replaced by either *R4m-DIN-A4* or *R4m-DIN-A5*. Deprecated in JDF 1.0 **Allowed values are from:** ▸ Appendix L Hole Pattern Catalog**.**
Shape ? Modified in JDF 1.1	enumeration	Shape of the holes if @*HoleType* is not "Explicit". **Default value is:** value implied by @*HoleType*. **Allowed values are:** Elliptic Round Rectangular
Hole *	element	Description of individual ***Hole*** elements.
HoleLine * New in JDF 1.1	element	Description of ***HoleLine*** elements.
RegisterMark ? New in JDF 1.1	refelement	Reference to the registration mark that defines the coordinate system origin for **HoleMaking**.

8.69 IdentificationField

This resource contains information about a mark on a document (e.g., a bar code) used for OCR-based verification purposes or document separation.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***Abstract PhysicalResource***, ***Disjointing***, ***EmbossingParams***/***Emboss***, ***Layout***/***MarkObject***, ***LayoutElementProductionParams***/***LayoutElementPart***/***BarcodeProductionParams***

Input of Processes: Verification

Table 8.107: IdentificationField Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BoundingBox ?	rectangle	Box that provides the boundaries of the mark that indicates where the ***IdentificationField*** is placed. If the ***IdentificationField*** is specified in a ***Layout***, the coordinate system is defined by the ***MarkObject*** containing the ***IdentificationField***. If no ***Layout*** context is available, the origin of the coordinate system is defined as the lower left corner of the resource surface that @*Position* specifies when the specified surface is viewed in its natural orientation. Each item in the list below specifies a value of @*Position* and the corner that is the origin for the specified value when the viewer is positioned in front of the front surface. For example, when @*Position* = "Left", the origin is the bottom-back corner of left surface when viewed from the front surface of the resource and lower-left corner when viewed from the left surface. • "Front" – Bottom-left corner • "Left" – Bottom-back corner • "Back" – Bottom-right corner • "Right" – Bottom-front corner • "Top" – Front-left corner • "Bottom" – Back left corner If no @*BoundingBox* is defined and the ***IdentificationField*** is specified • outside the context of a ***Layout***, the complete visible surface SHALL be scanned for an appropriate bar code. • within the context of a ***Layout***, the implied @*BoundingBox* is specified by ***MarkObject***/@*ClipBox*. The @*BoundingBox* is used only as metadata when searching or scanning ***IdentificationField*** elements and not used when generating ***IdentificationField*** elements in a **LayoutElementProduction** process. **Modification note:** Starting with **JDF** 1.4, all text is new.
Encoding ? Modified in JDF 1.4	enumeration	Encoding of the information. **Allowed values are:** ASCII – Plain-text font. Barcode – Any bar code. New in JDF 1.3 BarCode1D – One-dimensional bar code. Deprecated in JDF 1.3 BarCode2D – Two-dimensional bar code. Deprecated in JDF 1.3 Braille – Braille text. New in JDF 1.4 RFID – Radio Frequency Identification tag. New in JDF 1.3
EncodingDetals ?	NMTOKEN	Details about the encoding type. An example is the bar code scheme. **Values include those from:** ▸ Table 8.108 EncodingDetails Attribute Values**.**
Format ? Modified in JDF 1.2	regExp	Regular expression that defines the expected format of the expression (e.g., the number of digits, alphanumeric or numeric). Note that this field MAY also be used to define constant fields (e.g., the end of document markers or packaging labels). If not specified, any expression is valid. Exactly one of @*Format*, @*Value* or the pair @*ValueFormat* and @*ValueTemplate* SHALL be specified.
Orientation ?	matrix	Orientation of the contents within the ***IdentificationField***. The coordinate system is defined in the system of the sheet or component where the ***IdentificationField*** resides. The @*Orientation* is used only as metadata when searching or scanning ***IdentificationField*** elements and not used when generating ***IdentificationField*** elements in a **LayoutElementProduction** process.
Page ?	integer	If @*Position* = "Page", this refers to the page where the ***IdentificationField*** can be found. Negative values denote an offset relative to the last page in a stack of pages.

Table 8.107: IdentificationField Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Position ?	enumeration	Position with respect to the instance document or ***PhysicalResource*** to which the resource refers. **Allowed values are:** Header – Sheet before the document. Trailer – Sheet after the document. Page – A page of the document. Top – The top of the resource. Bottom – The bottom of the resource. Left – The left side of the resource. Right – The right side of the resource. Front – The front side of the resource. Back – The back side of the resource. Any – Deprecated in JDF 1.2
Purpose ?	enumeration	Purpose defines the usage of the field. **Allowed values are:** Label – Used to mark a product or component. Separation – Used to separate documents. Verification – Used for verification of documents.
PurposeDetails ? New in JDF 1.3	NMTOKEN	More detail about the usage of the barcode. **Values include:** ProductIdentification – End product identification (e.g., scanning in the super market).
Value ? New in JDF 1.1	string	Fixed value of the ***IdentificationField*** (e.g., on a label). Exactly one of @*Format*, @*Value* or the pair @*ValueFormat* and @*ValueTemplate* SHALL be specified.
ValueFormat ? New in JDF 1.3	string	A formatting string used with @*ValueTemplate* to define fixed and/or variable content of barcodes or text. **Allowed values are from:** ▸ Appendix H Dynamic String Generation. **Constraint:** exactly one of @*Format*, @*Value* or the pair @*ValueFormat* and @*ValueTemplate* SHALL be specified.
ValueTemplate ? New in JDF 1.3	string	A list of values used with @*ValueFormat* to define fixed and/or variable content of barcodes or text. If ***MetadataMap*** elements are present, ***MetadataMap***/@*Name* SHALL be included in @*ValueTemplate* to select the data from the ***MetadataMap***. **Allowed values are from:** ▸ Appendix H Dynamic String Generation. **Constraint:** exactly one of @*Format*, @*Value* or the pair @*ValueFormat* and @*ValueTemplate* SHALL be specified.
BarcodeDetails ? New in JDF 1.3	element	Additional specification for complex barcodes.
ExtraValues ? New in JDF 1.3	element	Additional values encoded in the ***IdentificationField***.
MetadataMap * New in JDF 1.5	element	Describes the mapping of metadata that is encoded in an ***IdentificationField*** to @*PartIDKeys*. This allows for automated selective finishing based on bar codes.

8.69.1 EncodingDetails Attribute Values

The following list provides a sample of Barcode encoding details. Values that are not present in this list MAY be valid in a **JDF** workflow.

Table 8.108: EncodingDetails Attribute Values

VALUE	DESCRIPTION	VALUE	DESCRIPTION
BOBST		ITF_14	
BrailleASCII New in JDF 1.4	A binary representation for 6 dot Braille messages. See http://en.wikipedia.org/wiki/Braille_ASCII	ITF_6	
BrailleUnicode New in JDF 1.4	A binary representation for Braille messages. See http://www.unicode.org/charts/PDF/U2800.pdf#search=%22braille%20unicode%22	ITF_16	
CODABAR		KURANDT	
CODABAR_Tradional		LAETUS_PHARMA	
CODABLOCK		MSI	
CODABLOCK_F		NDC_HRI	
Code128		PARAF	
Code25		Plessey	
Code39		PDF417	
Code39_Extended		PZN	
DATAMATRIX Deprecated in JDF 1.3	**Deprecation note:** Starting with **JDF** 1.3, use "HIBC_DATAMATRIX"	QR	
EAN	includes Bookland_EAN and ISSN.	RSS_14	
EAN_13		RSS_14_Stacked	
EAN_8		RSS_14_Stacked_Omnidir	
EAN_Coupon		RSS_14_Truncated	
EAN_128		RSS_Limited	
HIBC_Code39		RSS_Expanded	
HIBC_Code128		RSS_Expanded_Stacked	
HIBC_Code39_2		UPC_A	
HIBC_CODABLOCK_F		UPC_Coupon	
HIBC_QR		UPC_E	
HIBC_DATAMATRIX		UPC_SCS	
Interleave25			

8.69.2 BarcodeDetails

Table 8.109: BarcodeDetails Element

NAME	DATA TYPE	DESCRIPTION
BarcodeVersion ?	NMTOKEN	The version of a barcode. **Values include those from:** ▸ Table 8.112 BarcodeVersion Values – for HIBC_DATAMATRIX. **Values include those from:** ▸ Table 8.113 BarcodeVersion Values – for QR barcodes.
ErrorCorrectionLevel ?	NMTOKEN	Error correction level for barcodes having a separately definable error correction level. Each value can be used only for certain values of ***IdentificationField***/@*EncodingDetails*. **Values include:** PDF417_EC_0 – for @*EncodingDetails* = "PDF417" PDF417_EC_1 – for @*EncodingDetails* = "PDF417" PDF417_EC_2 – for @*EncodingDetails* = "PDF417" PDF417_EC_3 – for @*EncodingDetails* = "PDF417" PDF417_EC_4 – for @*EncodingDetails* = "PDF417" PDF417_EC_5 – for @*EncodingDetails* = "PDF417" PDF417_EC_6 – for @*EncodingDetails* = "PDF417" PDF417_EC_7 – for @*EncodingDetails* = "PDF417" PDF417_EC_8 – for @*EncodingDetails* = "PDF417" QR_EC_L – for @*EncodingDetails* = "QR" QR_EC_M – for @*EncodingDetails* = "QR" QR_EC_Q – for @*EncodingDetails* = "QR" QR_EC_H – for @*EncodingDetails* = "QR"
XCells ?	integer	The number of cells in x direction of a matrix barcode. For "DATAMATRIX" this field can be omitted since @*BarcodeVersion* already defines this. For "PDF417" this is the number of codewords/row.
YCells ?	integer	The number of cells in y direction of a matrix barcode For "DATAMATRIX" this field can be omitted since @*BarcodeVersion* already defines this. For "PDF417" this is the number of rows.

8.69.3 ExtraValues

Table 8.110: ExtraValues Element

NAME	DATA TYPE	DESCRIPTION
Usage	NMTOKEN	The usage of the value. **Values include:** Supplemental – UPC supplemental 2/5 digit symbology CompositeCode – This is applicable for barcodes like RSS-14 that have an optional composite code part. Coupon – The additional message for the EAN128 part of a UPC or EAN coupon.
Value	string	Additional value of the ***IdentificationField*** as specified in @*Usage*.

8.69.4 Usage of barcode Attributes

The following table specifies whether the attributes @*Height*, @*Magnification* and @*Ratio* are applicable for a given barcode type that is specified by @*EncodingDetails*.

Table 8.111: Usage of Barcode Attributes for Certain Barcode Types

ENCODINGDETAILS VALUES (BARCODE TYPES)	HEIGHT	MAGNIFICATION	RATIO
Code25 Code39 Code39_Extended Interleave25 MSI Plessey	Used	Used	Used
CODABAR Code128 EAN_13 EAN_8 EAN_128 HIBC_Code39 HIBC_Code128 ITF_14 ITF_16 NDC_HRI PARAF UPC_A UPC_E UPC_SCS UPC_SCS	Used	Used	Not used
BOBST KURANDT LAETUS_PHARMA	Used	Not used	Not used
RSS_14 RSS_14_Stacked RSS_14_Stacked_Omnidir RSS_14_Truncated RSS_Limited RSS_Expanded RSS_Expanded_Stacked	Not used	Used	Not used
PZN	Not used	Not used	Not used

8.69.4.1 BarcodeVersion Attribute Values – for HIBC_DATAMATRIX

The following table specifies valid values of ***BarcodeDetails***/@*BarcodeVersion* for a "HIBC_DATAMATRIX" barcode:
Modification note: Starting with **JDF** 1.3, these values are for "HIBC_DATAMATRIX" rather than "DATAMATRIX"

Table 8.112: BarcodeVersion Values – for HIBC_DATAMATRIX (Sheet 1 of 2)

VALUES			
DM_8_by_18	DM_16_by_16	DM_26_by_26	DM_72_by_72
DM_8_by_32	DM_16_by_36	DM_32_by_32	DM_80_by_80
DM_10_by_10	DM_16_by_48	DM_40_by_40	DM_88_by_88
DM_12_by_12	DM_18_by_18	DM_44_by_44	DM_96_by_96
DM_12_by_26	DM_20_by_20	DM_48_by_48	DM_104_by_104

Table 8.112: BarcodeVersion Values – for HIBC_DATAMATRIX (Sheet 2 of 2)

VALUES			
DM_12_by_36	DM_22_by_22	DM_52_by_52	DM_120_by_120
DM_14_by_14	DM_24_by_24	DM_64_by_64	DM_132_by_132
			DM_144_by_144

8.69.4.2 BarcodeVersion Attribute Values – for QR barcodes

The following table specifies valid values of ***BarcodeDetails***/@*BarcodeVersion* for a QR barcode.

Table 8.113: BarcodeVersion Values – for QR barcodes

VALUES							
QR_1	QR_6	QR_11	QR_16	QR_21	QR_26	QR_31	QR_36
QR_2	QR_7	QR_12	QR_17	QR_22	QR_27	QR_32	QR_37
QR_3	QR_8	QR_13	QR_18	QR_23	QR_28	QR_33	QR_38
QR_4	QR_9	QR_14	QR_19	QR_24	QR_29	QR_34	QR_39
QR_5	QR_10	QR_15	QR_20	QR_25	QR_30	QR_35	QR_40

Example 8.15: Barcode

The following example illustrates the description of a barcode in a **LayoutElementProduction** process:

```
<LayoutElementProductionParams Class="Parameter" ID="BarcodeParams"
       Status="Available">
   <LayoutElementPart>
      <BarcodeProductionParams>
         <IdentificationField Encoding="Barcode" EncodingDetails="EAN_13"
               Purpose="Label" PurposeDetails="ProductIdentification"
               Value="0123456789128"/>
         <BarcodeReproParams Height="73.50" Magnification="1.0">
            <BarcodeCompParams CompensationProcess="Printing"
                  CompensationValue="10.0"/>
         </BarcodeReproParams>
      </BarcodeProductionParams>
   </LayoutElementPart>
</LayoutElementProductionParams>
```

8.70 IDPrintingParams

Deprecated in JDF 1.1

See ▸ Section N.7.10 IDPrintingParams for details of this deprecated resource.

8.71 ImageCompressionParams

Prior to **JDF** 1.2 the filtering in ImageCompressionParams was based on the terminology in PostScript and PDF. Many image compression and decompression filters require additional information in the form of a filter parameter dictionary, and additional filter parameters have been added to meet this need.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	***ContentList***/***ContentData***, ***FormatConversionParams***, ***LayoutElement***, ***PageList***, ***PageList***/***PageData***
Example Partition:	"DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetName", "Side", "SignatureName"

Input of Processes: ImageReplacement, PDLCreation, PSToPDFConversion

Table 8.114: ImageCompressionParams Resource

NAME	DATA TYPE	DESCRIPTION
ImageCompression *	element	Specifies how images are to be compressed.

8.71.1 ImageCompression

Table 8.115: ImageCompression Element (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
AntiAliasImages = "false"	boolean	If "true", anti-aliasing is permitted on images. If "false", anti-aliasing is not permitted. Anti-aliasing increases the number of bits per component in downsampled images to preserve some of the information that is otherwise lost by downsampling. Anti-aliasing is only performed if the image is actually downsampled and if @*ImageDepth* has a value greater than the number of bits per color component in the input image.
AutoFilterImages = "true" Modified in JDF 1.2	boolean	SHALL NOT be specified unless @*EncodeImages* is "true". This attribute is not used if @*ImageType* = "Monochrome". If "true", the filter defined by @*ImageAutoFilterStrategy* is applied to photos and the "FlateEncode" filter is applied to screen shots. If "false", the @*ImageFilter* compression method is applied to all images.
ConvertImagesToIndexed ?	boolean	If "true", the application converts images that use fewer than 257 colors to an indexed color space for compactness. This attribute is used only when @*ImageType* = "Color".
DCTQuality = "0"	double	A value between 0 and 1 that indicates "how much" the process SHALL compress images when using a "DCTEncode" filter. 0.0 means "do as loss-less compression as possible." 1.0 means "do the maximum compression possible."
DownsampleImages = "false" Modified in JDF 1.1A	boolean	If "true", sampled color images are downsampled using the resolution specified by @*ImageResolution*. If "false", downsampling is not carried out and the image resolution in the PDF file is the same as that in the source file.
EncodeColorImages ? Deprecated in JDF 1.1	boolean	If "true", color images are encoded using the compression filter specified by the value of the @*ImageFilter* key. If "false", no compression filters are applied to color sampled images.
EncodeImages = "false" New in JDF 1.1 Modified in JDF 1.1A	boolean	If "true", images are encoded using the compression filter specified by the value of the @*ImageFilter* key. If "false", no compression filters are applied to sampled images.
ImageAutoFilterStrategy ? New in JDF 1.2	NMTOKEN	Selects what image compression strategy to employ if passing through an image that is not already compressed. **Values include:** JPEG – Lossy JPEG compression for low-frequency images and lossless Flate compression for high-frequency images. JPEG2000 – Lossy JPEG2000 compression for low-frequency images and lossless JPEG2000 compression for high-frequency images.
ImageDepth ?	integer	Specifies the number of bits per component in the downsampled image when @*DownsampleImages* = "true". If not specified, the downsampled image has the same number of bits per sample as the original image.

Table 8.115: ImageCompression Element (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
ImageDownsampleThreshold = "2.0"	double	Sets the image downsample threshold for images. This is the ratio of image resolution to output resolution above which downsampling can be performed. The following short examples provide a hypothetical configuration: To use @*ImageDownsampleThreshold*, set the following attributes to the values indicated: @*ImageResolution* = 72 @*ImageDownsampleThreshold* = 1.5 The input image would not be downsampled unless it has a resolution greater than (72 * 1.5) = 108 dpi
ImageDownsampleType ?	enumeration	Downsampling algorithm for images. **Allowed values are:** Average – The program averages groups of samples to get the new downsampled value. Bicubic – The program uses bicubic interpolation on a group of samples to get a new downsampled value. Subsample – The program picks the middle sample from a group of samples to get the new downsampled value.
ImageFilter ? Modified in JDF 1.3	NMTOKEN	Specifies the compression filter to be used for images. Ignored if @*AutoFilterImages* = "true" or if @*EncodeImages* = "false". **Values include:** CCITTFaxEncode – Used to select CCITT Group 3 or 4 facsimile encoding. Used only if @*ImageType* = "Monochrome". DCTEncode – Used to select JPEG compression. FlateEncode – Used to select ZIP compression. JBIG2Encode – Used to select JBIG2 encoding. Used only if @*ImageType* = "Monochrome". New in JDF 1.3 JPEG2000 – Used to select JPEG2000/Wavelet compression. New in JDF 1.2 LZWEncode – LZW compression. PackBits – A simple byte-oriented run length scheme. **Modification note:** Starting with **JDF** 1.1, the data type changes from enumeration to NMTOKEN in order to allow for extensions.
ImageResolution ?	double	Specifies the minimum resolution for downsampled color images in dots per inch. This value is used only when @*DownsampleImages* = "true". The application downsamples only images that are above that resolution to that actual resolution.
ImageType Modified in JDF 1.5	enumerations	Specifies the kind of images that are to be manipulated. **Allowed values are:** All – image compression is applied to all image types. New in JDF 1.5 Color Grayscale Monochrome
JPXQuality ? New in JDF 1.2	integer	Specifies the image quality. Valid values are greater than or equal to one (1) and less than or equal to 100. One (1) means lowest quality (highest compression), 99 means visually lossless compression, and 100 means numerically lossless compression.
CCITTFaxParams ? New in JDF 1.2	element	The equivalent of the PostScript *Rows* and *BlackIs1* parameters, which are implicit in the raster data to be compressed.
DCTParams ? New in JDF 1.2	element	Provides the equivalents of the PostScript *Columns*, *Rows* and *Colors* attributes, which are assumed to be implicit in the raster data to be compressed.
FlateParams ? New in JDF 1.2	element	The equivalent of the PostScript *Columns*, *BitsPerComponent* and *Colors* parameters, which are implicit in the raster data to be compressed.

Table 8.115: ImageCompression Element (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
JBIG2Params ? New in JDF 1.3	element	Provides the JBIG2 compression parameters.
JPEG2000Params ? New in JDF 1.3	element	Provides the JPEG2000 compression parameters.
LZWParams ? New in JDF 1.2	element	The equivalent of the PostScript *Columns*, *BitsPerComponent* and *Colors* parameters, which are implicit in the raster data to be compressed

8.71.2 CCITTFaxParams

New in JDF 1.2

Table 8.116: CCITTFaxParams Element

NAME	DATA TYPE	DESCRIPTION
EncodedByteAlign ?	boolean	A flag indicating whether the CCITTFaxEncode filter inserts an extra 0 bits before each encoded line so that the line begins on a byte boundary.
EndOfBlock ?	boolean	A flag indicating whether the CCITTFaxEncode filter appends an end-of-block pattern to the encoded data
EndOfLine ?	boolean	A flag indicating whether the CCITTFaxEncode filter prefixes an end-of-line bit pattern to each line of encoded data.
K = "0"	integer	An integer that selects the encoding scheme to be used. < 0 – Pure two-dimensional encoding (Group 4, TIFF compression = 4) = 0 – Pure one-dimensional encoding (Group 3, 1-D, TIFF compression = 2) > 0 – Mixed one- and two-dimensional encoding (Group 3, 2-D, TIFF compression = 3), in which a line encoded one-dimensionally can be followed by at most K – 1 lines encoded two-dimensionally
Uncompressed = "false"	boolean	A flag to indicate whether the file generated can use uncompressed encoding when advantageous.

8.71.3 DCTParams

New in JDF 1.2

Table 8.117: DCTParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ColorTransform = "Automatic"	enumeration	Color transformation algorithm. **Allowed values are:** None – Colors are not to be transformed. YUV – RGB raster values are to be transformed to YUV before encoding and from YUV to RGB after decoding. If four channels are present, transform CMYK values to YUVK before encoding and from YUVK to CMYK after decoding. Automatic – "YUV" for 3-channel raster data, "None" otherwise. **Note**: YUV is equivalent to YCbCr in TIFF terminology.
HSamples ?	IntegerList	A sequence of horizontal sampling factors—one entry per color channel in the raster data. If not specified, the implied default is "1" for every channel.
HuffTable ?	DoubleList	Huffman tables for DC and AC components. If present, there SHALL be at least one HuffTable element for each color channel.
QFactor = "1.0"	double	A scale factor applied to the elements of @*QuantTable*.
QuantTable ?	DoubleList	Quantization tables. If present, there SHALL be one @*QuantTable* entry for each color channel.

Table 8.117: DCTParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
SourceCSs Deprecated in JDF 1.6	enumerations	Identifies which of the incoming color spaces will be operated on. **Allowed values are from:** ▸ Table 9.10 SourceCS Attribute Values**.** **Note: JDF** 1.1 defined that CalRGB be treated as RGB, CalGray as Gray, and ICC-Based color spaces as one of Gray, RGB or CMYK depending on the number of channels. **Note:** In **JDF** 1.2, the data type was erroneously specified as enumeration, not enumerations.
VSamples ?	IntegerList	A sequence of vertical sampling factors—one entry per color channel in the raster data. If not specified, the implied default is "1" for every channel.

When the ***DCTParams*** element is a Subelement of ***ImageCompressionParams*** used in a **Rendering** process to generate TIFF files, YUV is equivalent to YCbCr in TIFF terminology. The HSamples and VSamples values are used to set YCbCrSubSampling or CIELabSubSampling. This means that they are only relevant for data supplied as Lab, or data where @*ColorTransform* is "YUV"; that the first element SHALL be 1 in each case; that the fourth element SHALL be 1 where CMYK data is to be compressed; and that the second and third elements SHALL equal each other.

8.71.4 FlateParams

New in JDF 1.2

Table 8.118: FlateParams Element

NAME	DATA TYPE	DESCRIPTION
Effort ?	integer	A code controlling the amount of memory used and the execution speed for Flate compression. Allowed values range from 0 to 9. A value of 0 compresses rapidly but not tightly, using little auxiliary memory. A value of 9 compresses slowly but as tightly as possible, using a large amount of auxiliary memory.
Predictor = "1"	integer	A code that selects the predictor function: **Note:** On 1X PNG predictors, these values select the specific PNG predictor function(s) to be used, as indicated above. When decoding the predictor function is explicitly encoded in the incoming data. **Values include:** 1 – No predictor (normal encoding or decoding). 2 – TIFF Predictor 2. 10 – PNG predictor, None function. 11 – PNG predictor, Sub function. 12 – PNG predictor, Up function. 13 – PNG predictor, Average function. 14 – PNG predictor, Path function. 15 – PNG predictor in which the encoding filter automatically chooses the optimum function separately for each row.

8.71.5 JBIG2Params

New in JDF 1.3

Table 8.119: JBIG2Params Element

NAME	DATA TYPE	DESCRIPTION
JBIG2Lossless ?	boolean	If "true" requires JBIG2 compressed images to retain the exact representation of the original image without loss.

8.71.6 JPEG2000Params

New in JDF 1.3

Table 8.120: JPEG2000Params Element

NAME	DATA TYPE	DESCRIPTION
CodeBlockSize ?	integer	The nominal code block width and height. SHALL be a power of 2.
LayerRates ?	DoubleList	Compression bit ratio for each layer. If specified, there SHALL be the same number of doubles in this list as @*LayersPerTile* in ascending order. Small values correspond to maximum compression and 1.0 corresponds to no compression (lossless). If available, @*LayerRates* SHOULD be supplied.
LayersPerTile = "1"	integer	Specifies the number of quality layers per tile at the same resolution.
NumResolutions ?	integer	The number of resolution levels encoded in the file.
ProgressionOrder ?	enumeration	Per tile progression order. **Allowed values are:** LRCP – layer-resolution-component-position progressive (i.e., rate scalable). RLCP – Resolution-layer-component-position progressive (i.e., resolution scalable). RPCL – Resolution-position-component-layer progressive. PCRL – Position-component-resolution-layer progressive. CPRL – Component-position-resolution-layer progressive.
TileSize ?	XYPair	The width and height of each encoding tile. If not specified the image is considered to be a single tile.

8.71.7 LZWParams

New in JDF 1.2

Table 8.121: LZWParams Element

NAME	DATA TYPE	DESCRIPTION
EarlyChange = "1"	integer	A code indicating when to increase the code word length. The TIFF specification can be interpreted to imply that code word length increases are postponed as long as possible. However, some existing implementations of LZW increase the code word length one code word earlier than necessary. The PostScript language supports both interpretations. If @*EarlyChange* is "0", code word length increases are postponed as long as possible. If it is "1", they occur one code word early. **Note:** The default SHOULD NOT be used when this ***LZWParams*** element is in ***ImageCompressionParams*** used as an Input resource to a ***FormatConversion*** process that is creating TIFF files.
Predictor = "1"	integer	A code that selects the predictor function: 1 – No predictor (normal encoding or decoding). 2 – TIFF Predictor 2. 10 – PNG predictor, None function. 11 – PNG predictor, Sub function. 12 – PNG predictor, Up function. 13 – PNG predictor, Average function. 14 – PNG predictor, Path function. 15 – PNG predictor in which the encoding filter automatically chooses the optimum function separately for each row. **Note:** On 1X PNG predictors, these values select the specific PNG predictor function(s) to be used, as indicated above. When decoding, the predictor function is explicitly encoded in the incoming data.

8.72 ImageEnhancementParams

ImageEnhancementParams describes the controls for manipulating images.

New in JDF 1.5

Resource Properties

Resource Class: Parameter

Input of Processes: **ImageEnhancement**

Table 8.122: ImageEnhancementParams Resource

NAME	DATA TYPE	DESCRIPTION
ImageEnhancementOp *	element	Each ***ImageEnhancementOp*** describes an individual enhancement operation. The XML order of ***ImageEnhancementOp*** elements is significant. Multiple elements that apply to the same object SHALL be applied in that XML order.

8.72.1 ImageEnhancementOp

New in JDF 1.5

Table 8.123: ImageEnhancementOp Element

NAME	DATA TYPE	DESCRIPTION
ObjectTags ?	NMTOKENS	Tags associated with individual objects that this ***ImageEnhancementOp*** SHALL be applied to. Each tag specified in @*ObjectTags* is logically anded with the object type(s) specified by @*SourceObjects*, enabling first qualification by object type (such as image), and then tags associated with those objects. The values of @*ObjectTags* depends on the PDL that the color correction is applied to. @*ObjectTags* SHALL apply only to objects whose tag pool includes all the tags in the value of @*ObjectTags*.
Operation	NMTOKEN	Individual enhancement operation name. **Values include:** Sharpening – Image Sharpening Blurring – Image Blurring RedEyeRemoval BestGuess – Best guess automated improvements based on image analysis
OperationDetails ?	string	Additional details of the @*Operation*. The values are implementation specific.
SourceObjects ?	enumeration	Identifies which class(es) of incoming graphical objects will be operated on. Allowed values are from: ▸ SourceObjects

8.73 ImageReplacementParams

This resource specifies parameters to control image replacement within production workflows.

Resource Properties

Resource Class: Parameter

Example Partition: "DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetName", "Side", "SignatureName"

Input of Processes: ImageReplacement

Table 8.124: ImageReplacementParams Resource

NAME	DATA TYPE	DESCRIPTION
IgnoreExtensions ?	NMTOKENS	Identifies a set of filename extensions that will be trimmed during searches for high-resolution images. These extensions are what will be stripped from the end of an image name to find a base name. The leading dot "." is included. The values are examples: **Values include:** .lay .e .samp
ImagePreScanStrategy ? New in JDF 1.2	NMTOKEN	Specifies the image pre-scanning strategy to be used on the input document data before starting the **RIPing** . **Values include:** NoPreScan – Do not pre-scan the document looking for references to images. PreScan – Pre-scan the document looking for references to images and making sure the data are accessible now so that the RIP will not encounter a fault later. PreScanAndGather – Pre-scan the document looking for references to images, and copy the data to a temporary place so that the RIP will be able to access the data with a predictable and small well-bounded delay later.
ImageReplacementStrategy	enumeration	Identifies how externally referenced images will be handled within the associated process. **Allowed values are:** Omit – Complete process maintaining only references to external data. Proxy – Complete process using available proxy images. Replace – Replace external references with image data during processing. AttemptReplacement – Attempt to replace external references with image data during processing. If replacement fails, complete the process using available proxy images.
MaxResolution ? Deprecated in JDF 1.1	double	Reduces the resolution of images with a resolution higher than @*MaxResolution*. Replaced with a link to ***ImageCompressionParams*** in the process.
MaxSearchRecursion ?	integer	Identifies how many levels of recursion in the search path will be traversed while trying to locate images. A value of 0 indicates that no recursion is desired.
MinResolution ?	double	Specifies the minimum resolution that an image SHALL have in order to be embedded. If not specified, images of any resolution can be embedded.
ResolutionReductionStrategy ? Deprecated in JDF 1.1	enumeration	Identifies the mechanism used for reducing the image resolution. **Allowed values are:** Downsample Subsample Bicubic **Deprecation note:** Starting with **JDF** 1.1, use a link to ***ImageCompressionParams*** in the process.
FileSpec (SearchPath) + New in JDF 1.1	refelement	Specification of the paths to search when trying to locate the referenced data. The ***FileSpec*** replaces the ***SearchPath*** text element.
SearchPath * Deprecated in JDF 1.1	text element	String that identifies the paths to search when trying to locate the referenced data.

8.74 ImageSetterParams

This resource specifies the settings for the imagesetter. A number of settings are OEM-specific, while others are so widely used they MAY be supported between vendors. Both filmsetter settings and platesetter settings are described with this resource.

Resource Properties

Resource Class: Parameter

Resource referenced by: *PreviewGenerationParams*

Intent Pairing: *ProofingIntent*

Input of Processes: ImageSetting

Table 8.125: ImageSetterParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AdvanceDistance ?	double	Additional media advancement beyond the media dimensions on a Web-Fed device.
BurnOutArea ? New in JDF 1.1	XYPair	Size of the burnout area. The area defined by @*BurnOutArea* is exposed, regardless of the size of the image. If not specified or "0 0", only the area defined by the image is exposed.
CenterAcross ?	enumeration	Specifies the axis around which a device SHALL center an image if the device is capable of doing so. **Allowed value is from:** ▸ Axis.
CutMedia ?	boolean	Indicates whether or not to cut the media (Web-Fed).
ManualFeed ? New in JDF 1.2	boolean	Indicates whether the media will be fed manually.
MirrorAround ="None"	enumeration	This attribute specifies the axis around which a device SHALL mirror an image if the device is capable of doing so. **Allowed value is from:** ▸ Axis.
NonPrintableMarginBottom ? New in JDF 1.3	double	The width in points of the bottom margin measured inward from the edge of the ***Media*** with respect to the idealized process coordinate system of the **ImageSetting** process. The ***Media*** origin is unaffected by @*NonPrintableMarginBottom*. These margins are independent of the PDL content.
NonPrintableMarginLeft ? New in JDF 1.3	double	Same as @*NonPrintableMarginBottom* except for the left margin.
NonPrintableMarginRight ? New in JDF 1.3	double	Same as @*NonPrintableMarginBottom* except for the right margin.
NonPrintableMarginTop ? New in JDF 1.3	double	Same as @*NonPrintableMarginBottom* except for the top margin.
Polarity = "Positive"	enumeration	Some Devices can invert the image (in hardware). **Allowed value is from:** ▸ Polarity
Punch ? Deprecated in JDF 1.3	boolean	If "true", indicates that the device SHALL create registration punch holes. Use a combined process with a **Bending** process to specify punching in **JDF** 1.3 and beyond.
PunchType ? Deprecated in JDF 1.3	string	Name of the registration punch scheme (e.g., *Bacher*). Use a combined process a **Bending** process to specify punching in **JDF** 1.3 and beyond.
Resolution ?	XYPair	Resolution of the output. If not specified, the default is taken from the resolution of the input ByteMap.
RollCut ?	double	Length of media to be cut off of a Roll, in points.
Sides = "OneSided-Front" New in JDF 1.2	enumeration	Indicates whether the content layout SHALL be imaged on one or both sides of the media. @*Sides* SHALL NOT be used unless ***ImageSetterParams*** describes output to a proofer. **Allowed values are from:** ▸ Table 8.126 Sides Attribute Values**.**

Table 8.125: ImageSetterParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
SourceWorkStyle ? New in JDF 1.2	enumeration	When proofing in a "RIP once, output many" (ROOM) workflow, @*SourceWorkStyle* specifies the direction in which the bytemaps have been prepared for press. The device SHALL use this information to calculate a transformation that results in a proof that is identical to the press Sheet. **Allowed value is from:** ▸ WorkStyle**.**
TransferCurve ?	Transfer-Function	Area coverage correction of the device.
FitPolicy ? New in JDF 1.2	element	Describes the hardware image fitting algorithms. Allows printing even if the size of the imageable area of the media does not match the requirements of the data.

8.74.1 Sides Attribute Values

*Table 8.126: **Sides** Attribute Values*

VALUE	DESCRIPTION
OneSidedBackFlipX	Page content is imaged on the back side of media so that the corresponding page cells back up to a blank front cell when flipping around the X axis. Equivalent to "WorkAndTumble" with a blank front side.
OneSidedBackFlipY	Page content is imaged on the back side of media so that the corresponding page cells back up to a blank front cell when flipping around the Y axis. Equivalent to "WorkAndTurn" with a blank front side.
OneSidedFront	Page content is imaged on the front side of media. This is the only value that is valid for filmsetting and platesetting. The default.
TwoSidedFlipX	Page content is imaged on both the front and back sides of media Sheets so that the corresponding page cells back up to each other when flipping around the X axis. Equivalent to "WorkAndTumble".
TwoSidedFlipY	Page content is imaged on both the front and back sides of media Sheets so that the corresponding page cells back up to each other when flipping around the Y axis. Equivalent to "WorkAndTurn".

8.75 Ink

Ink describes the ink, primer, toner or varnish that is applied to a substrate when printing or varnishing. Whereas ***Color*** describes the visual properties of a colorant, Ink describes the physical material that is applied to the substrate. The default unit of measurement for ***Ink*** is @*Unit* = "g" (gram).

Resource Properties

Resource Class: Consumable

Example Partition: "FountainNumber", "Separation", "SheetName", "Side", "SignatureName", "WebName"

Input of Processes: **ConventionalPrinting**, **DigitalPrinting**, **Varnishing**

Table 8.127: Ink Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ColorName ? Deprecated in JDF 1.4	string	Link to a definition of the color specifics. The value of @*ColorName* color SHOULD match the @*Name* attribute of a ***Color*** defined in a ***ColorPool*** resource that is linked to the process that is using the ***Ink*** resource. Instead of linking the ***ColorPool*** resource directly, it MAY be referenced by another resource that is linked to the process. **Note:** A @*ColorName* attribute is used differently in other resources where it refers to a @*NamedColor* as defined in ▸ Section A.3.22 NamedColor. **Deprecation note:** Starting with **JDF** 1.4, use @*Separation* Partition Key.

Table 8.127: Ink Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Family ?	NMTOKEN	Ink family. **Values include:** Ink – any ink that is used as a colorant. Primer – any ink that is used as a primer. Varnish – liquid that is similar to ink Toner – liquid that is similar to ink
InkName ?	string	The fully qualified ink name including the ink @*Family* name. For instance, "PANTONE 138 C" is a member of the PANTONE family.
SpecialInk ? Modified in JDF 1.5	NMTOKENS	Specific ink attributes. **Values include those from:** ▸ Section C.6 Media Coating Names.
SpecificYield ?	double	Weight per area at total coverage in g/m^2.

8.76 InkZoneCalculationParams

This resource specifies the parameters for the **InkZoneCalculation** process.

Resource Properties

Resource Class: Parameter

Example Partition: "TileID", "WebName"

Input of Processes: **InkZoneCalculation**

Table 8.128: InkZoneCalculationParams Resource

NAME	DATA TYPE	DESCRIPTION
FountainPositions ?	DoubleList	Even number of positions. Each pair specifies the begin and end of the ink slides belonging to a certain fountain. The positions are in coordinates of the printable width along the cylinder axis. The first pair is associated to the first fountain position (corresponds to the Partition @*FountainNumber* = "0"), the second to the second position (@*FountainNumber* = "1"), etc.
PrintableArea ?	rectangle	Position and size of the printable area of the print cylinder in the coordinates of the ***Preview*** resource. The Partition @*TileID* SHALL be used for each plate together with this attribute in case of multiple plates per cylinder. Multiple plates per cylinder MAY be used in Web Printing. The default case SHALL specify a rectangle that encompasses the complete image to be printed.
ZoneHeight ?	double	The width of one zone in the feed direction of the printing Machine being used.
Zones ? Modified in JDF 1.2	integer	The number of ink zones of the press.
ZonesY ?	integer	Number of ink zones in feed direction of the press.
ZoneWidth ? Modified in JDF 1.2	double	The width of one zone of the printing Machine being used. Typically, the width of a zone is the width of an ink slide.
Device ? New in JDF 1.2	refelement	***Device*** provides a reference to the press that the ***InkZoneProfile*** is defined for and is used to gather information about ink zone geometry.

8.77 InkZoneProfile

This resource specifies ink zone settings that are specific to the geometry of the printing device being used. ***InkZoneProfile*** elements are independent of the device details.

Resource Properties

Resource Class: Parameter

Example Partition: "FountainNumber", "Separation", "SheetName", "Side", "SignatureName", "WebName"

Input of Processes: **ConventionalPrinting**

Output of Processes: **InkZoneCalculation**

Table 8.129: InkZoneProfile Resource

NAME	DATA TYPE	DESCRIPTION
ZoneHeight ?	double	The width of one zone in the feed direction of the printing Machine being used.
ZoneSettingsX	DoubleList	Each entry of the @*ZoneSettingsX* attribute is the value of one ink zone. The first entry is the first zone, and the number of entries equals the number of zones of the printing device being used. Allowed values are in the range [0,1] where 0 is no ink and 1 is 100% coverage.
ZoneSettingsY ?	DoubleList	Each entry of the @*ZoneSettingsY* attribute is the value of one ink zone in Y Direction. The first entry is the first zone, and the number of entries equals the number of zones of the printing device being used. Allowed values are in the range [0,1] where 0 is no ink and 1 is 100% coverage.
ZoneWidth	double	The width of one zone of the printing Machine being used. Typically, the width of a zone is the width of an ink slide.

8.78 InsertingParams

This resource specifies the parameters for the **Inserting** process. Figure 7.13 shows the various components involved in an inserting process, and how they interact.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***InsertingIntent***

Input of Processes: **Inserting**

Table 8.130: InsertingParams Resource

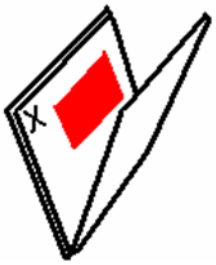
NAME	DATA TYPE	DESCRIPTION
FinishedPage ? New in JDF 1.2	integer	Finished Page number of the mother ***Component*** on which the child ***Component*** has to be placed. @*FinishedPage* SHALL NOT be specified unless @*InsertLocation* = "FinishedPage". Corresponds to ***InsertingIntent***/@Folio.
InsertLocation Modified in JDF 1.2	enumeration	Where to place the "child" Sheet. **Allowed values are:** Back FinishedPage – Place the child exactly onto the page specified in @*FinishedPage*. New in JDF 1.2 Front Overfold – Place onto the overfold. Replaces "OverfoldLeft" and "OverfoldRight". New in JDF 1.2 OverfoldLeft – Deprecated in JDF 1.2 OverfoldRight – Deprecated in JDF 1.2 **Modification note:** Starting with **JDF** 1.2, this attribute is renamed from @*Location* due to a name clash with the @*Location* Partition Key.
Method = "BlowIn"	enumeration	Inserting method. **Allowed values are:** BindIn – Apply glue to fasten the insert. BlowIn – Loose insert.
SheetOffset ? Deprecated in JDF 1.1	XYPair	Offset between the Sheet to be inserted and the "mother" Sheet. @*SheetOffset* is implied by the Transformation matrix in ***ResourceLink***/@*Transformation* of the child's ***ComponentLink***.
GlueLine *	element	Array of all ***GlueLine*** elements. The coordinate system is defined by the mother ***Component***.

Location of Inserts

New in JDF 1.2

The following graphics depict the various values of ***InsertingParams***/@*InsertLocation:*

Table 8.131: Location of Inserts

FRONT	BACK	OVERFOLD	FINISHED PAGE
			
Child on *Front* of mother component — is used for fixed inserts (e.g., gluing of inserts and so forth on Signatures).	Child on *Back* of mother component — is used for fixed inserts (e.g., gluing of inserts on Signatures).	The mother component is opened at the overfold and the child is placed in the center of the of the mother. *Overfold* is used for loose inserts (e.g., inserts into newspapers).	Child on "FinishedPage" X of mother component — can be used for loose and fixed inserts.

8.79 InterpretedPDLData

Represents the results of the **Interpreting** or **RasterReading** process. The details of this resource are not specified, as it is assumed to be implementation dependent.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***RunList***

8.80 InterpretingParams

The ***InterpretingParams*** resource contains the parameters needed to interpret PDL pages. The resource itself is a generic resource that contains attributes that are relevant to all PDLs. PDL-specific instances of ***InterpretingParams*** resources MAY be included as Subelements of this generic resource. This specification defines one additional PDL-specific resource instance: ***PDFInterpretingParams***.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***ProofingIntent***

Example Partition: "DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetName", "Side", "SignatureName"

Input of Processes: **Interpreting**

Table 8.132: InterpretingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Center = "false"	boolean	Indicates whether or not the finished page image SHALL be centered within the imageable area of the media. The @*Center* is ignored if ***FitPolicy***/@*SizePolicy* = "ClipToMaxPage" and clipping is specified.
FitToPage ? Deprecated in JDF 1.1	boolean	Specifies whether the finished page contents SHALL be scaled to fit the media. In **JDF** 1.1 and beyond, use ***FitPolicy***.
MirrorAround = "None"	enumeration	This attribute specifies the axis around which a RIP SHALL mirror an image. **Note:** This is mirroring in the RIP and not in the hardware of the output device. **Allowed value is from:** ▸ Axis**.**

Table 8.132: InterpretingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Polarity = "Positive"	enumeration	The image SHALL be RIPed in the specified polarity. Note that this is a polarity change in the RIP and not a polarity change in the hardware of the output device. **Allowed value is from:** ▸ Polarity.
Poster ? Deprecated in JDF 1.5	XYPair	Specifies whether the page contents SHALL be expanded such that each page covers X by Y pieces of media. **Deprecation note:** Starting with **JDF** 1.5, use **Tiling** instead of @*Poster* and @*PosterOverlap*.
PosterOverlap ? Deprecated in JDF 1.5	XYPair	This pair of real numbers identifies the amounts of overlap in points for the poster tiles across the horizontal and vertical axes, respectively. **Deprecation note:** Starting with **JDF** 1.5, use **Tiling** instead of @*Poster* and @*PosterOverlap*.
PrintQuality = "Normal" New in JDF 1.1	enumeration	Generic switch for setting the quality of an otherwise inaccessible Device. **Allowed values are:** High – Highest quality available on the printer. Normal – The default quality provided by the printer. Draft – Lowest quality available on the printer.
Scaling ?	XYPair	A pair of positive real values that indicates the scaling factor for the page contents. Values between 0 and 1 specify that the contents are to be reduced, while values greater than 1 specify that the contents are to be expanded. This attribute is ignored if @*FitToPage* = "true". Any scaling defined in ***FitPolicy*** SHALL be applied after the scaling defined by this attribute.
ScalingOrigin ?	XYPair	A pair of real values that identifies the point in the unscaled PDL page that remains at the same position after scaling. This point is defined in the coordinate system of the PDL page. For example, The @*ScalingOrigin* of a PDL page with dimensions "300 400" scaled from the PDL page center would be "150 200", regardless of the value of @*Scaling*. **Modification note:** Starting with **JDF** 1.4, 1) the default value MAY be set to an implementation defined value; the default value is no longer specified as "0 0" in this document; 2) the phrase "PDL page" replaces "Page"; 3) this attribute specifies the point which is not shifted when scaling is applied and doesn't specify a new Origin (i.e., lower left of the page).
FitPolicy ? New in JDF 1.1	element	Allows printing even if the size of the imageable area of the media does not match the requirements of the data. This replaces the deprecated @*FitToPage* attribute. This ***FitPolicy*** resource SHALL be ignored in a combined process with **LayoutPreparation**.
InterpretingDetails ? New in JDF 1.5	element	Container for interpreter-specific details.
Media * New in JDF 1.1 Modified in JDF 1.2	refelement	This resource provides a description of the physical media which will be marked. The physical characteristics of the media MAY affect decisions made during **Interpreting**. The cardinality was changed to "*" in **JDF** 1.2 in order support description of multiple media types (e.g., Film, Plate and Paper). If multiple ***Media*** are specified, ***Media***/@*MediaType* defines the type of ***Media***. If multiple ***Media*** with ***Media***/@*MediaType* = "Paper" are specified in a proofing environment, the first ***Media*** is the proofer paper and the second ***Media*** is the final device paper.
ObjectResolution *	element	Indicates the resolution at which the PDL contents will be interpreted in DPI. These elements MAY be different from the ***ObjectResolution*** elements provided in the resource.
PDFInterpretingParams ? New in JDF 1.1	element	Details of interpreting for PDF. Note that this is a Subelement in **JDF** 1.1 and beyond, and not an instance as in **JDF** 1.0.

8.80.1 InterpretingDetails

New in JDF 1.5

InterpretingDetails contains PDL-specific instructions for an interpreter.

Table 8.133: InterpretingDetails Element

NAME	DATA TYPE	DESCRIPTION
MinLineWidth ?	double	If present, this attribute specifies the minimum width in points for PDL line objects. If a line is defined with a width smaller than this value it SHALL be adjusted to a line width equal to this value. **Note:** This attribute is useful for managing the consistency of thin lines across different digital printing systems that have varying imaging resolutions.

8.80.2 PDFInterpretingParams

New in JDF 1.1

Table 8.134: PDFInterpretingParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
EmitPDFBG = "true"	boolean	Indicates whether BlackGeneration functions are to be emitted.
EmitPDFHalftones = "true"	boolean	Indicates whether Halftones are to be emitted.
EmitPDFTransfers = "true"	boolean	Indicates whether Transfer functions are to be emitted
EmitPDFUCR = "true"	boolean	Indicates whether UnderColorRemoval functions are to be emitted.
HonorPDFOverprint = "true"	boolean	Indicates whether or not overprint settings in the file will be honored. If "true", the setting for overprint will be honored. If "false", it is expected that the device does not directly support overprint and that the PDF is prepro-cessed to simulate the effect of the overprint settings
ICCColorAsDeviceColor = "false"	boolean	Indicates whether colors specified by ICC color spaces are to be treated as device colorants.
OCGDefault = "FromPDF" New in JDF 1.3	enumeration	Specifies whether optional Content Groups (OCGs or layers) in the PDF being interpreted and not explicitly listed in subsidiary ***OCGControl*** Subelements, are to be included in the ***InterpretedPDLData*** produced by the **Interpreting** process. **Allowed values are:** Exclude – All layers not explicitly listed are to be excluded. FromPDF – The guidelines in the PDF reference are to be used to determine whether to include each layer that is not explicitly listed. Include – All layers not explicitly listed are to be included.
OCGIntent ? New in JDF 1.3	NMTOKEN	If @*OCGDefault* = "FromPDF", then the value of @*OCGIntent* sets the intent for which OCGs are to be selected. **Values include:** Design – as described in ▸[PDF1.6]. View – as described in ▸[PDF1.6].
OCGProcess ? New in JDF 1.3	enumeration	If @*OCGDefault* = "FromPDF", then the value of @*OCGProcess* sets the purpose for which the **Interpreting** process is being performed. This, in turn, sets which value from a relevant optional content usage dictionary SHALL be used to determine whether each OCG is included in the ***InterpretedPDLData***. **Allowed values are:** Export – PDF ExportState in the Export subdictionary. Print – PDF PrintState in the Print subdictionary View – PDF ViewState in the View subdictionary. Additional values are defined in ▸[ISO19593-1:2016] or MAY be site specific.

Table 8.134: PDFInterpretingParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
OCGZoom = "1.0" New in JDF 1.3	double	If @*OCGDefault* = "FromPDF", then the value of @*OCGZoom* sets the magnification to be assumed in comparisons with the Zoom dictionary in a relevant optional content usage dictionary to determine whether each OCG is included in the ***InterpretedPDLData***. A @*OCGZoom* value of 1.0 is assumed to be a magnification of 100%.
PrintPDFAnnotations = "false" Modified in JDF 1.3	boolean	Indicates whether the contents of annotations on PDF pages SHALL be included in the output. This only refers to annotations that are set to print in the PDF file excluding trap annotations. Trap annotations are controlled with @*PrintTrapAnnotations*.
PrintTrapAnnotations ? New in JDF 1.3	boolean	Indicates whether the contents of trap annotations on PDF pages SHALL be included in the output.
TransparencyRenderingQuality ?	double	Values are 0 to 1. A value of 0 represents the lowest allowable quality; 1 represents the highest desired quality.
OCGControl * New in JDF 1.3	element	***OCGControl***provides a list of the OCGs (layers) that are to be explicitly included or excluded in the ***InterpretedPDLData***. Any OCGs not listed in an ***OCGControl*** element SHALL follow the rules set by @*OCGDefault*.
ReferenceXObjParams ? New in JDF 1.4	element	Describes how the interpreter should handle PDF Reference XObjects

8.80.3 OCGControl

New in JDF 1.3

Table 8.135: OCGControl Element

NAME	DATA TYPE	DESCRIPTION
IncludeOCG = "true"	boolean	Defines whether the optional content group(s) identified by @*OCGName* are to be included in the ***InterpretedPDLData***. If "true", then the layer SHALL be included. If "false", it SHALL NOT. **Note:** The contents stream of excluded OCGs SHALL still be interpreted so that changes to CTM, etc., are acted on. The objects drawn in excluded OCGs SHALL NOT be rendered.
OCGName ?	string	The name of the optional content group(s) that SHALL be included or excluded. Exactly one of OCGName or ProcStepsGroup SHALL be present. **Note:** The @*Name* attribute of an optional content group entry is encoded as a PDF text string, and @*OCGName* is encoded with the Unicode variant identified in the **JDF** file header; names SHALL be re-encoded as necessary for comparison. Using a value for @*OCGName* that does not match any OCG in the referenced PDF file is an error (subject to @*SettingsPolicy*), independent of the value of @*IncludeOCG*.
ProcStepsGroup ? New in JDF 1.6	NMTOKEN	An OCG is selected, if @*ProcStepsGroup* matches the value of GTS_ProcStepsGroup in the GTS_Metadata dictionary of the OCG of a PDF that complies with ▸ [ISO19593-1:2016].
ProcStepsType ? New in JDF 1.6	NMTOKEN	If specified, An OCG is selected, if @*ProcStepsType* matches the value of GTS_ProcStepsType in the GTS_Metadata dictionary of the OCG of a PDF that complies with ▸ [ISO19593-1:2016]. @*ProcStepsType* SHALL NOT be specified unless @*ProcStepsGroup* is present.

8.80.4 ReferenceXObjParams

New in JDF 1.4

Table 8.136: ReferenceXObjParams Element

NAME	DATA TYPE	DESCRIPTION
Mode	NMTOKEN	Specifies how to handle a Reference XObject's reference. **Values include:** Ignore – the reference is ignored, and no content is imaged for that Reference XObject. If proxy content is supplied with the Reference XObject, it is imaged. ResolveAlways – an attempt is made to resolve the reference, and image the graphics described by that reference. ResolveIfPDFX5 – an attempt is made to resolve the reference ONLY if the PDF file is a valid PDF/X-5 file, AND the referenced file passes the criteria stated in section 8.4 of ISO 15930-8 (PDF/X-5).
FileSpec (SearchPath) *	refelement	An ordered list of search paths to search when an XObject provides a relative file specification for its target file. If not specified, then the directory that contains the PDF file being interpreted will be searched, and SHALL NOT be searched recursively.

8.80.5 More about PDFInterpretingParams

8.80.5.1 PDF Optional Content Groups

The order of ***OCGControl*** elements has no effect; the Z-order of graphic elements that make up each optional content group (the term layer is misleading in this regard) within the PDF file defines the drawing order of those graphic elements.

Any preferences recorded in OCGs within the PDF file as to whether that OCG are to be displayed or not will be ignored if that OCG is referenced from an ***OCGControl*** element, or if @*OCGDefault* is either "Include" or "Exclude"; PDF preferences are only applied when @*OCGDefault* = "FromPDF".

If @*OCGDefault* = "FromPDF", the state of all OCGs explicitly referenced from ***OCGControl*** elements SHALL be set before determining the state of any remaining OCGs.

All controls for OCGs in **JDF** address OCGs directly, and not Optional Content Member Dictionaries (OCMDs do not have unique names).

Note: ▸ [PDF1.6] does not state that all OCGs SHALL have unique names. It is therefore possible for a single PDF file to contain multiple OCGs with the same name. When ***OCGControl***/@*OCGName* refers to multiple OCGs in a file, they will all be explicitly included or excluded together.

8.81 JacketingParams

New in JDF 1.1

Description of the setup of the jacketing machinery. Jacket height and width (1 and 4 in the ▸ Figure 8-37: Setup of the Jacketing Machinery) are specified within the ***Component*** that describes the jacket.

Figure 8-37: Setup of the Jacketing Machinery

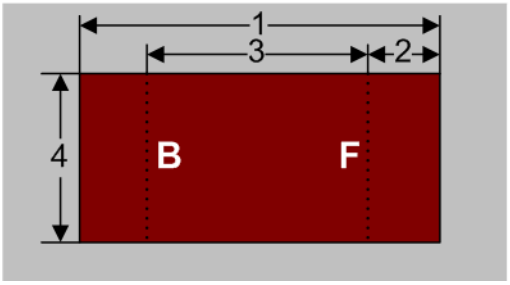
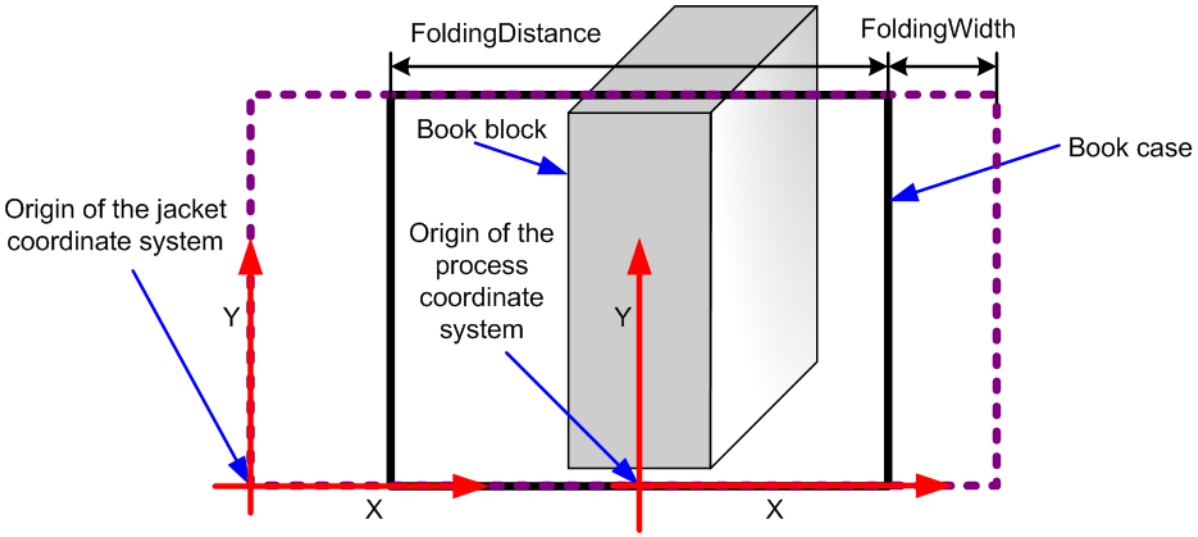


Figure 8-38: Parameters and coordinate system for jacketing



Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **Jacketing**

Table 8.137: JacketingParams Resource

NAME	DATA TYPE	DESCRIPTION
FoldingDistance ? New in JDF 1.4	double	Distance from the fold at @*FoldingWidth* to the other fold. If not specified, it defaults to width of the Jacket minus two times @*FoldingWidth* (symmetrical folds).
FoldingWidth	double	Definition of the dimension of the folding width of the front cover fold (see @*FoldingWidth* in the picture above). All other measurements are implied by the dimensions of the book.

8.82 LabelingParams

New in JDF 1.1

LabelingParams defines the details of the **Labeling** process.

Resource Properties

Resource Class: Parameter

Input of Processes: **Labeling**

Table 8.138: LabelingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Application ?	NMTOKEN	Application method of the label. **Values include:** Glue – Glued onto the component. Loose – Loosely laid onto the component. SelfAdhesive – Self adhesive label. Staple – Stapled onto the component.
CTM ? Deprecated in JDF 1.6	matrix	Position and orientation of the label lower-left-corner relative to the lower left corner of the component surface as defined by @*Position*. **Deprecation note:** Use @*Offset* and ***ComponentLink***(Label)/@*Orientation*.
Face ? New in JDF 1.6	enumeration	Position of the label on the bundle. **Allowed value is from:** ▸ Face.

Table 8.138: LabelingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Offset ?	XYPair	Position of the lower-left-corner of the label after applying the rotation defined in ComponentLink(Label)/@*Orientation* relative to the lower left corner of the component surface as defined by @*Position*.
Position ?	enumeration	Position of the label on the bundle. **Allowed value is from:** ▸ Face.
FileSpec ? New in JDF 1.5	refelement	A ***FileSpec*** resource pointing to an address list. The format of the referenced mailing list is implementation dependent.

8.83 LaminatingParams

New in JDF 1.1

This resource specifies the parameters needed for laminating.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***LaminatingIntent***

Example Partition: "SheetName", "Side"

Input of Processes: **Laminating**

Table 8.139: LaminatingParams Resource

NAME	DATA TYPE	DESCRIPTION
AdhesiveType ?	string	Type of adhesive used. Valid only when @*LaminatingMethod* = "DispersionGlue".
GapList ?	DoubleList	List of non-laminated gap positions in the X direction of the laminating tool in the coordinate system of the ***Component***. The zero-based even entries define the absolute position of the start of a gap, and the odd entries define the end of a gap. If not specified, the complete area defined by @*LaminatingBox* is laminated.
HardenerType ?	string	Type of hardener used. Valid only when @*LaminatingMethod* = "DispersionGlue".
LaminatingBox ? Modified in JDF 1.4	rectangle	Area on the ***Component*** to be laminated. **Modification note:** Starting with **JDF** 1.4, @*LaminatingBox* becomes optional to enable **Laminating** yes/no style definitions.
LaminatingMethod ?	enumeration	Laminating technology that is applied. **Allowed values are:** CompoundFoil DispersionGlue Fusing – New in JDF 1.3 Unknown – Deprecated in JDF 1.2
ModuleIndex ? New in JDF 1.4	integer	Index of the laminating module in the press. See ***ConventionalPrintingParams***. In a combined process, all modules of the device, including press modules, finishing modules and varnishing modules are counted to calculate @*ModuleIndex*.
NipWidth ? New in JDF 1.3	double	Width of the nip in points to be formed between the fusing rollers and the component in the **Laminating** process.
Temperature ?	double	Temperature used in the **Laminating** process, in ° Centigrade.

8.84 Layout

Represents the root of the layout structure. The ***Layout*** is used both for fixed-layout and for automated printing.

Resource Properties

Resource Class: Parameter

RESOURCES

Resource referenced by: ***LayoutIntent***, ***Component***, ***CylinderLayout***, ***InsertSheet***

Example Partition: "SignatureName", "WebName", "RibbonName", "SheetName", "Side", "PartVersion"

Input of Processes: **ConventionalPrinting**, **CylinderLayoutPreparation**, **DigitalPrinting**, **Imposition**, **InkZoneCalculation**

Output of Processes: **LayoutPreparation**, **Stripping**

Table 8.140: Layout Resource (Sheet 1 of 4)

NAME	DATA TYPE	DESCRIPTION
Automated ?	boolean	If "true", the **Imposition** process is expected to perform automated imposition. ***Layout***/@*Automated* SHALL only be specified in the root partition of ***Layout***. **Default value is:** "false" in the root Partition of ***Layout***
BaseOrdReset = "PagePool" New in JDF 1.4	enumeration	Policy about how the @*Ord* attribute of an entry SHALL be calculated when extracting a page from a ***RunList*** and positioning it in the ***Layout***. **Allowed values are:** PagePool – The ▸ Base Ord is reset to point to the first page entry of the ▸ Page Pool at the beginning of each ▸ Page Pool processed by the ▸ Imposition Template. PagePoolList – At the beginning of processing of the ▸ Imposition Template, the ▸ Base Ord is reset to point to the first page entry of the first ▸ Page Pool to be processed by the ▸ Imposition Template. This results in all ▸ Page Pools that will be processed by the ▸ Imposition Template to be treated as a ▸ Page Pool List.
LockOrigins ?= "false" New in JDF 1.3	boolean	Determines the relationship of the coordinate systems for front and back surfaces. When "false", all contents for all surfaces are transformed into the first quadrant, in which the origin is at the lower left corner of the surface. When "true", contents for the front surface are imaged into the first quadrant (as above), but contents for the back surface are imaged into the second quadrant, in which the origin is at the lower right. This allows the front and back origins to be aligned even if the exact media size is unknown. The @*LockOrigins* was copied from the deprecated ***Sheet*** resource.
MaxCollect ? New in JDF 1.4	integer	Maximum number of Sheets that will be collected into a signature. @*MaxCollect* modifies the pagination when automated imposition is selected. Specifying @*MaxCollect* can effectively cause a ▸ Page Pool or ▸ Page Pool List to be broken into "sub" ▸ Page Pools. Each of these "sub" ▸ Page Pools provides the set of pages mapped onto a single Collect, and are processed sequentially out of the "parent" ▸ Page Pool (or ▸ Page Pool List). Thus each sub- ▸ Page Pool effectively restarts the ord counting within the ▸ Imposition Template (i.e., treat a sub-Page Pool as if a new ▸ Page Pool were being started with the Imposition Template). If not specified, all sheets SHALL be collected.
MaxDocOrd = "1" New in JDF 1.1 Deprecated in JDF 1.4	integer	Zero-based maximum number of Instance Documents that are consumed from a ***RunList*** each time the ***Layout*** is executed, assuming the **Imposition** process is automated. **Deprecation note:** See @*MaxOrd*.
MaxOrd ? Deprecated in JDF 1.4	integer	Zero-based maximum number of placed objects that are consumed from a ***RunList*** each time the ***Layout*** is executed, assuming the **Imposition** process is automated. If not specified, it SHALL be calculated from the @*Ord* values of the ***ContentObject*** elements in the ***Layout***. **Deprecation note:** @*MaxOrd* has no meaning if negative @*Ord* values exist in an automated ***Layout***. The consumer SHALL calculate the implied 2 values for increasing and decreasing the explicit @*Ord* values in an automated ***Layout*** by evaluating the actual values of ***ContentObject***/@*Ord*. Increment from Front = 1+max(***ContentObject***/@*Ord*_+) where "@*Ord*_+" specifies positive values of @*Ord*; Decrement from Back = max (abs ((***ContentObject***/@*Ord*_-)) where "@*Ord*_-" specifies negative values of @*Ord*. See ▸ Section 6.3.18 Imposition for details.

Table 8.140: Layout Resource (Sheet 2 of 4)

NAME	DATA TYPE	DESCRIPTION
MaxSetOrd = "1" New in JDF 1.1 Deprecated in JDF 1.4	integer	Zero-based maximum number of Document Sets that are consumed from a ***RunList*** each time the ***Layout*** is executed, assuming the **Imposition** process is automated. **Deprecation note:** See @*MaxOrd*.
MinCollect ? New in JDF 1.4	integer	Minimum number of Sheets that will be collected into a signature. @*MinCollect* modifies the pagination when automated imposition is selected.
Name ? New in JDF 1.1 Deprecated in JDF 1.4	string	Unique name of the ***Layout***. The @*Name* is used for external reference to a ***Layout***. **Deprecation note:** Starting with **JDF** 1.4, use @*DescriptiveName*.
NonPrintableMargin Bottom ?	double	The width in points of the bottom margin measured inward from the edge of the paper ***Media*** with respect to the idealized process coordinate system of the final printing process. The ***Media*** origin is unaffected by @*NonPrintableMarginBottom*. These margins are independent of the PDL content.
NonPrintableMargin Left ?	double	Same as @*NonPrintableMarginBottom* except for the left margin.
NonPrintableMargin Right ?	double	Same as @*NonPrintableMarginBottom* except for the right margin.
NonPrintableMargin Top ?	double	Same as @*NonPrintableMarginBottom* except for the top margin.
OrdsConsumed ? New in JDF 1.4	IntegerRangeList	Range of @*Ord* values of the ***RunList*** (Document) that are consumed by this ***Layout*** section. SHALL NOT be specified unless @*Automated* = "true".
SheetCountReset ? New in JDF 1.4	enumeration	Policy as to when the automated imposition variables @*SheetCount* and @*TotalSheetCount* are reset. See ▸ Section 6.3.18.2 Variables for Automated Imposition. **Allowed values are:** Continue – @*SheetCount* continues to increment for each sheet generated by the current ▸ Imposition Template. PagePool – @*SheetCount* is reset to zero upon start of processing of a new ▸ Page Pool and @*TotalSheetCount* is determined for that new ▸ Page Pool.. PagePoolList – @*SheetCount* is reset to zero upon start of processing of an ▸ Imposition Template, and @*TotalSheetCount* is recalculated. Note that the value of @*TotalSheetCount* may depend on the sheets generated from successive ▸ Imposition Templates (for example, if the current ▸ Imposition Template has @*SheetCountReset* = "PagePoolList", and the subsequent ▸ Imposition Template has @*SheetCountReset* = "Continue", @*TotalSheetCount* will include the sheets generated by both ▸ Imposition Templates. **Note:** @*SheetCount* and @*TotalSheetCount* are always reset to zero at the beginning of processing of a set regardless of the value of ***Layout***/@*SheetCountReset*.
SheetNameFormat ? New in JDF 1.4	string	A formatting string used with @*SheetNameTemplate* to algorithmically construct @*SheetName*. @*SheetNameFormat* and @*SheetNameTemplate* are used to identify individual parts of the ***Layout*** in an automated environment. **Allowed values are from:** ▸ Appendix H Dynamic String Generation.
SheetNameTemplate ? New in JDF 1.4	string	A list of values used with @*SheetNameFormat* to algorithmically construct @*SheetName*. @*SheetNameFormat* and @*SheetNameTemplate* are used to identify individual parts of the ***Layout*** in an automated environment. **Allowed values are from:** ▸ Appendix H Dynamic String Generation.

Table 8.140: Layout Resource (Sheet 3 of 4)

NAME	DATA TYPE	DESCRIPTION
SourceWorkStyle? New in JDF 1.3	enumeration	Indicates which @*WorkStyle* was used to create the ***Layout***. This is only informative and can be useful when creating double sided proofs. **Allowed value is from:** ▸ WorkStyle**.**
SurfaceContentsBox ? New in JDF 1.3	rectangle	This box, specified in ***Layout***-coordinate space, defines the area into which ***MarkObject*** or ***ContentObject*** elements are distributed.The lower left corner of the rectangle specified by the value of this attribute establishes the coordinate system into which the content is mapped and SHOULD have a value of "0 0". @*SurfaceContentsBox* MAY imply clipping. This attribute SHOULD be supplied in order to get predictable placement of content. If this attribute is not supplied, a rectangle with the origin at "0 0" and an extent that MAY be dependent on the dimensions of one of the ***Media*** is implied.
TemplateType = "Normal" New in JDF 1.4	enumeration	Specifies the type of automated ▸ Imposition Template being defined. If @*TemplateType* ="ConditionalSheets", then this ▸ Imposition Template SHALL only specify conditional sheet definitions (see ***Layout***/***SheetCondition***). Typically, such an ▸ Imposition Template defines conditional sheets to be generated at the beginning and/or end of job and/or set. SHALL ONLY be specified if ***Layout***/@*Automated*="true". **Allowed values are:** ConditionalSheets – the ▸ Imposition Template contains ONLY conditional sheet definitions Normal – the ▸ Imposition Template contains at least one sheet definition that consumes pages from the ***RunList*** (Document), and may contain conditional sheet definitions. If this value is specified in a partition leaf, this leaf SHALL NOT contain conditional sheet definitions.
InsertSheet * Deprecated in JDF 1.4	refelement	Additional Sheets that are to be inserted before and/or after a document. Depending on which Partition level the ***InsertSheet*** is defined, it specifies how to complete the Sheet or surface in an automated printing environment. **Deprecation note:** Starting with **JDF** 1.4, use ***Layout***/***PageCondition*** for "FillSheet", "FillSurface", and "FillSignature" operations; use ***Layout***/***SheetCondition*** for an insert sheet.
LayerList ? New in JDF 1.1	element	List of ***LayerDetails*** elements.
LogicalStackParams ? New in JDF 1.4	element	When specified, configures the imposition engine to place content onto one or more ▸ Logical Stacks distributed on a common set of sheets. ***Layout***/***LogicalStackParams*** SHALL only be specified in the root ***Layout*** element AND only when ***Layout***/@*Automated* = "true". All ▸ Logical Stacks defined by ***LogicalStackParams*** SHALL be used in all ▸ Imposition Templates. See ▸ Section 6.3.18.4.1 Using Logical Stacks.
Media * New in JDF 1.1 Modified in JDF 1.3	refelement	Describes the media to be used. If multiple ***Media*** are specified, ***Media***/@*MediaType* species the type of ***Media***, typically Paper, Plate or Film. Multiple ***Media*** with the same ***Media***/@*MediaType* SHALL NOT be specified in one ***Layout***. Note that at least one ***Media*** SHALL be specified in the Partitioned ***Layout*** tree in **JDF** 1.3 or above.
MediaSource ? Deprecated in JDF 1.1	refelement	Describes the media to be used. Replaced by ***Media*** in **JDF** 1.1.
PageCondition * New in JDF 1.4	element	The ***PageCondition*** elements are used only with automated imposition. They define restrictions on which page content may be placed in a ***Layout***/***ContentObject*** and. If any ***PageCondition*** restricts placing a page into a ***ContentObject***, the Page SHALL NOT be filled into that ***ContentObject***.

Table 8.140: Layout Resource (Sheet 4 of 4)

NAME	DATA TYPE	DESCRIPTION
PlacedObject * New in JDF 1.3	element	Provides a list of the ***PlacedObject*** (i.e., ***ContentObject*** and ***MarkObject***) elements to be placed on to the surface. Contains the marks on the surface in rendering order. All ***PlacedObject*** elements SHALL be specified in the partition leaves of the ***Layout***. See ▸ Section 8.84.16.1.2 Position of PlacedObject Elements in Layout. **Note:** ***PlacedObject*** is not a container but an abstract type.
SheetCondition ? New in JDF 1.4	element	Specifies the conditions under which the optional sheet defined by this ***Layout*** is produced. @***SheetCondition*** SHALL only be present when ***Layout***/@*Automated*="true".
Signature * Deprecated in JDF 1.3	element	The ***Signature*** element has been replaced by a ***Layout*** Partition, namely ***Layout*** [@*SignatureName*]. In **JDF** 1.3 and beyond, ***Signature***/@*Name* has been replaced by the Partition Key ***Layout***/@*SignatureName*.
TransferCurvePool ? New in JDF 1.1	refelement	Describes the relationship of transfer curves and coordinate systems within the various processes.

8.84.1 LayerList

New in JDF 1.1

This element provides a container for an ordered list of ***LayerDetails*** elements. The individual elements are referenced by their zero-based index in the ***LayerList*** using the @*LayerIDs* Partition Key.

Table 8.141: LayerList Element

NAME	DATA TYPE	DESCRIPTION
LayerDetails *	element	Details of the individual layers.

8.84.2 LayerDetails

New in JDF 1.1

This element provides information about individual layers.

Table 8.142: LayerDetails Element

NAME	DATA TYPE	DESCRIPTION
Name ?	string	Unique name of the layer.

8.84.3 LogicalStackParams

New in JDF 1.4

Table 8.143: LogicalStackParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
MaxStackDepth ?	integer	Maximum number of imposed sheets to generate as an ▸ Imposed Sheet Set (the size of the ▸ Logical Stack). Implementations SHALL generate the minimum stack size to accommodate the available number of ▸ Logical Sheets if the total number of required sheets for the last stack is smaller than @*MaxStackDepth*.

Table 8.143: LogicalStackParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Restrictions = "None"	enumeration	Describes any restrictions set on the placement of a ▸ Recipient Set's ▸ Logical Sheets within or across ▸ Imposed Sheet Sets. **Allowed values are:** None – a recipient set's ▸ Logical Sheets may be placed across both ▸ Logical Stacks and ▸ Imposed Sheet Sets. WithinImposedSheetSet – a ▸ Recipient Set's ▸ Logical Sheets SHALL be placed within a single ▸ Imposed Sheet Set WithinLogicalStack – a ▸ Recipient Set's ▸ Logical Sheets SHALL be placed within a single ▸ Logical Stack.
Stack +	element	Describes parameters to control the sequencing of ▸ Logical Sheets onto individual ▸ Logical Stacks.

8.84.4 Stack

New in JDF 1.4

Table 8.144: Stack Element

NAME	DATA TYPE	DESCRIPTION
LogicalStackOrd	integer	0-based ▸ Logical Stack identifier that specifies which ▸ Logical Stack is controlled by this ***Stack*** element. The value of ***Stack***/@*LogicalStackOrd* SHALL correspond to a ***PlacedObject***/@*LogicalStackOrd* value.
LogicalStackSequence = "SheetIndex"	enumeration	Specifies how ▸ Logical Sheets SHALL be placed onto the ▸ Logical Stack. **Allowed values are:** SheetIndex – ▸ Logical Sheets are placed in the order of ascending @*SheetIndex*. DescendingSheetIndex – ▸ Logical Sheets are placed in the order of descending @*SheetIndex*.

8.84.5 PageCondition

New in JDF 1.4

The ***PageCondition*** element defines restrictions on when page content SHALL NOT be placed in a***ContentObject*** of a ***Layout***. Before placing page content from a ***RunList*** into a ***ContentObject*** the ***PageCondition***/@*RestrictedContentObjects* attribute SHALL be checked for the @*Ord* of the ***ContentObject***. If the @*Ord* of the ***ContentObject*** is in the @*RestrictedContentObjects* attribute value, the alternate content, if any, SHALL be placed in the ***ContentObject***. After skipping a restricted ***ContentObject***, the **Imposition** process SHALL then place the current page content into the location defined by the next ***ContentObject*** (after that specified by the @*RestrictedContentObject*). This corresponds to incrementing the effective @*Ord* value of the page in the ***RunList*** by 1, effectively incrementing the total number of pages of the ***RunList***. If the next ***ContentObject*** is also restricted then the process is repeated. ***PageCondition*** elements are processed in their XML order.

Table 8.145: PageCondition Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Condition ?	enumeration	Specifies the conditions when the ***PageCondition*** applies. Condition SHALL NOT be specified if ***Part*** elements are present. **Allowed values are:** PagePoolStart – the condition is true when the @*Ord* refers to the first page of a ▸ Page Pool in the ***RunList***. PagePoolEnd – after processing of the ▸ Page Pool is completed, the condition is true for all unused @*Ord* positions in the current Collect. PagePoolListStart – the condition is true when the @*Ord* refers to the first page of an aggregated set of ▸ Page Pools in the ***RunList***. PagePoolListEnd – after processing of the ▸ Page Pool list is completed, the condition is true for all unused Ord positions in the current Collect.

Table 8.145: PageCondition Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
RestrictedContentObjects	IntegerList	List of @*Ord* values of those ***ContentObject*** elements into which page content that matches the conditions as specified in ***Part*** or ***PageCondition***/@*Condition* SHALL NOT be placed.
Part *	element	Specifies the conditions when the ***PageCondition*** applies. Multiple ***Part*** elements specify alternate page conditions (ORing of them). ***Part*** elements SHALL NOT be specified if @*Condition* is present.
RunList ?	refelement	Alternate page content that SHALL be placed into the ***ContentObject*** elements that are specified in @*RestrictedContentObjects* when the ***PageCondition*** evaluates to "true". The first page of the referenced ***RunList*** SHALL be used. **Note:** The behavior of providing alternate content using ***RunList*** is defined only if @*Condition* is specified.

Example 8.16: PageCondition

New in JDF 1.4

```
<Layout Class="Parameter" ID="L000004" Status="Available"
          PartIDKeys="SheetName Side" BaseOrdReset="PagePoolList">
    <PageCondition RestrictedContentObjects="1 -1">
    <!--
       This example assumes that the pages of a sequence of documents of the
       RunList are to be treated as an aggregate page pool, and the pages are
       to be saddle stitch imposed onto a continuous sequence of sheets. Some
       documents of the sequence represent a start of a new chapter where their
       DocTag is set to the value 'Chapter'. These chapter starts force the
       first page of each chapter to be placed on the right side finished page.
    -->
       <Part DocTags="Chapter" DocRunIndex="0"/>
    </PageCondition>
    <Layout SheetName="Mysheet">
       <Layout Side="Front">
          <ContentObject CTM="1 0 0 1 0 0" Ord="-1"/>   <!-- Outside left -->
          <ContentObject CTM="1 0 0 1 595 0" Ord="0" /> <!-- outside right -->
       </Layout>
       <Layout Side="Back">
          <ContentObject CTM="1 0 0 1 0 0" Ord="1"/>    <!-- inside left-->
          <ContentObject CTM="1 0 0 1 595 0" Ord="-2"/> <!-- inside right-->
       </Layout>
    </Layout>
</Layout>
```

8.84.6 PlacedObject

The marks that are to be placed on the designated surface of a ***Layout*** come in two varieties: ***ContentObject*** or ***MarkObject*** elements. All inherit characteristics from the ***AbstractPlacedObject*** which is described below.

8.84.6.1 AbstractPlacedObject

Table 8.146: Abstract PlacedObject Element (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
Anchor ? New in JDF 1.4	enumeration	Specifies the anchor point of the ***PlacedObject*** that remains in place on the surface when the value of @*TrimSize* changes. @*Anchor* is specified in the coordinate system of the ***PlacedObject*** prior to application of the @*CTM*. **Note:** The @*Anchor* attribute is metadata used to identify to an Imposition generation utility a fixed anchor point reference to an abstract content page. This may occur when a **JDF** ***Layout*** resource is used as a template for that utility. This attribute has no effect on processing when a ***Layout*** resource is input to the **Imposition** process.**Allowed value is from:** ‣ Anchor**.**

Table 8.146: Abstract PlacedObject Element (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
AssemblyIDs ? New in JDF 1.5	NMTOKENS	Identification of the ***Assembly*** elements if **Stripping** describes an imposition scheme for multiple ***Assembly*** elements. @*AssemblyIDs* MAY contain multiple NMTOKENS, when the ***Assembly*** resource specifies an intermediate product that contains multiple final assemblies***BinderySignature*** .
ClipBox ?	rectangle	Clipping rectangle in the coordinates of the @*SurfaceContentsBox*. @*ClipBox* SHALL NOT be present if ***PlacedObject***/@*ClipBoxFormat* is supplied.
ClipBoxFormat ? New in JDF 1.4	string	A formatting string used with @*ClipBoxTemplate* to algorithmically construct @*ClipBox*. @*ClipBoxFormat* SHALL ONLY be present if ***PlacedObject***/@*ClipBox* is not supplied and ***Layout***/@*Automated* = "true". **Allowed values are from:** ▸ Appendix H Dynamic String Generation**.**
ClipBoxTemplate ? New in JDF 1.4	string	A list of values used with @*ClipBoxFormat* to algorithmically construct @*ClipBox*. @*ClipBoxTemplate* SHALL ONLY be present if ***PlacedObject***/@*ClipBox* is not supplied and ***Layout***/@*Automated* = "true". **Allowed values are from:** ▸ Appendix H Dynamic String Generation**.**
ClipPath ? New in JDF 1.3	PDFPath	Clip path for the ***PlacedObject*** in the coordinates of the @*SurfaceContentsBox* (lower left of @*SurfaceContentsBox* is used as reference zero point, same as for @*ClipBox*). The actual clip region is the intersection of @*ClipBox* and @*ClipPath* or the intersection of @*ClipBox* and @*SourceClipPath*. Thus both clip paths are applied sequentially and the resulting clip region is smaller than each individual clip box or Path. @*ClipPath* and @*SourceClipPath* SHALL NOT be specified in the same ***PlacedObject***. @*ClipPath* SHOULD be specified when both @*ClipPath* and @*SourceClipPath* are known because @*ClipPath* provides a more stable coordinate system (not sensitive to shifts caused by editing the page).
CompensationCTMFormat ? New in JDF 1.4	string	A formatting string used with @*CompensationCTMTemplate* to algorithmically construct a compensation CTM that SHALL be concatenated to CTM. @*CompensationCTMFormat* MAY be present if ***Layout***/@*Automated* = "true". **Allowed values are from:** ▸ Appendix H Dynamic String Generation**.**
CompensationCTMTemplate ? New in JDF 1.4	string	A list of values used with @*CompensationCTMFormat* to algorithmically construct a compensation CTM that SHALL be concatenated to CTM. @*CompensationCTMTemplate* MAY be present if ***Layout***/@*Automated* = "true". **Allowed values are from:** ▸ Appendix H Dynamic String Generation**.**
CTM	matrix	The coordinate transformation matrix (CTM — a Postscript term) of the object in the @*SurfaceContentsBox*. For details, see ▸ Figure 2-8: Equation for Surface Coordinate System Transformations. The origin of the source coordinate system is the lower left (expressed in the source coordinate system) of the object and the origin of the destination coordinate system is lower left of the @*SurfaceContentsBox*. For details, see ▸ Section 2.5.1.1 Source Coordinate Systems. **Note:** @*CTM* SHALL be recalculated if the object is replaced afterwards with a new object with different dimensions.
HalfTonePhaseOrigin = "0 0"	XYPair	Location of the origin for screening of this ***ContentObject***. Specified in the coordinate systems of @*SurfaceContentsBox*.
LayerID ? New in JDF 1.1	integer	If a ***Layout*** supports layering (e.g., for versioning), @*LayerID* specifies the index of the ***Layout***/***LayerList***/***LayerDetails*** element in ***Layout***/***LayerList*** that a ***ContentObject***belongs to (e.g., the language layer version). The details of the layers are specified in the ***Layout***/***LayerList***/***LayerDetails*** element.
LogicalStackOrd ? New in JDF 1.4	integer	0-based ▸ Logical Stack identifier that this ***PlacedObject*** belongs to. @*LogicalStackOrd* SHALL match the @*LogicalStackOrd* of an entry in ***Layout***/***LogicalStackParams***/***Stack***.
OrdID ? New in JDF 1.1	integer	If a ***Layout*** supports layering (e.g., for versioning), elements that belong to the same final page SHOULD have a matching @*OrdID*.

Table 8.146: Abstract PlacedObject Element (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
SourceClipPath ? Modified in JDF 1.3	PDFPath	Clip path for the ***PlacedObject*** in the source coordinate system. @*SourceClipPath* is applied to the referenced source object in addition to any clipping that is internal to the object. Internal transformation of the source object (Rotation key in PDF, Orientation Tag in TIFF etc.) SHALL be applied prior to applying @*SourceClipPath*. @*ClipPath* and @*SourceClipPath* SHALL NOT be specified in the same ***PlacedObject***. See ▸ Section 2.5.1.1 Source Coordinate Systems for definitions of source coordinate systems.
TrimClipPath ? New in JDF 1.4	PDFPath	The die cutting path for the ***PlacedObject*** in the coordinates of the @*SurfaceContentsBox* (lower left of @*SurfaceContentsBox* is used as reference zero point, same as for @*ClipBox*). That path can be used for proofing purpose. **Note:** The @*TrimClipPath* attribute may be used by an Imposition generation utility when a **JDF** ***Layout*** resource is used as a template for that utility. This attribute has no effect on processing when a ***Layout*** resource is input to the **Imposition** process.
TrimCTM ? New in JDF 1.1	matrix	The transformation matrix of the trim box to be applied to the object's referenced content in the coordinate system of @*SurfaceContentsBox*. Note that imposition programs that execute the ***Layout*** SHALL recalculate the @*CTM* in case the referenced content is replaced with new referenced content having different dimensions, otherwise the position of the content inside the trim box will shift. This recalculation is based on @*Anchor*, @*TrimCTM*, @*TrimSize* and trim box. **Note:** The @*TrimCTM* attribute may be used by an Imposition generation utility when a **JDF** ***Layout*** resource is used as a template for that utility. This attribute has no effect on processing when a ***Layout*** resource is input to the **Imposition** process.
TrimSize ? New in JDF 1.2 Modified in JDF 1.4	XYPair	The size of the object's trim box as viewed in the object source coordinates (@*TrimCTM* scaling and rotation NOT applied). Modified in JDF 1.4 @*TrimSize* is needed when replacing the object by a new object with a different dimension. When a ***Layout*** resource is input to the **Imposition** process, @*TrimSize* specifies the bounding box to be used for text layout when processing a ***MarkObject***/***DeviceMark*** or for scaling and rotation when processing ***PlacedObject***/***FitPolicy***. **Note:** Recalculation of ***PlacedObject***/@*CTM* is only necessary when the **Stripping** process or application needs to replace some pages from the provided ***RunList*** (using the ***Layout*** as a kind of imposition "template"). To ensure correct placement of a new page in the ***Layout***, ***PlacedObject***/@*CTM* recalculations SHOULD always be done according to ***PlacedObject***/@*TrimCTM* and ***PlacedObject***/@*TrimSize*. Together, these two attributes represent the trimming information of the imposition software page, which is not always the same as the original ***RunList*** page trimming information (= ***LayoutElement***/@*SourceTrimBox* when real trim box of the object is known). Usage of both ***PlacedObject*** elements @*TrimCTM* and @*TrimSize* attributes will allow page replacements on any type of imposition ***Layout***.
Type ? Deprecated in JDF 1.1	enumeration	Describes the kind of ***PlacedObject***. **Allowed values are:** Content Mark
FitPolicy ? New in JDF 1.4	element	SHALL NOT be present when ***Layout***/@*Automated* = "false". Specifies automated fit policy for the page cell described by the ***PlacedObject***. When present, ***PlacedObject***/@*TrimSize* SHALL also be present in the ***PlacedObject***, and represents the cell size for this ***PlacedObject***.

8.84.7 ContentObject

ContentObject elements describe containers for page content on a surface. They are filled from the content ***RunList*** of the **Imposition** process. For print applications where page count varies from Instance Document to Instance Document, imposition templates can automatically assign pages to the correct surface and ***PlacedObject*** position.

Table 8.147: ContentObject Element

NAME	DATA TYPE	DESCRIPTION
DocOrd ? New in JDF 1.1	integer	Reference to an index of an Instance Document in the content ***RunList***. This references an Instance Document with an index module. ***Layout***/@*MaxDocOrd* equals @*DocOrd* in an automated layout scenario. The index can either be known explicitly from a variable ***RunList*** or implicitly from the index within an indexable content definition language (e.g., PPML).
ID ? New in JDF 1.5	ID	Identifier for referencing this ***ContentObject*** from ***MarkObject***/@*ContentRef*.
Ord ? Modified in JDF 1.4	integer	A zero-based reference to an index in the content ***RunList***. The index is incremented for every page of the ***RunList*** with @*IsPage* = "true". The @*Ord* value of the first page of a ***RunList*** has the value "0". If ***Layout***/@*Automated* = "true", @*Ord* MAY be a negative integer in a ***ContentObject***. In this case, the explicit @*Ord* for each iteration of the automated ***Layout*** is calculated by subtracting the appropriate number of @*Ord* values from the back of the document. For details on automated ***Layout***, see ▸ Section 6.3.18 Imposition.
OrdExpression ?	string	Function to calculate an @*Ord* value dynamically, using a value of *s* for signature number and *n* for total number of pages in the instance document. The @*Ord* or @*DocOrd* and @*OrdExpression* are mutually exclusive in one ***PlacedObject***. **Value format is from:** ▸ Section 8.84.16.5 Using Expressions in the OrdExpression Attribute**.**
SetOrd ? New in JDF 1.1	integer	A non-negative, zero-based reference to an index of a Document Set in the content ***RunList***. This references an Instance Document with an index module. ***Layout***/@*MaxSetOrd* = @*SetOrd* in an automated layout scenario. The index can either be known explicitly from a variable ***RunList*** or implicitly from the index within an indexable content definition language (e.g., PPML).

8.84.8 MarkObject

MarkObject elements describe containers for page marks on a surface. The PDL for the marks SHOULD exist prior to imposingand SHOULD be filled from the ***RunList*** (Marks) of the **Imposition** process. An individual ***MarkObject*** represents the content data of the Marks. The content data in individual ***MarkObject*** elements MAY contain multiple logical marks.

Table 8.148: MarkObject Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ContentRef ? New in JDF 1.4	IDREF	@*ContentRef* refers to the ***ContentObject*** that this ***MarkObject*** is related to. @*ContentRef* is used to define the object that metadata for generating dynamic marks MAY be extracted from.
LayoutElementPageNum ? New in JDF 1.1 Modified in JDF 1.3 Deprecated in JDF 1.4	integer	Page number to use from the PDL file described by ***LayoutElement***. **Modification note:** Starting with **JDF** 1.3, the default value of "0" is removed. **Deprecation note:** Starting with **JDF** 1.4, PDL for Marks SHOULD be referenced via ***RunList*** (Marks).
Ord ? Modified in JDF 1.4	integer	A non-negative reference to an index in the ***RunList*** (Marks). The index is incremented for every page of the ***RunList*** with @*IsPage* = "true". The first page of a ***RunList*** has the value 0. **Modification note:** Starting with **JDF** 1.4, at most one of @*Ord* or ***DeviceMark*** SHALL be specified. For **JDF** 1.3 only, at most one of ***LayoutElement***, @*Ord* or ***JobField*** SHALL be specified.

Table 8.148: MarkObject Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
CIELABMeasuringField *	element	Specific information about this kind of mark object. See below for information regarding dynamically generated marks.
ColorControlStrip * Modified in JDF 1.1	refelement	Specific information about this kind of mark object. See below for information regarding dynamically generated marks.
CutMark * Modified in JDF 1.1	refelement	Specific information about this kind of mark object. See below for information regarding dynamically generated marks.
DeviceMark ? New in JDF 1.4	element	***DeviceMark*** specifies all formatting options for dynamic Marks. ***JobField***/***DeviceMark*** specifies the formatting parameters of the ***JobField*** and all other dynamically generated marks are positioned with @*CTM*. **Constraint:** at most one of @*Ord* or ***DeviceMark*** SHALL be specified. **Creation note:** Starting with **JDF** 1.4, ***DeviceMark*** is back after being deprecated in **JDF** 1.3.
DynamicField *	element	Definition of text replacement for a ***MarkObject***. ***MarkObject***/***DynamicField*** specifies text replacement within an existing PDL mark.
FillMark * New in JDF 1.5	element	Specifies marks that define a fill layer, e.g., for backlit displays.
IdentificationField *	refelement	Specific information about this kind of mark object. See below for information regarding dynamically generated marks.
JobField ? New in JDF 1.1 Modified in JDF 1.4	element	***JobField*** specifies the metadata of a given dynamic slug line. **Modification note:** Starting with **JDF** 1.4, the maximum number of ***JobField*** elements per ***MarkObject*** is limited to 1; previously, there was no limit. For **JDF** 1.3 only, at most one of ***LayoutElement***, @*Ord* or ***JobField*** SHOULD be specified.
LayoutElement ? Deprecated in JDF 1.4	refelement	PDL description of the mark. The ***LayoutElement*** and @*Ord* are mutually exclusive within one ***MarkObject***. **Modification note:** For **JDF** 1.3 only, at most one of ***LayoutElement***, @*Ord* or ***JobField*** SHALL be specified. **Deprecation note:** Starting with **JDF** 1.4, PDL for Marks SHALL be referenced via ***RunList***(Marks).
MarkActivation * New in JDF 1.4	element	Rules about when to apply the mark in an automated ***Layout***. If no ***MarkActivation*** is specified, the ***MarkObject*** is unconditionally active. If multiple ***MarkActivation*** elements are specified, all conditions SHALL be met for the mark to be active. ***MarkActivation*** SHALL NOT be specified unless ***Layout***/@*Automated* = "true".
RefAnchor ? New in JDF 1.4	element	Details of the coordinate system that this mark is placed relative to. This MAY be either the sheet coordinate system or the coordinate system of a referenced ***PlacedObject***. If the anchor point in the referenced object (***PlacedObject*** or sheet surface) is modified (e.g., due to a change in @*TrimSize*), the CTM of the placed object of this ***DeviceMark*** SHALL be modified accordingly. **Note:** ***RefAnchor*** does NOT modify the origin of the CTM of this ***PlacedObject***. It is only used to recalculate relative shifts.
RegisterMark * Modified in JDF 1.1	refelement	Specific information about this kind of mark object. See below for information regarding dynamically generated marks.
ScavengerArea * New in JDF 1.1	refelement	Specific information about this kind of mark object See below for information regarding dynamically generated marks.

8.84.9 CIELABMeasuringField

Information about a color measuring field. The color is specified as CIE-L*a*b* value.

Table 8.149: CIELABMeasuringField Element

NAME	DATA TYPE	DESCRIPTION
Center	XYPair	Position of the center of the color measuring field in the coordinates of the ***MarkObject*** that contains this mark. If the measuring field is defined within a ***ColorControlStrip***, @*Center* refers to the rectangle defined by @*Center* and @*Size* of the ***ColorControlStrip***.
CIELab	LabColor	L*a*b* color specification.
DensityStandard ? Deprecated in JDF 1.1	enumeration	Density filter standard used during density measurements. **Allowed values are:** ANSIA – ANSI Status A ANSIE – ANSI Status E ANSII – ANSI Status I ANSIT – ANSI Status T. DIN16536 DIN16536NB **Deprecation note:** Starting with **JDF** 1.1, use ***ColorMeasurementConditions***/@*DensityStandard*.
Diameter ? Modified in JDF 1.1	double	Diameter of the measuring field.
Light Deprecated in JDF 1.1	NMTOKEN	Type of light. **Values include:** D50 D65
Observer ? Deprecated in JDF 1.1	integer	Observer in degree (2 or 10). In **JDF** 1.1 and beyond, use ***ColorMeasurementConditions***/@*Observer*.
Percentages ?	DoubleList	Percentage values for each separation. The number of array elements SHALL match the number of separations.
ScreenRuling ?	DoubleList	Screen ruling values in lines per inch for each separation. The number of array elements SHALL match the number of separations.
ScreenShape ?	string	Shape of screening dots.
Setup ? Deprecated in JDF 1.1	string	Description of measurement setup. **Deprecation note:** Starting with **JDF** 1.1, use details from ***ColorMeasurementConditions***
Tolerance ? Modified in JDF 1.1	double	Tolerance in ΔE.
ColorMeasurementConditions ? New in JDF 1.1	refelement	Detailed description of the measurement conditions for color measurements.

8.84.10 ColorControlStrip

This resource describes a color control strip. The type of the color control strip is given in the @*StripType* attribute. The lower left corner of the control strip box is used as the origin of the coordinate system used for the definition of the measuring fields. It can be calculated using the following formula:

$$x_0 = x - \frac{w}{2}\cos(\varphi) + \frac{h}{2}\sin(\varphi)$$

$$y_0 = y - \frac{w}{2}\sin(\varphi) + \frac{h}{2}\cos(\varphi)$$

Where:

x = X element of the @*Center* attribute
y = Y element of the @*Center* attribute
w = X element of the @*Size* attribute

h = Y element of the @*Size* attribute
φ = Value of the @*Rotation* attribute

Table 8.150: ColorControlStrip Element

NAME	DATA TYPE	DESCRIPTION
Center ? Modified in JDF 1.4	XYPair	Position of the center of the color control strip in the coordinates of the ***MarkObject*** that contains this mark. **Modification note:** Starting with **JDF** 1.4, @*Center* is optional.
Rotation ?	double	Rotation in degrees. Positive graduation figures indicate counter-clockwise rotation; negative figures indicate clockwise rotation.
Size ? Modified in JDF 1.4	XYPair	Size, in points, of the color control strip. **Modification note:** Starting with **JDF** 1.4, @*Size* is optional.
StripType ? Modified in JDF 1.5	string	Type of color control strip. This attribute MAY be used for specifying a pre-defined, company-specific color control strip. **Modification note:** Starting with **JDF** 1.5, the data type of this attribute was changed from NMTOKEN to string.
CIELABMeasuringField * New in JDF 1.1	element	Details of a CIELAB measuring field that is part of this ***ColorControlStrip***.
DensityMeasuringField * New in JDF 1.1	refelement	Details of a density measuring field that is part of this ***ColorControlStrip***.
SeparationSpec * New in JDF 1.4	element	Ordered list of separations that comprise the ***ColorControlStrip***. If neither ***CIELABMeasuringField*** nor ***DensityMeasuringField*** are specified, the geometry is implied by the value of @*StripType*.

8.84.11 DensityMeasuringField

This resource contains information about a density measuring field.

Table 8.151: DensityMeasuringField Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Center	XYPair	Position of the center of the density measuring field in the coordinates of the ***MarkObject*** that contains this mark. If the measuring field is defined within a ***ColorControlStrip***, @*Center* SHALL refer to the rectangle defined by @*Center* and @*Size* of the ***ColorControlStrip***.
Density Modified in JDF 1.1A	DoubleList	Density value for each process color measured with filter. The data type was modified to NumberList in **JDF** 1.1A in order to accommodate density values >1.0. The sequence of colors is C M Y K, as in the data type CMYKColor.
Diameter	double	Diameter of measuring field.
DotGain	double	Percentage of dot gain.
Percentage	double	Film percentage or equivalent.
Screen	string	Description of the screen.
Separation	string	Reference to a separation that this ***DensityMeasuringField*** applies to. When ***DensityMeasuringField*** is used as an element, it is a standard attribute, otherwise when ***DensityMeasuringField*** is used as a resource, @*Separation* SHALL be defined as a @*Separation* Partition Key.
Setup ?	string	Description of measurement setup.
ToleranceBlack	XYPair	Upper and lower black measurement limits (in density units).

Table 8.151: DensityMeasuringField Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ToleranceCyan	XYPair	Upper and lower cyan measurement limits (in density units).
ToleranceDotGain	XYPair	Upper and lower measurement limits (in%).
ToleranceMagenta	XYPair	Upper and lower magenta measurement limits (in density units).
ToleranceYellow	XYPair	Upper and lower yellow measurement limits (in density units).
ColorMeasurementConditions ? New in JDF 1.1	refelement	Detailed description of the measurement conditions for color measurements.

8.84.12 DynamicField

DynamicField provides a description of dynamic text replacements for a ***MarkObject*** element. This element SHALL be used for production purposes such as defining bar codes for variable data printing. ***DynamicField*** elements are not intended as a placeholders for actual content such as addresses. Rather, they are marks with dynamic data such as time stamps and database information. Dynamic objects are ***MarkObject*** elements with additional OPTIONAL ***DynamicField*** elements that define text replacement.

Table 8.152: DynamicField Element

NAME	DATA TYPE	DESCRIPTION
Format	string	Format string in C printf format that defines the replacement. **Values are from:** ▸ Appendix H Dynamic String Generation**.**
InputField ? Deprecated in JDF 1.1	string	String that SHALL be replaced by the ***DynamicInput*** element in the contents ***RunList*** referenced by @*Ord* or @*OrdExpression*.
Ord ? Deprecated in JDF 1.4	integer	Reference to an index in the contents ***RunList*** that contains ***DynamicInput*** elements. **Constraint:** at most one of @*Ord* or @*OrdExpression* SHALL be specified. **Deprecation note:** Starting with **JDF** 1.4, @*Ord* SHALL be specified in the parent ***MarkObject*** element.
OrdExpression ? Deprecated in JDF 1.4	string	Expression to calculate the reference to an index in the contents ***RunList*** that contains ***DynamicInput*** fields. **Values include those from:** ***ContentObject*****/@***OrdExpression* **Constraint:** at most one of @*Ord* or @*OrdExpression* SHALL be specified.
ReplaceField ?	string	String that SHALL be replaced by the instantiated text expression as defined by the @*Format* and @*Template* attributes in the file referenced by ***MarkObject***/@*Ord*, ***MarkObject***/@*OrdExpression* or ***MarkObject***/***LayoutElement***. If @*ReplaceField* is not specified, the device that processes the ***DynamicField*** SHALL format the ***DynamicField***.
Template	string	Template to define a sequence of variables consumed by @*Format*. **Allowed values are from:** ▸ Appendix H Dynamic String Generation**.** **Deprecation note:** Starting with **JDF** 1.4, ***RunList***/***DynamicInput***/@*Name* (mentioned here in **JDF** 1.3) no longer defines further variables because ***DynamicInput*** has been deprecated.
DeviceMark ? New in JDF 1.1 Deprecated in JDF 1.4	element	***DeviceMark*** defines the formatting parameters for the mark. If not specified, the ***DeviceMark*** settings defined in ***LayoutPreparationParams*** or in the ***Layout*** tree are assumed.

Example 8.17: Layout: DynamicField Element

In this example the text "____xxx____" in the file MyReplace.pdf would be replaced by the sentence "Replacement Text for Joe and John go in here at 14:00 on Mar-31-2000". MyReplace.pdf is placed at the position defined by the @*CTM* of the ***MarkObject*** and Variable.pdf is placed at the position defined by the @*CTM* of the ***ContentObject***.

```
<RunList Class="Parameter" ID="L3" PartIDKeys="Run" Status="Available">
    <MetadataMap DataType="string" Name="i1" ValueFormat="%s"
          ValueTemplate="s1">
       <!--This expression maps the value of /Dokument/Rezipient/@Name to a
          variable "s1"-->
       <Expr Name="s1" Path="/Dokument/Rezipient/@Name"/>
    </MetadataMap>
    <LayoutElement ElementType="Graphic">
       <FileSpec URL="File:///Variable.pdf"/>
    </LayoutElement>
</RunList>
<Layout Class="Parameter" ID="Link0003" Status="Available">
    <!--The MarkObject in the Layout hierarchy: -->
    <ContentObject CTM="1 0 0 1 0 0" Ord="0"/>
    <MarkObject CTM="1 0 0 1 10 10">
       <LayoutElement ElementType="Graphic">
          <FileSpec URL="File:///MyReplace.pdf"/>
       </LayoutElement>
       <DynamicField
             Format="Replacement Text for %s goes in here at %s on %s"
             Ord="0" ReplaceField="___xxx___" Template="i1,Time,Date"/>
       <DynamicField Format="More Replacement Text for %s go in here"
             Ord="0" ReplaceField="___yyy___" Template="SignatureName"/>
    </MarkObject>
</Layout>
```

8.84.13 FillMark

New in JDF 1.5

Table 8.153: FillMark Element

NAME	DATA TYPE	DESCRIPTION
KnockoutBleed ?	double	Bleed in points that the fill should grow into (positive values) the knockout area. **Note:** This attribute implies the same bleed for all separations.
KnockoutRefs ?	IDREFS	Reference to the ***PlacedObject*** elements that SHALL not be filled by this ***FillMark***. The knockout boundaries are implied by the value of @*KnockoutSource*.
KnockoutSource	enumeration	Definition of the source of the knockout from the referenced ***PlacedObject*** elements. **Allowed values are:** ClipPath – Use the Clip Path as defined by the referenced ***PlacedObject***/@*ClipPath*. SourceClipPath – Use the Clip Path as defined by the referenced ***PlacedObject***/@*SourceClipPath*. TrimClipPath – Use the Clip Path as defined by the referenced ***PlacedObject***/@*TrimClipPath*. TrimBox – Use the Clip Path as defined by the referenced ***PlacedObject***/@*TrimCTM* and ***PlacedObject***/@*TrimSize*.
MarkColor *	element	Definition of the separations used to fill the mark.

8.84.14 MarkActivation

New in JDF 1.4

MarkActivation specifies condition when to apply the mark in an automated ***Layout***.

Table 8.154: MarkActivation Element

NAME	DATA TYPE	DESCRIPTION
Context	NMTOKEN	The context in which the iteration is counted. **Values include:** CollectSheetIndex – a parameter maintained by the imposition engine to count sheets (e.g., in the context of a signature). Its value starts at 0 and is incremented by one for each sheet. If ***Layout***/@*MaxCollect* is specified, its maximum value is one less than ***Layout***/@*MaxCollect*. Otherwise, it continues to increment per sheet until completion of the page-pool/page-pool-list processing through the ▸ Imposition Template. See ▸ Section 6.3.18 Imposition. DocIndex – a Partition Key. SetDocIndex – a Partition Key. SetIndex – a Partition Key. SheetIndex – a Partition Key. SubDocIndex0, ... – a parameter maintained by the imposition engine. See ▸ Section 6.3.18 Imposition.
Index	IntegerRangeList	The enclosing ***MarkObject*** is active and its specified Mark SHALL be imaged if the value of the variable specified by @*Context* is equal to one of the values of this attribute.

8.84.14.1 Dynamic Marks

JobField and @*Ord* are mutually exclusive within one ***MarkObject***.

The elements marked as Dynamic marks in the table above can be used for three purposes:

- If @*Ord* is specified, the PDL of the mark is provided by the ***RunList*** (Marks) and the dynamic mark Subelements provide metadata about the mark to a press controller or bindery equipment. This is the usual behavior of existing imposition engines. A single ***MarkObject*** SHALL NOT contain multiple mark Subelements that are represented by the same PDL, for instance there MAY be only one Marks layer for an entire surface.
- If @*Ord* is not present, but ***JobField*** is present, an Imposition device SHOULD dynamically generate the mark based on information in ***JobField***.
- If none of @*Ord* and ***JobField*** are present, a mark SHOULD be dynamically drawn based on the information within the Subelement. The marks are positioned relative to the @*CTM* of the ***MarkObject***. A single ***MarkObject*** SHOULD NOT contain multiple dynamic mark Subelements. Note that the **JDF** specification of dynamic marks other than ***JobField*** are in flux and that the behavior described here might change in future versions of **JDF**.

8.84.15 SheetCondition

New in JDF 1.4

Table 8.155: SheetCondition Element

NAME	DATA TYPE	DESCRIPTION
Condition ?	enumerations	When present, defines an optional sheet by specifying each condition (equivalent to a logical or) under which the optional sheet is produced. **Values include:** Begin – At beginning of imposition processing (all sets in case of multiple ▸ Recipient Sets) End – At the end of imposition processing. BeginSet – At beginning of processing of an individual ▸ Recipient Set. EndSet – At end of processing of an individual ▸ Recipient Set. PagePoolBegin – At beginning of processing of a ▸ Page Pool. PagePoolEnd – At the end of processing of a ▸ Page Pool. PagePoolListBegin – At beginning of processing of a ▸ Page Pool List PagePoolListEnd – At end of processing of a ▸ Page Pool List LogicalStackBegin – adds a ▸ Logical Sheet to the beginning of each ▸ Logical Stack generated as part of an ▸ Imposed Sheet Set. LogicalStackEnd – adds a ▸ Logical Sheet to the end of each ▸ Logical Stack generated as part of an ▸ Imposed Sheet Set. LogicalStackSetBegin – At beginning of generation of a set of ▸ Logical Stacks. Note that @*LogicalStackOrd* SHALL be used to indicate the ▸ Logical Stack on which the conditional sheet is placed. LogicalStackSetEnd – At end of generation of a set of ▸ Logical Stacks. Note that @*LogicalStackOrd* SHALL be used to indicate the ▸ Logical Stack on which the conditional sheet is placed. ImposedSheetSetBegin – At beginning of generation of an ▸ Imposed Sheet Set. Note that this generates a separator sheet not counted as part of the ▸ Imposed Sheet Set. Any ***MarkObject*** elements specifying @*LogicalStackOrd* are ignored. ImposedSheetSetEnd – At end of generation of an ▸ Imposed Sheet Set. Note that this generates a separator sheet not counted as part of the ▸ Imposed Sheet Set. Any ***MarkObject*** elements specifying @*LogicalStackOrd* are ignored.
RunList ?	refelement	Supplies content for any ***ContentObject*** elements specified within the optional sheet definition. All ***ContentObject*** elements in the optional sheet definition SHALL reference content supplied by this ***RunList***.

8.84.16 More about Layout

8.84.16.1 Migrating from a Pre-JDF 1.3 Layout to a Partitioned Layout

New in JDF 1.3

The ***Layout*** resource was significantly modified in **JDF** 1.3. This section describes how a pre-**JDF** 1.3 ***Layout*** can be transformed into a **JDF** 1.3 ***Layout*** and what restrictions MAY be applied to a **JDF** 1.3 ***Layout*** so that it can be easily transformed into a pre-**JDF** 1.3 ***Layout*** or a PJTF Layout.
Note: This section is not applicable when ***Layout***/@*Automated* = "true" for any Partitions.

8.84.16.1.1 Partition Key restrictions:

If "SignatureName", "SheetName" or "Side" are specified in @*PartIDKeys*, the order SHALL be specified as "SignatureName SheetName Side".

Only a ***Layout*** with exactly @*PartIDKeys* = "SignatureName SheetName Side" can be translated into a **JDF** 1.2 ***Layout*** or a PJTF. Thus, it is highly RECOMMENDED to use exactly this Partitioning of the ***Layout*** in **JDF** 1.3 whenever possible. Any other Partitioning will make consumption by existing products very unlikely.

8.84.16.1.2 Position of PlacedObject Elements in Layout

In order to avoid ambiguities in the layering order, ***MarkObject*** elements and ***ContentObject*** elements SHALL only be specified in the leaves of Partitioned resources.

Example 8.18: Invalid MarkObject

The following INVALID example is correct according to ▸ Section 3.10.5.1 Subelements in Partitioned Resources. If standard Partitioning inheritance were permitted for ***MarkObject*** elements and ***ContentObject*** elements it would be unclear whether the ***ContentObject*** in Sheet01 is layered over or under <**MarkObject** **Ord**="1">:

```
<Layout Class="Parameter" ID="L3" Status="Available"
        PartIDKeys="SignatureName SheetName Side">
    <!-- INVALID, this PlacedObject is not in a leaf partition and not used -->
    <!-- since it is overwritten by <MarkObject Ord="1"> -->
    <MarkObject Ord="0" CTM="0.0 1.0 -1.0 0.0 176.69 23.62" >
        <RegisterMark Center="0.0 0.0" MarkType="Cross" MarkUsage="PaperPath" />
    </MarkObject>
    <Layout SignatureName="Sig00">
        <!-- INVALID, this PlacedObject is not in a leaf partition -->
        <MarkObject Ord="1" CTM="0.0 1.0 -1.0 0.0 176.69 23.62" >
            <RegisterMark Center="0.0 0.0" MarkType="Cross"
                  MarkUsage="PaperPath" />
        </MarkObject>
        <Layout SheetName="Sheet00">
            <Layout Side="Front">
                <MarkObject Ord="2" CTM="0.0 1.0 -1.0 0.0 176.69 23.62" >
                    <RegisterMark Center="0.0 0.0" MarkType="Arc"
                       MarkUsage="PaperPath" />
                </MarkObject>
                <ContentObject CTM="0.0 1.0 -1.0 0.0 176.69 23.62" Ord="0" />
            </Layout>
        </Layout>
        <Layout SheetName="Sheet01">
            <Layout Side="Front">
                <!-- Not clear whether this is layered over or under
                    <MarkObject Ord="0">
                -->
                <ContentObject CTM="0.0 1.0 -1.0 0.0 176.69 23.62" Ord="0" />
            </Layout>
        </Layout>
    </Layout>
</Layout>
```

Example 8.19: MarkObject

This VALID example is contains the same ***PlacedObject*** elements as the previous example but they are correctly specified in the leaves of the Partitioned ***Layout***.

```
<Layout Class="Parameter" ID="L3" Status="Available"
        PartIDKeys="SignatureName SheetName Side">
    <Layout SignatureName="Sig00">
        <Layout SheetName="Sheet00">
            <Layout Side="Front">
                <MarkObject Ord="2" CTM="0.0 1.0 -1.0 0.0 176.69 23.62" >
                    <RegisterMark Center="0.0 0.0" MarkType="Arc"
                          MarkUsage="PaperPath"/>
                </MarkObject>
                <ContentObject CTM="0.0 1.0 -1.0 0.0 176.69 23.62" Ord="0" />
            </Layout>
        </Layout>
        <Layout SheetName="Sheet01">
            <Layout Side="Front">
                <MarkObject Ord="1" CTM="0.0 1.0 -1.0 0.0 176.69 23.62" >
                    <RegisterMark Center="0.0 0.0" MarkType="Cross"
                          MarkUsage="PaperPath" />
                </MarkObject>
                <ContentObject CTM="0.0 1.0 -1.0 0.0 176.69 23.62" Ord="0" />
            </Layout>
        </Layout>
    </Layout>
</Layout>
```

8.84.16.2 CTM Definitions

New in JDF 1.2

The following are explanations of the terms used in this section and beyond:

- **Dimensions of object** – The width and height of either the box defined to include all drawings for this file format, or the artificial box that includes these drawings for file formats that have no clearly defined box for this.
- **Trim box of the signature page** – A rectangle that indicates where the trim box of object SHALL be positioned. This is the equivalent to the area the user is intended to see in the final product. Positioning the trim box of the object inside the trim box of the signature page is implementation-specific (usually it is centered).
- **Trim box of the object** – A rectangle that is PDL-specific that indicates the area of the object that indicates the intended trimming area.

8.84.16.3 Finding the Trim Box of an Object

The ***LayoutElement***/@*SourceTrimBox* always takes precedence over boxes defined inside the file. Make sure that ***LayoutElement***/@*SourceTrimBox* is updated after replacing elements. The following is a list of names used for the real trim box in various file formats:

- PostScript (PS) – **PageSize**
- Encapsulated PostScript (EPS) – CropBox
- Portable Document Format (PDF) – **TrimBox**
- Raster files – entire area

If this information is not available, alternative sources for trim box information can include (but these boxes might not be correct in all cases):

- EPS – HiResBoundingBox then BoundingBox
- PDF – **CropBox** then **MediaBox**

8.84.16.4 Using Ord to Reference Elements in RunList Resources

New in JDF 1.1A

The @*Ord* attribute in ***ContentObject*** or ***MarkObject*** elements represents a reference to a *logical* element in a ***RunList***. The index is incremented for every page of the ***RunList*** with @*IsPage* = "true". The reference is not changed by repartitioning the ***RunList***. The content and marks ***RunList*** are referenced independently. The following examples illustrate the usage of @*Ord*.

Example 8.20: RunList: Simple Multi-File Unseparated RunList

This example specifies all pages contained in File1.pdf and File2.pdf. File 1 has 6 pages, file 2 has an unknown number of pages.

```
<RunList Class="Parameter" ID="L3" PartIDKeys="Run" Status="Available">
    <RunList NPage="6" Pages="0 ~ 5" Run="1">
        <LayoutElement>
            <FileSpec URL="File:///File1.pdf"/>
        </LayoutElement>
    </RunList>
    <RunList Pages="0 ~ -1" Run="2">
        <LayoutElement>
            <FileSpec URL="File:///File2.pdf"/>
        </LayoutElement>
    </RunList>
</RunList>
```

Table 8.156: Example (1) of Ord Attribute in PlacedObject Elements

ORD	FILE	PAGE		ORD	FILE	PAGE
0	File1	0		1	File1	1
2	File1	2		3	File1	3
4	File1	4		5	File1	5
6	File2	0		7	File2	1
8	File2	2		(n)	File2	(n - 6)

Example 8.21: RunList: Simple Multi-File Separated RunList

This example specifies two pages contained in Presep.pdf and following that, pages 1, 3 and 5 of each preseparated file.

```
<RunList Class="Parameter" ID="Link0003" PartIDKeys="Run Separation"
        Status="Available">
    <RunList NPage="2" Run="1" SkipPage="3">
        <LayoutElement>
            <FileSpec URL="File:///Presep.pdf"/>
        </LayoutElement>
        <RunList FirstPage="0" IsPage="false" Separation="Cyan"/>
        <RunList FirstPage="1" IsPage="false" Separation="Magenta"/>
        <RunList FirstPage="2" IsPage="false" Separation="Yellow"/>
        <RunList FirstPage="3" IsPage="false" Separation="Black"/>
    </RunList>
    <RunList IsPage="true" Pages="1 3 5" Run="2">
        <RunList IsPage="false" Separation="Cyan">
            <LayoutElement>
                <FileSpec URL="File:///Cyan2.pdf"/>
            </LayoutElement>
        </RunList>
        <RunList IsPage="false" Separation="Magenta">
            <LayoutElement>
                <FileSpec URL="File:///Magenta2.pdf"/>
            </LayoutElement>
        </RunList>
        <RunList IsPage="false" Separation="Yellow">
            <LayoutElement>
                <FileSpec URL="File:///Yellow2.pdf"/>
            </LayoutElement>
        </RunList>
        <RunList IsPage="false" Separation="Black">
            <LayoutElement>
                <FileSpec URL="File:///Black2.pdf"/>
            </LayoutElement>
        </RunList>
    </RunList>
</RunList>
```

Table 8.157: Example (2) of Ord Attribute in PlacedObject Elements

ORD	FILE	PAGE	SEPARATION	ORD	FILE	PAGE	SEPARATION
0	PreSep	0	Cyan	0	Presep	1	Magenta
0	PreSep	2	Yellow	0	Presep	3	Black
1	PreSep	4	Cyan	1	Presep	5	Magenta
1	PreSep	6	Yellow	1	Presep	7	Black
2	Cyan2	1	Cyan	2	Magenta2	1	Magenta
2	Yellow2	1	Yellow	2	Black2	1	Black
3	Cyan2	3	Cyan	3	Magenta2	3	Magenta
3	Yellow2	3	Yellow	3	Black2	3	Black
4	Cyan2	5	Cyan	4	Magenta2	5	Magenta
4	Yellow2	5	Yellow	4	Black2	5	Black

.
.
.

8.84.16.5 Using Expressions in the OrdExpression Attribute

Expressions can use the operators +, –, *, /,% and parentheses, operating on integers and two variables: *s* for signature number (starting at 0) and *n* for number of pages to be imposed in one document. Signature number denotes the number of times that a complete set of placed objects has been filled with content from the run list. The operators have the same meaning as in the C programming language. Expressions are evaluated with normal "C" operator precedence. Multiplication SHALL be expressed by explicitly including the * operator (i.e., use "2*s", not "2 s"). Remainders are discarded.

Example 8.22: OrdExpression

Saddle stitched booklet for variable page length documents.

The following describes the OrdExpressions for a booklet with varying page lengths. The example page assignments are for a book of 13-16 pages.

Table 8.158: OrdExpresson for Varying Page Length Booklet

ORDEXPRESSION	ITERATION 1	ITERATION 2	ITERATION 3	ITERATION 4
2*s	0	2	4	6
4*((n+3)/4)-(s*2)-1	15	13	11	9
2*s+1	1	3	5	7
4*((n+3)/4)-(s*2)-2	14	12	10	8

Example 8.23: DocOrd Usage

Two-sided business cards 4/Sheet

The following describes the Ord + DocOrd usage for a 4-up step + repeat business card.

Table 8.159: DocOrd Usage

MAXDOCORD	SIDE	ORD	DOCORD
4	Front	0	0
4	Front	0	1
4	Front	0	2
4	Front	0	3
4	Back	1	0
4	Back	1	1
4	Back	1	2
4	Back	1	3

8.84.17 Signature

Deprecated in JDF 1.3

The table defining the deprecated ***Signature*** Subelement has been moved to ▸ Section N.7.12.1 Signature. All attributes that were defined in ***Signature*** have been moved into ***Layout***.

8.85 LayoutElement

This resource is needed for ***LayoutElementProduction***. It describes some text, an image, one or more pages or anything else that is used in the production of the layout of a product.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***LayoutElement***/***Dependencies***, ***LayoutElementProductionParams***/***LayoutElementPart***, ***RunList***

Example Partition: "PageNumber"

Input of Processes: **LayoutElementProduction**, **ShapeDefProduction**

Output of Processes: **LayoutElementProduction**

Table 8.160: LayoutElement Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ClipPath ? Modified in JDF 1.2	PDFPath	Path that describes the outline of the ***LayoutElement*** in the coordinate space of the ***LayoutElement*** of @*ElementType* = "Page" that results from the **LayoutElementProduction** process. The default case is that there is no clip path. @*ClipPath*, @*SourceClipBox*, ***PlacedObject***/@*SourceClipPath* and ***PlacedObject***/@*ClipBox* if supplied, SHALL be concatenated.
ContentDataRefs ? New in JDF 1.4	IDREFS	IDs of ***ContentData*** elements in the referenced ***ContentList***. ***ContentData*** elements provide Metadata related to the product to be published. @*ContentDataRefs* SHALL NOT be specified if no ***ContentList*** is specified.
ElementType ? Modified in JDF 1.3	enumeration	Describes the content type for this ***LayoutElement***. **Allowed values are from:** ▸ Table 8.161 ElementType Attribute Values**.**
HasBleeds ? Modified in JDF 1.2	boolean	If "true", the file has bleeds. If not specified, the set of values of ***PageList***/***PageData***/@*HasBleeds* selected by @*PageListIndex* is applied.
IgnorePDLCopies = "false" New in JDF 1.1	boolean	If "true", any PDL defined copy count SHALL be ignored.
IgnorePDLImposition = "true" New in JDF 1.1	boolean	If "true", any PDL defined imposition definition SHALL be ignored. Examples are PDF with embedded PJTF or PPML with a PRINT_LAYOUT. If @*IgnorePDLImposition* = "false" and **JDF** also defines imposition, the imposed Sheets of the PDL are treated as pages in the context of **JDF** imposition. The front and back surfaces of the PDL and **JDF** imposition SHOULD be matched. Note that it is strongly discouraged to specify imposition both in the PDL and **JDF**, and that this might result in undesired behavior.
IsBlank ? New in JDF 1.2	boolean	If "true", the ***LayoutElement*** has no content marks and is blank. If not specified, the set of values of ***PageList***/***PageData***/@*IsBlank* selected by @*PageListIndex* is applied. Note that in **JDF** 1.2 the description erroneously stated that @*IsBlank* = "false" specifies a blank page.
IsPrintable ? Modified in JDF 1.2	boolean	If "true", the file is a PDL file and can be printed. Possible files types include PCL, PDF or PostScript files. Application files such as MS Word have @*IsPrintable* = "false". If not specified, the set of values of ***PageList***/***PageData***/@*IsPrintable* selected by @*PageListIndex* is applied.
IsTrapped ? Modified in JDF 1.2	boolean	If "true", the file has been trapped. If not specified, the set of values of ***PageList***/***PageData***/@*IsTrapped* selected by @*PageListIndex* is applied.
PageListIndex ? New in JDF 1.2	IntegerRangeList	List of the indices of the ***PageData*** elements of the ***PageList*** specified in this ***LayoutElement***. Note that this list MAY be overridden by the ***RunList*** that contains this ***LayoutElement*** and refers to a subset of this ***LayoutElement***. ***PageList*** SHALL be specified if @*PageListIndex* is specified.
SetLevel ? New in JDF 1.4	XPath	Specifies the mapping for the structure of a document of type MultiSet to the structure processed by the PDL processor. If specified, the XPath expression selects a node set from the Structured PDL's hierarchy. Each node of that node set is processed by the PDL processor as a **JDF** set. If not specified, the nodes that are processed as a set by the PDL processor SHALL be defined by the PDL. If the PDL does not define which nodes represent sets, then which nodes represent sets is undefined. **Note:** An example of a PDL that can define which nodes represent sets is ISO 16612-2 (PDF/VT), where the ***DPartRoot***/@*RecordLevel* can provide that mapping.
SourceBleedBox ? Modified in JDF 1.2	rectangle	A rectangle that describes the bleed area of the element to be included. This rectangle is expressed in the source coordinate system of the object. If not specified, the set of values of ***PageList***/***PageData***/@*SourceBleedBox* selected by @*PageListIndex* is applied.

Table 8.160: LayoutElement Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
SourceClipBox ? Modified in JDF 1.2	rectangle	A rectangle that defines the region of the element to be included. This rectangle is expressed in the source coordinate system of the object. If not specified, the set of values of ***PageList***/***PageData***/@*SourceClipBox* selected by @*PageListIndex* is applied.
SourceMediaBox ? New in JDF 1.4	rectangle	The MediaBox of the ***LayoutElement***.
SourceTrimBox ? Modified in JDF 1.2	rectangle	A rectangle that describes the intended trimmed size of the element to be included. This rectangle is expressed in the source coordinate system of the object. If not specified, the set of values of ***PageList***/***PageData***/@*SourceTrimBox* selected by @*PageListIndex* is applied.
Template ? Modified in JDF 1.2	boolean	@*Template* is "false" when this layout element is self-contained. This attribute is "true" if the ***LayoutElement*** represents a template that SHALL be completed with information from a database. If not specified, the value of ***PageList***/***PageData***/@*Template* is applied.
ColorPool ? New in JDF 1.2	refelement	Definition of the color details.
ContentList ? New in JDF 1.4	refelement	***ContentList*** with additional metadata. **Constraint:** at most one of ***ContentList*** and ***PageList*** SHALL be specified.
Dependencies ? New in JDF 1.2	element	List of dependent references (e.g., fonts, external images, etc.).
ElementColorParams ? New in JDF 1.2	refelement	Color details of the ***LayoutElement***. If not specified, the value of ***PageList***/***PageData***/***ElementColorParams*** is applied.
FileSpec ? Modified in JDF 1.2	refelement	URL plus metadata about the physical characteristics of a file representing the ***LayoutElement***. If not present, then only metadata is known but not the content file.
ImageCompressionParams ? New in JDF 1.2	refelement	Specification of the image compression properties. If not specified, the value of ***PageList***/***PageData***/***ImageCompressionParams*** is applied.
PageList ? New in JDF 1.2	refelement	Specification of page metadata for pages described by this ***LayoutElement***. **Constraint:** at most one of ***ContentList*** and ***PageList*** SHALL be specified.
ScreeningParams ? New in JDF 1.2	refelement	Specification of the screening properties. If not specified, the value of ***PageList***/***PageData***/***ScreeningParams*** is applied.
SeparationSpec * Modified in JDF 1.2	element	List of used separation names. If not specified, the value of ***PageList***/***PageData***/***SeparationSpec*** applies.

8.85.1 ElementType Attribute Values

*Table 8.161: **ElementType** Attribute Values (Sheet 1 of 2)*

VALUE	DESCRIPTION
Auxiliary	Any type of file that is needed to complete a layout but not explicitly displayed (e.g., ICC profiles or fonts).
Barcode New in JDF 1.3	A barcode.
Composed	Combination of elements that define an element that is not bound to a document page.

Table 8.161: ElementType Attribute Values (Sheet 2 of 2)

VALUE	DESCRIPTION
Document	An ordered set of one or more pages.
Graphic	Line art.
IdentificationField New in JDF 1.3	A general identification field excluding bar codes.
Image	Bitmap image.
MultiDocument	An ordered set of one or more Documents including document breaks (e.g., PPML, PPML/VDX, MIME Multipart/Related).
MultiSet	An ordered set of one or more document sets, including document set breaks, document breaks and sub document breaks (e.g., PPML, PPML/VDX, ISO 16612-2 PDF/VT). **Modification note:** Starting with **JDF** 1.4, 3 kinds of breaks are added.
Page	Representation of one document page.
Reservation	Empty element. Content for this area of the page might be provided by a subsequent process.
Surface	Representation of an imposed surface.
Text	Formatted or unformatted text.
Tile	Representation of the contents of one tile.
Unknown Deprecated in JDF 1.2	

8.85.2 Dependencies

New in JDF 1.2

This element provides a container for dependent references of the ***LayoutElement***.

Table 8.162: Dependencies Element

NAME	DATA TYPE	DESCRIPTION
LayoutElement *	refelement	Description of dependent elements (e.g., fonts, images, etc.).

8.86 LayoutElementProductionParams

New in JDF 1.3

This resource is needed for **LayoutElementProduction**. This resource contains detailed information about the type of ***LayoutElement*** to be produced. Before **JDF** 1.4, it only contains information for automated production of barcodes. Starting with **JDF** 1.4, the description of positioning of the graphics is added.

Resource Properties

Resource Class: Parameter

Input of Processes: **LayoutElementProduction**

Table 8.163: LayoutElementProductionParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ActionPool ? New in JDF 1.4	element	A pool of ***Action*** elements that describe the restrictions that are applied to the created output
FileSpec (DataList) ? New in JDF 1.5	refelement	References a data list containing record information for variable data production. The format of the referenced data is implementation specific.

Table 8.163: LayoutElementProductionParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
LayoutElementPart *	element	Description of the specific parameters for generating a ***LayoutElement***.
ShapeDef ? New in JDF 1.4	refelement	A resource describing the shape of the ***LayoutElement*** to be produced.
TestPool ? New in JDF 1.4	element	Container for zero or more Test elements that are referenced from ***Action*** elements in the ***ActionPool***. ***TestPool*** SHALL be supplied if ***ActionPool*** is present.

Example 8.24: LayoutElementProductionParams: Page Shape

The following example requests four pages with a trim size of A4.

```
<!-- Page Shape Sample
       Date: Aug 2, 2007 Version: 2
       A page with a certain size  -->
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n001"
       Type="LayoutElementProduction"
       Status="Waiting" DescriptiveName="Page sample for shape"
       JobPartID="ID34" Version="1.4">
    <ResourcePool>
       <LayoutElementProductionParams Class="Parameter" ID="LEPParams"
              Status="Available" />
       <LayoutElement Class="Parameter" ID="LayElOut" Status="Unavailable"
              SourceMediaBox="0 0 595.27 822.05"
              SourceTrimBox="28.34 28.34 566.93 793.71"/>
    </ResourcePool>
    <ResourceLinkPool>
       <LayoutElementProductionParamsLink rRef="LEPParams" Usage="Input"/>
       <LayoutElementLink rRef="LayElOut" Usage="Output"/>
    </ResourceLinkPool>
    <AuditPool>
       <Created AgentName="XYZ Corporation" TimeStamp="2006-01-09T09:00:00+01:00"/>
    </AuditPool>
</JDF>
```

Example 8.25: LayoutElementProductionParams: Label Shape

The following example requests a label shape with an explicit cut path.

```
<!-- Shape Sample for a label with a cut line
       Date: Jan 9, 2005 Version: 1.00
       A page with a certain size  -->
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n001"
       Type="LayoutElementProduction"
       Status="Waiting" DescriptiveName="Page sample for shape"
       JobPartID="ID400" Version="1.4">
    <ResourcePool>
       <LayoutElementProductionParams Class="Parameter" ID="LEPParams"
              Status="Available" >
          <ShapeDef>
             <Shape ShapeType="Path" DDESCutType="101" CutPath="..." />
          </ShapeDef>
       </LayoutElementProductionParams>
       <LayoutElement Class="Parameter" ID="LayElOut" Status="Unavailable"
              SourceMediaBox="0 0 595.27 822.05"
              SourceTrimBox="28.34 28.34 566.93 793.71"/>
    </ResourcePool>
    <ResourceLinkPool>
       <LayoutElementProductionParamsLink rRef="LEPParams" Usage="Input"/>
       <LayoutElementLink rRef="LayElOut" Usage="Output"/>
    </ResourceLinkPool>
    <AuditPool>
       <Created AgentName="ABC-Corporation" TimeStamp="2006-01-09T09:00:00+01:00"/>
    </AuditPool>
</JDF>
```

Example 8.26: LayoutElementProductionParams: Box Shape

The following example requests a box that is described by an external CAD file.

```
<!-- Shape Sample for a box defined by a CAD file
       Date: Jan 9, 2005 Version: 1.00
       A page with a certain size  -->
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n001"
       Type="LayoutElementProduction"
       Status="Waiting" JobPartID="ID100"
       DescriptiveName="Page sample for shape" Version="1.4">
    <ResourcePool>
       <LayoutElementProductionParams Class="Parameter" ID="LEPParams"
              Status="Available">
          <ShapeDef>
             <FileSpec URL="file://myserver/myshare/olive.dd3"/>
          </ShapeDef>
       </LayoutElementProductionParams>
       <LayoutElement Class="Parameter" ID="LayElOut" Status="Unavailable"/>
    </ResourcePool>
    <ResourceLinkPool>
       <LayoutElementProductionParamsLink rRef="LEPParams" Usage="Input"/>
       <LayoutElementLink rRef="LayElOut" Usage="Output"/>
    </ResourceLinkPool>
    <AuditPool>
       <Created AgentName="ZYX Corporation" TimeStamp="2006-01-09T09:00:00+01:00"/>
    </AuditPool>
</JDF>
```

8.86.1 LayoutElementPart

LayoutElementPart is a generic placeholder for specifying details of **LayoutElementProduction**. Currently only details of barcode production have been fleshed out but additional processes are anticipated. Note that the ordering of ***LayoutElementPart*** elements might become significant in future versions.

Table 8.164: LayoutElementPart Element

NAME	DATA TYPE	DESCRIPTION
ID ? New in JDF 1.4	ID	ID of the ***LayoutElementPart***.
Comment ?	element	Human readable instructions how this ***LayoutElementPart*** should be processed.
BarcodeProductionParams ?	element	Description of the specific parameters for barcode production.
ColorCorrectionParams ? New in JDF 1.5	refelement	Parameters of **ColorCorrection** that have been applied to this ***LayoutElementPart***.
ImageCompressionParams ? New in JDF 1.5	refelement	Image compression that has been applied to this ***LayoutElementPart***.
ImageEnhancementParams ? New in JDF 1.5	refelement	Image enhancment operations that have been applied to this ***LayoutElementPart***
LayoutElement ? New in JDF 1.4	refelement	Specification of an existing ***LayoutElement*** that is used to initially populate this ***LayoutElementPart***. Any ***LayoutElement*** resources that are specified here SHALL also be specified as Input resources to the **LayoutElementProduction** process.
PositionObj ? New in JDF 1.4	element	Definition of the size and position of this ***LayoutElementPart***

8.86.2 BarcodeProductionParams

BarcodeProductionParams describes of the specific parameters for barcode production.

Table 8.165: BarcodeProductionParams Element

NAME	DATA TYPE	DESCRIPTION
BarcodeReproParams ?	element	Description of the formatting and reproduction parameters for barcode production.
IdentificationField	refelement	Description of the barcode metadata.

8.86.3 PositionObj

New in JDF 1.4

PositionObj describes the size and position of the ***LayoutElementPart***.

Table 8.166: PositionObj Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Anchor ?	enumeration	*@Anchor* specifies the origin (0,0) of the coordinate system in the unrotated ***LayoutElementPart***. **Allowed value is from:** ▸ Anchor.
CTM ?	matrix	Transformation matrix of the origin of ***LayoutElementPart*** as specified by *@Anchor*. Not that this is not necessarily the actual CTM that will position a given ***LayoutElementPart***. The actual CTM SHALL be recalculated based on the values of *@Anchor* and *@Size*.
PageRange ?	IntegerRangeList	Reader Page index in the ***PageList***.

Table 8.166: PositionObj Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
PositionPolicy ?	enumeration	Specifies the level of freedom when applying the values specified in ***PositionObj***. **Allowed value is from:** ▸ PositionPolicy.
RelativeSize ?	XYPair	Specifies the size of the unrotated and unscaled object, relative to the parent specified in ***RefAnchor***.
RotationPolicy ?	enumeration	Specifies the level of freedom when applying the values specified in ***PositionObj***. **Allowed value is from:** ▸ PositionPolicy.
Size ?	XYPair	Specifies the size of the unrotated and unscaled object, in points.
SizePolicy ?	enumeration	Specifies the level of freedom when applying the values specified in ***PositionObj***. **Allowed value is from:** ▸ PositionPolicy.
RefAnchor ?	element	Reference to a ***LayoutElementPart*** that this ***LayoutElementPart*** is positioned relative to. If ***RefAnchor*** is not specified, ***PositionObj*** refers to the lower left of the first page specified in page Range.

Example 8.27: LayoutElementProductionParams: PositionObj

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n000002"
      JobPartID="n000002" Status="Waiting" Type="LayoutElementProduction"
      Version="1.4"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="LayoutElementProduction">
   <!--Generated by the CIP4 Java open source JDF Library version :
      CIP4 JDF Writer Java 1.3 BLD 46-->
   <AuditPool>
      <Created AgentName="CIP4 JDF Writer Java" AgentVersion="1.3 BLD 46"
            ID="a000003"
            TimeStamp="2007-09-05T18:20:31+02:00"/>
   </AuditPool>
   <ResourcePool>
      <RunList Class="Parameter" ID="r000004" Status="Unavailable">
         <LayoutElement Class="Parameter">
            <FileSpec Class="Parameter" MimeType="application/pdf" URL="output.pdf"/>
         </LayoutElement>
      </RunList>
      <LayoutElementProductionParams Class="Parameter" ID="r000005"
            Status="Unavailable">
         <!--This is a "well placed" CTM defined mark
               The anchor defines the 0,0 point to be transformed
               The element to be placed is referenced by LayoutElement/FileSpec/URL
         -->
         <LayoutElementPart>
            <PositionObj Anchor="BottomLeft" CTM="1 0 0 1 0 0" PageRange="0"
                  PositionPolicy="Exact">
               <RefAnchor Anchor="BottomLeft" AnchorType="Parent"/>
            </PositionObj>
            <LayoutElement Class="Parameter">
               <FileSpec Class="Parameter" MimeType="application/pdf"
                     URL="bkg.pdf"/>
            </LayoutElement>
         </LayoutElementPart>
         <!--This is a "roughly placed" reservation in the middle of the page-->
         <LayoutElementPart ID="l1000006">
            <PositionObj Anchor="Center" PageRange="0" PositionPolicy="Free">
               <RefAnchor Anchor="Center" AnchorType="Parent"/>
            </PositionObj>
            <LayoutElement Class="Parameter" ElementType="Image">
               <Comment ID="c000007">
                  Please add an image of a palm tree on a beach here!
               </Comment>
            </LayoutElement>
         </LayoutElementPart>
```

```
<!--This is a "roughly placed" reservation 36 points below the previous
    image; NextPosition points from Anchor on this to NextAnchor on next,
    i.e. a positive vector specifies that next is shifted in the positive
    direction in the parent (in this case page) coordinate system
-->
<LayoutElementPart>
    <PositionObj Anchor="TopCenter" CTM="1 0 0 1 0 36"
           PageRange="0" PositionPolicy="Free">
        <RefAnchor Anchor="BottomCenter" AnchorType="Sibling"
              rRef="l1000006"/>
    </PositionObj>
    <LayoutElement Class="Parameter" ElementType="Image">
        <Comment ID="c000008">
            Please add an image of a beach ball below the palm tree!
        </Comment>
    </LayoutElement>
</LayoutElementPart>
<!--This is a "well placed" CTM defined mark. The anchor defines the
    0,0 point used as the RefAnchor for the element to be transformed
-->
<LayoutElementPart>
    <PositionObj Anchor="BottomLeft" CTM="1 0 0 1 2 3" PageRange="0"
           PositionPolicy="Exact">
        <RefAnchor Anchor="BottomLeft" AnchorType="Parent"/>
    </PositionObj>
    <BarcodeProductionParams>
        <!--barcode details here-->
        <IdentificationField Encoding="Barcode" EncodingDetails="CODABAR" />
    </BarcodeProductionParams>
</LayoutElementPart>
<LayoutElementPart>
    <PositionObj Anchor="TopRight" PageRange="0" PositionPolicy="Exact">
        <RefAnchor Anchor="TopRight" AnchorType="Parent"/>
        <!--This is a "roughly placed" mark.
            The anchor at top right is placed at the right (=1.0) top(=1.0)
            position of the page. No rotation is specified
        -->
    </PositionObj>
    <BarcodeProductionParams>
        <!--barcode details here-->
        <IdentificationField Encoding="Barcode" EncodingDetails="CODABAR" />
    </BarcodeProductionParams>
</LayoutElementPart>
<!--This is a "roughly placed" container for marks
    The anchor at top left is defined in the !Unrotated! orientation.
    It is placed at the left (=0.0) bottom(=0.0) position of the page.
    The text flows bottom to top (=Rotate 90 = counterclockwise)
    do we need margins?
-->
<LayoutElementPart ID="l1000009">
    <PositionObj Anchor="TopLeft" CTM="0 1 -1 0 0 0"
           PageRange="1" PositionPolicy="Free">
        <RefAnchor Anchor="BottomCenter" AnchorType="Parent"/>
    </PositionObj>
</LayoutElementPart>
```

```
        <!--This is a barcode inside the previous container
            The anchor at bottom left is defined in the !Unrotated! orientation.
            It is placed at the left (=0.0) bottom(=0.0) position of the container.-->
        <LayoutElementPart ID="l1000010">
            <PositionObj Anchor="BottomLeft" CTM="1 0 0 1 0 0">
                <RefAnchor Anchor="BottomLeft" AnchorType="Parent" rRef="l1000009"/>
            </PositionObj>
            <BarcodeProductionParams>
                <!--barcode details here-->
                <IdentificationField Encoding="Barcode" EncodingDetails="CODABAR" />
            </BarcodeProductionParams>
        </LayoutElementPart>
        <!--This is a disclaimer text inside the previous container
            The anchor at top left is defined in the !Unrotated! orientation.
            The barcode and text are justified with their top margins and spaced
            by 72 points which corresponds to the left of the page because the
            container is rotated 90Â° AbsoluteSize specifies the size of the
            object in points -->
        <LayoutElementPart>
            <PositionObj Size="300 200" Anchor="TopLeft" CTM="1 0 0 1 -72 0">
                <RefAnchor Anchor="TopRight" AnchorType="Sibling" rRef="l1000010"/>
            </PositionObj>
            <LayoutElement Class="Parameter" ElementType="Text">
                <FileSpec Class="Parameter"
                       URL="file://myServer/disclaimers/de/aspirin.txt"/>
            </LayoutElement>
        </LayoutElementPart>
        <!--This is a "VERY roughly placed" piece of text somewhere on pages 2-3
            RelativeSize specifies the size of the object as a ratio of the size
            of the container -->
        <LayoutElementPart>
            <PositionObj PageRange="1 ~ 2" RelativeSize="0.8 0.5"/>
            <LayoutElement Class="Parameter" ElementType="Text">
                <Comment ID="c000011" Name="Instructions">
                    Please add some text about
                    the image of a palm tree on a beach here!
                </Comment>
            </LayoutElement>
        </LayoutElementPart>
        <!--This is another "VERY roughly placed" piece of text somewhere on
            pages 2-3; the text source is the JDF-->
        <LayoutElementPart>
            <PositionObj PageRange="1 ~ 2"/>
            <LayoutElement Class="Parameter" ElementType="Text">
                <Comment ID="c000012" Name="TextInput">
                    Laurum Ipsum Blah blah blah!
                    btw. this is unformatted plain text and nothing else!
                </Comment>
            </LayoutElement>
        </LayoutElementPart>
    </LayoutElementProductionParams>
  </ResourcePool>
  <ResourceLinkPool>
    <RunListLink Usage="Output" rRef="r000004"/>
    <LayoutElementProductionParamsLink Usage="Input" rRef="r000005"/>
  </ResourceLinkPool>
</JDF>
```

Example 8.28: LayoutElementProductionParams: Preflight

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n000002"
       JobPartID="n000002" Status="Completed" Type="LayoutElementProduction"
       Version="1.4" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:type="LayoutElementProduction">
    <!--Generated by the CIP4 Java open source JDF Library version : CIP4 JDF
          Writer Java 1.3 BLD 47-->
    <AuditPool>
       <Created AgentName="CIP4 JDF Writer Java" AgentVersion="1.3 BLD 47"
             ID="a000003"
             TimeStamp="2007-10-11T20:23:18+02:00"/>
       <PhaseTime AgentName="CIP4 JDF Writer Java"
             AgentVersion="1.3 BLD 47"
             End="2007-10-11T20:23:23+02:00" ID="a000020"
             Start="2007-10-11T20:23:21+02:00" Status="InProgress"
             StatusDetails="Creative Work" TimeStamp="2007-10-11T20:23:21+02:00"/>
       <ProcessRun AgentName="CIP4 JDF Writer Java"
             AgentVersion="1.3 BLD 47"
             Duration="PT2S" End="2007-10-11T20:23:23+02:00"
             EndStatus="Completed" ID="a000024"
             Start="2007-10-11T20:23:21+02:00" TimeStamp="2007-10-11T20:23:23+02:00"/>
    </AuditPool>
    <ResourcePool>
       <RunList Class="Parameter" ID="r000004" Status="Unavailable">
          <LayoutElement Class="Parameter">
             <FileSpec Class="Parameter" MimeType="application/pdf" URL="output.pdf"/>
          </LayoutElement>
       </RunList>
       <LayoutElementProductionParams Class="Parameter" ID="r000005"
             Status="Unavailable">
          <Comment ID="c000006" Name="Instruction">
             Add any human readable instructions here
          </Comment>
          <ActionPool>
             <Action DescriptiveName="set number of pages to 4" ID="A000007"
                   Severity="Error" TestRef="T000008"/>
             <Action
                   DescriptiveName="set number of separations to 6 on page 0 and 3"
                   ID="A000009" Severity="Error" TestRef="T000010">
                <PreflightAction SetRef="T000011"/>
             </Action>
             <Action
                   DescriptiveName="separation to black only on page 1 and 2"
                   ID="A000012" Severity="Error" TestRef="T000013">
                <PreflightAction SetRef="T000014"/>
             </Action>
             <Action DescriptiveName="set TrimBox to 8.5*11 Method 2"
                   ID="A000015" Severity="Error" TestRef="T000016">
                <PreflightAction SetRef="T000017"/>
             </Action>
             <Action
                   DescriptiveName="Warn when effective resolution&lt;300 dpi"
                   ID="A000018" Severity="Warning" TestRef="T000019"/>
          </ActionPool>
```

```
            <TestPool>
                <Test ID="T000008">
                    <not>
                        <IntegerEvaluation ValueList="4">
                            <BasicPreflightTest Name="NumberOfPages"/>
                        </IntegerEvaluation>
                    </not>
                </Test>
                <Test ID="T000010">
                    <not>
                        <StringEvaluation>
                            <BasicPreflightTest ListType="UniqueList" MaxOccurs="6"
                                  MinOccurs="6" Name="SeparationList"/>
                        </StringEvaluation>
                    </not>
                </Test>
                <Test ID="T000011">
                    <IntegerEvaluation ValueList="0 3">
                        <BasicPreflightTest Name="PageNumber"/>
                    </IntegerEvaluation>
                </Test>
                <Test ID="T000013">
                    <not>
                        <StringEvaluation>
                            <BasicPreflightTest Name="SeparationList"/>
                            <Value Value="Black"/>
                        </StringEvaluation>
                    </not>
                </Test>
                <Test ID="T000014">
                    <IntegerEvaluation ValueList="1 ~ 2">
                        <BasicPreflightTest Name="PageNumber"/>
                    </IntegerEvaluation>
                </Test>
                <Test ID="T000016">
                    <not>
                        <RectangleEvaluation ValueList="0 0 612 792">
                            <BasicPreflightTest Name="PageBoxSize"/>
                        </RectangleEvaluation>
                    </not>
                </Test>
                <Test ID="T000017">
                    <EnumerationEvaluation ValueList="TrimBox">
                        <BasicPreflightTest Name="PageBoxName"/>
                    </EnumerationEvaluation>
                </Test>
                <Test ID="T000019">
                    <XYPairEvaluation ValueList="0 0 ~ 300 300">
                        <BasicPreflightTest Name="EffectiveResolution"/>
                    </XYPairEvaluation>
                </Test>
            </TestPool>
        </LayoutElementProductionParams>
    </ResourcePool>
    <ResourceLinkPool>
        <RunListLink Usage="Output" rRef="r000004"/>
        <LayoutElementProductionParamsLink Usage="Input" rRef="r000005"/>
    </ResourceLinkPool>
</JDF>
```

8.87 LayoutPreparationParams

New in JDF 1.1

This resource provides the parameters of the **LayoutPreparation** process, which provides the details of how finished page contents will be imaged onto media. This resource has a provision for specifying either a multi-up grid of content page cells or an imposition layout of finished pages. The **LayoutPreparation** also provides means to specify creeping gutters for booklet imposition. In the case where attributes of ***LayoutPreparationParams*** used to explicitly control creep are spec-

ified, the @*MinGutter* and @*GutterPolicy* attributes of ***FitPolicy***, which affect the adjustment of gutter widths, SHALL NOT be specified.

A multi-up grid of pages can be step and repeated across, down, or through a stack of Sheets in any axis order. Note that for all resources, the coordinate system for all parameters is defined with respect to the process coordinate system as defined in ▸ Section 2.5.3 Coordinate Systems of Resources and Processes. The process coordinate system for **LayoutPreparation** is defined by the ***Layout*** resource coordinate system.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***LayoutIntent***

Example Partition: "DocIndex", "DocRunIndex", "RunIndex", "SetIndex", "SheetName"

Input of Processes: **LayoutPreparation**

Table 8.167: LayoutPreparationParams Resource (Sheet 1 of 7)

NAME	DATA TYPE	DESCRIPTION
BindingEdge ? New in JDF 1.3	enumeration	Indicates which finished page edge should be bound. The binding edge is defined relative to the orientation of the page cell containing the first Reader Page in the finished print component with content on it. **Allowed values are:** Left Right Top Bottom None
BackMarkList ?	NMTOKENS	List of marks that are to be marked on each back surface. The appearance of the marks are defined by the process implementation. **Values include:** CIELABMeasuringField ColorControlStrip ColorRegisterMark CutMark DensityMeasuringField IdentificationField JobField PaperPathRegisterMark RegisterMark ScavengerArea
CreepValue ?	XYPair	This parameter specifies horizontal and vertical creep compensation value in points. The first value specifies the creep compensation of all horizontal gutters, and the second value specifies the creep compensation of all vertical gutters. The numbers specify the distance in points by which the respective explicitly creeping gutter either increments (positive values) or decrements (negative values) in width from one Sheet to the next for a given sequence of Sheets related to the same bound component. If not specified, it MAY be calculated based on the information taken from Media.

Table 8.167: LayoutPreparationParams Resource (Sheet 2 of 7)

NAME	DATA TYPE	DESCRIPTION
FinishingOrder = "GatherFold"	enumeration	Specifies the order of operations for finishing a bound booklet created from multiple imposed Sheets. The **LayoutPreparation** process needs this information in order to completely determine content page distribution onto the sequence of Sheets comprising the pages of a single booklet under consideration of the values of the @*PageDistributionScheme* and @*FoldCatalog* attributes. **Allowed values are:** FoldGather – The Sheets of a document are first folded according to the value of the @*FoldCatalog* attribute and then gathered on a pile. Usually applies to finishing of perfect-bound documents. FoldCollect – The Sheets of a document are first folded, according to the value of the @*FoldCatalog* attribute, and then collected on a saddle. Usually applies to finishing of both perfect-bound and saddle-stitched booklets. Gather – The Sheets of a document are gathered on a pile. No folding is assumed. GatherFold – The Sheets of a document are first gathered on a pile then folded according to the value of the @*FoldCatalog* attribute. Usually applies to finishing of both perfect-bound and saddle-stitched booklets.
FoldCatalog ?	string	Description of the type of fold that will be applied to all printed Sheets according to the folding catalog in ▸ Figure 8-33: Fold catalog part 1 and ▸ Figure 8-34: Fold catalog part 2. **Value format is:** "Fn-i" where "n" is the number of finished pages and "i" is either an integer, which identifies a particular fold or the letter "X", which identifies a generic fold (e.g., "F6-2" describes a Z-fold of 6 finished pages, and "F6-X" describes a generic fold with 6 finished pages). The **LayoutPreparation** process uses the fold description specified by this attribute in the determination of the proper distribution of pages onto the surfaces of the Sheets in the context of the values of both the @*PageDistributionScheme* and @*FinishingOrder* attributes. If not present, no folding other than the folding that is implied by @*PageDistributionScheme* = "Saddle" is assumed.
FoldCatalogOrientation = "Rotate0" New in JDF 1.3	enumeration	This attribute specifies the orientation of how the identified fold catalog entry SHALL be interpreted for the purposes of mapping input pages into the imposition layout (not for purposes of performing the folding, if any, or orienting the Sheet). **Allowed value is from:** ▸ Orientation.
FrontMarkList ?	NMTOKENS	List of marks that are to be marked on each front surface. The appearance of the marks are defined by the process implementation. **Values include:** **See @*BackMarkList*..**

Table 8.167: LayoutPreparationParams Resource (Sheet 3 of 7)

NAME	DATA TYPE	DESCRIPTION
Gutter ? Modified in JDF 1.2	XYPair	Width in points of the horizontal and vertical gutters formed between rows and columns of page cells of a multi-up Sheet layout. The gutter width is defined as the distance between the ***PageCell***/@*TrimSize* defined trim boxes of adjacent page cells. The first value specifies the width of all horizontal gutters, and the second value specifies the width of all vertical gutters. If no gutters are defined because either the @*NumberUp* attribute is not specified or its explicit values are equal to one, this attribute SHALL be ignored. In the case where a gutter is identified as creeping by either @*VerticalCreep* or @*HorizontalCreep*, then the values of @*Gutter* specify the initial width of explicitly creeping gutters where the gutter width may increment or decrement depending on the @*CreepValue* attribute. If a value of @*CreepValue* is negative then @*Gutter* SHALL be interpreted as the starting gutter width of the outermost Sheet, otherwise it SHALL be interpreted as the starting gutter width of the innermost Sheet. @*Gutter* is applied in addition to any @*Border* specified in the ***PageCell***.
GutterMinimumLimit ? New in JDF 1.3	XYPair	Specifies the minimum width in points of explicitly creeping horizontal and vertical gutter(s). If an explicitly creeping gutter shrinks to a width equal to or less than this value, all subsequent gutters SHALL be set to this value. If @*GutterMinimumLimit* is specified and neither @*Gutter* nor @*CreepValue* is specified, the device SHOULD calculate creep in a device specific manner.
HorizontalCreep ? Modified in JDF 1.2	IntegerList	Specifies which horizontal gutters creep. The allowed values are zero-based indexes that reference horizontal gutters formed by multiple rows of pages in a multi-up page layout specified by the second value of @*NumberUp*. The value for an entry in this list SHALL be between zero and two (2) less then the second value of @*NumberUp*. If not specified, then horizontal gutters SHALL NOT creep. Gutters identified by this attribute are known as explicitly creeping gutters whereas those not identified are known as implicitly creeping gutters. Note: In order preserve the absolute position of the center lines of all gutters across all Sheets, only specify alternating gutters starting with gutter index zero.
ImplicitGutter ? New in JDF 1.3	XYPair	Specifies the initial gutter width in points for implicitly creeping horizontal and vertical gutters. The first number corresponds to horizontal gutters and the second number corresponds to vertical gutters. The particular Sheet to which this initial gutter applies (innermost or outermost) depends upon the polarity of the creep increment specified by @*CreepValue* (see @*Gutter*).
ImplicitGutterMinimumLimit ? New in JDF 1.3	XYPair	Specifies the minimum width in points of implicitly creeping vertical and horizontal gutter(s). If an implicitly creeping gutter shrinks to a width equal to or less than this value, all subsequent gutters SHALL be set to this value.

Table 8.167: LayoutPreparationParams Resource (Sheet 4 of 7)

NAME	DATA TYPE	DESCRIPTION
NumberUp ?	XYPair	Specifies a regular, multi-up grid of ***PageCell*** elements into which content finished pages are mapped. The first value specifies the number of columns of page cells and the second value specifies the number of rows of page cells in the multi-up grid (both numbers are integers). The relative positioning of the page cells within the multi-up grid are defined by the explicit or implied values of the @*Gutter*, @*HorizontalCreep*, @*VerticalCreep* and @*CreepValue* attributes. The distribution of content pages from the content ***RunList*** into the page cells is defined by the explicit or implied values of the @*PageDistributionScheme*, @*PresentationDirection*, @*Sides*, @*FinishingOrder* and @*FoldCatalog* attributes and the implicit number of Sheets comprising the bound component.
PageDistributionScheme = "Sequential" Modified in JDF 1.5	NMTOKEN	Specifies how finished pages are to be distributed onto a multi-up grid of finished ***PageCell*** elements defined by the values of the @*NumberUp* attribute. **Values include those from:** ▸ Table 8.168 PageDistributionScheme Attribute Values **Note:** Page distribution ordering depends upon the implied number of Sheets per finished ***Component*** and how the imposed Sheets are to be folded during finishing as well as the order of gathering and folding. Refer to the @*FoldCatalog* and @*FinishingOrder* attributes. **Note:** The @*NumberUp* attribute SHALL always specify a multi-up layout appropriate for a given finished page distribution ordering and @*FoldCatalog*. Setting this attribute does not imply the multi-up grid dimensions are appropriate for the selected page distribution scheme. **Note:** In all cases, the order of finished pages as represented by the content ***RunList*** SHALL be either in reader order or in an order appropriate for multi-up saddle stitching. Refer to the @*PageOrder* attribute. **Modification note:** Starting with **JDF** 1.5, there are new values.
PageOrder = "Reader"	NMTOKEN	The assumed ordering of the finished pages in the ***RunList***. **Values include:** Booklet – The finished pages are ordered in the ***RunList*** and SHALL be processed exactly in the order as specified by @*PresentationDirection*. @*NumberUp* SHALL still be set to the appropriate value and is not implied by specifying @*PageOrder* = "Booklet". @*PageOrder* = "Booklet" SHALL NOT be used in conjunction with @*FoldCatalog*. Reader – The finished pages are in reader order in the ***RunList***.

Table 8.167: LayoutPreparationParams Resource (Sheet 5 of 7)

NAME	DATA TYPE	DESCRIPTION
PresentationDirection ?	enumeration	Indicates the order in which finished pages will be distributed into the page cells of the @*NumberUp* layout. If @*PageDistributionScheme* = "Saddle", @*PresentationDirection* applies to sets of two adjacent pages. This allows positioning of multiple page pairs for SaddleStitching onto one Sheet. **Allowed values are:** FoldCatalog – Finished Pages are imaged so that the result is compatible with a finished product produced from the folding catalog as specified in @*FoldCatalog*. XYZ – Permutations of the letters XYZ and xyz so that exactly one of upper or lower case of x, y and z define the order in which finished pages are flowed along each axis with respect to the coordinate system of the front side of the Sheet. The first letter of the triplet specifies the initial axis of flow. The second letter of the triplet specifies the second axis of flow and so on. • X – Specifies flowing left to right across a Sheet surface. • x – Specifies flowing right to left across a Sheet surface. • Y – Specifies flowing bottom to top vertically across a Sheet surface. • y – Specifies flowing top to bottom vertically across a Sheet surface. • Z – Specifies flowing bottom of stack to top of it through the stack. • z – Specifies flowing top of stack to bottom of it through the stack. **Examples:** The following table specifies how cells are ordered on a simplex 4-up layout for a 2-Sheet stack depending on @*PresentationDirection*. In each example, the left set of 4 numbers represent the top Sheet and the right set of 4 numbers represent the bottom Sheet of the 2-Sheet stack.

Xyz	xyz	XYZ	Zxy	yxZ
1 2 5 6 3 4 7 8	2 1 6 5 4 3 8 7	7 8 3 4 5 6 1 2	4 2 3 1 8 6 7 5	7 5 3 1 8 6 4 2

Rotate = "Rotate0"	enumeration	Orthogonal rotation including the implied translation to be applied to the grid of ***PageCell*** elements on the entire surface relative to the process coordinate system. **Allowed values are:** Rotate0 Rotate90 – 90° counterclockwise rotation. Rotate180 – 180° rotation. Rotate270 – 90° clockwise rotation. **Note:** For details of orthogonal rotations, refer to ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component. If a @*RotatePolicy* value other than "NoRotate" is specified in ***FitPolicy***, the actual rotation specified in @*Rotate* MAY be modified accordingly. **Note:** A rotation of the grid also rotates the gutters (i.e., it is applied after all other parameters have been evaluated and applied).

Table 8.167: LayoutPreparationParams Resource (Sheet 6 of 7)

<table>
<tr><th>NAME</th><th>DATA TYPE</th><th>DESCRIPTION</th></tr>
<tr><td>Sides = "OneSidedFront"</td><td>enumeration</td><td>Indicates whether the content layout is to be imaged on one or both sides of the media. When the content layout consists of multiple input RunList pages to be imposed on a single surface, @Sides applies to the entire unfolded Sheet.
When a different value for the @Sides attribute is encountered, it SHALL force a new Sheet. However, when the same value for the @Sides attribute is restated for consecutive pages, it is the same as if that restatement was not present.
Allowed values are from: ▸ Table 8.126 Sides Attribute Values.</td></tr>
<tr><td>StackDepth ?</td><td>integer</td><td>The number of Sheets in a stack that are processed when imposing down the Z axis. If not specified, the entire Job defines one stack.</td></tr>
<tr><td>StepDocs ?
Modified in JDF 1.2</td><td>XYPair</td><td>A list of two integers that species how to impose multiple Instance Documents on one Sheet. The first value specifies the document repeats along the X axis, the second value specifies the repeats along the Y axis. Each entry of @NumberUp SHALL be an integer multiple of @StepRepeat * @StepDocs. Positive values define grouped step and repeat whereas negative values define alternating step and repeat. The following examples, where documents are denoted A and B while pages are denoted 1 and 2, have @PresentationDirection = "Xyz", @NumberUp = "4 4" and @StepRepeat = "2 2 1" and @StepDocs =:
<table>
<tr><td>"2 1" (2 documents in X, 1 in Y)
A1 A1 B1 B1
A1 A1 B1 B1
A2 A2 B2 B2
A2 A2 B2 B2</td><td>"1 2" (1 document in X, 2 in Y)

A1 A1 A2 A2
A1 A1 A2 A2
B1 B1 B2 B2
B1 B1 B2 B2</td></tr>
</table></td></tr>
<tr><td>StepRepeat ?</td><td>IntegerList</td><td>A list of three integers that specifies the number of identical pages to impose. The first value specifies the repeats along the X axis, the second value specifies the repeats along the Y axis, and the third value specifies the repeats down the stack — the Z axis. Each entry of @NumberUp SHALL be an integer multiple of @StepRepeat * @StepDocs. Positive values define grouped step and repeat, whereas negative values define alternating step and repeat. Note that negative values are illegal for the third component, since the total depth of the stack might be unknown. The following examples have @PresentationDirection = "Xyz", @NumberUp = "4 4" and @StepRepeat =:
<table>
<tr><td>"2 2 1"
1 1 2 2
1 1 2 2
3 3 4 4
3 3 4 4</td><td>"-2 2 1"
1 2 1 2
1 2 1 2
3 4 3 4
3 4 3 4</td><td>"-2 –2 1"
1 2 1 2
3 4 3 4
1 2 1 2
3 4 3 4</td><td>"2 –2 1"
1 1 2 2
3 3 4 4
1 1 2 2
3 3 4 4</td><td>"1 4 1"
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4</td></tr>
</table></td></tr>
<tr><td>SurfaceContentsBox ?
Modified in JDF 1.1A</td><td>rectangle</td><td>This box, specified in Layout coordinate space, defines the area into which PageCell elements are distributed. The lower left corner of the rectangle specified by the value of this attribute establishes the coordinate system into which the content is mapped and SHOULD have a value of "0 0". @SurfaceContentsBox MAY imply clipping. This attribute SHOULD be supplied in order to get predicable placement of content. If this attribute is not supplied, a rectangle with the origin at "0 0" and an extent that MAY be dependent on the dimensions of the Media is implied.</td></tr>
</table>

Table 8.167: LayoutPreparationParams Resource (Sheet 7 of 7)

NAME	DATA TYPE	DESCRIPTION
VerticalCreep ?	IntegerList	Specifies which vertical gutters creep. The allowed values are zero-based indexes that reference vertical gutters formed by multiple columns of pages in a multi-up page layout specified by the first value of @*NumberUp*. The value for an entry in this list SHALL be between zero and two (2) less then the first value of @*NumberUp*. An index value outside of this range is ignored. If not specified then vertical gutters SHALL NOT creep. Gutters identified by this attribute are known as explicitly creeping gutters whereas those not identified are known as implicitly creeping gutters. Note: In order preserve the absolute position of the center lines of all gutters across all Sheets, only specify alternating gutters starting with gutter index zero.
DeviceMark ?	element	Details how device-dependent marks are to be generated. If not specified, the marks are device-dependent.
ExternalImpositionTemplate ? New in JDF 1.3	refelement	Reference to an external imposition template in a proprietary format. ***LayoutPreparationParams*** SHOULD NOT contain information that overlaps information specified in ***ExternalImpositionTemplate***. Information specified in ***LayoutPreparationParams*** overrides parameters specified in ***ExternalImpositionTemplate***.
FitPolicy ?	element	Details how to fit the grid of ***PageCell*** elements onto the @*SurfaceContentsBox*.
ImageShift ?	element	Details how to place the grid of ***PageCell*** elements into the @*SurfaceContentsBox*. ***ImageShift*** SHALL be applied before any transformations of the grid of ***PageCell*** elements as specified by @*Rotate* or ***FitPolicy***. The reference origin of the grid of page cells is the lower left corner of the trim box of the lower left page cell of the grid of the first Sheet prior to applying any creep. Note that ***ImageShift*** will generally be required to allow for space when @*CreepValue* is positive.
InsertSheet *	refelement	Additional Sheets to be inserted before, after or within a Job.
JobField *	element	Specific information about this kind of mark object.
Media ?	refelement	Specific information about the media.
PageCell ? Modified in JDF 1.1A	element	***PageCell*** elements describe how page contents will be imaged onto individual page cells. At most one ***PageCell*** SHALL be specified and it is applied to all page cells on both surfaces of a Sheet.

8.88 PageDistributionScheme Attribute Values

Table 8.168: PageDistributionScheme Attribute Values

VALUE	DESCRIPTION
Perfect	Distribute finished pages onto a sequence of one or more Signatures in proper order for perfect binding. For this page distribution scheme, creep is usually not used.
PerfectFront New in JDF 1.5	Distribute finished pages onto a sequence of one or more Signatures in proper order for perfect binding where only the reader order front pages respective of the finished product are placed in the signature layout. For left hand binding, only right facing page cells contain pages and left facing page cells are empty. For right hand binding, only left facing page cells contain pages and right facing page cells are empty. For top binding, only bottom facing page cells contain pages and top facing page cells are empty. For bottom binding, only top facing page cells contain pages and bottom facing page cells are empty. For this page distribution scheme, creep is usually not used.
Saddle	Distribute finished pages onto a sequence of one or more imposition layouts in proper order for saddle stitch binding. For this page distribution scheme, creep SHALL be applied only to odd-numbered vertical gutters where any even-numbered gutters SHALL automatically creep in the opposite direction.
SaddleFront New in JDF 1.5	Distribute finished pages onto a sequence of one or more imposition layouts in proper order for saddle stitch binding where only the reader order front pages respective of the finished product are placed in the signature layout. For left hand binding, only right facing page cells contain pages and left facing page cells are empty. For right hand binding, only left facing page cells contain pages and right facing page cells are empty. For top binding, only bottom facing page cells contain pages and top facing page cells are empty. For bottom binding, only top facing page cells contain pages and bottom facing page cells are empty. For this page distribution scheme, creep SHALL be applied only to odd-numbered vertical gutters where any even-numbered gutters SHALL automatically creep in the opposite direction.
Sequential	The finished pages are distributed onto the multi-up layout according to the value of the @*PresentationDirection* attribute.

8.88.1 PageCell

*Table 8.169: **PageCell** Element (Sheet 1 of 2)*

NAME	DATA TYPE	DESCRIPTION
Border ? Modified in JDF 1.1A	double	A number indicating the width in points of a drawn border line, that appears around the trim region specified by the explicit or implied value of @*TrimSize*. A value of "0" specifies no border. If the value of this attribute is non-zero and positive, then a border of that specified width will be drawn to the outside of the page cell whose inside dimension is the same as the explicit or implied value of the @*TrimSize* attribute. The border marks SHALL NOT overwrite the page contents of the trimmed page. Note that when the page cells are distributed evenly over the area of the @*SurfaceContentsBox*, the page cells position and/or size can be adjusted to accommodate the border. If the value of this attribute is non-zero and negative, then a border of a width specified by the absolute value of this attribute will be drawn to the inside of the page cell whose outside dimension is the same as the explicit or implied value of the @*TrimSize* attribute. The border marks MAY overwrite the page contents of the trimmed page. The rectangle defined by the inside edge of the border defines a @*ClipBox* beyond which no content will be imaged.
ClipBox ?	rectangle	Defines a rectangle with an origin relative to the lower left corner of the page cell rectangle defined by the explicit or implied value of the @*TrimSize* attribute. Page content data imaged outside of the region defined by this rectangle SHALL be clipped. If @*ClipBox* is larger than @*TrimSize*, it is used to specify a bleed region. If not specified, its default value is "0 0 X Y" where X and Y are the explicit or implied values of @*TrimSize*.

*Table 8.169: **PageCell** Element (Sheet 2 of 2)*

NAME	DATA TYPE	DESCRIPTION
MarkList ?	NMTOKENS	List of marks that are to be marked on each page cell. The appearance of the marks are defined by the process implementation. **Values include:** CIELABMeasuringField ColorControlStrip ColorRegisterMark CutMark DensityMeasuringField IdentificationField JobField PaperPathRegisterMark RegisterMark ScavengerArea
Rotate = "Rotate0"	enumeration	Orthogonal rotation to be applied to the contents in each page cell. **Allowed values are:** Rotate0 Rotate90 – 90° counterclockwise rotation. Rotate180 – 180° rotation. Rotate270 – 90° clockwise rotation. **Note:** For details of orthogonal rotation, refer to ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component. If a @*RotatePolicy* value other than "NoRotate" is specified in ***FitPolicy***, the actual rotation specified in @*Rotate* MAY be modified accordingly.
TrimSize ? Modified in JDF 1.1A	XYPair	Defines the dimensions of the page cell. The lower left corner of the rectangle specified by the value of this attribute establishes the coordinate system into which the page content is mapped. If not specified, @*TrimSize* is calculated by subtracting the gutters from the ***LayoutPreparationParams***/@*SurfaceContentsBox* and dividing by the appropriate @*NumberUp* value.
Color ?	refelement	Color of the border.
DeviceMark ?	element	Details how device dependent marks are to be generated. Defaults to the value of ***DeviceMark*** in the parent ***LayoutPreparationParams***.
FitPolicy ?	element	Details how page content is fit into the page cells. If the dimensions of the page contents vary, ***FitPolicy*** is applied to the contents of each cell individually.
ImageShift ?	element	Element which describes how content SHALL be placed into the page cells. X and Y are specified in the coordinate system of the ***PageCell***.

8.88.2 ImageShift

ImageShift elements describe how the grid of page cells will be imaged onto media, when ***ImageShift*** is specified in the context of ***LayoutPreparationParams***. When ***ImageShift*** is specified in the context of a ***PageCell***, it specifies how content is imaged into the respective page cells.

Table 8.170: ImageShift Element

NAME	DATA TYPE	DESCRIPTION
PositionX ?	enumeration	Indicates how content SHALL be positioned horizontally. The @*ShiftBack* and @*ShiftFront* are applied after @*PositionX* and @*PositionY*. **Allowed values are:** Center – Center the content horizontally without regard to limitations of the receiving container. Left – Position the left edge of the content so that it is coincident with the left edge of the receiving container. Right – Position the right edge of the content so that it is coincident with the right edge of the receiving container. Spine – Position the content so that it is coincident with the vertical binding edge of the receiving container. New in JDF 1.2 None – Place the content wherever the print data specify. Deprecated in JDF 1.3
PositionY ?	enumeration	Indicates how content SHALL be positioned vertically. The @*ShiftBack* and @*ShiftFront* are applied after @*PositionX* and @*PositionY*. **Allowed values are:** Bottom – Position the bottom edge of the content so that it is coincident with the bottom edge of the receiving container. Center – Center the content horizontally without regard to limitations of the receiving container. Top – Position the top edge of the content so that it is coincident with the top edge of the receiving container. Spine – Position the content so that it is coincident with the horizontal binding edge of the receiving container. New in JDF 1.2 None – Place the content wherever the print data specify. Deprecated in JDF 1.3
ShiftBack ?	XYPair	The amount in X and Y direction by which the content SHALL be shifted on the back side of the receiving container. If not specified, @*ShiftBack* SHALL be calculated from @*ShiftFront* so that the content remains aligned.
ShiftFront ="0 0"	XYPair	The amount in X and Y direction by which the content SHALL be shifted on the front side of the receiving container.

Figure 8-39: Diagram of a 4-up cross-folded saddle-stitch imposition with vertical gutter creep

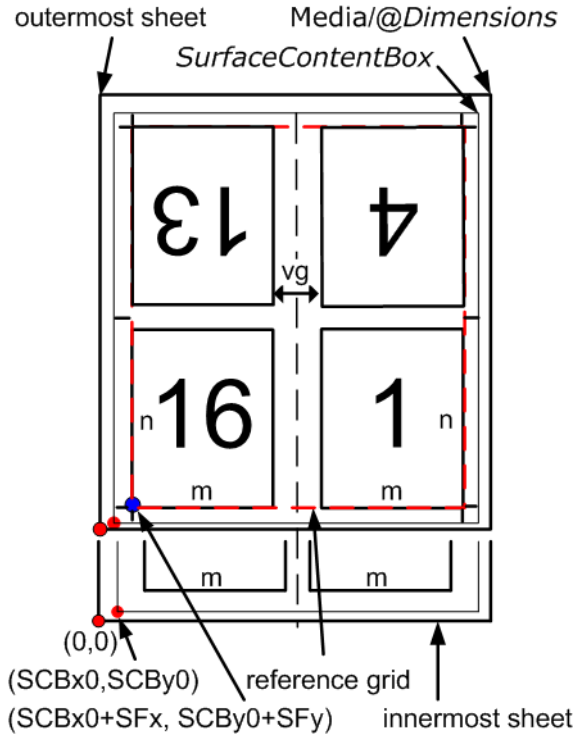
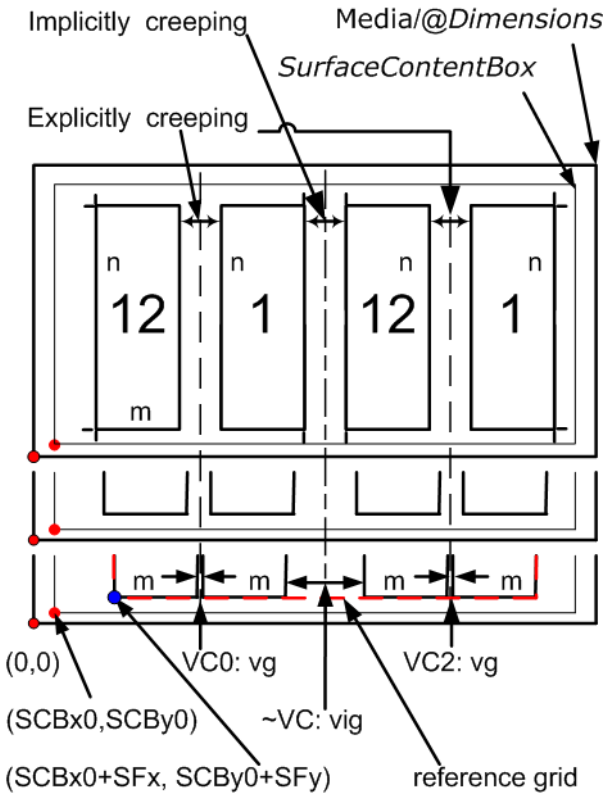


Figure 8-40: Diagram of a step-and-repeat 2-up saddle-stitch imposition with vertical spine gutter creep



Description for Figure 8-39: Diagram of a 4-up cross-folded saddle-stitch imposition with vertical gutter creep

The following terms are used in Figure 8-39: Diagram of a 4-up cross-folded saddle-stitch imposition with vertical gutter creep.

- **reference grid** in Figure 8-39: Diagram of a 4-up cross-folded saddle-stitch imposition with vertical gutter creep refers to the dashed red box around page cells of outermost Sheet, which indicates the size of the reference grid used in calculating grid placement relative to the @*SurfaceContentsBox* origin using ***LayoutPreparationParams***/@*ImageShift*
- **SCBx0**, **SCBy0**, **SFx**, **SFy**, **m**, **n** and **vg** are used in the **JDF** below.

Figure 8-39: Diagram of a 4-up cross-folded saddle-stitch imposition with vertical gutter creep illustrates the **JDF** below. The **JDF** assumes that the dimensions of the ***RunList*** page's trim rectangle matches ***PageCell***/@*TrimSize*, whose dimensions are m by n (width and height) in the **JDF** example below. The Sheet with the widest creep gutter is on the top of the logical Sheet stack.

Example 8.29: LayoutPreparationParams: JDF for ▸ Figure 8-39: Diagram of a 4-up cross-folded saddle-stitch imposition with vertical gutter creep

```
<LayoutPreparationParams Status="Available" Class="Parameter" ID="LPP_2"
      NumberUp="2 2" PageDistributionScheme="Saddle" FoldCatalog="F8-7"
      FoldCatalogOrientation="Flip270" Sides="TwoSidedFlipY"
      StepRepeat="1 1 1" SurfaceContentsBox="0 0 612 792" BindingEdge="Left"
      VerticalCreep="0" GutterMinimumLimit="5 5" CreepValue="0 -5"
      Gutter="20 20" FinishingOrder="FoldCollect" FrontMarkList="CutMark">
   <!-- Note: the value of some attributes in LayoutPreparationParams and
      subElements relate to symbols in the above Figure:
         SurfaceContentsBox="SCBx0 SCBy0 SCBx1 SCBy1"
         GutterMinimumLimit="hml vml"
         CreepValue="0 -vc"
         Gutter="hg vg"
         TrimSize="m n"
         ShiftFront="SFx SFy"
   -->
   <PageCell TrimSize="612 792">
      <ImageShift PositionX="Spine" PositionY="Center" />
   </PageCell>
   <ImageShift PositionY="Bottom" PositionX="Left" ShiftFront="20 20"/>
</LayoutPreparationParams>
```

Description for ▸ Figure 8-40: Diagram of a step-and-repeat 2-up saddle-stitch imposition with vertical spine gutter creep

The following terms are used in ▸ Figure 8-40: Diagram of a step-and-repeat 2-up saddle-stitch imposition with vertical spine gutter creep.

- **reference grid** in ▸ Figure 8-40: Diagram of a step-and-repeat 2-up saddle-stitch imposition with vertical spine gutter creep refers to the dashed red box around page cells of innermost Sheet, which indicates the size of the reference grid used in calculating grid placement relative to the @*SurfaceContentsBox* origin using ***LayoutPreparationParams***/@*ImageShift*
- **SCBx0**, **SCBy0**, **SFx**, **SFy**, **m**, **n**, **vg** and **vig** are used in the **JDF** below.

▸ Figure 8-40: Diagram of a step-and-repeat 2-up saddle-stitch imposition with vertical spine gutter creep illustrates the **JDF** below. The **JDF** assumes that the dimensions of source content page rectangle matches ***PageCell***/@*TrimSize*, whose dimensions are m by n (width and height) in the **JDF** example below.

Example 8.30: LayoutPreparationParams: JDF for

▸ Figure 8-40: Diagram of a step-and-repeat 2-up saddle-stitch imposition with vertical spine gutter creep

```
<LayoutPreparationParams Class="Parameter" ID="LPP_1" Status="Available"
      NumberUp="4 1" PageDistributionScheme="Saddle" FoldCatalog="F4-1"
      FoldCatalogOrientation="Flip0" Sides="TwoSidedFlipY" StepRepeat="2 1 1"
      SurfaceContentsBox="0 0 612 792" VerticalCreep="0 2"
      ImplicitGutter="0 30" ImplicitGutterMinimumLimit="0 20" CreepValue="0 5"
      Gutter="0 10" FinishingOrder="GatherFold" FrontMarkList="CutMark">
      <!--Note: folding pattern F4-1 applies to each of the two 2x1
                signatures
            Note: step and repeat by two in X direction logically divides grid
                into two 2x1 signatures
            Note: first (VC0) and third (VC2) vertical gutters are explicitly
                creeping and the rest (~VC) are implicitly creeping
            Note: Positive vertical creep value indicates initial gutter
                Widths of inner most Sheet
            Note: cut marks are located relative to largest page cell grid
                trim box
            Note: the value of some attributes in LayoutPreparationParams and
                subElements relate to symbols in the above Figure:
                   SurfaceContentsBox="SCBx0 SCBx1 SCBy0 SCBy1"
                   ImplicitGutter="0 vig"
                   ImplicitGutterMinimumLimit="0 vig1"
                   CreepValue="0 +vc"
                   Gutter="0 vg"
                   TrimSize="m n"
                   ShiftFront="SFx SFy"
      -->
   <PageCell TrimSize="612 792">
      <ImageShift PositionX="Spine" PositionY="Bottom"/>
   </PageCell>
   <ImageShift PositionY="Bottom" PositionX="Left" ShiftFront="20 20"/>
</LayoutPreparationParams>
```

8.89 LayoutShift

New in JDF 1.4.

Resource Properties

Resource Class: Parameter

Example Partition: @*SheetName*, @*Side*, @*Separation*

Input of Processes: **LayoutShifting**

Table 8.171: LayoutShift Resource

NAME	DATA TYPE	DESCRIPTION
ShiftPoint +	element	Description of separation dependent transformations for a given point on the ***Layout***.

8.89.1 ShiftPoint

Table 8.172: ShiftPoint Element

NAME	DATA TYPE	DESCRIPTION
CTM	matrix	@*CTM* that SHALL be applied to the Separation after all other transformations.
Position	XYPair	Point that this ***ShiftPoint*** applies to. **Note:** The interpolation algorithm between ***ShiftPoint*** positions is implementation dependent.

Example 8.31: LayoutShift

New in JDF 1.4.

Example of absolute positions with @*Position*

```
<!--LayoutShift SHOULD be partitioned: at least Side and Separation
      will make sense -->
<LayoutShift ID="r000005" Class="Parameter" Status="Unavailable"
      PartIDKeys="Side Separation" >
   <!--LayoutShift SHOULD be partitioned: at least Side and Separation
         will make sense-->
   <!--Note that the interpolation algorithm between positions is
         implementation dependent-->
   <LayoutShift Side="Front">
      <LayoutShift Separation="Cyan">
         <ShiftPoint CTM="1 0 0 1 0 0" Position="360 500"/>
         <ShiftPoint CTM="1 0 0 1 0 2" Position="1800 500"/>
         <ShiftPoint CTM="1 0 0 1 1 0" Position="360 1500"/>
         <ShiftPoint CTM="1 0 0 1 1 2" Position="1800 1500"/>
         <ShiftPoint CTM="1 0 0 1 2 0" Position="360 2500"/>
         <ShiftPoint CTM="1 0 0 1 2 2" Position="1800 2500"/>
         <ShiftPoint CTM="1 0 0 1 3 0" Position="360 3500"/>
         <ShiftPoint CTM="1 0 0 1 3 2" Position="1800 3500"/>
      </LayoutShift>
      <LayoutShift Separation="Magenta">
         <ShiftPoint CTM="1 0 0 1 1 1" Position="360 500"/>
         <ShiftPoint CTM="1 0 0 1 1 3" Position="1800 500"/>
         <ShiftPoint CTM="1 0 0 1 2 1" Position="360 1500"/>
         <ShiftPoint CTM="1 0 0 1 2 3" Position="1800 1500"/>
         <ShiftPoint CTM="1 0 0 1 3 1" Position="360 2500"/>
         <ShiftPoint CTM="1 0 0 1 3 3" Position="1800 2500"/>
         <ShiftPoint CTM="1 0 0 1 4 1" Position="360 3500"/>
         <ShiftPoint CTM="1 0 0 1 4 3" Position="1800 3500"/>
      </LayoutShift>
      <LayoutShift Separation="Yellow">
         <ShiftPoint CTM="1 0 0 1 2 2" Position="360 500"/>
         <ShiftPoint CTM="1 0 0 1 2 4" Position="1800 500"/>
         <ShiftPoint CTM="1 0 0 1 3 2" Position="360 1500"/>
         <ShiftPoint CTM="1 0 0 1 3 4" Position="1800 1500"/>
         <ShiftPoint CTM="1 0 0 1 4 2" Position="360 2500"/>
         <ShiftPoint CTM="1 0 0 1 4 4" Position="1800 2500"/>
         <ShiftPoint CTM="1 0 0 1 5 2" Position="360 3500"/>
         <ShiftPoint CTM="1 0 0 1 5 4" Position="1800 3500"/>
      </LayoutShift>
      <LayoutShift Separation="Black">
         <ShiftPoint CTM="1 0 0 1 3 3" Position="360 500"/>
         <ShiftPoint CTM="1 0 0 1 3 5" Position="1800 500"/>
         <ShiftPoint CTM="1 0 0 1 4 3" Position="360 1500"/>
         <ShiftPoint CTM="1 0 0 1 4 5" Position="1800 1500"/>
         <ShiftPoint CTM="1 0 0 1 5 3" Position="360 2500"/>
         <ShiftPoint CTM="1 0 0 1 5 5" Position="1800 2500"/>
         <ShiftPoint CTM="1 0 0 1 6 3" Position="360 3500"/>
         <ShiftPoint CTM="1 0 0 1 6 5" Position="1800 3500"/>
      </LayoutShift>
   </LayoutShift>
</LayoutShift>
```

8.90 LongitudinalRibbonOperationParams

Deprecated in JDF 1.1.

See ▸ Section N.7.13 LongitudinalRibbonOperationParams for details of this deprecated resource.

8.91 ManualLaborParams

New in JDF 1.1

This resource describes the parameters to qualify generic manual work within graphic arts production. Additional ***Comment*** elements will generally be needed to describe the work in human readable form.

Resource Properties

Resource Class: Parameter

Input of Processes: **ManualLabor**

Table 8.173: ManualLaborParams Resource

NAME	DATA TYPE	DESCRIPTION
LaborType Modified in JDF 1.4	NMTOKEN	Type of manual labor that is performed. **Values include:** CreateCoatingForm – create a form to apply coatings during or after printing EditArt – Unspecific art editing (for work on specific files **LayoutElementProduction** SHALL be used) EditMarks – Marks editing EditTraps – Traps editing ManageJob – General work on the Job. PhoneCallToCustomer – Phone calls to ask/inform the customer. SeparateBlanks – Manual separation of blanks from a sheet after die cutting. New in JDF 1.4 **Modification note:** Starting with **JDF** 1.3, the data type is changed from the erroneous NMTOKENS.

8.92 Media

This resource describes a physical element that represents a raw, unexposed printable surface such as sheet, film or plate. @*Gloss*, @*MediaColorName* and @*Opacity* attributes provide media characteristics pertinent to color management.

Resource Properties

Resource Class: Consumable

Resource referenced by: ***Color***/***Media***, ***Media***, ***DieLayout***, ***DigitalPrintingParams***, ***EmbossingParams***/***Emboss***, ***ExposedMedia***, ***FeedingParams***/***Feeder***, ***FeedingParams***/***CollatingItem***, ***ImageSetterParams***,***InterpretingParams***, ***Layout***, ***LayoutPreparationParams***, ***Media***/***MediaLayers***, ***RasterReadingParams***, ***ShapeDef***, ***StrippingParams***, ***Tile***

Intent Pairing: ***MediaIntent***, ***HoleMakingIntent***, ***ProofingIntent***

Example Partition: "Location", "SheetName", "Side", "SignatureName", "TileID", "WebName"

Input of Processes: **Bending**, **BoxPacking**, **Bundling**, **CaseMaking**, **ConventionalPrinting**, **ContactCopying**, **Cutting**, **DigitalPrinting**, **Embossing**, **Feeding**, **ImageSetting**, **Laminating**, **Varnishing**, **Wrapping**

Output of Processes: **Cutting**, **Feeding**

Table 8.174: Media Resource (Sheet 1 of 7)

NAME	DATA TYPE	DESCRIPTION
BackBrightness ? New in JDF 1.5	double	Equivalent to @*Brightness* (see below), but applied to the back surface of the Media. If not specified, the value of @*Brightness* applies to the Front and Back surfaces of the Media.
BackCoatingDetail ? New in JDF 1.4	NMTOKEN	Identical to @*FrontCoatingDetail* (see below), but applied to the back surface of the media. **Default value is from:** @*FrontCoatingDetail*. **Allowed values are from: @***FrontCoatingDetail***.**
BackCoatings ?	enumeration	Identical to @*FrontCoatings* (see below), but applied to the back surface of the media. **Allowed value is from:** ▸ Coating.
BackGlossValue ? New in JDF 1.2	double	Gloss of the back surface of the media in gloss units as defined by ▸ [ISO8254-1:1999]. When not known, @*BackGlossValue* defaults to the value of @*FrontGlossValue*.
BackISOPaperSubstrate ? New in JDF 1.6	enumeration	@*ISOPaperSubstrate* SHALL specify the back surface of paper material defined in accordance with the print substrate set forth in ▸ [ISO12647-2:2013]. If not specified, the value of @*ISOPaperSubstrate* SHALL be applied. **Allowed value is from:** ▸ ISOPaperSubstrate**.**

Table 8.174: Media Resource (Sheet 2 of 7)

NAME	DATA TYPE	DESCRIPTION
Brightness ? Modified in JDF 1.5	double	Reflectance percentage of diffuse blue reflectance as defined by ▸ [ISO2470:1999]. The reflectance is reported per ▸ [ISO2470:1999] as the diffuse blue reflectance factor of the Media in percent to the nearest 0.5% reflectance factor.See also @*BackBrightness*. **Modification note:** Starting with **JDF** 1.5, the brightness MAY be specified separately for the Front and Back surfaces by specifying both @*Brightness* and @*BackBrightness*.
CIETint ? New in JDF 1.2	double	Average CIE tint value. Average CIE tint is calculated according to equations given in ▸ [TAPPI T560].
CIEWhiteness ? New in JDF 1.2	double	Average CIE whiteness value. Average CIE whiteness is calculated according to equations given in ▸ [TAPPI T560].
ColorName ? New in JDF 1.1 Deprecated in JDF 1.2	string	Link to a definition of the color specifics. The value of @*ColorName* color SHOULD match the @*Name* attribute of a ***Color*** defined in a ***ColorPool*** resource that is linked to the process using this ***Media*** resource. **Deprecation note:** Starting with **JDF** 1.2, use @*MediaColorName* and @*MediaColorNameDetails*.
CoreWeight ? New in JDF 1.3	double	Weight of the core of a Roll, in grams [g]
Dimension ? Modified in JDF 1.4	XYPair	The X and Y dimensions of the chosen medium, measured in points. @*Dimension* specifies the outer bounding box of the ***Media***. The X, Y values of @*Dimension* establishes the user coordinate system into which content is mapped (i.e., the origin is in the lower left corner of the rectangle defined by 0 0 X Y). In case of "Roll" media, the X coordinate specifies the reel width and the Y coordinate specifies the length of the Web in points. If a @*Dimension* coordinate is unknown, the value SHALL be "0". If not specified, the dimension is unknown. If either or both X or Y = "0" (i.e., unknown), the default orientation is assumed to be portrait (i.e., Y > X). **Values include those from:** ▸ Table F.1 Media Sizes. New in JDF 1.4 **Modification note:** Starting with **JDF** 1.4, the description states that @*Dimension* specifies the outer bounding box of the ***Media*** and new values are specified.
Flute ? New in JDF 1.3	NMTOKEN	Single, capital letter that specifies the Flute type of corrugated media. Although the classification of flutes using a letter code “A”, “B”, etc., are used very frequently (e.g., in the specification of the order for a box), there seems to be no agreement on the exact numerical specification of those categories. Slightly varying numbers for flute size and frequency can be found between regions (European versus US) and between vendors. **Values include:** A B C
FluteDirection ? New in JDF 1.3	enumeration	Direction of the flute of corrugated Media in the coordinate system of the ***Media***. **Allowed value is from: Allowed values are:** LongEdge – Along the longer axis as defined by @*Dimension*. ShortEdge – Along the shorter axis as defined by @*Dimension*. XDirection – Along the X-axis of the Media coordinate system YDirection – Along the Y-axis of the Media coordinate system
FrontCoatingDetail ? New in JDF 1.4	NMTOKEN	Describes (beyond @*FrontCoatings*) the coating to the front surface of the media and possibly the technology used to apply the coating. **Values include:** Cast
FrontCoatings ? Modified in JDF 1.4	enumeration	What pre-process coating has been applied to the front surface of the media. **Allowed value is from:** ▸ Coating.

Table 8.174: Media Resource (Sheet 3 of 7)

NAME	DATA TYPE	DESCRIPTION
FrontGlossValue ? New in JDF 1.2	double	Gloss of the front side of the of the media in gloss units as defined by ▸ [ISO8254-1:1999]. Refer also to ▸ [TAPPI T480] for examples of gloss calculation.
Grade ? Modified in JDF 1.5 Deprecated in JDF 1.6	integer	The @*Grade* of the media on a scale of 1 through 5. The @*Grade* is ignored if @*MediaType* is not "Paper". @*Grade* of paper material is defined in accordance with the paper "types" set forth in ▸ [ISO12647-2:2004]. If a workflow supports @*ISOPaperSubstrate*, and both @*Grade* and @*ISOPaperSubstrate* are present, it SHALL use @*ISOPaperSubstrate*. **Note:** ▸ [ISO12647-2:2004] paper type attribute Values do NOT align with U.S. GRACOL paper grade attribute Values (e.g., ▸ [ISO12647-2:2004] type 1 does not equal U.S. GRACOL grade 1). The values define offset printing paper types. **Allowed values are:** 1 – Gloss-coated paper. 2 – Matt-coated paper. 3 – Gloss-coated, Web paper. 4 – Uncoated, white paper. 5 – Uncoated, yellowish paper. **Deprecation note:** Use ISOPaperSubstrate and apply the table in ▸ Section E.3 Paper Grade.
GrainDirection ? New in JDF 1.1 Modified in JDF 1.3	enumeration	Direction of the grain in the coordinate system defined by @*Dimension*. **Allowed values are:** LongEdge – Along the longer axis as defined by @*Dimension*. ShortEdge – Along the shorter axis as defined by @*Dimension*. XDirection – Along the X-axis of the Media coordinate system. New in JDF 1.3 YDirection – Along the Y-axis of the Media coordinate system. New in JDF 1.3
HoleCount ? Deprecated in JDF 1.1	integer	The number of holes that are to be punched in the media (either pre- or post-punched). In **JDF**/1.1, use @*HoleType*, ***Hole*** or ***HoleLine***, which includes the number of holes.
HoleType ="None" New in JDF 1.1	enumerations	Predefined hole pattern. Multiple hole patterns are allowed (e.g., 3-hole ring binding and 4-hole ring binding holes on one piece of media). For details of the hole types, refer to ▸ Appendix L Hole Pattern Catalog. **Allowed values are:** None – No holes. Explicit – Holes are defined in a ***HoleList***. **Allowed values are from:** ▸ Appendix L Hole Pattern Catalog**.**
ImagableSide ?	enumeration	Side of the chosen medium that are to be marked. **Allowed values are:** Front Back Both Neither
InnerCoreDiameter ? New in JDF 1.4	double	Specifies the inner diameter of the core of a Roll, in points. See also @*OuterCoreDiameter* and @*RollDiameter*.
InsideLoss ? New in JDF 1.3	double	The inside loss of corrugated board material in microns [µm]. Note: @*InsideLoss* + @*OutsideGain* need not be exactly equal to thickness.

Table 8.174: Media Resource (Sheet 4 of 7)

NAME	DATA TYPE	DESCRIPTION
ISOPaperSubstrate ? New in JDF 1.5	enumeration	@*ISOPaperSubstrate* supersedes @*Grade* and adds new values to allow for improved papers. If a workflow supports @*ISOPaperSubstrate*, and both @*Grade* and @*ISOPaperSubstrate* are present, it SHALL use @*ISOPaperSubstrate*. @*ISOPaperSubstrate* SHALL specify the type of paper material defined in accordance with the print substrate set forth in ▸ [ISO12647-2:2013]. **Allowed value is from:** ▸ ISOPaperSubstrate. Note: See ▸ Section E.3 Paper Grade for a mapping to the paper grade values defined in ISO12647-2:2004
LabColorValue ? New in JDF 1.2	LabColor	@*LabColorValue* is the CIELAB color value of the media, computed as specified in ▸ [TAPPI T527].
MediaColorName ? Modified in JDF 1.1	enumeration	A name for the color. If more specific, specialized or site-defined media color names are needed, use @*MediaColorNameDetails*. **Allowed value is from:** ▸ NamedColor.
MediaColorNameDetails ? New in JDF 1.2	string	A more specific, specialized or site-defined name for the media color. If @*MediaColorNameDetails* is supplied, @*MediaColorName* SHOULD also be supplied.
MediaQuality ? New in JDF 1.4	string	Named quality description of the media. For folding carton quality, multiple named quality description systems are in use (e.g., GC1, SBB, etc.). For an overview, see ▸ [Pro Carton].
MediaSetCount ?	integer	When the input media is grouped in sets, identifies the number of pieces of media in each set. For example, if the @*MediaTypeDetails* is "PreCutTabs", a @*MediaSetCount* of "5" would indicate that each set includes five tab Sheets.
MediaType ? Modified in JDF 1.5	NMTOKEN	Describes the medium being employed. **Values include:** Blanket – A blanket that is used for varnishing. New in JDF 1.3 CorrugatedBoard New in JDF 1.3 Disc – CD or DVD disc to be printed on. EndBoard – end board used in the **Bundling** process. EmbossingFoil Film Foil GravureCylinder – gravure cylinder. New in JDF 1.3 ImagingCylinder – reusable direct imaging cylinder in a press. New in JDF 1.3 LaminatingFoil MountingTape – for flexo plate mounting tape. New in JDF 1.4 Other – not one of the defined values. Paper Plate Screen – used for Screen Printing. New in JDF 1.4 SelfAdhesive – New in JDF 1.3 Sleeve – for flexo sleeves New in JDF 1.4 ShrinkFoil Textile New in JDF 1.5 Transparency Unknown – Deprecated in JDF 1.2 Vinyl New in JDF 1.5
MediaTypeDetails ? Modified in JDF 1.5	NMTOKEN	Additional details of the chosen medium. **Constraint:** If @*MediaTypeDetails* is specified, @*MediaType* SHALL be specified. **Values include those from:** ▸ Table 8.175 MediaTypeDetails Attribute Values**.**

Table 8.174: Media Resource (Sheet 5 of 7)

NAME	DATA TYPE	DESCRIPTION
MediaUnit = "Sheet" Modified in JDF 1.2	enumeration	Describes the format of the media as it is delivered to the device. **Allowed values are:** Continuous – Continuously connected Sheets which can be fan folded. New in JDF 1.2 Roll Sheet – Individual cut Sheets.
Opacity ? Modified in JDF 1.2	enumeration	The opacity of the media. See @*OpacityLevel* to specify the degree of opacity for any of these values. **Allowed value is from:** ▸ Opacity.
OpacityLevel ? New in JDF 1.2	double	Normalized TAPPI opacity (Cn), as defined and computed in ▸ [ISO2471:1998]. Refer also to ▸ [TAPPI T519] for calculation examples.
OuterCoreDiameter ? New in JDF 1.3	double	Specifies the outer diameter of the core of a Roll, in points. See also @*InnerCoreDiameter* and @*RollDiameter*.
OutsideGain ? New in JDF 1.3	double	The outside gain of corrugated board material in microns [µm].
PlateTechnology ? New in JDF 1.3 Modified in JDF 1.4	enumeration	Exposure technology of the plates. **Allowed values are:** FlexoAnalogSolvent New in JDF 1.4 FlexoAnalogThermal New in JDF 1.4 FlexoDigitalSolvent New in JDF 1.4 FlexoDigitalThermal New in JDF 1.4 FlexoDirectEngraving New in JDF 1.4 InkJet – Exposure with inkjet technology. Note that @*FrontCoatings* = "Inkjet" specifies inkjet specific coating of paper or transparency Media, not of plates. Thermal – Thermal exposure UV – Ultraviolet exposure Visible – Visible light exposure
Polarity ?	enumeration	Polarity of the chosen medium. **Allowed value is from:** ▸ Polarity
PrePrinted = "false"	boolean	Indicates whether the media is preprinted.
PrintingTechnology ? New in JDF 1.4 Modified in JDF 1.5	NMTOKEN	Describes the printing technology that the media or coatings on the media are intended for or optimized for. **Values include:** DyeSublimation Electrostatic InkJet Laser Latex New in JDF 1.5 Offset Thermal UV New in JDF 1.5
Recycled ? Deprecated in JDF 1.2	boolean	If "true", recycled media is requested. If not specified, the ***Media*** might have recycled content. In **JDF** 1.2 and beyond, use @*RecycledPercentage*.
RecycledPercentage ? New in JDF 1.2	double	The percentage, between 0 and 100, of recycled material that the media SHALL contain.
ReliefThickness ? New in JDF 1.4	double	The thickness of the relief, measured in microns [µm]. The floor thickness can be calculated as (@*Thickness* - @*ReliefThickness*). See ▸ Figure 8-42: Relief and Floor Thickness for a Flexo Plate or Flexo Sleeve.

Table 8.174: Media Resource (Sheet 6 of 7)

NAME	DATA TYPE	DESCRIPTION
RollDiameter ?	double	Specifies diameter of a Roll, in points. See also @*InnerCoreDiameter* and @*OuterCoreDiameter*.
ShrinkIndex ? New in JDF 1.1	XYPair	Specifies the ratio of the media linear dimension after shrinking to prior shrinking. The X Value specifies index in the major shrink axis, whereas the Y Value specifies the index in the minor shrink axis. Used to describe shrink wrap media.
SleeveInterlock ? New in JDF 1.4	NMTOKEN	The type of interlock (or notch) to use for a flexo sleeve. **Values include:** Type01 – see ▸ Figure 8-43: Types of Interlocks for Flexo Sleeve. ... Type20 – see ▸ Figure 8-43: Types of Interlocks for Flexo Sleeve.
StockType ? New in JDF 1.1 Modified in JDF 1.4	NMTOKEN	Strings describing the available stock. @*StockType* defines the base size when calculating North American or Japanese paper weights. See ▸ Appendix E Paper Weight and Grade. New in JDF 1.4 Values with Kanji names support Japanese media. **Values include:** Bible – equivalent to "Book". New in JDF 1.4 Book – Base size: 25" x 38" New in JDF 1.4 Bond – Base size: 17" x 22" Bristol – Base size: 22½" x 28½" Coated – equivalent to "Book". New in JDF 1.4 Cover – Base size: 20" x 26" Index – Base size: 25½" x 30½" Ledger – equivalent to "Bond". New in JDF 1.4 Manifold – equivalent to "Bond". New in JDF 1.4 Newsprint – Base size: 24" x 36" Offset – This includes book stock, equivalent to "Book". Tag – equivalent to "Newsprint". Text – equivalent to "Book". Aatoposutoshi – **アートスポト紙** ("art-post paper") is cover stock coated on one side. New in JDF 1.4 Aatoshi – **アート紙** ("art paper") is machine coated paper, available in top quality and medium quality (Joushitsu and Chuushitsu). New in JDF 1.4 Chuushitsushi – **中質紙** ("medium-quality paper") contains a minimum of 70% chemical pulp. New in JDF 1.4 Joushitsushi – **上質紙** ("top-quality paper") contains 100% chemical pulp. New in JDF 1.4 Mashinkootoshi –**マシンコート紙** ("machine coated paper"), also called Kootoshi (**コート紙**), is machine coated paper given only a thin coat of clay. New in JDF 1.4
Texture ? New in JDF 1.1 Modified in JDF 1.6	NMTOKEN	The texture of paper media. **Values include:** Antique – Rougher than vellum surface. Calendared – Extra smooth or polished, uncoated paper. IPP:Coarse – Generic value for coarse finish.New in JDF 1.6 IPP:Fine – Generic value for fine finish.New in JDF 1.6 Glossy – Glossy media. New in JDF 1.6 Linen – Texture of coarse woven cloth. Matte – Matte media. New in JDF 1.6 IPP:Medium – Generic value for finish that is neither IPP:Fine nor IPP:Coarse. New in JDF 1.6 Smooth – Generic term for smooth paper. Stipple – Fine pebble finish. Vellum – Slightly rough surface. **Note:** Values of the form IPP:xxx are provided for mapping to IPP.

Table 8.174: Media Resource (Sheet 7 of 7)

NAME	DATA TYPE	DESCRIPTION
Thickness ?	double	The thickness of the chosen medium, measured in microns [µm]. Note: Thickness is often referred to as caliper.
UserMediaType ? Deprecated in JDF 1.1	NMTOKEN	A human-readable description of the type of media. The value can be used by an operator to select the correct media to load. The semantics of the values will be site-specific. **Deprecation note:** Starting with **JDF** 1.1, @*UserMediaType* has been merged into @*MediaTypeDetails*.
Weight ?	double	Weight of the chosen medium, measured in grams per square meter [g/m^2]. See ▸ Appendix E Paper Weight and Grade for details on converting North American paper weights to g/m^2.
WrapperWeight ? New in JDF 1.3	double	Weight of the wrapper of a roll, in grams [g]
Certification * New in JDF 1.6	element	***Certification*** SHALL specify a paper certification level that the paper fulfills.
Color ? Deprecated in JDF 1.1	refelement	A ***Color*** resource that provides the color of the chosen medium.
ColorMeasurementConditions ? New in JDF 1.2	refelement	Detailed description of the measurement conditions for color measurements used to measure @*LabColorValue*.
HoleList ? New in JDF 1.3	refelement	Explicit list of holes. ***HoleList*** SHALL be specified if @*HoleType* = "Explicit".
MediaLayers ? New in JDF 1.3	element	Subelement describing the layer structure of media such as corrugated or self adhesive materials.
TabDimensions ? New in JDF 1.4	element	Specifies the dimensions of the tabs when @*MediaTypeDetails* = "TabStock", "PreCutTabs" or "FullCutTabs". **Note:** See ***BindingIntent***/***Tabs*** (▸ Table 7.34 Tabs Element) (rather than ***MediaIntent***) for how tabbed media is specified in Product Intent.

8.92.1 MediaTypeDetails Attribute Values

Table 8.175: MediaTypeDetails Attribute Values (Sheet 1 of 2)

VALUE	DESCRIPTION
Aluminum Modified in JDF 1.3	Conventional or CtP press plate.
Backlit New in JDF 1.5	Any media that is designed to be illuminated from the back side.
Cardboard	
CD New in JDF 1.3	CD disc to be printed on.
Cloth New in JDF 1.6	Cloth; e.g., for a hard cover book case.
ContinuousLong	Continuously connected Sheets of an opaque material connected along the long edge.
ContinuousShort	Continuously connected Sheets of an opaque material connected along the short edge.
CtPVisiblePhotoPolymer Deprecated in JDF 1.3	Visible light CtP plate with photo polymer process.
CtPVisibleSilver Deprecated in JDF 1.3	Visible light CtP plate with silver halide process.
CtPThermal Deprecated in JDF 1.3	Thermal CtP plate.
DoubleWall New in JDF 1.3	Double wall corrugated board
DVD New in JDF 1.3	DVD disc to be printed on.
DryFilm	
Envelope	Envelopes that can be used for conventional mailing purposes.
EnvelopePlain	Envelopes that are not preprinted and have no windows.
EnvelopeWindow	Envelopes that have windows for addressing purposes.
EnvelopeWindowLeft New in JDF 1.6	Envelopes that have windows on the left for addressing purposes.
EnvelopeWindowRight New in JDF 1.6	Envelopes that have windows on the right for addressing purposes.
FlexoBase New in JDF 1.4	For the base layer of flexo plates.
FlexoPhotoPolymer New in JDF 1.4	For the photopolymer layer of flexo plates.
Flute	Flute layer of a corrugated board
FullCutTabs	Media with a tab that runs the full length of the medium so that only one tab is visible extending out beyond the edge of non-tabbed media.
ImageSetterPaper	Contact paper as replacement for film.
Labels	Label stock (e.g., a Sheet of peel-off labels).

Table 8.175: MediaTypeDetails Attribute Values (Sheet 2 of 2)

VALUE	DESCRIPTION
Leather New in JDF 1.6	Leather; e.g., for a hard cover book case.
Letterhead	Separately cut Sheets of an opaque material including a letterhead.
MultiLayer	Form medium composed of multiple layers which are preattached to one another (e.g., for use with impact printers).
MultiPartForm	Form medium composed of multiple layers not preattached to one another; each Sheet might be drawn separately from an input source.
Photographic	Separately cut Sheets of an opaque material to produce photographic quality images.
PlateUV Deprecated in JDF 1.3	Press plate for the UV process.
Polyester Modified in JDF 1.3	Conventional or CtP press plate.
PreCutTabs	Media with tabs that are cut so that more than one tab is visible extending out beyond the edge of non-tabbed media.
ScrimBanner New in JDF 1.5	Specific type of vinyl.
SingleFace New in JDF 1.3	Single face corrugated board.
SingleWall New in JDF 1.3	Single wall corrugated board.
Stationery	Separately cut Sheets of an opaque material, includes generic paper.
TabStock	Media with tabs, either precut or full-cut.–
Tractor	Tractor feed with holes.
TripleWall New in JDF 1.3	Triple wall corrugated board
WallPaper New in JDF 1.5	Details of Paper.
WetFilm	Conventional photographic film.

Figure 8-41: Paper Roll with some Roll-specific Information

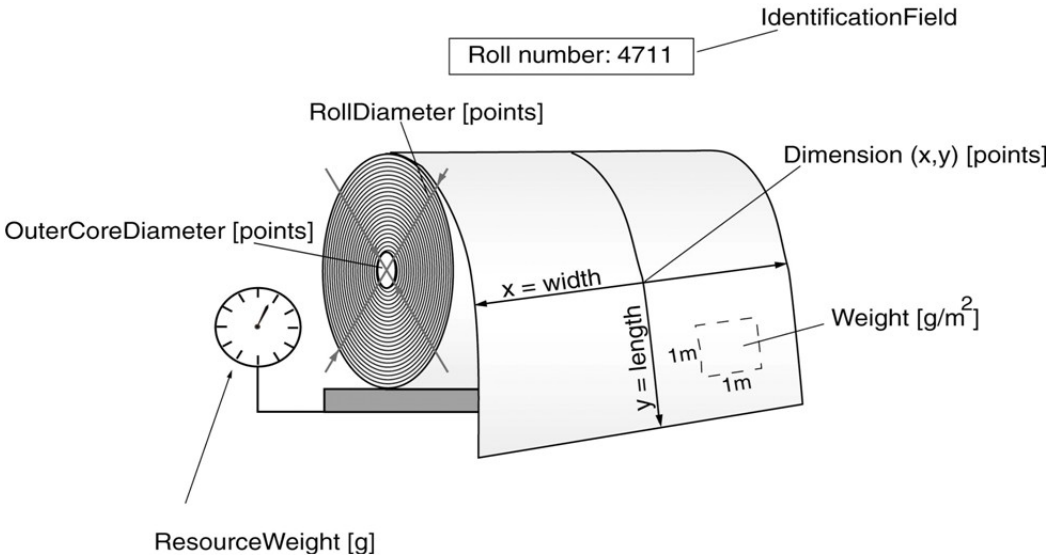


Figure 8-42: Relief and Floor Thickness for a Flexo Plate or Flexo Sleeve

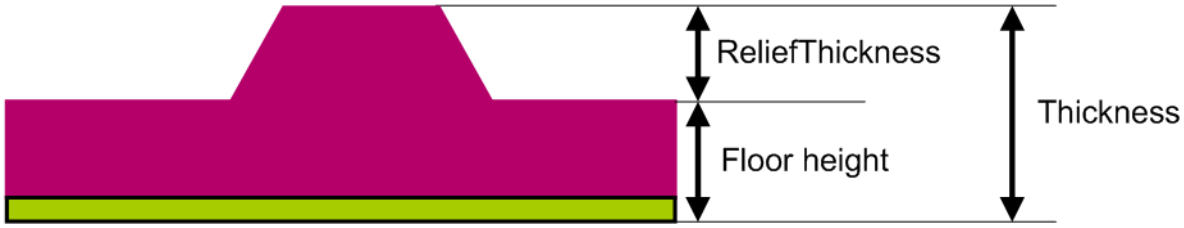
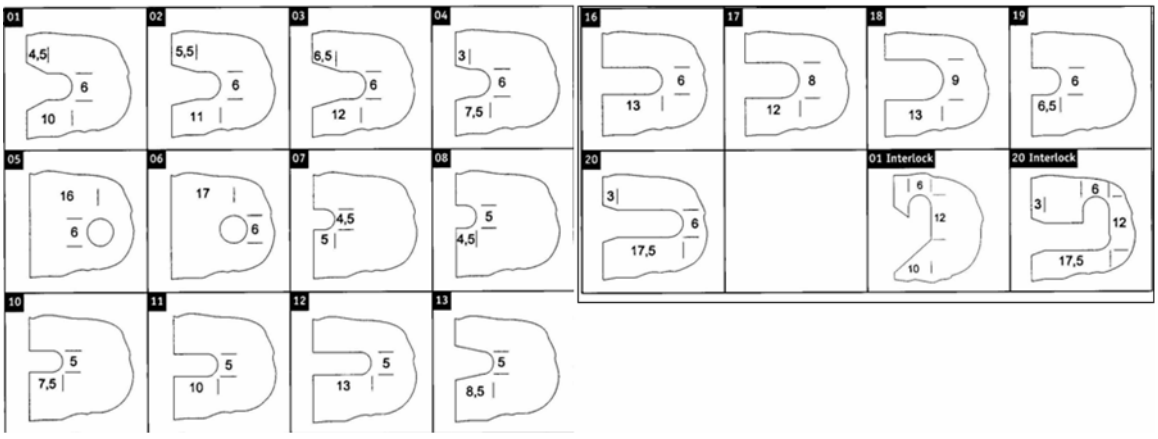


Figure 8-43: Types of Interlocks for Flexo Sleeve



8.92.2 TabDimensions

New in JDF 1.4

Specifies the size and placement of tabs in a bank and in a set of tab stock.

Table 8.176: TabDimensions Element

NAME	DATA TYPE	DESCRIPTION
TabEdge ?	enumeration	Indicates which edge of the media has tabs. Sets the coordinate system for @*TabOffset*, @*TabExtensionDistance*, and @*TabWidth*. **Allowed value is :** ▸ Edge.
TabExtensionDistance ?	double	The positive distance in points that the tab extends beyond the body of the other media. **Note:** Same as ***BindingIntent***/***Tabs***/@*TabExtensionDistance*. **Note:** This value is always included in the value of the overall extent of the ***Media*** defined by ***Media***/@*Dimension*. See ▸ Figure 8-45: Diagram of a Single Bank of Tabs.
TabOffset ?	double	Specifies the magnitude of the distance in points from the two corners to the edge of the first "tab pitch" point of the first tab in the bank along the @*TabEdge*. This distance is the same on both ends of the bank of tabs. See ▸ Figure 8-45: Diagram of a Single Bank of Tabs.
TabSetCollationOrder ?	NMTOKEN	Collation order of media provided in sets. Applicable to sets of pre-cut tabs. See ▸ Figure 8-44: TabSetCollationOrder Attribute Values. **Values include:** Forward – first tab is towards top of stack Reverse – first tab is toward bottom of stack.
TabsPerBank ?	integer	Specifies the number of equal-sized tabs in a single bank if all positions were filled. **Note:** Banks can have tabs only in some of the possible positions. **Note:** Same as ***BindingIntent***/***Tabs***/@*TabsPerBank*. ***Media***/@*MediaSetCount* specifies the number of tabs per set. A set can consist of one or more banks. If ***Media***/@*MediaSetCount* is not an even multiple of @*TabsPerBank*, the last bank in each set is partially filled.
TabWidth ?	double	The width along the @*TabEdge* of each tab as measured along the mid-line of the tab. Each tab is centered within a space called the "tab pitch". See ▸ Figure 8-45: Diagram of a Single Bank of Tabs.

Figure 8-44: TabSetCollationOrder Attribute Values

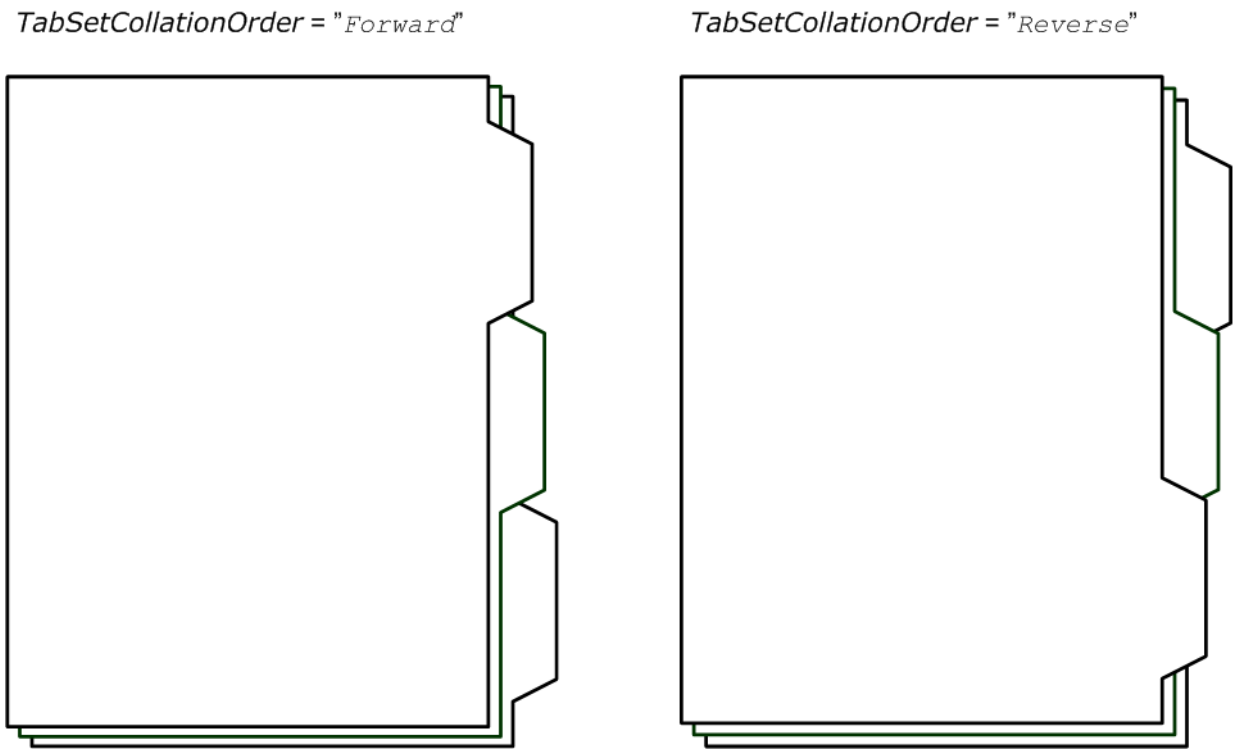
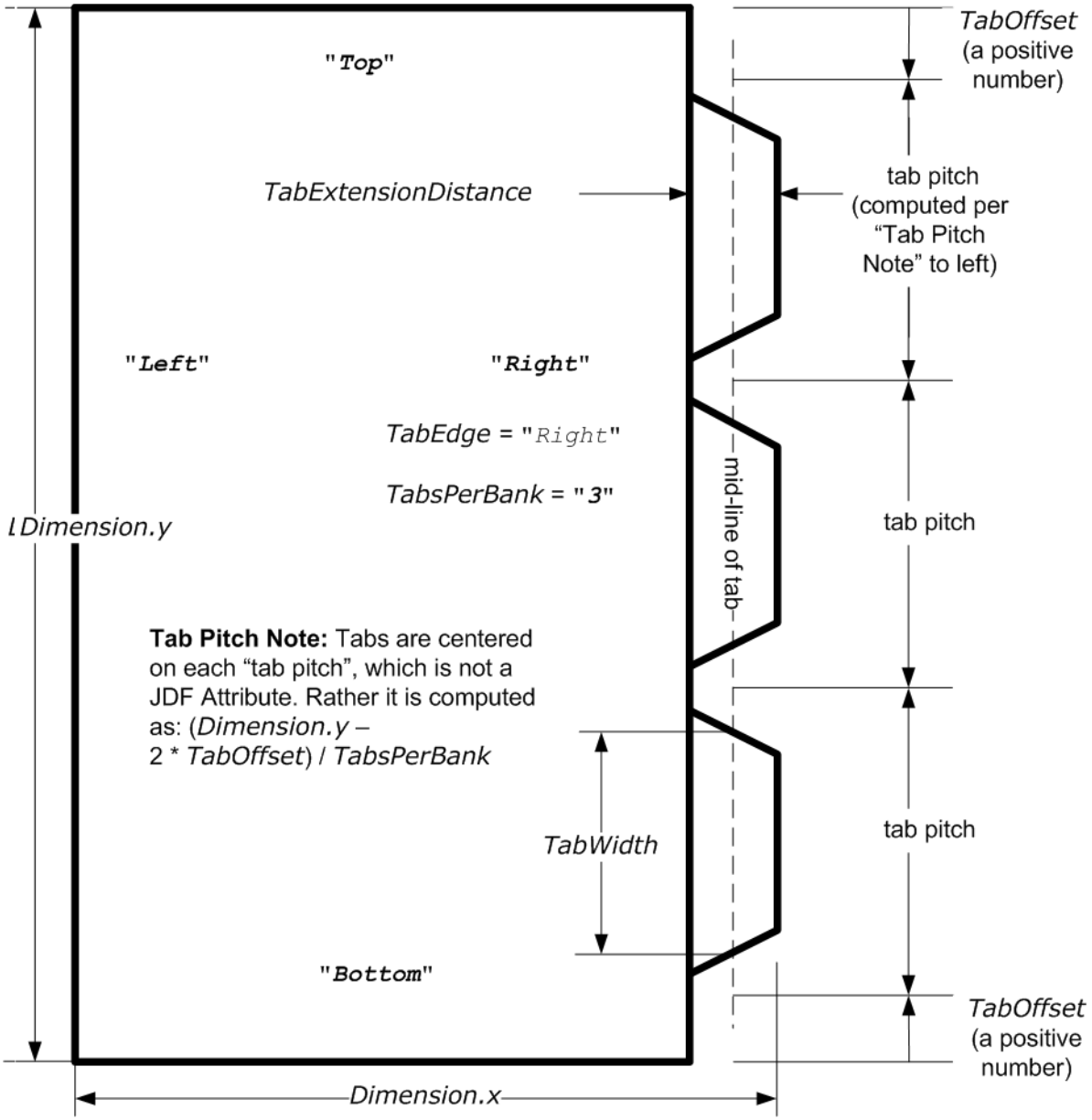


Figure 8-45: Diagram of a Single Bank of Tabs

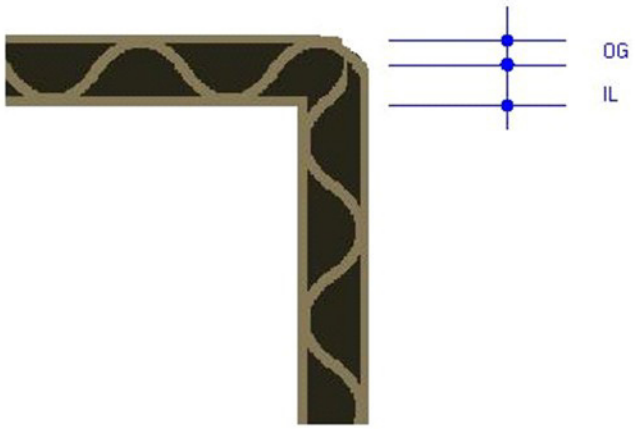


8.92.3 More about Media

8.92.3.1 Inside Loss and Outside Gain

Inside loss and outside gain: dimensional values used in the mechanical design phase of a box.
Note: IL + OG is not exactly equal to thickness. Thickness is most often referred to as caliper.

Figure 8-46: Inside Loss, Outside Gain



8.92.3.2 Corrugated Media:

Corrugated material consists of multiple Sheets of paper (called liners) with fluted material in between. For background information on corrugated media, see ▸ [Corrugated Packaging]. Corrugated media comes in different variants.

- Number of layers:
  - single face (1 liner, 1 flute),
  - single wall (2 liners, 1 flute),
  - double wall (3 liners, 2 flutes),
  - triple wall (4 liners, 3 flutes)
- Flute size and frequency: A, B, C, E, F flute. See ▸ [Corrugated Packaging].

Example 8.32: Media: Corrugated

```
<Media Class="Consumable" ID="M123456" ProductID="B190Y180D1050x120"
       Status="Available" DescriptiveName="B Flute 190Y 180D 1050x1210"
       Dimension="1050.0 120.0" MediaType="CorrugatedBoard"
       MediaTypeDetails="SingleWall" MediaUnit="Sheet" Thickness="2382.0"
       InsideLoss="1000.0" OutsideGain="1380.0" Weight="600">
    <MediaLayers>
       <!-- FrontLiner -->
       <Media DescriptiveName="190gsm clay coated" MediaType="Paper"
              Weight="190" FrontCoatings="Coated"/>
       <!-- Flute -->
       <Media DescriptiveName="Flute" MediaType="Paper" Weight="180"
              FluteDirection="ShortEdge" Flute="B" MediaTypeDetails="Flute"/>
       <!-- BackLiner -->
       <Media DescriptiveName="180gsm white top" MediaType="Paper"
              Weight="180"/>
    </MediaLayers>
</Media>
```

8.92.3.3 Self adhesive Media

Self adhesive media is described as ***MediaLayers*** elements with nested ***Media*** and ***GlueLine*** elements.

Example 8.33: Media: Self Adhesive

```
<Media Class="Consumable" ID="M123456" ProductID="7890123" Status="Available"
       DescriptiveName="40# Fasson coated label stock" Dimension="1134.0 0"
       MediaType="SelfAdhesive" MediaUnit="Roll" Thickness="1000.0"
       Weight="150">
   <MediaLayers>
      <!-- Front -->
      <Media DescriptiveName="Antique Cream Smooth WS IL" MediaType="Paper"
             Weight="90"/>
      <!-- Glue -->
      <GlueLine DescriptiveName="Permanent 91A" AreaGlue="true"
             GlueType="Hotmelt" GlueBrand="Uhu"/>
      <!-- Back -->
      <Media DescriptiveName="Blue Glassine 50" MediaType="Paper" Weight="50"/>
   </MediaLayers>
</Media>
```

8.92.3.4 Flexo Plate Media

A sample of a flexo plate with dimensions of 900 mm x 1200 mm, a base of 177 microns and a total thickness of 1143 microns.

A raw plate can contain several separations from multiple jobs. The real printing dimensions can only be determined when all elements of the mounting process are known: circumference of the sleeve on which the flat plate will be mounted, thickness of the mounting tape, thickness of base and thickness of the photopolymer.

Example 8.34: Media: Flexo Plate

```
<Media Class="Consumable" ID="M123456" ProductID="FlexoPlate"
       Status="Available" DescriptiveName="" Dimension="2551.181 3401.574"
       MediaType="Plate" PlateTechnology="FlexoDigitalThermal"
       Manufacturer="PlateManufactererA" Brand="BrandB" BatchID="Batch 12345"
       Thickness="1143" ReliefThickness="500">
   <!--MediaLayers contains 2 items: the base layer and the
         photopolymer layer of the flexo plate -->
   <MediaLayers>
      <Media DescriptiveName="Base" MediaType="Plate"
             MediaTypeDetails="FlexoPlateBase" Thickness="177"/>
      <Media DescriptiveName="Photopolymer Layer" MediaType="Plate"
             MediaTypeDetails="FlexoPlatePhotopolymer" Thickness="966"/>
   </MediaLayers>
</Media>
```

8.92.3.5 Flexo Sleeve Media

The flexo sleeve has dimensions of 500 x 250 mm, a base of 1249 microns and a total thickness of 2810 microns. The sleeve dimensions are identical to printing dimensions (no distortion).

Example 8.35: Media: Flexo Sleeve

```
<Media Class="Consumable" ID="M123456" ProductID="FlexoSleeve"
       Status="Available"
       DescriptiveName="Sleeve" Dimension="1417.32 750.0" MediaType="Sleeve"
       PlateTechnology="FlexoDigitalSolvent" Manufacturer="PlateManufactererB"
       Brand="BrandB" BatchID="Batch 6789" Thickness="2810"
       ReliefThickness="500">
    <!--MediaLayers contains 2 items: the base layer and the
          photopolymer layer of the flexo plate -->
    <MediaLayers>
       <Media DescriptiveName="Base" MediaType="Plate"
              MediaTypeDetails="FlexoPlateBase" Thickness="1249"/>
       <Media DescriptiveName="Photopolymer Layer" MediaType="Plate"
              MediaTypeDetails="FlexoPhotopolymer" Thickness="1570"/>
    </MediaLayers>
</Media>
```

8.93 MediaSource

Deprecated in JDF 1.1

See ▸ Section N.7.14 MediaSource for details of this deprecated resource.

8.94 MiscConsumable

New in JDF 1.3

The ***MiscConsumable*** resource is intended for cost accounting, inventory control and availability scheduling of supplies used in the production workflow where a more detailed parameterization of the resource is not necessary. ***MiscConsumable*** is limited to modeling consumables not already more specifically defined in **JDF** (***Ink***, ***Media***, ***Pallet***, ***RegisterRibbon***, ***Strap*** or ***UsageCounter***).

MiscConsumable resources MAY appear as inputs to any **JDF** process. The default unit for amounts of ***MiscConsumable*** is Countable Objects.

Certain types of ***MiscConsumable*** elements such as ***MiscConsumable***[*@ConsumableType* = "WasteContainer"] are typically "consumed" by being filled. The sense of the *@Amount* attribute for such resources shall be the quantity of unused or empty waste containers that are available. If *@Unit* is a volume, distance or weight instead of Countable Objects, such *@Amount* will still represent the remaining unused capacity of the waste container.

Resource Properties

Resource Class: Consumable

Input of Processes: **Any Process**

Table 8.177: MiscConsumable Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Color ? New in JDF 1.6	enumeration	*@Color* specifies the machine readable color of the consumable. **Allowed value is from:** ▸ NamedColor**.**
ColorDetails ? New in JDF 1.6	string	*@ColorDetails* specifies additional details of the color of the consumable which MAY be site specific and MAY be human readable. *@Color* SHOULD be specified if *@ColorDetails* is supplied.

Table 8.177: MiscConsumable Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ConsumableType ? Modified in JDF 1.5	NMTOKEN	Identifies the type of ***MiscConsumable*** (machine-readable). A human-readable (possibly localized) description of the consumable SHOULD also be supplied in @*DescriptiveName*. Additional machine readable details MAY be provided in @*TypeDetails*. **Values include:** BackReinforcement – Strip of material that is used to reinforce the spines of a hard cover book. New in JDF 1.6 Developer – Chemicals used in filmsetters and platesetters. Electricity – Electrical energy. Typically monitored for CO_2 tracking. Measured in kWh. New in JDF 1.5 FuserOil – Silicon Oil Gas – Natural Gas. Typically monitored for CO_2 tracking. Measured in m^3. New in JDF 1.5 Glue Grommet – Specifies an eyelet-like shape placed in a hole.New in JDF 1.6 Hardener – Glue hardener used to two part glues. New in JDF 1.6 Headband – Head band for a hard cover book. New in JDF 1.4 RegistrationRibbon New in JDF 1.5 PaperBand Paperclips PaperWrap PlasticBand RubberBand ShrinkWrap Staples Strap - Straps are used in a **Strapping** process. New in JDF 1.6 Thread – Thread used for **ThreadSealing** or **ThreadSewing**. New in JDF 1.6 WasteContainer – Waste toner bottle. Wire – Bulk wire used for forming staples or other binding.
TypeDetails ? New in JDF 1.6	NMTOKEN	Additional details of the consumable such as material.
IdentificationField * New in JDF 1.6	element	***IdentificationField*** associates bar codes or labels with this ***MiscConsumable***.

8.95 NodeInfo

The ***NodeInfo*** resource contains information about planned scheduling and message routing. It allows MIS to plan, schedule and invoice jobs or job parts. Prior to **JDF** 1.3, ***NodeInfo*** was a direct subelement of the **JDF** node and not a resource.

Modification note: Starting with **JDF** 1.3, ***NodeInfo*** is a resource that SHALL be linked (via ***NodeInfoLink***) like any other resource; there is no "inheritance". However, a node MAY link to the same ***NodeInfo*** resource as its parent

Note: The NORMATIVE ***NodeInfo*** is specified by a linked resource. An Informative ***NodeInfo*** MAY be retrieved by searching the ***NodeInfo*** of parent nodes or ***Ancestor*** elements.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***Ancestor***

Input of Processes: **Any Process**

Table 8.178: NodeInfo Resource (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
CleanupDuration ?	duration	Estimated duration of the clean-up phase of the process.

Table 8.178: NodeInfo Resource (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
DueLevel ?	enumeration	Description of the severity of a missed deadline. **Allowed values are:** Unknown – Consequences of missing the deadline are not known. Deprecated in JDF 1.2 Trivial – Missing the deadline has minor or no consequences. Penalty – Missing the deadline incurs a penalty. JobCancelled – The Job is cancelled if the deadline is missed.
End ?	dateTime	Date and time at which the process is scheduled to end.
FirstEnd ?	dateTime	Earliest date and time at which the process SHALL end.
FirstStart ?	dateTime	Earliest date and time at which the process SHALL begin.
IPPVersion ? New in JDF 1.1	XYPair	A pair of numbers (as integers) indicating the version of the IPP protocol to use when communicating to IPP Devices. The X value is the major version number.
JobPriority = "50" New in JDF 1.1	integer	The scheduling priority for the node where 100 is the highest and 0 is the lowest. Amongst the nodes that can be processed in the **JDF** Instance, all higher priority nodes are to be processed before any lower priority ones. If one or more of the deadline oriented attributes (e.g., @*FirstStart* or @*LastEnd*) is specified, such attribute(s) SHALL be honored before considering @*JobPriority*. The priority from **JDF** (***QueueSubmissionParams***/@*Priority* or ***QueueEntryPriParams***/@*Priority*) takes precedence over ***NodeInfo***/@*JobPriority*. **Modification note:** Starting with **JDF** 1.4, scheduling priority in the first paragraph is described in terms of the node rather than the job
LastEnd ?	dateTime	Latest date and time at which the process SHALL end. This is the deadline to which @*DueLevel* refers.
LastStart ?	dateTime	Latest date and time at which the process SHALL begin.
MergeTarget ? Deprecated in JDF 1.1	boolean	If @*MergeTarget* = "true" and this node has been spawned, it SHALL be merged with its direct ancestor by the controller that executes this node. The path of the ancestor is specified in the last ***Ancestor*** element located in the ***AncestorPool*** of this node. It is an error to specify both @*MergeTarget* and @*TargetRoute* in one node. **Note:** @*MergeTarget* has been deprecated in **JDF** 1.1 because avoiding concurrent access to the ancestor node is ill defined and cannot be implemented in an open system without proprietary locking mechanisms.
NaturalLang ? New in JDF 1.6	language	Language selected for human readable communication. If not specified, the operating systems language SHALL be used.
NodeStatus ? New in JDF 1.3	enumeration	Identifies the status of an individual part of the node. **Default value is from:** ***JDF***/@*Status*. **Allowed values are from:** ▸ Table 8.179 NodeStatus Attribute Values**.**
NodeStatusDetails ? New in JDF 1.3	string	Description of the status that provides details beyond the enumerative values given by @*NodeStatus*. **Default value is from:** ***JDF***/@*StatusDetails*. **Values include those from:** ▸ Section C.11 StatusDetails Supported Strings**.**
Route ?	URL	The URL of the controller or device that SHALL execute this node. If @*Route* is not specified, the routing controller SHALL determine a potential target controller or device independently. For details, see ▸ Section 4.2 Process Routing. Note that the receiving device SHALL NOT use @*Route* to determine whether to execute the node. Rather a device SHALL use a ***Media*** Input resource (if specified) to determine whether to execute the node.

Table 8.178: NodeInfo Resource (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
rRefs ? Deprecated in JDF 1.2	IDREFS	Array of IDs of any elements that are specified as ***ResourceRef*** elements. In version 1.1, @*rRefs* contained the IDREF of an ***Employee***. In **JDF** 1.2 and beyond, it is up to the implementation to maintain references.
SetupDuration ?	duration	Estimated duration of the setup phase of the process.
Start ?	dateTime	Date and time of the planned process start.
TargetRoute ?	URL	The URL where the **JDF** SHALL be sent after completion. If @*TargetRoute* is not specified, it defaults to the input @*Route* attribute of the subsequent node in the process chain. If this is also not known (e.g., because the node is spawned), the **JDF** node SHALL be sent to the processor default output URL. If @*TargetRoute* specifies a file-schemed URL, it SHALL be the exact file name and NOT just the directory of the resulting **JDF**. ***JMF***/***QueueSubmissionParams***/@*ReturnURL* takes precedence over ***NodeInfo***/@*TargetRoute* of the **JDF** that is processed.
TotalDuration ?	duration	Estimated total duration of the process, including setup and cleanup.
WorkStepID ? New in JDF 1.4	string	ID of an individual work step (e.g., a Press Run). If ***NodeInfo*** is not Partitioned, or all Partitions are executed simultaneously, @*WorkStepID* corresponds to @*JobPartID*.
BusinessInfo ? Deprecated in JDF 1.6	element	Container for business related information. It is expected that **JDF** will be utilized in conjunction with other e-commerce standards, and this container is provided to store the e-commerce information within **JDF** in case a workflow with **JDF** as the root level document is desired. When **JDF** is used as part of an e-commerce solution such as PrintTalk, the information given in the envelope document overrides the information in ***BusinessInfo***.
Employee ?	refelement	The internal administrator or supervisor that is responsible for the product or process defined in this node.
GangSource * New in JDF 1.6	element	If present, each ***GangSource*** SHALL represent the source jobs that are being processed as a gang job.
JMF * Deprecated in JDF 1.5	element	Represents **JMF** Query Messages that set up a persistent channel, as described in ▸ Section 5.3.4 Persistent Channels. These Message elements define the receiver that is designated to track Jobs via **JMF** Messages. These Message elements SHOULD be honored by any **JMF**-capable controller or device that executes this node. When these Messages are honored, a persistent communication channel is established that allows Devices to transmit (e.g., the status of the Job as **JMF** ***Signal*** Messages). The **JMF** specified in this ***NodeInfo*** SHALL be restricted in scope to the containing **JDF** element. Typically this will be achieved by explicitly stating @*JobID* in the appropriate ***QueryTypeObj***. **Deprecation note:** Starting with **JDF** 1.5, subscriptions SHOULD only be specified as root **JMF**.
MISDetails? New in JDF 1.2	element	Definition how the costs for the execution of this node are to be charged.
NotificationFilter * Deprecated in JDF 1.6	element	Defines the set of ***NotificationFilter*** elements that are to be logged in the ***AuditPool***. This provides a logging method for Devices that do no not support **JMF** messaging. For details of the ***NotificationFilter*** element, see ▸ Section 5.34.1 NotificationFilter.

8.95.1 NodeStatus Attribute Values

*Table 8.179: **NodeStatus** Attribute Values*

VALUE	DESCRIPTION
Aborted	Indicates that the process executing the node has been aborted, which means that execution will not be resumed again. For ***QueueEntry***.New in JDF 1.2
Cleanup	The process represented by this node is currently being cleaned up.
Completed	Indicates that the node or queue entry has been executed correctly, and is finished. For ***QueueEntry***.New in JDF 1.2
FailedTestRun	An error occurred during the test run. Error information is logged in the ***Notification*** element, which is an OPTIONAL Subelement of the ***AuditPool*** element described in ▸ Section 3.11 AuditPool and Audit.
InProgress	The node is currently executing.
Part New in JDF 1.3	Indicates that the node is processing Partitioned resources and that the Status varies depending on the Partition Keys. Details are provided in the ***NodeInfo*** resource of the node.
Pool Deprecated in JDF 1.3	Indicates that the node processes Partitioned resources and that the @*Status* varies depending on the Partition Keys. Details are provided in the ***StatusPool*** element of the node.
Ready	As indicated by the successful completion of a test run; all ***ResourceLink*** elements are correct; REQUIRED resources are available, and the parameters of resources are valid. The node is ready to start.
Setup	The process represented by this node is currently being set up.
Spawned	The node is spawned in the form of a separate spawned **JDF**. The status Spawned can only be assigned to the original instance of the spawned **JDF**. For details, see ▸ Section 4.4 Spawning and Merging.
Stopped	Execution has been stopped. If a Job is "Stopped", running can be resumed later. This status can indicate a break, a pause, maintenance or a breakdown — in short, any pause that does not lead the Job to be aborted.
Suspended New in JDF 1.3 Modified in JDF 1.4	Execution has been stopped. If a Job is "Suspended", running will be resumed later. Unlike "Stopped" this status indicates that the Job has been taken off the device to execute another Job or perform some other action that is not related to this Job. When resumed, the Job MAY go into @*Status* = "Setup" before changing to "InProgress" again. The value "Suspended" is also used to describe iterations. In an iterative environment, "Suspended" specifies that at least one iteration cycle has completed but additional iteration cycles MAY still occur. In this use case, @*StatusDetails* SHOULD be set to "IterationPaused"
TestRunInProgress	The node is currently executing a test run.
Waiting	The node can be executed, but it has not completed a test run.

8.96 NumberingParams

Deprecated in JDF 1.5

See ▸ Section N.7.15 NumberingParams for details of this deprecated resource.

8.97 OrderingParams

Deprecated in JDF 1.5

See ▸ Section N.7.17 OrderingParams for details of this deprecated resource.

8.98 PackingParams

Deprecated in JDF 1.1

The PackingParams resource has been deprecated in **JDF** 1.1 and beyond. It is replaced by the individual resources used by the processes defined in ▸ Section 6.6.5 Packaging Processes. See ▸ Section N.7.18 PackingParams for details of this deprecated resource.

8.99 PageAssignParams

New in JDF 1.4

This resource is an empty container for future extensions

Resource Properties

Resource Class: Parameter

Input of Processes: **PageAssigning**

Table 8.180: PageAssignParams Resource

NAME	DATA TYPE	DESCRIPTION

8.100 PageList

New in JDF 1.2

PageList defines the additional metadata of individual finished pages such as pagination details. ***PageList*** references the finished page regardless of the page's position in a PDL file or ***RunList***.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***Assembly***, ***Component***, ***ExposedMedia***, ***LayoutElement***, ***RunList***

Example Partition: "PartVersion"

Table 8.181: PageList Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AssemblyID ? Deprecated in JDF 1.3	string	ID of the ***Assembly*** or ***AssemblySection*** that this finished page belongs to.
AssemblyIDs ? New in JDF 1.3	NMTOKENS	IDs of the ***Assembly*** elements, ***AssemblySection*** elements or ***StrippingParams***[@*BinderySignatureName*] that the finished pages specified by this ***PageList*** belong to.
HasBleeds ?	boolean	If "true", the file has bleeds.
IsBlank ?	boolean	If "true", the ***PageList*** has no content marks and is blank. Note that in **JDF** 1.2, the description erroneously stated that @*IsBlank* = "false" specifies a blank page.
IsPrintable ?	boolean	If "true", the file is a PDL file and can be printed. Possible files types include PCL, PDF or PostScript files. Application files such as MS Word have @*IsPrintable* = "false".
IsTrapped ?	boolean	If "true", the file has been trapped.
JobID ?	string	ID of the Job that this finished page belongs to.
PageLabelPrefix ?	string	Prefix of the identification of the Reader Page as it is displayed on the finished page. For instance "C-", if the Reader Pages are labeled "C-1", "C-2", etc.
PageLabelSuffix ?	string	Suffix of the identification of the Reader Page as it is displayed on the finished page. For instance "-a", if the pages are labeled "C-1-a", "C-2-a", etc.
SourceBleedBox ?	rectangle	A rectangle that describes the bleed area of the page to be included. This rectangle is expressed in the source coordinate system of the object. If not specified, use defined bleed box of element (or no bleed box if element does not supply a bleed box).
SourceClipBox ?	rectangle	A rectangle that defines the region of the finished page to be included. This rectangle is expressed in the source coordinate system of the object. If not specified, use defined clip box of element (or no clip box if element does not supply a clip box).

Table 8.181: PageList Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
SourceTrimBox ?	rectangle	A rectangle that describes the intended trimmed size of the finished page to be included. This rectangle is expressed in the source coordinate system of the object. If not specified, use defined trim box of element (or no trim box if element does not supply a trim box).
Template = "false"	boolean	Template is "false" when this page is self-contained. This attribute is "true" if the ***PageList*** represents a template that SHALL be completed with information from a database.
Assembly ? New in JDF 1.3	refelement	***Assembly*** that is referred to by @*AssemblyIDs* or contains the ***AssemblySection*** that is referred to by @*AssemblyIDs*.
ColorPool ?	refelement	Definition of the color details.
ContentList ? New in JDF 1.3	refelement	List of ***ContentData*** elements that describe individual pieces of content on the pages.
ElementColorParams ?	refelement	Color details of the page list.
ImageCompressionParams ?	refelement	Specification of the image compression properties.
PageData *	element	Details of the individual finished page. The ***PageData*** elements are referred to by the values of ***PageData***/@*PageIndex* (if present), or otherwise, their index in the ***PageList***. In the latter case, the ***PageData*** elements SHOULD, therefore, not be removed or inserted in a position other than the end of the list. **Modification note:** See the Modification note in the section on the ***PageData*** element below.
ScreeningParams ?	refelement	Specification of the screening properties.
SeparationSpec *	element	List of separation names defined in the ***PageList***.

8.100.1 PageData

PageData defines the additional metadata of individual finished pages or sets of finished pages with common properties, such as pagination details.

If @*PageIndex* is not present in ***PageData*** elements, ***PageData*** elements are referred to by index of the ***PageData*** in the ***PageList***. If @*PageIndex* is present, it explicitly specifies the indices within the ***PageList***. Either all or no ***PageData*** elements in a ***PageList*** SHALL have @*PageIndex*. If a Page is not represented by a ***PageData***, the attributes of the ***PageList*** itself apply.

Modification note: Starting with **JDF** 1.4, ***PageData***/@*PageIndex* is added It allows ***PageData*** to describe multiple finished pages and to explicitly specify the index of a ***PageData*** element within a ***PageList***. The explicit index allows a ***PageList*** to contain a ***PageData*** for a particular index (e.g., 100) without the need for PageData elements for all indices that are lower (e.g., 0 to 99). Without @*PageIndex*, the position of ***PageData*** within ***PageList*** implicitly specifies its index.

If the ***PageList*** is partitioned, the index refers to ***PageData*** elements in the respective leaves of the partitioned ***PageList***. The index restarts at 0 with each partitioned leaf.

*Table 8.182: **PageData** Element (Sheet 1 of 3)*

NAME	DATA TYPE	DESCRIPTION
AssemblyID ? Deprecated in JDF 1.3	string	ID of the ***Assembly*** or ***AssemblySection*** that this finished page belongs to. **Default value is from:** ***PageList***/@*AssemblyID*.
AssemblyIDs ? New in JDF 1.3	NMTOKENS	IDs of the ***Assembly*** elements, ***AssemblySection*** elements or ***StrippingParams***[@*BinderySignatureName*] that this finished page belongs to. **Default value is from:** ***PageList***/@*AssemblyIDs*.
CatalogID ?	string	Identification of the resource (e.g., in a catalog environment). **Default value is from:** ***PageList***/@*CatalogID*.

*Table 8.182: **PageData** Element (Sheet 2 of 3)*

NAME	DATA TYPE	DESCRIPTION
CatalogDetails ?	string	Additional details of a resource in a catalog environment. **Default value is from:** ***PageList***/@*CatalogDetails*.
FoldOutPages ?	IntegerList	Page indices in the ***PageList*** of the file pages forming a content page that flows over multiple finished pages (e.g., foldout, centerfold). The list does not include the index of this ***PageData***. **Default behavior:** ***PageData*** does not describe a part of a foldout.
HasBleeds ?	boolean	If "true", the file has bleeds. **Default value is from:** ***PageList***/@*HasBleeds*.
IsBlank ?	boolean	If "true", the ***PageData*** has no content marks and is blank. Note that in **JDF** 1.2 the description erroneously stated that @*IsBlank* = "false" specifies a blank page. **Default value is from:** ***PageList***/@*IsBlank*.
IsPrintable ?	boolean	If "true", the file is a PDL file and can be printed. Possible files types include PCL, PDF or PostScript files. Application files such as MS Word have @*IsPrintable* = "false". **Default value is from:** ***PageList***/@*IsPrintable*.
IsTrapped ?	boolean	If "true", the file has been trapped. **Default value is from:** ***PageList***/@*IsTrapped*.
JobID ?	string	ID of the Job that this finished page belongs to. **Default value is from:** ***PageList***/@*JobID*.
PageFormat ? New in JDF 1.3	NMTOKEN	Defines the format of the page in a production workflow. **Values include:** Broadsheet – One single page that will be mounted on a broadsheet plate (one page goes on one (broadsheet) plate). Tabloid – One single page that will be paired with a second tabloid page. Later, the page pair will be mounted on a broadsheet plate. Newspaper4up – Four pages will be mounted on one plate. Newspaper8up – Eight pages will be mounted on one plate. **Note:** The values are for a newspaper workflow.
PageIndex ? New in JDF 1.4	IntegerRangeList	List of pages the ***PageData*** element represents. A Page number SHALL NOT appear more than once in the ***PageList***.
PageLabel ?	string	Complete identification of the finished page including @*PageLabelPrefix* and @*PageLabelSuffix* as it is displayed on the finished page, For instance "1", "iv" or "C-1". Note that this might be different from the position of the page in the finished document.
PageLabelPrefix ?	string	Prefix of the identification of the Reader Page as it is displayed on the finished page. For instance "C-", if the Reader Pages are labeled "C-1", "C-2", etc. **Default value is from:** ***PageList***/@*PageLabelPrefix*.
PageLabelSuffix ?	string	Suffix of the identification of the Reader Page as it is displayed on the finished page. For instance "-a", if the pages are labeled "C-1-a", "C-2-a", etc. **Default value is from:** ***PageList***/@*PageLabelSuffix*.
PageStatus ? New in JDF 1.3	NMTOKENS	Status of a single ***PageData*** element. **Values include those from:** ▸ Table C.12 MessageEvents ContentStatus and MilestoneType Values**.**
ProductID ?	string	An ID of the page as defined in the MIS system. **Default value is from:** ***PageList***/@*ProductID*.
SourceBleedBox ?	rectangle	A rectangle that describes the bleed area of the page to be included. This rectangle is expressed in the source coordinate system of the object. **Default value is from:** ***PageList***/@*SourceBleedBox*.

Table 8.182: PageData Element (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
SourceClipBox ?	rectangle	A rectangle that defines the region of the finished page to be included. This rectangle is expressed in the source coordinate system of the object. **Default value is from:** ***PageList***/@*SourceClipBox*.
SourceTrimBox ?	rectangle	A rectangle that describes the intended trimmed size of the finished page to be included. This rectangle is expressed in the source coordinate system of the object. **Default value is from:** ***PageList***/@*SourceTrimBox*.
Template ?	boolean	Template is "false" when this page is self-contained. This attribute is "true" if the ***PageList*** represents a template that SHALL be completed with information from a database. **Default value is from:** ***PageList***/@*Template*.
ElementColorParams ?	refelement	Color details of the ***PageData*** element. **Default value is from:** ***PageList***/***ElementColorParams***
ImageCompressionParams ?	refelement	Specification of the image compression properties. **Default value is from:** ***PageList***/***ImageCompressionParams***
PageElement * New in JDF 1.3	element	Describes an individual element on a page. This might be a part of an image, text, advertisement, editorial, etc.
ScreeningParams ?	refelement	Specification of the screening properties. **Default value is from:** ***PageList***/***ScreeningParams***
SeparationSpec *	element	List of separation names defined in the element. **Default value is from:** ***PageList***/***SeparationSpec***

8.100.2 PageElement

New in JDF 1.3

PageElement defines the positioning of ***ContentData*** on a Page or ***PageElement*** and additional metadata of individual elements within a page.

Table 8.183: PageElement Element

NAME	DATA TYPE	DESCRIPTION
ContentDataRefs ? New in JDF 1.4	IDREFS	***ContentData*** provides metadata of the element that is independent of the page position. ID of the ***ContentData*** elements in the referenced ***ContentList***. ***ContentData*** elements provide Metadata related to this ***PageData***. @*ContentDataRefs* SHALL NOT be specified if no ***ContentData*** is specified in the grand-parent ***PageList*** element.
ContentListIndex ? Deprecated in JDF 1.4	integer	Index into a ***ContentList***/***ContentData*** element. If neither @*ContentListIndex* nor ***PageElement*** are specified, this ***PageElement*** is a reservation. **Deprecation note:** Starting with **JDF** 1.4, use @*ContentDataRefs*.
ContentType ?	NMTOKEN	Type of content that is placed in this ***PageElement***. **Values include those from:** ***ContentList*****v/*****ContentData*****/@***ContentType***.**
ElementPages ?	IntegerRangeList	List of Pages that this ***PageElement*** traverses. Multiple values designate, e.g., fold out pages or multi-page ads.
RelativeBox ?	Rectangle	Position of the ***PageElement*** in the coordinate system of the parent ***PageData***.
PageElement *	element	Further sub-page elements that comprise this ***PageElement***.

8.101 Pallet

New in JDF 1.1

A ***Pallet*** represents the pallet used in packing goods.

Resource Properties

Resource Class: Consumable

Input of Processes: **Palletizing**

Table 8.184: Pallet Resource

NAME	DATA TYPE	DESCRIPTION
PalletType Modified in JDF 1.4	NMTOKEN	Type of pallet used. **Values include:** 2Way – Two-way entry. 4Way – Four-way entry. Euro – Standard 1*1 m Euro pallet.Deprecated in JDF 1.4 Euro800x600 – 800x600mm according to 15146-4 (equals half Euro pallet). New in JDF 1.4 Euro800x1200 – 800x1200mm according to DIN EN 13698-1 (equals Euro pallet). New in JDF 1.4 Euro1000x1200 – 1000x1200mm according to DIN EN 13698-2 (flat pallet).New in JDF 1.4 Euro1200x1200 – 1200x1200mm no norm, but in use in the field.New in JDF 1.4
Size ?	XYPair	Describes the length and width of the pallet, in points (e.g., 3500 3500). If not specified, the size is defined by @*PalletType*.

8.102 PalletizingParams

New in JDF 1.1

PalletizingParams defines the details of **Palletizing**. Details of the actual pallet used for **Palletizing** can be found in the ***Pallet*** resource that is also an input of the **Palletizing** process.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***PackingIntent***

Input of Processes: **Palletizing**

Table 8.185: PalletizingParams Resource

NAME	DATA TYPE	DESCRIPTION
LayerAmount ? New in JDF 1.4	IntegerList	Ordered number of input components in a layer. The first number is the first layer on the bottom. If there are more layers than entries in the list, counting restarts at the first entry.
MaxHeight ?	double	Maximum height of a loaded pallet in points.
MaxWeight ?	double	Maximum weight of a loaded pallet in grams.
Overhang ? New in JDF 1.4	XYPair	Overhang in *x* and *y* direction on each side.
OverhangOffset ? New in JDF 1.4	XYPair	Overhang offset if overhang is not centered.
Pattern ?	string	Name of the palletizing pattern. Used to store a predefined pattern that defines the layers and positioning of individual component on the pallet.
Bundle ? New in JDF 1.4	refelement	Describes additional properties, such as the number of individual products, and describes the list of the individual products on the pallet.

8.103 PDFToPSConversionParams

This resource specifies a set of configurable options that can be used by processes that read PDF and generate PostScript files. It is RECOMMENDED to describe reading of arbitrary PDL documents as a combination of the **Interpreting** and **PDL-Creation** processes.

Some descriptions below mention attributes or structures in specific source formats, such as PDF. Appropriate equivalent actions should be taken when converting from other source formats that have equivalent attributes or structures. A small number of parameters apply only to PDF sources.

Font controls are applied in the following order:

1 *@IncludeBaseFonts*
2 *@IncludeEmbeddedFonts*
3 *@IncludeType1Fonts*
4 *@IncludeType3Fonts*
5 *@IncludeTrueTypeFonts*
6 *@IncludeCIDFonts*

For example, an embedded Type-1 font follows the rule for embedded fonts, not the rule for Type-1 fonts. In other words, if *@IncludeEmbeddedFonts* is "true", and *@IncludeType1Fonts* is "false", embedded Type-1 fonts would be included in the PostScript stream.

Resource Properties

Resource Class: Parameter

Resources referenced: ***PDLCreationParams***

Example Partition: "DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetName", "Side", "SignatureName"

Input of Processes: **PDFToPSConversion**

Table 8.186: PDFToPSConversionParams Resource (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
BinaryOK = "true"	boolean	If "true", binary data are to be included in the PostScript stream.
BoundingBox ?	rectangle	It is used for **BoundingBox** DSC comment in *@CenterCropBox* calculations and for PostScript's **setpage**Device.
CenterCropBox = "true"	boolean	If "true", the **CropBox** from the source document is centered on the page when the **CropBox** is smaller than **MediaBox**.
GeneratePageStreams = "false"	boolean	If "true", the process emits individual streams of data for each page in the ***RunList***.
IgnoreAnnotForms = "false"	boolean	If "true", ignores annotations that contain a PDF **XObject** form. (PDF source only).
IgnoreBG = "true" New in JDF 1.1	boolean	Ignores the **BG**, **BG2** parameters in the PDF **ExtGState** dictionary, and the operand of any calls to the PostScript **setblackgeneration** operator.
IgnoreColorSeps = "false"	boolean	If "true", ignores images for Level-1 separations.
IgnoreDeviceExtGState ? Deprecated in JDF 1.1	boolean	If "true", ignores all device-dependent extended graphic state parameters. This overrides *@IgnoreHalftones*. The following parameters are to be ignored: OP – Overprint parameter. OPM – Overprint mode. BG, BG2 – Black generation. UCR, UCR2 – Undercolor removal. TR, TR2 – Transfer functions. HT – Halftone dictionary. FL – Flatness tolerance. SA – Automatic stroke adjustment.
IgnoreDSC = "true"	boolean	If "true", ignores DSC (Document Structuring Conventions).
IgnoreExternStreamRef = "false"	boolean	If a PDF image resource uses an external stream and *@IgnoreExternStreamRef* = "true", ignores code that points to the external file. (PDF source only). Note: *@IgnoreExternStreamRef* was misspelled as *@IgnoreExternSreamRef* prior to **JDF** 1.3.
IgnoreHalftones = "false"	boolean	If "true", ignores any halftone screening in the source file.

Table 8.186: PDFToPSConversionParams Resource (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
IgnoreOverprint = "true" New in JDF 1.1	boolean	Ignores **OP** parameters in a source PDF **ExtGState** dictionary **setoverprint** in a source PostScript file, etc.
IgnorePageRotation = "false"	boolean	If "true", ignores a "concatenation" provided at the beginning of each page that orients the page so that it is properly rotated. Used when emitting EPS.
IgnoreRawData = "false"	boolean	If "true", no unnecessary filters are to be added when emitting image data.
IgnoreSeparableImagesOnly = "false"	boolean	If "true", and if emitting EPS, ignores only CMYK and gray images.
IgnoreShowPage = "false"	boolean	If "true", ignores save-and-restore **showpage** in PostScript files
IgnoreTransfers = "true" New in JDF 1.1	boolean	Ignores **TR**, **TR2** parameters in a source PDF **ExtGState** dictionary, **settransfer** and **setcolortransfer** in a source PostScript file, etc.
IgnoreTTFontsFirst = "false"	boolean	If "true", ignores TrueType fonts before any other fonts.
IgnoreUCR = "true" New in JDF 1.1	boolean	Ignores **UCR**, **UCR2** parameters in a source PDF **ExtGState** dictionary, **setundercolorremoval** in a source PostScript file, etc.
IncludeBaseFonts = "IncludeNever"	enumeration	Determines when to embed the base fonts. The base fonts are "Symbol" and the plain, bold, italic and bold-italic faces of "Courier", "Times", and "Helvetica". **Allowed value is from:** ▸ IncludeResources.
IncludeCIDFonts = "IncludeOncePerDoc"	enumeration	Determines when to embed CID fonts. **Allowed value is from:** ▸ IncludeResources.
IncludeEmbeddedFonts = "IncludeOncePerDoc"	enumeration	Determines when to embed fonts in the document that are embedded in the source file. This attribute overrides the @*IncludeType1Fonts*, @*IncludeTrueTypeFonts* and @*IncludeCIDFonts* attributes. **Allowed value is from:** ▸ IncludeResources.
IncludeOtherResources = "IncludeOncePerDoc"	enumeration	Determines when to include all other types of resources in the file. **Allowed value is from:** ▸ IncludeResources.
IncludeProcSets = "IncludeOncePerDoc"	enumeration	Determines when to include ProcSets in the file. **Allowed value is from:** ▸ IncludeResources.
IncludeTrueTypeFonts = "IncludeOncePerDoc"	enumeration	Determines when to embed TrueType fonts. **Allowed value is from:** ▸ IncludeResources.
IncludeType1Fonts = "IncludeOncePerDoc"	enumeration	Determines when to embed Type-1 fonts. **Allowed value is from:** ▸ IncludeResources.
IncludeType3Fonts = "IncludeOncePerPage"	enumeration	Determines when to embed Type-3 fonts. It is included here to complete the precedence hierarchy. It has only one value. **Allowed values are:** "IncludeOncePerPage"
OutputType = "PostScript"	enumeration	Describes the kind of output to be generated. **Allowed values are:** PostScript EPS
PSLevel = "2"	integer	Number that indicates the PostScript level. Values include "1", "2" or "3".

Table 8.186: PDFToPSConversionParams Resource (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
Scale = "100"	double	Number that indicates the wide-scale factor of documents. Full size = "100".
SetPageSize = "false"	boolean	(PostScript Levels 2 and 3 only) If "true", sets page size on each page automatically. For PDF source, use MediaBox for outputting PostScript files and CropBox for EPS.
SetupProcsets = "true"	boolean	If "true", indicates that if ProcSets are included, the init/term code is also included.
ShrinkToFit = "false"	boolean	If "true", the page is scaled to fit the printer page size. This field overrides scale
SuppressCenter = "false"	boolean	If "true", suppresses automatic centering of page contents whose crop box is smaller than the page size.
SuppressRotate = "false"	boolean	If "true", suppresses automatic rotation of pages when their dimensions are better suited to landscape orientation. More specifically, the application that generates the PostScript compares the dimensions of the page. If the width is greater than the height, then pages are not rotated if @*SuppressRotate* = "true". On the other hand, if @*SuppressRotate* = "false", the orientation of each source page (e.g., as set by the PDF **Rotate** key) is honored, regardless of the dimensions of the pages (as defined by the **MediaBox** attribute).
TTasT42 = "false"	boolean	If including TrueType fonts, converts to Type-42 instead of Type-1 fonts when @*TTasT42* = "true".
UseFontAliasNames = "false"	boolean	If "true", font alias names are used when printing with system fonts.

8.104 PDLCreationParams

New in JDF 1.3

This resource is used to encapsulate the PDL output parameters for the supported output PDL types used in the **PDLCreation** process.

Resource Properties

Resource Class: Parameter

Input of Processes: **PDLCreation**

Table 8.187: PDLCreationParams Resource

NAME	DATA TYPE	DESCRIPTION
MimeType	string	This resource identifies the MIME type associated with this output file format. For example "application/pdf".
FontParams ? New in JDF 1.6	refelement	*Fontparams* describes how fonts SHALL be handled when creating PDL.
PDFToPSConversionParams ?	refelement	Postscript specific ***Parameter Resource*** for the output. SHALL NOT be specified unless @*MimeType* = "application/postscript".
PSToPDFConversionParams ?	refelement	PDF specific ***Parameter Resource*** for the output. It SHALL NOT be specified unless @*MimeType* = "application/pdf".

8.105 PDLResourceAlias

This resource provides a mechanism for referencing resources that occur in files, or that are expected to be provided by Devices. Prepress and printing processes have traditionally used the word "Resource" to refer to reusable data structures that are needed to perform processes. Examples of such resources include fonts, halftones and functions. The formats of these Resources are defined within PDLs, and instances of these resources can occur within PDL files or can be provided by Devices.

JDF does not provide a syntax for defining such resources directly within a Job. Instead, resources continue to occur within PDL files and continue to be provided by Devices. However, since it is necessary to be able to refer to these resources from **JDF** Jobs, the ***PDLResourceAlias*** resource is provided to fulfill this need.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***ColorantControl***/***ColorSpaceSubstitute***

Input of Processes: **Interpreting**

Table 8.188: PDLResourceAlias Resource

NAME	DATA TYPE	DESCRIPTION
ResourceType	string	The type of PDL resource that is referenced. The semantic of this attribute is defined by the PDL.
SourceName ?	string	The name of the resource in the file referenced by the ***FileSpec*** or by the device.
FileSpec ?	refelement	Location of the file containing the PDL resource. If ***FileSpec*** is absent, the device is expected to provide the resource defined by this ***PDLResourceAlias*** resource.

8.106 PerforatingParams

New in JDF 1.1

PerforatingParams define the parameters for perforating a Sheet.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***FoldingIntent***

Input of Processes: **Perforating**

Table 8.189: PerforatingParams Resource

NAME	DATA TYPE	DESCRIPTION
Perforate *	element	Defines one or more ***Perforate*** lines.

8.107 PlaceHolderResource

Deprecated in JDF 1.5

See ▸ Section N.7.19 PlaceHolderResource for details of this deprecated resource.

8.108 PlasticCombBindingParams

This resource describes the details of the **PlasticCombBinding** process.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **PlasticCombBinding**

Table 8.190: PlasticCombBindingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Brand ?	string	The name of the comb manufacturer and the name of the specific item.
Color ?	NamedColor	Determines the color of the plastic comb.
ColorDetails ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If @*ColorDetails* is supplied, @*Color* SHOULD also be supplied.
Diameter ?	double	The comb diameter is determined by the height of the block of Sheets to be bound.
Thickness ?	double	The material thickness of the comb.

Table 8.190: PlasticCombBindingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Type ? Modified in JDF 1.1 Deprecated in JDF 1.2	enumeration	The distance between the "teeth" and the distance between the holes of the prepunched Sheets SHALL be the same. The following values from the hole type catalog in ▸ Appendix L Hole Pattern Catalog exist: **Allowed values are:** P12m-rect-02 – Distance = 12 mm; Holes = 7 mm x 3 mm P16-9i-rect-0t – Distance = 14.28 mm; Holes = 8 mm x 3 mm Euro – (Distance = 12 mm; Holes = 7 mm x 3 mm) Deprecated in JDF 1.1. USA1 – (Distance = 14.28 mm; Holes = 8 mm x 3 mm) Deprecated in JDF 1.1. **Deprecation note:** Starting with **JDF** 1.2, use the value implied by ***HoleMakingParams***/@*HoleType*.
HoleMakingParams ?	refelement	Details of the holes to be made. Note that ***HoleMakingParams***/@*Shape* is always rectangular by design of the plastic combs.

8.109 PlateCopyParams

Deprecated in JDF 1.1

See ▸ Section N.7.20 PlateCopyParams for details of this deprecated resource.

8.110 PreflightAnalysis

Deprecated in JDF 1.2

This resource was deprecated as a result of a major revision to the **Preflight** process and its associated resources. For details of this deprecated resource see ▸ Section N.5.7 PreflightAnalysis.

8.111 PreflightInventory

Deprecated in JDF 1.2

This resource was deprecated as a result of a major revision to the **Preflight** process and its associated resources. For details of this deprecated resource see ▸ Section N.5.8 PreflightInventory.

8.112 PreflightParams

New in JDF 1.2

The ***PreflightParams*** resource specifies the tests for the **Preflight** process to run. These tests are defined using ▸ Section 10.2.2 ActionPool, which defines a list of reporting actions to have for given document object tests defined into a ***Test***. (See ▸ Section 10.2.12 TestPool). This section makes use of elements and attributes defined in ▸ Section Device Capabilities. It is suggested that readers familiarize themselves with that section and ▸ Section 10.3 Concept of the Preflight Process.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***PreflightReport***

Input of Processes: **Preflight**

Table 8.191: PreflightParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ActionPool * Modified in JDF 1.4	element	A set of ***ActionPool*** elements. Multiple ***ActionPool*** elements are equivalent to one ***ActionPool*** that contains all ***Action*** elements of the individual ***ActionPool*** elements. ***ActionPool*** and ***TestPool*** SHALL both be supplied or both be absent. **Modification note:** Starting with **JDF** 1.4, ***ActionPool*** becomes optional.
FileSpec ? New in JDF 1.4	refelement	File that describes the preflight actions in a non-**JDF** format.

Table 8.191: PreflightParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
TestPool ? New in JDF 1.3 Modified in JDF 1.4	element	Container for zero or more ***Test*** elements that are referenced from ***Action*** elements in the ***ActionPool***. ***ActionPool*** and ***TestPool*** SHALL both be supplied or both be absent. **Modification note:** Starting with **JDF** 1.4, ***TestPool*** becomes optional. It was REQUIRED when it was added in **JDF** 1.3 because ***ActionPool*** implicitly requires a parallel ***TestPool*** as a container for the referenced ***Test*** elements that are defined in ***Action***/@*TestRef*.

The ***ActionPool***, as defined in ▸ Section 10.2.2 ActionPool, has ***Action*** Subelements, which can reference a ***Test*** with a given action type. The ***Action*** element includes a ***PreflightAction*** Subelement, defined below, which can be used to define how tests are to be applied in **Preflight** processes.

8.112.1 PreflightAction

Table 8.192: PreflightAction Element

NAME	DATA TYPE	DESCRIPTION
SetRef ?	IDREF	A reference to a preflight ***Test*** ID used to filter a set of objects before applying the tests referenced by preflight ***Action***. When @*SetRef* is not defined, the ***Test*** is applied to all the objects.
SetSplitBy = "RunList"	enumeration	This is used to group objects in different ways. **Allowed values are:** Page – Tests are applied on objects page per page. Document – Tests are applied on objects document per document. RunList – All objects of all pages included in all documents are processed together. **Note:** @*SetSplitBy* is only used when @*SetRef* is defined in order to create sets on a page-per-page or document-per-document basis. For instance, if you want to get the list of separations per page, @*SetSplitBy* is set to "Page". In such a case, the report's content (as long as the @*PRItem* is defined properly for the ***Action***) will be grouped by page.

Test elements make use of ***Evaluation*** Subelements that define various basic preflight testing functions that can be combined together in order to build preflight test. In order to specify basic preflight tests using ***Evaluation***, the Subelement ***BasicPreflightTest*** is used.
Note: The ***BasicPreflightTest*** includes a ***PreflightArgument*** subelement that is defined below.

8.112.2 BasicPreflightTest

The ***BasicPreflightTest*** element defines a named preflight test that can be evaluated by a preflight application. The result of the test can be compared with the values defined in the explicit ***Evaluation*** elements in order to filter the objects within the file to be tested. The following table describes the ***BasicPreflightTest*** element.

Table 8.193: BasicPreflightTest Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Classes ? New in JDF 1.4	NMTOKENS	List of object classes that the test SHALL be applied to. It is strongly recommended to supply @*Classes*.
ClassName ? New in JDF 1.4	NMTOKEN	This tag can be used to directly command the test to specifically apply on a given class of object. The two purposes of this change are 1) to simply preflight engine processors, and 2) to simplify Test rules. **Allowed values are from:** ▸ Table 10.70 Object Classes for a Document**.**
DevNS = "http://www.CIP4.org/JDFSchema_1_1"	URI	Namespace of the test that is described by @*Name* in this ***BasicPreflightTest*** element.

Table 8.193: BasicPreflightTest Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ListType = "Single-Value" Modified in JDF 1.4	enumeration	Specifies what type of list or object the basic preflight test describes. **Allowed values are from:** ***State*/@*ListType*** **(** ▸ Table 10.11 ListType Attribute Values**).** **Modification note:** Starting with **JDF** 1.4, @*ListType* has a specified default value.
MaxOccurs = "1"	integer	Maximum number of elements in the list described by this ***BasicPreflightTest*** (e.g., the maximum number of integers in an integer list). If @*MaxOccurs* is not "1", the ***BasicPreflightTest*** element refers to a list or RangeList of values (e.g., a ***NameEvaluation*** will allow a list of NMTOKENS).
MinOccurs = "1"	integer	Minimum number of elements in the list described by this ***BasicPreflightTest***. Default = "1" (i.e., it is an individual value). If MinOccurs is not "1", the ***BasicPreflightTest*** element refers to a list or RangeList of values (e.g., a ***NameEvaluation*** will allow a list of NMTOKENS).
Name Modified in JDF 1.4	NMTOKEN	Local name of the preflight constraint that is evaluated by this ***BasicPreflightTest***. A valid @*Name* value for the **JDF** namespace is any property name defined in any of the Properties tables in ▸ Section 10.3.2 Properties Preflight tests are defined through the use of constraints. **Modification note:** Starting with **JDF** 1.4, @*Name* is no longer optional.
PreflightArgument ?	element	Additional arguments for the preflight test. For details see ▸ Section 8.112 PreflightParams for the definition of ***PreflightArgument*** and constraints upon which preflight tests are defined.

8.112.3 PreflightArgument

This Subelement is used by ***BasicPreflightTest*** when additional data are needed to determine object property.

Table 8.194: PreflightArgument Element

NAME	DATA TYPE	DESCRIPTION
BoxArgument ?	element	Used if ***BasicPreflightTest***/@*Name* has a value of either "InsideBox" and "OutsideBox". Used for tests with the same two names.
BoxToBoxDifference ?	element	Used by the ***BoxToBoxDifference*** test.

8.112.4 BoxArgument

Table 8.195: BoxArgument Element

NAME	DATA TYPE	DESCRIPTION
Box	enumeration	The box type used to verify inclusion or exclusion. **Allowed values are from:** ▸ Table 8.196 Box Attribute Values**.**
MirrorMargins ?	enumeration	The @*MirrorMargins* attribute allows the flip of the @*Offset* value depending on the ***RunList*** index. When the index is even, the original @*Offset* value is preserved. When the index is odd, the @*Offset* value is flipped. **Default behavior:** the value of @*Offset* is not changed (if unspecified). **Allowed values are:** Vertical – turns [l b r t] into [r b l t]. Horizontal – turns [l b r t] into [l t r b].
Offset ?	rectangle	The offset to build real rectangle to which test is made.
Overlap = "false"	boolean	Explains if overlap is allowed to check inclusion or exclusion.

8.112.4.1 Box Attribute Values

Table 8.196: Box Attribute Values

BOX TYPE	DESCRIPTION
ArtBox	Defines the extent of the page's meaningful content (including potential white space) as intended by the page's creator.
BleedBox	Defines the region to which the contents of the page SHALL be clipped when output in a production environment. This might include any extra "bleed area" needed to accommodate the physical limitations of cutting, folding and trimming equipment. The actual printed page might include printing marks that fall outside the bleed box.
CropBox	Defines the region to which the contents of the page are to be clipped (cropped) when displayed or printed. Unlike the other boxes, the crop box has no defined meaning in terms of physical page geometry or intended use — it merely imposes clipping on the page contents. However, in the absence of additional information, the crop box will determine how the page's contents are to be positioned on the output medium.
MarginsBox	Defines the trim box minus the margins.
MediaBox	Defines the boundaries of the physical medium on which the page SHALL be printed. It might include any extended area surrounding the finished page for bleed, printing marks or other such purposes. It might also include areas close to the edges of the medium that cannot be marked because of physical limitations of the output device. Content falling outside this boundary can safely be discarded without affecting the meaning of the file.
SlugBox	Defines an area where document related information and objects that will not be on the final document could be printed.
TrimBox	Defines the intended dimensions of the finished page after trimming. It can be smaller than the media box, to allow for production-related content such as printing instructions, cut marks or color bars. In another type of document than PDF, this box represents the page size.

8.112.5 BoxToBoxDifference

Table 8.197: BoxToBoxDifference Element

NAME	DATA TYPE	DESCRIPTION
FromBox ?	enumeration	The "From" box used for ***BoxToBoxDifference*** calculation. **Allowed values are from: *BoxArgument*/@*Box*.**
ToBox ?	enumeration	The "To" box used for ***BoxToBoxDifference*** calculation. **Allowed values are from: *BoxArgument*/@*Box*.**

Example 8.36: Test with InsideBox and a BoxArgument Subelement

The following is an example of ***Test*** using @*InsideBox* and a ***BoxArgument*** Subelement:

```
<PreflightParams Class="Parameter" ID="PP001" Status="Available">
   <TestPool>
      <Test ID="PT01">
         <BooleanEvaluation ValueList="true">
            <BasicPreflightTest Name="InsideBox">
               <PreflightArgument>
                  <BoxArgument Box="TrimBox" Overlap="true"/>
               </PreflightArgument>
            </BasicPreflightTest>
         </BooleanEvaluation>
      </Test>
   </TestPool>
   <ActionPool/>
</PreflightParams>
```

8.113 PreflightProfile

Deprecated in JDF 1.2

This resource was deprecated as a result of a major revision to the **Preflight** process and its associated resources. For details of this deprecated resource see ▸ Section N.5.9 PreflightProfile.

8.114 PreflightReport

New in JDF 1.2

The ***PreflightReport*** resource describes the results of the preflight tests specified in ***PreflightParams***. This section makes use of elements and attributes defined in ▸ Section Device Capabilities. It is suggested that reader's familiarize themselves with that section and ▸ Section 10.3 Concept of the Preflight Process.

Resource Properties

Resource Class: Parameter

Input of Processes: **Any Process**

Output of Processes: **Preflight**

Table 8.198: PreflightReport Resource

NAME	DATA TYPE	DESCRIPTION
ErrorCount ? Modified in JDF 1.4	integer	The count of errors that were encountered while preflighting the Job. **Modification note:** Starting with **JDF** 1.4, @*ErrorCount* becomes optional.
ErrorState ?	enumerations	Describes the type of errors that occurred during preflighting when the **Preflight** process does not understand certain preflight tests or cannot apply them to the given objects. **Default behavior:** no errors occurred (if not specified). **Allowed values are:** TestNotSupported TestWrongPDL
WarningCount ? Modified in JDF 1.4	integer	The count of warnings that were encountered while preflighting the Job. **Modification note:** Starting with **JDF** 1.4, @*WarningCount* becomes optional.
FileSpec ? New in JDF 1.4	refelement	References a readable preflight report
PreflightParams	refelement	References the ***PreflightParams*** that was used to create this report.
PreflightReportRulePool ? Modified in JDF 1.4	refelement	References the ***PreflightReportRulePool*** that was used to create this report. This resource SHALL be provided if the containing ***PreflightReport*** is an input resource. **Modification note:** Starting with **JDF** 1.4, ***PreflightReportRulePool*** becomes optional.
PRItem *	element	Describes the ***Action*** elements that produced an error or a warning.
RunList	refelement	References the ***RunList*** of documents that were used to create this report.

8.114.1 PRItem

The ***PRItem*** structure is used to describe the errors that occurred during the execution of one ***Action***. When a ***Test*** could not be evaluated during the **Preflight** process, this is reported as a ***PRError***.

Objects that fail the preflight test are grouped together as described by a @*PRRule*. During the **Preflight** process, the number of objects and groups that are reported are limited to the maximum numbers defined in the @*PRRule*.

When a ***PreflightReport*** is copied from one **JDF** document to another (e.g., a **JDF** writer might reduce the size of the ***PreflightReport*** by removing ***PRGroup*** and ***PROccurrence*** items within a ***PRGroup***), this will not invalidate the ***PreflightReport***.

Table 8.199: PRItem Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ActionRef	IDREF	References the ***PreflightParams***/***ActionPool***/***Action*** that triggered this ***PRItem***.

Table 8.199: PRItem Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Occurrences	integer	The number of occurrences of objects that failed the ***Action***. When the ***Action*** describes a set-test, this is the number of set-objects that failed the test.
PageSet ?	Inte-gerRangeList	All run indices where there is an object that gives an error on that page.
PRError *	element	Describes the errors that were found while running this preflight test.
PRGroup *	element	Describes the ***Action*** elements that produced an error or a warning.

8.114.2 PRError

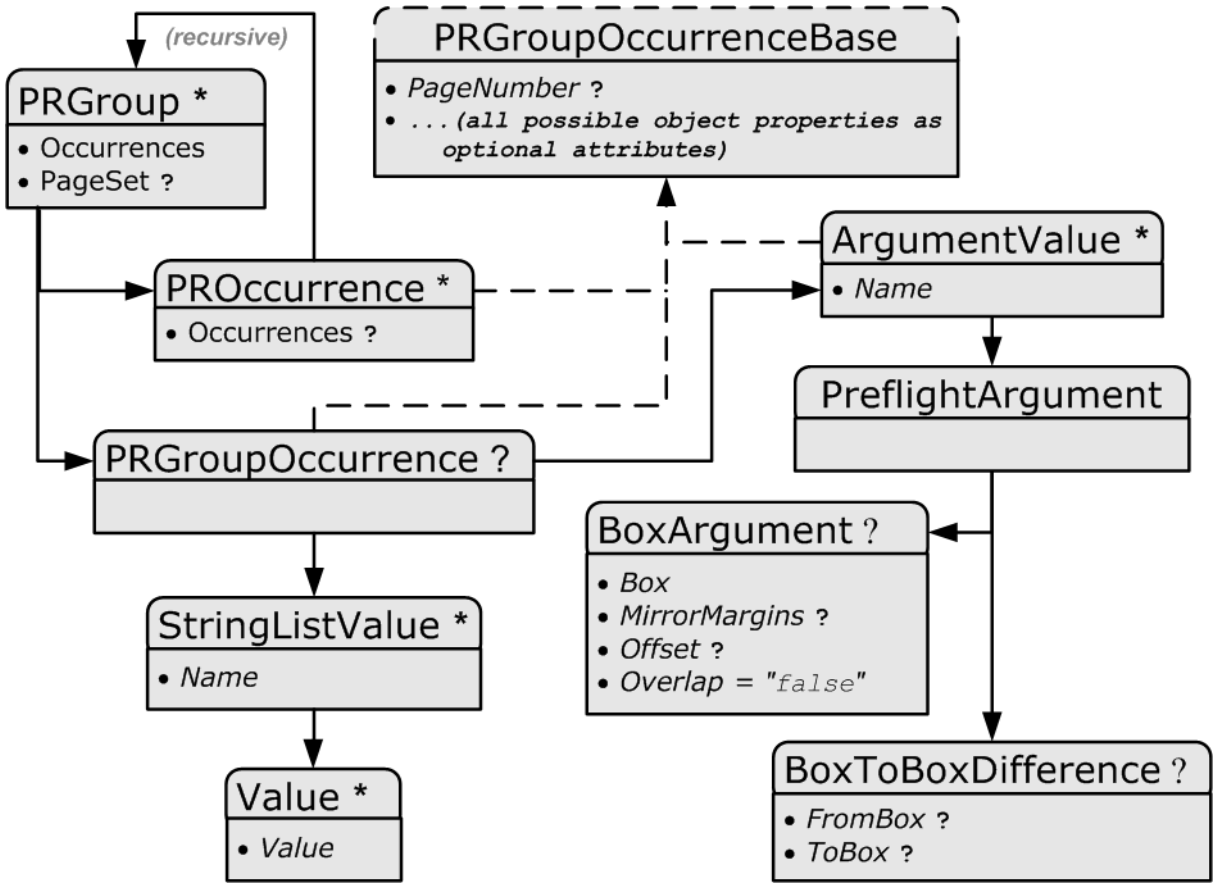
The ***PRError*** structure is used to describe generic errors that occurred while evaluating an object property while executing a ***Test***.

Table 8.200: PRError Element

NAME	DATA TYPE	DESCRIPTION
ErrorType	enumeration	**Allowed values are:** TestWrongPDL TestNotSupported
Value	NMTOKEN	The name of the object property that was being tested when the process error occurred.

PRGroup

Figure 8-47: PRGroup – a diagram of its structure



8.114.3 PRGroup

The ***PRGroup*** structure is used to describe a group of document objects that share common properties and that failed the ***Action***.

Table 8.201: PRGroup Element

NAME	DATA TYPE	DESCRIPTION
Occurrences	integer	The number of occurrences of objects of this group that failed the ***Action***. When the ***Action*** elements describes a set-test, this is the number of set-objects.
PageSet ?	IntegerRangeList	All run indices where there is an object of this group that gives an error on that page.
PRGroupOccurrence ?	element	The properties that are shared by all elements of the group as defined by ***PreflightReportRulePool***/***PRRule***/@*GroupBy*.
PROccurrence *	element	An object that failed the ***Action***.

Depending on the test in the ***Action***, the ***PRGroup*** is used in two different ways:

- When the test is not a set-test, there will be one level of ***PRGroup*** and ***PROccurrence*** elements. These are used to describe all the document objects that failed the preflight test. The ***PROccurrence*** describes the actual object while ***PRGroup*** is used to group those objects that share common properties.
- When the test is a set-test, there will be two levels of ***PRGroup*** and ***PROccurrence*** elements whereby the second level occurs as a child element of ***PROccurrence***.
  - The top level describes the set objects that failed the preflight test. Just as in the non-set-test case, ***PROccurrence*** describes the actual set-objects while ***PRGroup*** is used to group those sets that share common properties. In the example below there are four page sets that failed the test (e.g., pages 1, 4, 8 and 12).
  - The second level, which is a child element of the top level ***PROccurrence***, describes the document objects that are part of the set. These document objects are grouped as well. In the example below page one consists of 20 objects: five text objects and 15 image objects.

Example 8.37: PRItem

```
<PreflightReport Class="Parameter" ID="PP001" Status="Available" ErrorCount="0" WarningCount="0"
>
  <PRItem Occurrences="4" ActionRef="A001">
    <PRGroup Occurrences="1">
      <PRGroupOccurrence PageNumber="1"/>
      <PROccurrence Occurrences="20">
        <PRGroup Occurrences="5">
          <PRGroupOccurrence/>
          <PROccurrence TextSize="12"/>
        </PRGroup>
        <PRGroup Occurrences="15">
          <PRGroupOccurrence/>
          <PROccurrence EffectiveResolution="300 300"/>
        </PRGroup>
      </PROccurrence>
    </PRGroup>
    <PRGroup Occurrences="1">
      <PRGroupOccurrence PageNumber="4"/>
      <PROccurrence Occurrences="20">
        <PRGroup Occurrences="7">
          <PRGroupOccurrence/>
          <PROccurrence NumberOfPathPoints="4"/>
        </PRGroup>
        <PRGroup Occurrences="13">
          <PRGroupOccurrence/>
          <PROccurrence EffectiveResolution="300 300"/>
        </PRGroup>
      </PROccurrence>
    </PRGroup>
    <PRGroup Occurrences="1">
      <PRGroupOccurrence PageNumber="8"/>
    </PRGroup>
    <PRGroup Occurrences="1">
      <PRGroupOccurrence PageNumber="12"/>
    </PRGroup>
  </PRItem>
  <PreflightParams>
    <TestPool>
      <Test ID="T001">
        <BooleanEvaluation ValueList="true"/>
      </Test>
    </TestPool>
    <ActionPool>
      <Action ID="A001" TestRef="T001"/>
    </ActionPool>
  </PreflightParams>
  <PreflightReportRulePool/>
  <RunList/>
</PreflightReport>
```

8.114.4 Abstract PRGroupOccurrenceBase

Abstract PRGroupOccurrenceBase is an abstract element that serves as container for properties that were evaluated during the **Preflight** process.

Table 8.202: Abstract PRGroupOccurrenceBase Element

NAME	DATA TYPE	DESCRIPTION
All possible object properties as OPTIONAL *attributes.*	*As defined by the object property.*	An example is given above. See also section ▸ Section 10.3.2 Properties and following.
PageNumber ?	integer	Example of an integer attribute. The same format applies to boolean, Number, Name, NameList, enumeration, enumerations and string data types.

8.114.5 PRGroupOccurrenceBase

The following elements are derived from the ***Abstract PRGroupOccurrenceBase*** element

Table 8.203: List of PRGroupOccurrenceBase Elements

NAME	PAGE	DESCRIPTION
ArgumentValue	page 562	For additional arguments for a ***PRGroupOccurrence***.
PRGroupOccurrence	page 562	Specifies shared Properties of all ***PROccurrence*** elements in a ***PRGroup***
PROccurrence	page 562	Describes an individual occurrence of a preflight action failure

8.114.6 ArgumentValue

ArgumentValue specifies a value that is specified with additional arguments. ***ArgumentValue*** is derived from ***Abstract PRGroupOccurrenceBase***:

Table 8.204: ArgumentValue Element

NAME	DATA TYPE	DESCRIPTION
Name	NMTOKEN	The name of the subject property.
PreflightArgument	element	The argument that was used to evaluate this property. This is a ***PreflightArgument*** element. See ▸ Section 8.112.3 PreflightArgument.

8.114.7 PRGroupOccurrence

PRGroupOccurrence specifies the shared properties of all ***PROccurrence*** elements in a ***PRGroup***. When the object does not support a certain property, the corresponding attribute SHALL NOT be specified in ***PRGroupOccurrence***. ***PRGroupOccurrence*** is derived from ***Abstract PRGroupOccurrenceBase***.

Table 8.205: PRGroupOccurrence Element

NAME	DATA TYPE	DESCRIPTION
ArgumentValue *	element	Describes the value of a property that is enhanced with additional arguments.
StringListValue *	element	Describes the values of a StringList property.

8.114.8 StringListValue

StringListValue specifies a type that returns a set of strings.

Table 8.206: StringListValue Element

NAME	DATA TYPE	DESCRIPTION
Name	NMTOKEN	The name of the subject property.
Value *	element	Element of type ***StringEvaluation***/***Value***. See ▸ Section 10.2.13.8.2.13 StringEvaluation.

8.114.9 PROccurrence

PROccurrence describes an individual occurrence of a preflight action failure. When the object does not support a certain property, the corresponding attribute SHALL NOT be specified in ***PROccurrence***. ***PROccurrence*** is derived from ***Abstract PRGroupOccurrenceBase***.

Table 8.207: PROccurrence Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Occurrences ?	integer	Only used when the subject occurrence is a set-object. It describes the number of objects in the set.

Table 8.207: PROccurrence Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
PRGroup *	element	When this occurrence describes a set-object, the ***PRGroup*** elements describe the objects that are part of the set.

8.115 PreflightReportRulePool

New in JDF 1.2

The ***PreflightReportRulePool*** resource specifies how the ***PreflightReport*** SHALL log the errors that were found during the **Preflight** process. This section makes use of elements and attributes defined in ▸ Section Device Capabilities. It is suggested that reader's familiarize themselves with that section and ▸ Section 10.3 Concept of the Preflight Process.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***PreflightReport***

Input of Processes: **Preflight**

Table 8.208: PreflightReportRulePool Resource

NAME	DATA TYPE	DESCRIPTION
ActionPools Deprecated in JDF 1.3	IDREFS	References the ***ActionPool*** whose reporting are defined by this rule. **Deprecation note:** Starting with **JDF** 1.3 Errata, @*ActionPools* is deprecated because ***PRRule***/@*ActionRefs* has the same role.
MaxOccurrences ?	integer	An upper bound to the maximum number of ***PROccurrence*** elements that are to be logged in the ***PreflightReport***.
PRRule *	element	A list of available ***PRRule*** elements.
PRRuleAttr ?	element	Defines the default behavior of all ***PRRule*** when not defined inside of a ***PRRule*** Subelement.

8.115.1 PRRule

The ***PRRule*** structure is used to define how the ***PreflightReport*** SHALL log the events that were found during the execution of one ***Action***.

Table 8.209: PRRule Element

NAME	DATA TYPE	DESCRIPTION
ActionRefs	IDREFS	References the action for which the report behavior is defined in ***PRRule***.
PRRuleAttr	element	Defines the way to report this specific rule(s).

The format of the ***PreflightReport*** is defined by specifying ***PRRule*** elements for specific ***Action*** elements. Because @*ActionRefs* can refer to multiple ***Action*** elements, a single rule applies to all referenced ***Action*** elements (e.g., all color-related ***Action*** elements will use similar reporting).

8.115.2 PRRuleAttr

Table 8.210: PRRuleAttr Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
GroupBy = "Tested"	NMTOKENS	Group objects having the same N-pair of attributes listed here. **Values include those from: @*ReportAttr*.**
LogErrors ?	integer	When the **Preflight** process does not understand or cannot apply certain tests, that error SHALL be logged when the associated type is logged here. The value is the sum of "TestWrongPDL" and "TestNotSupported" (these two returned values are explained in ▸ Section 10.3 Concept of the Preflight Process).

Table 8.210: PRRuleAttr Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
MaxGroups ?	integer	The maximum number of groups allowed in the report for this problem. When an object is encountered that fails the preflight test and it belongs to none of the existing groups and there are already @*MaxGroups*, that occurrence is no longer reported individually and no new group is created, although it is added to the @*Occurrences* count and the @*PageSet*.
MaxPerGroup ?	integer	The maximum number of individual occurrences reported per group for this problem. When an object is encountered that fails the preflight test and it belongs to a group that already contains @*MaxPerGroup* elements, that occurrence is no longer reported individually, although it is added to the @*Occurrences* count and the @*PageSet*.
ReportAttr = "Tested Filename PageNumber"	NMTOKENS	When individual items are reported, these attributes are also reported. Attributes which are also being referred by @*GroupBy* are ignored. **Values include those from:** ▸ Table 8.211 ReportAttr Attribute Values**.**

8.115.2.1 ReportAttr Attribute Values

Table 8.211: ReportAttr Attribute Values

VALUE	DESCRIPTION
<Property Attribute>	An object-specific attribute (e.g., @*ColorSpace*, @*FontName*, etc.). At the time that we define the ***Test***, we will almost automatically define these attributes.
BriefAppSpecific	Refers to a small list of attributes that the preflight agent (with preflight agent-specific logic) finds interesting for the ***Test*** element(s) used by the ***Action*** element(s) listed in @*ActionRefs*.
Tested	Refers to all the attributes that are referred to in the ***Test*** element(s) used by the ***Action*** element(s) listed in the @*ActionRefs*.
TestRelated	Refers to all the attributes referred in the ***Test*** element(s) used by the ***Action*** element(s) listed in @*ActionRefs* and the ones that belong to the group of properties in which the tested property was found. For instance, if the @*Creator* basic test was made, then all other document properties will be reported as well.
VerboseAppSpecific	Refers to a large list of attributes that the preflight agent (with preflight agent-specific logic) finds interesting for the ***Test*** element(s) used by the ***Action*** element(s) listed in @*ActionRefs*.

When the report is generated, the "Tested", "VerboseAppSpecific" and "BriefAppSpecific" terms are expanded depending on the context (i.e., the specific test and the specific preflight agent) so that the list of attributes only contain object specific attributes.

Note: The "VerboseAppSpecific" and "BriefAppSpecific" tokens can be dependent on the context of a specific test. It is expected that a preflight agent will have a default list of tokens that will always be added (e.g., "PageNumber"). In addition it is expected that a preflight agent will define separate lists for specific domains (e.g., color, font). When a specific test covers some of these specific domains, the attributes of these lists are also added. When @*ReportAttr* = "Tested BriefAppSpecific PageNumber", the attributes that are reported are dependent on the ***Test*** element(s) used by the ***Action*** element(s) and on the preflight agent as demonstrated in the table below.

Table 8.212: Contingent Report Behavior (Sheet 1 of 2)

PREFLIGHT AGENT	FOR COLORSPACE TEST	FOR FONTEMBEDDED TEST	BEHAVIOR
Preflight agent 1	@*ColorSpace* @*PageNumber*	@*FontEmbedded* @*PageNumber* @*FontName*	@*PageNumber* is always added. For color-related tests, @*ColorSpace* is added. For font-related tests, @*FontName* is added

Table 8.212: Contingent Report Behavior (Sheet 2 of 2)

PREFLIGHT AGENT	FOR COLORSPACE TEST	FOR FONTEMBEDDED TEST	BEHAVIOR
Preflight agent 2	@*ColorSpace* @*PageNumber* @*BoundingBox*	@*FontEmbedded* @*PageNumber* @*BoundingBox* @*FontSubset*	@*PageNumber* and @*BoundingBox* are always added. For color-related tests, @*ColorSpace* is added. For font-related tests, @*FontName*, @*FontEmbedded* and @*FontSubset* are added.

When such an attribute is evaluated against an object and when the attribute is a property of the object, value will be recorded as an attribute of the ***PROccurrence*** and ***PRGroupOccurrence*** elements. When the attribute is not a property of the object, no attribute will be added to the ***PROccurrence*** and ***PRGroupOccurrence*** elements. For example: @*TextSize* on a text object would give <**PROccurrence** **TextSize**="12"/> (assuming @*TextSize* is defined as returning the size in points), but @*TextSize* on an image would correspond to **<PROccurrence/>**.

8.116 Preview

The preview of the content of a surface. It can be used for the calculation of the ink coverage (@*PreviewUsage*="Separation") or as a preview of what is currently processed in a device (@*PreviewUsage*="Viewable" or @*PreviewUsage*="ThumbNail"). When the preview is of @*PreviewUsage*="Separation" or @*PreviewUsage*="SeparationRaw", a gray value of "0" represents full ink, while a value of "255" represents no ink (for more information, see DeviceGray color model chapter 4.8.2 of the *PostScript Language Reference Manual*) ▸ [PS].

Resource Properties

Resource Class: Parameter

Resource referenced by: ▸ Any Element (generic content), ***QueueEntry***

Example Partition: "PreviewType", "Separation", "SheetName", "Side", "TileID", "WebName", "RibbonName"

Input of Processes: **InkZoneCalculation**, **PreviewGeneration**

Output of Processes: **PreviewGeneration**

Table 8.213: Preview Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Compensation ? Modified in JDF 1.2	enumeration	Compensation of the image to reflect the application of transfer curves to the image. **Allowed value is from:** ▸ Compensation.
CTM ? New in JDF 1.1 Modified in JDF 1.3	matrix	Orientation of the ***Preview*** with respect to the ***Layout*** coordinate system. CTM is applied after any transformation defined within the referenced image file (e.g., the transformation defined in the **CIP3PreviewImageMatrix** of a PPF file). In case of PPF, @*CTM* is applied to the native Postscript coordinate system of the preview. In case of PNG, the origin of the object is defined as the lower left corner of the image.
Directory ? New in JDF 1.1	URL	Defines a base URL for the files that represent this ***Preview***. If @*Directory* is specified, it SHALL be an Absolute URI ▸ [RFC3986] that implicitly also specifies a Base URI which is used to resolve any relative URL of ***Preview***. See ▸ Appendix J Resolving Directory URL References and ▸ [FileURL] for examples.
MimeTypeDetails ? New in JDF 1.4	string	Specifies additional details of the preview's MIME type in case the value of @*PreviewFileType* is a MIME type.

Table 8.213: Preview Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
PreviewFileType = "PNG" New in JDF 1.2 Modified in JDF 1.4	string	The file type of the preview. **Values include:** PNG – The Portable Network Graphics format. CIP3Multiple – The format as defined in ▸[CIP3 - PPF]. One or more previews per CIP3 file are supported. CIP3Single – The format as defined in ▸[CIP3 - PPF]. Only one preview per CIP3 file is supported. **Values include also: any MIME media type. See** ▸Appendix G MimeTypes**.** New in JDF 1.4 **Note:** The CIP3 formats were added in **JDF** 1.2 only for backwards compatibility since many systems only support CIP3 format. The CIP3 formats SHALL NOT be used except in ***Preview*** resources that are used as Input resources to **InkZoneCalculation**. **Modification note:** Starting with **JDF** 1.4, the Data Type is changed from enumeration to string because MIME media types are added as values.
PreviewType ? Deprecated in JDF 1.2	enumeration	Type of the preview. **Allowed values are:** Separation – Separated preview in medium resolution. SeparationRaw – Separated preview in medium resolution. SeparatedThumbNail – Very low resolution separated preview. ThumbNail – Very low resolution RGB preview. Viewable – RGB preview in medium resolution. **Deprecation note:** Starting with **JDF** 1.2, @*PreviewType* is still a Partition Key and SHALL be used only as such — as an attribute of ***Preview***, @*PreviewUsage* (below) replaces @*PreviewType*.
PreviewUsage = "Separation" New in JDF 1.2 Modified in JDF 1.5	enumeration	The kind of the preview. @*PreviewUsage* defines the semantics of the preview. **Constraint:** If both @*PreviewType* as a Partition Key and @*PreviewUsage* are specified, they SHALL match. **Allowed values are:** Animation – animated previews for 3D display. New in JDF 1.4 Identification – ***Preview*** is used as a visual help to identify one or more products, e.g. on a gang form. New in JDF 1.5 SeparatedThumbNail – Very low resolution separated preview. Separation – Separated preview in medium resolution. Separation is generally used in **InkZoneCalculation**. SeparationRaw – Separated preview in medium resolution. This is identical to "Separation" except that no compensation has been applied. "SeparationRaw" is generally used for closed loop color control. Static3D – static 3D model New in JDF 1.4 Modified in JDF1.5 ThumbNail – Very low resolution RGB preview. Viewable – RGB preview in medium resolution. **Modifcation note:** Starting with **JDF** 1.5, 3D was renamed to Static3D because enumerations SHALL NOT begin with a number in XML
URL Modified in JDF 1.2	URL	@*URL* identifying any preview file (e.g., the PNG image or ▸[CIP3 - PPF] file that represents this ***Preview***). See ▸[RFC3986] and ▸Appendix J Resolving Directory URL References and ▸Appendix M FileSpec Use Cases for the syntax and examples. For the "file" URL scheme see also ▸[RFC1738] and ▸[FileURL]. **Note:** A preview will generally be Partitioned by separation, unless it represents an RGB viewable image or thumbnail. PPF files with multiple images can contain multiple Separations. In this case, the separation names defined in **CIP3ADMSeparationNames** define the separations and SHALL match the @*Separation* partition keys used in the **JDF**.

8.117 PreviewGenerationParams

Parameters specifying the size and the type of the preview.

Resource Properties

Resource Class: Parameter

Example Partition: "PreviewType", "Separation", "SheetName", "Side", "TileID", "WebName", "RibbonName"

Input of Processes: **PreviewGeneration**

Table 8.214: PreviewGenerationParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AspectRatio = "Ignore" New in JDF 1.1	enumeration	Policy that defines how to define the preview size if the aspect ratio of the source and preview are different. Note that @*AspectRatio* only has an effect if @*Size* is specified. **Allowed values are:** CenterMax – Keep the aspect ratio and preview @*Size*, and center the image so that the preview has missing pixels at both sides of the larger dimension. CenterMin – Keep the aspect ratio and preview @*Size*, and center the image so that the preview has blank pixels at both sides of the smaller dimension. Crop – Keep the aspect ratio, and modify the preview size so that the image fits into a bounding rectangle defined by @*Size*. Expand – Keep the aspect ratio, and modify the preview size so that the smaller image dimension is defined by @*Size*. Ignore – Fill the preview completely, keeping @*Size*, even if this requires modifying the aspect ratio.
Compensation ? Modified in JDF 1.2	enumeration	Compensation of the image to reflect the application of transfer curves to the image. **Allowed value is from:** ▸ Compensation.
PreviewFileType = "PNG" New in JDF 1.2	enumeration	The file type of the preview to be generated. **Allowed values are:** PNG – The Portable Network Graphics format. CIP3Multiple – The format as defined in ▸ [CIP3 - PPF]. One or more previews per CIP3 file are supported. CIP3Single – The format as defined in ▸ [CIP3 - PPF]. Only one preview per CIP3 file is supported. **Note:** The CIP3 formats were added in **JDF** 1.2 only for backwards compatibility since many systems only support CIP3 format. The CIP3 formats SHALL NOT be used except in ***Preview*** resources that are used as Input resources to **InkZoneCalculation**.
PreviewType ? Deprecated in JDF 1.1	enumeration	The kind of preview to be generated. **Allowed values are:** Separation Viewable **Deprecation note:** Starting with **JDF** 1.1, @*PreviewType* is still a Partition Key and SHALL be used only as such — as an attribute of ***Preview***, @*PreviewUsage* (below) replaces @*PreviewType*.
PreviewUsage = "Separation" New in JDF 1.1 Modified in JDF 1.2	enumeration	The kind of preview to be generated. **Allowed values are:** Separation – Separated preview in medium resolution. SeparationRaw – Separated preview in medium resolution with no compensation. SeparatedThumbNail – Very low resolution separated preview. ThumbNail – Very low resolution RGB preview. Viewable – RGB preview in medium resolution. **Constraint:** @*PreviewUsage* defines the semantics of the preview. If both @*PreviewType* as a Partition Key and @*PreviewUsage* are specified, they SHALL match.
Resolution ?	XYPair	Resolution of the preview, in dpi. If @*PreviewUsage* = "Separation", the default is "50.8 50.8".

Table 8.214: PreviewGenerationParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Size ?	XYPair	Size of the preview, in pixels. If this attribute is present, the @*Resolution* attribute evaluated according to the policy defined in @*AspectRatio*. If @*Size* is not specified, it SHALL be calculated using the @*Resolution* attribute and the input image size.
ImageSetterParams ? New in JDF 1.1	refelement	Details of the **ImageSetting** process. Needed for accessing information about coordinate transformations that are performed by the imagesetter hardware.

8.118 PrintCondition

New in JDF 1.2

PrintCondition is a resource used to control the use of colorants when printing pages on a specific media. the attributes and elements of the ***PrintCondition*** resource describe the aim values for a given printing process.

Resource Properties

Resource Class: Parameter

Example Partition: "SignatureName", "SheetName", "Side", "Separation"

Input of Processes: **ConventionalPrinting**, **DigitalPrinting**

Table 8.215: PrintCondition Resource

NAME	DATA TYPE	DESCRIPTION
AimCurve ?	Transfer-Function	Describes the desired tone-value increase function. If not specified, it defaults to the media and printing machinespecific values.
Density ?	double	Density value of colorant (100% tint). Whereas ***Color***/@*NeutralDensity* describes measurements of inks on substrate with wide-band filter functions, @*Density* is derived from measurements of inks on substrate with special small band filter functions according to ANSI and DIN. If not specified, it defaults to the value of ***Color***// @*Density*.
Name	string	Name of the ***PrintCondition***. Used to reference a ***PrintCondition*** from a ***Color*** element.
ColorMeasurementConditions ?	refelement	Describes measurement conditions for color measurement and density measurement. If not specified, it defaults to the value of ***Color***// ***ColorMeasurementConditions***.
Device ?	refelement	Specifies the device or device group that this ***PrintCondition*** applies to.
FileSpec (TargetProfile) ?	refelement	A ***FileSpec*** resource pointing to an ICC profile that defines the target output device in case the object that uses the color has been color space converted to a device color space. If not specified, it defaults to the value of ***Color***// ***FileSpec*** (TargetProfile).

Example 8.38: PrintCondition

```
<ColorMeasurementConditions Class="Parameter" ID="MyColorMeasCond"
        Status="Available"/>
<PrintCondition Name="Standard" Class="Parameter" ID="PC"
        PartIDKeys="Side Separation" Status="Available">
    <ColorMeasurementConditionsRef rRef="MyColorMeasCond"/>
    <PrintCondition Side="Front">
        <PrintCondition AimCurve="0.0 0.0 0.5 0.66 1.0 1.0" Density="1.8"
            Separation="Black"/>
        <PrintCondition AimCurve="0.0 0.0 0.5 0.63 1.0 1.0" Density="1.4"
            Separation="Cyan"/>
    </PrintCondition>
</PrintCondition>
```

8.119 PrintRollingParams

New in JDF 1.2

Resource Properties

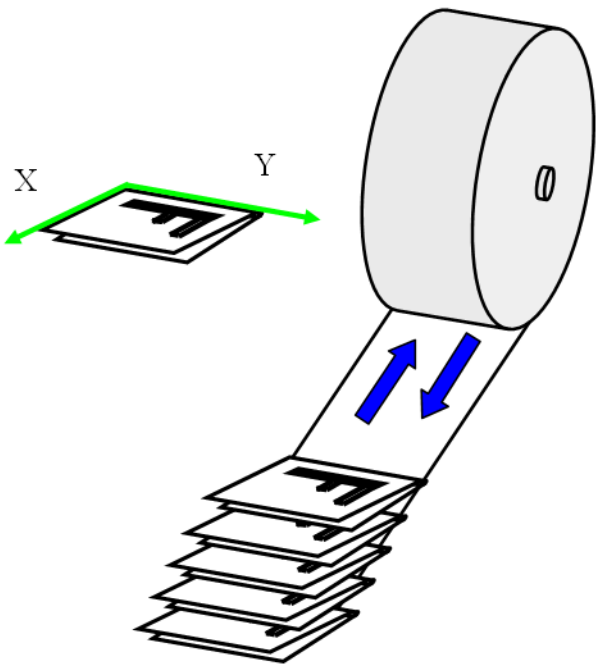
Resource Class: Parameter

Input of Processes: **PrintRolling**

Table 8.216: PrintRollingParams Resource

NAME	DATA TYPE	DESCRIPTION
Copies ?	integer	Number of copies on the Roll. @*Copies* SHALL NOT be specified if @*MaxDiameter* is present.
MaxDiameter ?	double	Maximal allowed diameter of Roll. @*MaxDiameter* SHALL NOT be specified if @*Copies* is present.

Figure 8-48: PrintRollingParams Coordinate System



8.120 ProductionPath

New in JDF 1.3

ProductionPath describes the individual paper path through the different modules of a web-press device, in order to produce a particular product.

Resource Properties

Resource Class: Parameter

Resource referenced by: *CylinderLayoutPreparationParams*

Example Partition: "RibbonName", "WebName"

Input of Processes: **WebInlineFinishing**

Table 8.217: ProductionPath Resource

NAME	DATA TYPE	DESCRIPTION
ProductionPathID ?	string	Identification of the entire production path. The @*ProductionPathID* SHALL be unique within the machine. If not specified, ***PrintingUnitWebPath*** SHALL be specified.
FolderSuperstructureWebPath ?	element	Describes the path through the folder super-structure. The Web will generally be cut into ribbons in this area of the production path.
PostPressComponentPath *	element	Describes the path through the inline postpress equipment. Folded Sheets (***Component***) will be processed in this area of the production path.
PrintingUnitWebPath ?	element	Describes the path through the printing units. If not specified, @*ProductionPathID* SHALL be specified.

8.120.1 FolderSuperstructureWebPath

This is a placeholder that might be filled with additional information in future versions of **JDF**. In **JDF** 1.3, paths are identified by ID only.

*Table 8.218: **FolderSuperstructureWebPath** Element*

NAME	DATA TYPE	DESCRIPTION
ProductionPathID ?	string	Unique identification of the part of the production path specified in this element.

8.120.2 PostPressComponentPath

This is a placeholder that might be filled with additional information in future versions of **JDF**. In **JDF** 1.3, paths are identified by ID only.

*Table 8.219: **PostPressComponentPath** Element*

NAME	DATA TYPE	DESCRIPTION
ProductionPathID ?	string	Unique identification of the part of the production path specified in this element.

8.120.3 PrintingUnitWebPath

This is a placeholder that might be filled with additional information in future versions of **JDF**. In **JDF** 1.3, paths are identified by ID only.

*Table 8.220: **PrintingUnitWebPath** Element*

NAME	DATA TYPE	DESCRIPTION
ProductionPathID ?	string	Unique identification of the part of the production path specified in this element.

Example 8.39: ProductionPath: on Path Level:

This example and the next illustrate the different Web path description levels:

```
<ProductionPath Class="Parameter" ID="F1" Status="Available"
    ProductionPathID="ID_2webproduction_64pages"/>
```

Example 8.40: ProductionPath: on Part Path Level:

This example and the previous illustrate the different Web path description levels:

```
<ProductionPath Class="Parameter" ID="F1" Status="Available"
            PartIDKeys="WebName">
    <ProductionPath WebName="1">
        <PrintingUnitWebPath ProductionPathID="ID_PrintingUnitWebPath"/>
        <FolderSuperstructureWebPath ProductionPathID="abcd"/>
        <PostPressComponentPath ProductionPathID="xyz"/>
    </ProductionPath>
</ProductionPath>
```

8.121 ProofingParams

Deprecated in JDF 1.2

In **JDF** 1.2 and beyond, proofing is handled as a combined process. For detail of this deprecated resource, see ▸ Section N.7.21 ProofingParams.

8.122 PSToPDFConversionParams

This resource contains the parameters that control the conversion any PDL to PDF documents. Prior to **JDF** 1.3, ***PSToPDFConversionParams*** was used only for converting PostScript streams to PDF. The name "***PSToPDFConversionParams***" was retained for backwards compatibility, although most parameters apply to PDF conversion from any source format.

Some descriptions below mention attributes or structures in specific source formats, such as PostScript. Appropriate equivalent actions should be taken when converting from other source formats that have equivalent attributes or structures. A small number of parameters apply only to PostScript sources.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***PDLCreationParams***

Example Partition: "DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetName", "Side", "SignatureName"

Input of Processes: **PSToPDFConversion**

Table 8.221: PSToPDFConversionParams Resource (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
AllowJBIG2Globals = "false"	boolean	This resource allows JBIG2 compressed images to share a single global dictionary in the resulting PDF file instead of a dictionary per image.
ASCII85EncodePages = "false"	boolean	If "true", binary streams (e.g., page contents streams, sampled images, and embedded fonts) are ASCII85-encoded, resulting in a PDF file that is almost pure ASCII. If "false", they are not, resulting in a PDF file that can contain substantial amounts of binary data.

Table 8.221: PSToPDFConversionParams Resource (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
AutoRotatePages ?	enumeration	Allows the device to try to orient pages based on the predominant text orientation. If the source is PostScript, this attribute is only used if the file does not contain "%%ViewingOrientation", "%%PageOrientation" or "%%Orientation" DSC comments. If the file does contain such DSC comments, it honors them. "%%ViewingOrientation" takes precedence over others, then "%%PageOrientation", then "%%Orientation". **Allowed values are:** None – Turns @*AutoRotatePages* off. All – Takes the predominant text orientation across all pages and rotates all pages the same way. PageByPage – Does the rotation on a page-by-page basis, rotating each page individually. Useful for documents that use both portrait and landscape orientations.
Binding = "Left"	enumeration	Determines how the printed pages would be bound. **Allowed values are:** Left – for left binding. Right – for right binding.
CompressPages ?	boolean	Enables compression of pages and other content streams like forms, patterns and Type 3 fonts. If "true", use Flate compression.
DefaultRenderingIntent ? Modified in JDF 1.2	enumeration	Selects the rendering intent for the current job. **Allowed value is from:** ▸ RenderingIntent.
DetectBlend = "true"	boolean	Enables or disables blend detection. If "true" and if @*PDFVersion* is 1.3 or higher, then blends will be converted to smooth shadings.
DoThumbnails = "true"	boolean	If "true", thumbnails are created.
EndPage ? Deprecated in JDF 1.3	integer	Number that indicates the last page that is displayed when the PDF file is viewed. @*EndPage* SHALL be either "-1" or greater than or equal to @*StartPage*. When combined with @*StartPage*, @*EndPage* selects a range of pages to be displayed. The entire file MAY be converted, but only @*StartPage* to @*EndPage* pages, inclusive, are opened and viewed in a PDF viewing application.
ImageMemory ? Deprecated in JDF 1.2	integer	Number of bytes in the buffer used in sample processing for color, grayscale and monochrome images. Its contents are written to disk when the buffer fills up. This attribute was deprecated because it is an internal application setting and not a parameter setting.
InitialPageSize ? New in JDF 1.1	XYPair	Defines the initial page dimensions, in points, that will be used to set MediaBox. This will be overridden by any page size attribute found in the source document, such as the PostScript **PageSize** page device parameter. The use of this attribute is strongly encouraged when processing EPS files (%%BoundingBox comments do not override @*InitialPageSize*).
InitialResolution ? New in JDF 1.1	XYPair	Defines the initial horizontal and vertical resolution, in dpi. This will be overridden by any resolution attribute found in the source document, such as the PostScript **HWResolution** page device parameter. The use of this attribute is strongly encouraged when processing EPS files.
Optimize = "true"	boolean	If "true", the PS-to-PDF converter optimizes the PDF file. See ▸ [PDF1.6] for more information on optimization.
OverPrintMode ?	integer	Controls the overprint mode strategy of the Job. Set to "0" for full overprint or "1" for non-zero overprint. For more information, see ▸ [Adb-TN5044].
PDFVersion ?	double	Specifies the version number of the PDF file produced. Values include all legal version designators (e.g., 1.2, 1.5).

Table 8.221: PSToPDFConversionParams Resource (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
StartPage ? Deprecated in JDF 1.3	integer	Sets the first page that is be displayed when the PDF file is opened with a PDF viewing application. @*StartPage* SHALL be greater than or equal to 1. @*EndPage* SHALL be either "-1" or greater than or equal to @*StartPage*.
AdvancedParams ?	element	Advanced parameters which control how certain features of PDF are handled.
PDFXParams ? New in JDF 1.2	element	PDF/X parameters.
ThinPDFParams ?	element	Parameters that control the optional content or form of PDF files that will be created.

8.122.1 AdvancedParams

Table 8.222: AdvancedParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AllowPSXObjects = "true" New in JDF 1.2	boolean	If "true", allows PostScript **XObject**s.
AllowTransparency = "false" New in JDF 1.2	boolean	If "true", allows transparency in the PDF.
AutoPositionEPSInfo = "true" Modified in JDF 1.1A	boolean	If "true", the process automatically resizes and centers information from EPS source files on the page. (EPS source only)
EmbedJobOptions = "false" New in JDF 1.2	boolean	If "true", the PDF settings used to create the PDF are embedded in the PDF.
EmitDSCWarnings = "false"	boolean	If "true", warning messages about questionable or incorrect DSC comments appear during the processing of the source PostScript file. (PostScript source only)
LockDistillerParams = "true"	boolean	If "true", any ***PSToPDFConversionParams*** settings configured by the source content (e.g., with **setdistillerparams** in a PostScript source document) are ignored. If "false", each parameter defined in the source document overrides that set in the **JDF**.
ParseDSCCommentForDocInfo = "true"	boolean	If "true", the process parses the DSC comments in a PostScript source file and extracts the document information. This information is recorded in the Info dictionary of the PDF file.
ParseDSCComments = "true"	boolean	If "true", the process parses the DSC comments in a PostScript source document for any information that might be helpful for converting the file or for information that is to be stored in the PDF file. If "false", the process treats the DSC comments as pure PS comments and ignores them. (PostScript source only)
PassThroughJPEGImages = "false" New in JDF 1.2	boolean	If "true", JPEG images are passed through without recompressing them.

Table 8.222: AdvancedParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
PreserveCopyPage = "true"	boolean	If "true", the **copypage** operator of PostScript Level 2 is maintained. If "false", the PostScript Level 3 definition of copypage operator is used. In PostScript Levels 1 and 2, the copypage operator transmits the page contents to the current output device (similar to **showpage**). However, **copypage** does not perform many of the re-initializations that **showpage** does. Many PostScript Level 1 and 2 programs used the **copypage** operator to perform such operations as printing multiple copies and implementing forms. These programs produce incorrect results when interpreted using the Level 3 **copypage** semantics. This attribute provides a mechanism to retain Level 2 compatibility for this operator. (PostScript source only)
PreserveEPSInfo = "true"	boolean	If "true", preserves the EPS information in a PostScript source file and stores it in the resulting PDF file. (PostScript source only)
PreserveHalftoneInfo = "false" New in JDF 1.1	boolean	If "true", passes halftone screen information (frequency, angle and spot function) into the PDF file. If "false", halftone information is not passed in.
PreserveOPIComments = "true"	boolean	If "true", encapsulates Open Prepress Interface (OPI) low resolution images as a form and preserves information for locating the high resolution images.
PreserveOverprintSettings = "true" New in JDF 1.1	boolean	If "true", passes the value of the **setoverprint** operator through to the PDF file. Otherwise, overprint is ignored.
TransferFunctionInfo = "Preserve" New in JDF 1.1	enumeration	Determines how transfer functions are handled. **Allowed values are:** Preserve – Transfer functions are passed into the PDF file. Remove – Transfer functions are ignored. They are neither applied to the color values nor passed into the PDF file. Apply – Transfer functions are used to modify the data that are written to the PDF file, instead of writing the transfer function itself to the file.
UCRandBGInfo = "Preserve" New in JDF 1.1	enumeration	Determines whether the under-color removal and black-generation parameters from the source document (e.g., the arguments to the PostScript commands **setundercolorremoval** and **setblackgeneration**) are passed into the PDF file. **Allowed values are:** Preserve – The arguments are passed into the PDF file. Remove – The arguments are ignored.
UsePrologue = "false"	boolean	If "true", the process SHALL append a PostScript prologue file before beginning of the Job and append a PostScript epilog file after the end the Job. Such files are used to control the PostScript environment for the conversion process. The expected location and allowable contents for these files is defined by the process implementation. (PostScript source only)

8.122.2 PDFXParams

New in JDF 1.2

Parameters for generating PDF/X files. Note that TrimBox, BleedBox, output intent and the Trapped state may be provided by the use of the **pdfmark** operator in a PostScript source file.

Table 8.223: PDFXParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
PDFX1aCheck = "false" Deprecated in JDF 1.5	boolean	If "true", checks compliance with the PDF/X-1a standard ▸ [ISO15930-1:2001]. **Deprecation note:** Use @*PDFXCheck* instead.

Table 8.223: PDFXParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
PDFX3Check = "false" Deprecated in JDF 1.5	boolean	If "true", checks compliance with the PDF/X-3 standard ▸ [ISO15930-3:2002]. **Deprecation note:** Use @*PDFXCheck* instead.
PDFXBleedBoxtoTrimBoxOffset ?	rectangle	If the **BleedBox** entry is not specified in the page object of the source document, **BleedBox** is set to PDF **TrimBox** with offsets. All numbers SHALL be greater than or equal to 0.0. PDF **BleedBox** will be completely outside PDF **TrimBox**.
PDFXCheck ? New in JDF 1.5	NMTOKENS	List of PDF/X versions that the output SHALL be compliant with. **Values include:** X1a – see the PDF/X-1a standard ▸ [ISO15930-1:2001]. X3 – see the PDF/X-3 standard ▸ [ISO15930-3:2002]. X4 – see the PDF/X-4 standard ▸ [ISO15930-7:2010]. X5 – see the PDF/X-5 standard ▸ [ISO15930-8:2010].
PDFXCompliantPDFOnly = "false"	boolean	If "true", produces a PDF document only if PDF/X compliance tests are passed.
PDFXNoTrimBoxError = "true"	boolean	If "true" and both **TrimBox** and **ArtBox** entries are not specified in the page object of the source document, the condition is reported as an error.
PDFXOutputCondition ?	string	The string is an optional comment which is added to the PDF file. It describes the intended printing condition in a form that ought to be meaningful to a human operator at the site receiving the PDF document.
PDFXOutputIntentProfile ?	string	If the source document does not specify an output intent name, then this value is used. **Values include those from:** ▸ Table 8.224 PDFXOutputIntentProfile Attribute Values**.**
PDFXRegistryName	URL	Indicates a location at which more information regarding the registry that defines the **OutputConditionIdentifier** can be obtained.
PDFXSetBleedBoxToMediaBox = "true"	boolean	If "true" and the **BleedBox** entry is not specified in the page object of the source document, **BleedBox** is set to **MediaBox**.
PDFXTrapped ?	enumeration	If a source document does not specify a **Trapped** state, then the value provided here is used. The value "Unknown" SHALL be used for workflows requiring 1) that the document specify a **Trapped** state and 2) that compliance checking fail if **Trapped** is not present in the document. **Allowed values are:** Unknown false true **Note:** "Unknown" is prohibited in PDF/X files.
PDFXTrimBoxToMediaBoxOffset?	rectangle	If both the **TrimBox** and **ArtBox** entries are not specified in the page object of the source document, **TrimBox** is set to **MediaBox** with offsets. All numbers SHALL be greater than or equal to 0.0. The **TrimBox** will be completely inside **MediaBox**.

8.122.2.1 PDFXOutputIntentProfile Attribute Values

*Table 8.224: **PDFXOutputIntentProfile** Attribute Values*

VALUE	DESCRIPTION
None	Used when it is REQUIRED that the source document specifies an intent; allows compliance checking to fail
Euroscale Coated v2	
Euroscale Uncoated v2	
Japan Color 2001 Coated	
Japan Color 2001 Uncoated	
Japan Standard v2	
Japan Web Coated (Ad)	
U.S. Sheetfed Coated v2	
U.S. Sheetfed Uncoated v2	
U.S. Web Coated (SWOP) v2	
U.S. Web Uncoated v2	
Photoshop 4 Default CMYK	
Photoshop 5 Default CMYK	

8.122.3 ThinPDFParams

Table 8.225: ThinPDFParams Element

NAME	DATA TYPE	DESCRIPTION
FilePerPage = "false"	boolean	If "true", the process generates 1 PDF file per page.
SidelineEPS = "false" New in JDF 1.2	boolean	If "true", embedded EPS files in PostScript source documents are not converted but are stored in external files in the same location as the PDF itself. (PostScript source only)
SidelineFonts = "false"	boolean	If "true", font data are stored in external files during PDF generation.
SidelineImages = "false"	boolean	If "true", image data are stored in an external stream during the PDF Generation phase. This prevents large amounts of image data from having to be passed through all phases of the code generation process.

8.123 QualityControlParams

New in JDF 1.2

QualityControlParams defines the set of parameters for the quality control process. The specific measurement conditions are defined in specialized subelements such as ***BindingQualityParams***. Parameters for **QualityControl** MAY also be referenced by providing a ***FileSpec***. Parameters for **QualityControl** in XML SHOULD be provided as subelements in a separate namespace. Examples include ▸ [ISO17972-1:2015] for color measurement data.

Resource Properties

Resource Class: Parameter

Input of Processes: **QualityControl**

Table 8.226: QualityControlParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
SampleInterval ?	integer	Interval in number of samples between tests.

Table 8.226: QualityControlParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
TimeInterval ?	duration	Time interval between individual tests.
BindingQualityParams ?	element	Specification of the binding quality measurements..
FileSpec ? New in JDF 1.6	element	Location of an external file that contains details of the quality control setup.

8.123.1 BindingQualityParams

The set of parameters in ***BindingQualityParams*** identifies how the quality of the binding is verified.

8.123.1.1 Pull test

In the pull test (sheet pulling test), a single sheet is subjected to slowly increasing tensile loading until it comes away from the glue film or the material breaks down. The load increases constantly during the automatic test procedure. It is applied evenly along the whole length of the glued seam.
Note: That is why the pull test is also described as a static test method.

8.123.1.2 Flex test

The page flex test (page turning test) is used more and more rarely in quality checking, not least because it takes a long time. In the page flex test a sheet is moved back and forth under varying tensile loads, usually at 1 N/cm, until it pulls out of the glue film, with the number of to and fro movements being measured automatically.
Note: As this test procedure involves a rapid turning movement, the flex test is called a dynamic test procedure.

Table 8.227: BindingQualityParams Element

NAME	DATA TYPE	DESCRIPTION
FlexValue ?	double	Flex quality parameter measured in [N/cm].
PullOutValue ?	double	Pull out quality parameter measured in [N/cm].

8.124 QualityControlResult

New in JDF 1.2

This set of parameters returns results of a **QualityControl** process. The ***QualityControlResult*** defines the set of results from the quality control process. The specific measurements are returned in specialized subelements such as ***BindingQualityParams***. **QualityControl** results may also be referenced by providing a ***FileSpec***. **QualityControl** measurement results in XML SHOULD be provided as subelements in a separate namespace. Examples include ▸ [ISO17972-1:2015] for color measurement data.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***Abstract Resource***

Output of Processes: **QualityControl**

Table 8.228: QualityControlResult Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
End ? New in JDF 1.6	dateTime	Date and time of the end of the measurement. If not specified, the value of @*Start* is applied.
Failed ?	integer	Total number of failed measurements.
Passed ?	integer	Total number of passed measurements.
Start ? New in JDF 1.6	dateTime	Date and time of the start of the measurement. If not specified, the measurement time is not known.
BindingQualityParams ?	element	Reference to the measurement setup definition.

Table 8.228: QualityControlResult Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
FileSpec ?	refelement	Location of an external file that contains details of the quality control measurement.
QualityMeasurement *	element	One individual measurement result.

8.124.1 QualityMeasurement

QualityMeasurement elements describe an individual measurement.

Table 8.229: QualityMeasurement Element

NAME	DATA TYPE	DESCRIPTION
Condition ?	NMTOKEN	Condition of the tested *Component*. If the *Component* passed the test, but the test itself destroyed the *Component*, the value SHALL be set to "destroyed". **Values include:** destroyed
End ?	dateTime	Date and time of the end of the measurement. If not specified, the value of @*Start* is applied.
Failed ?	integer	Total number of failed measurements.
Passed ?	integer	Total number of passed measurements.
Start ?	dateTime	Date and time of the start of the measurement. If not specified, the measurement time is not known.
BindingQualityMeasurement ?	element	Details of the *BindingQualityMeasurement*.

8.124.2 BindingQualityMeasurement

Table 8.230: BindingQualityMeasurement Element

NAME	DATA TYPE	DESCRIPTION
FlexValue ?	double	Flex quality parameter given in [N/cm].
PullOutValue ?	double	Pull out quality parameter given in [N/cm].

8.125 RasterReadingParams

New in JDF 1.3

This set of parameters specifies the details for **RasterReading**.

Resource Properties

Resource Class: Parameter

Input of Processes: **RasterReading**

Table 8.231: RasterReadingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Center = "false"	boolean	Indicates whether or not the finished page image SHALL be centered within the imageable area of the media. The @*Center* is ignored if *FitPolicy*/@*SizePolicy* = "ClipToMaxPage" and clipping is requested.
MirrorAround = "None"	enumeration	This attribute specifies the axis around which a raster reader SHALL mirror an image. **Allowed value is from:** ▸ Axis.

Table 8.231: RasterReadingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Polarity = "Positive"	enumeration	The image SHALL be RIPed in the polarity specified. Note that this is a polarity change in the RIP and not a polarity change in the hardware of the output device. **Allowed value is from:** ▸ Polarity.
Poster ? Deprecated in JDF 1.5	XYPair	Specifies whether the page contents SHALL be expanded such that each page covers X by Y pieces of media. **Deprecation note:** Starting with **JDF** 1.5, use **Tiling** (▸ Section 6.3.41 Tiling).
PosterOverlap ? Deprecated in JDF 1.5	XYPair	This pair of real numbers identifies the amounts of overlap in points, that specify the poster tiles across the horizontal and vertical axes, respectively. **Deprecation note:** Starting with **JDF** 1.5, use **Tiling** (▸ Section 6.3.41 Tiling).
Scaling ?	XYPair	A pair of positive real values that indicates the scaling factor for the page contents. Values between 0 and 1 specify that the contents are to be reduced, while values greater than 1 specify that the contents are to be expanded. This attribute is ignored if @*FitToPage* = "true" or if @*Poster* is present and has a value other than "1 1". Any scaling defined in ***FitPolicy*** SHALL be applied after the scaling defined by this attribute.
ScalingOrigin ?	XYPair	A pair of real values that identify the point in the unscaled page that SHALL become the origin of the new, scaled page image. This point is defined in the coordinate system of the unscaled page. If not specified, and scaling is requested, the @*ScalingOrigin* defaults to "0 0"
FitPolicy ? New in JDF 1.1	element	Allows printing even if the size of the imageable area of the media does not match the requirements of the data. This replaces the deprecated @*FitToPage* attribute. This ***FitPolicy*** resource SHALL be ignored in a combined process with **LayoutPreparation**.
Media * New in JDF 1.1 Modified in JDF 1.2	refelement	This resource provides a description of the physical media which will be marked. The physical characteristics of the media MAY affect decisions made during **RasterReading**. The cardinality was changed to "*" in **JDF** 1.2 in order support description of multiple media types (e.g., Film, Plate and Paper). If multiple ***Media*** are specified, The ***Media***/@*MediaType* defines the type of ***Media***. If multiple ***Media*** with ***Media***/@*MediaType* = "Paper" are specified in a proofing environment, the first ***Media*** is the proofer paper and the second ***Media*** is the final device paper.

8.126 RegisterMark

Defines a register mark, which can be used for setting up and monitoring color registration in a printing process. It can also be used to synchronize the Sheet position in a paper path. The position and rotation of each register mark can be specified with the help of the following attributes. It is important that the register marks are defined in such a way that their centers are on the point of origin of the coordinate system, as otherwise they are not positioned properly.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***HoleMakingParams***, ***Layout***/***MarkObject***

Table 8.232: RegisterMark Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Center	XYPair	Position of the center of the register mark in the coordinates of the ***MarkObject*** that contains this mark.
MarkType ? Modified in JDF 1.4	NMTOKENS	Type of ***RegisterMark***. **Values include:** Arc Circle Cross **Modification note:** Starting with **JDF** 1.4, the data type changes from NMTOKEN to NMTOKENS.

Table 8.232: RegisterMark Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
MarkUsage ? New in JDF 1.1 Modified in JDF 1.4	enumerations	Specifies the usage of the ***RegisterMark***. **Allowed values are:** Color – The mark is used for separation color registration. PaperPath – The mark is used for paper path synchronization. Tile – The mark is used to mark the position of tiles in Tiling. New in JDF 1.4
Rotation ?	double	Rotation in degrees. Positive graduation figures indicate counter-clockwise rotation; negative figures indicate clockwise rotation.
SeparationSpec * Modified in JDF 1.2	element	Set of separations to which the register mark is bound.

8.127 RenderingParams

This set of parameters identifies how the **Rendering** process SHALL operate. Specifically, these parameters define the expected output of the ***ByteMap*** resource that the **Rendering** process creates.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***ProofingIntent***

Example Partition: "DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetName", "Side", "SignatureName"

Input of Processes: **Rendering**

Table 8.233: RenderingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BandHeight ?	integer	Height of output bands expressed in lines. For a frame device, the band height is simply the full height of the frame.
BandOrdering ?	enumeration	Indicates whether output buffers are generated in "BandMajor" or "ColorMajor" order. **Allowed values are:** BandMajor – The position of the bands on the page is prioritized over the color. ColorMajor – All bands of a single color are played in order before progressing to the next plane. This is only possible with non-interleaved data.
BandWidth ?	integer	Width of output bands, in pixels.
ColorantDepth ?	integer	Number of bits per colorant. Determines whether the output is bitmaps or bytemaps.
Interleaved ?	boolean	If "true", the resulting colorant values are interleaved and @*BandOrdering* is ignored.
MimeType ? New in JDF 1.5	string	@*MimeType* identifies the MIME type associated with this output file format. For example "application/pdf".
AutomatedOverPrintParams ?	element	Controls for overprint substitutions. Defaults to no automated overprint generation.
Media ? New in JDF 1.1 Deprecated in JDF 1.2	refelement	This resource provides a description of the physical media which will be marked. The physical characteristics of the media MAY affect decisions made during **Rendering**. In **JDF** 1.2 and beyond, a RIP SHALL obtain ***Media*** information from ***InterpretingParams***/***Media***.
ObjectResolution * Modified in JDF 1.2	element	Elements which define the resolutions to render the contents at. More than one element MAY be used to specify different resolutions for different @*SourceObjects* types. If no ***ObjectResolution*** is specified, the value is implied from the input data.

Table 8.233: RenderingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
TIFFFormatParams ? New in JDF 1.5	element	Parameters specific to conversion of rasters to TIFF files.

8.127.1 TIFFFormatParams

New in JDF 1.2

Table 8.234: TIFFFormatParams Element

NAME	DATA TYPE	DESCRIPTION
ByteOrder ?	enumeration	Byte order of the TIFF file. **Allowed values are:** II – Low byte first. MM – high byte first. **Note:** The identifier values have been selected to match the identifier with the same purpose within the TIFF file itself.
Interleaving = "1"	integer	How the components of each pixel are stored. The values are taken from TIFF tag 284—*PlanarConfiguration*: **Allowed values are:** 1 – "Chunky" format, which is pixel interleaved. 2 – "Planar" format, which is strip interleaved.
RowsPerStrip ?	integer	The number of image scan lines per strip, encoded in the TIFF file as RowsPerStrip. This attribute is ignored if @*Segmentation*! = "Stripped". The default, when not known, is set by the processing system with the exception that when converting from ***ByteMap*** to TIFF, ***ByteMap***/@*BandHeight* is the default.
Segmentation ?	enumeration	How the image data are segmented. **Allowed values are:** SingleStrip – all data are included in one segment. This is encoded in the TIFF file by setting @*RowsPerStrip* to a number equal to or larger than the number of pixel rows in the image. Stripped – Data are segmented into strips. Tiled – Data are segmented into tiles.
SeparationNameTag = "270"	integer	When color separations are stored in individual TIFF files it is often useful to mark each with the name of the colorant that it represents, but there is no universally accepted way to do this. In order to avoid the need for explicit Partitioning, the tag to be used to encode the separation name (as a string) can be entered here as the TIFF tag number. If the same TIFF tag number is also supplied as a TIFFtag Subelement, then the TIFFtag element takes priority over @*SeparationNameTag*. The tag SHOULD only be put in the resulting TIFF files if the name of the separation is known.The default of "270" is the "TIFF" ImageDescription tag.
TileSize ?	XYPair	Two integers. The X value provides width of tiles, and the Y value provides height of tiles. This attribute is ignored if @*Segmentation* is not "Tiled".
WhiteIsZero = "true"	boolean	When writing monochrome or grayscale files, this flag indicates whether the data SHALL be written as "WhiteIsZero" or "BlackIsZero."
TIFFEmbeddedFile *	element	Files to be embedded within the created TIFF file. These might include an ICC profile, XMP data, etc.
TIFFtag *	element	Specific tag values for inclusion in the TIFF file.

The number of channels SHOULD be derived from the raster data to be converted.

When the **PhotometricInterpretation** tag = 5 and the **InkSet** tag = 2, it is strongly RECOMMENDED that the NumberOfInks and InkNames tags be completed—separation names MAY be obtained from the ***ColorPool*** resource .

Flate and JPEG compression in resulting TIFF files SHOULD use Compression = 8 and Compression = 7 respectively, as documented in [TIFFPS]. In particular, the JPEG encoding using Compression = 6, as described in [TIFF6] SHOULD NOT be used.

8.127.2 TIFFtag

New in JDF 1.2

Table 8.235: TIFFtag Element

NAME	DATA TYPE	DESCRIPTION
BinaryValue ?	hexBinary	If the type of the tag is UNDEFINED, then @*BinaryValue* is used to encode the data
IntegerValue ?	IntegerList	If the type of the tag is BYTE, SHORT, LONG, SBYTE, SSHORT or SLONG, then @*IntegerValue* is used to encode that data
NumberValue ?	DoubleList	If the type of the tag is RATIONAL, SRATIONAL, FLOAT or DOUBLE, then @*NumberValue* is used to encode that data
StringValue ?	string	If the type of the tag is ASCII, then @*StringValue* is used to encode the data.
TagNumber	integer	Tag number of the specified tag (e.g., 270 (decimal) for ImageDescription).
TagType	integer	The type of the tag as defined in [TIFF6] (1 = BYTE, 2 = SHORT, etc.).

Exactly one of @*IntegerValue*, @*NumberValue*, @*StringValue* or @*BinaryValue* SHALL be present, depending on the type of the TIFF tag to be carried. TIFFtag elements SHALL NOT be used for any tags related to the image data and its encoding (ImageWidth, Compression, etc.). TIFFtag elements MAY include informational tags such as OPIProxy, ImageID, Copyright, DateTime, ImageDescription, etc.

8.127.3 TIFFEmbeddedFile

New in JDF 1.2

Table 8.236: TIFFEmbeddedFile Element

NAME	DATA TYPE	DESCRIPTION
TagNumber	integer	Tag number of the specified tag (e.g., 34675 (decimal) for an ICC profile or 700 for XMP).
TagType	integer	The type of the tag as defined in [TIFF6]. This will usually be 1 (BYTE) or 7 (UNDEFINED).
FileSpec	refelement	Reference to the file to be embedded.

8.128 ResourceDefinitionParams

This set of parameters identifies how the **ResourceDefinition** process SHALL operate. Specifically, these parameters define how default parameters of applications and the Input resource are to be combined.

Resource Properties

Resource Class: Parameter

Input of Processes: **ResourceDefinition**

Table 8.237: ResourceDefinitionParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
DefaultID ? Deprecated in JDF 1.1	NMTOKEN	**JDF** ID of the default resource. If missing, it is assumed that the file specified by @*DefaultJDF* contains only a **JDF** resource element, not a complete **JDF**.
DefaultJDF ?	URL	Link to a **JDF** resource that defines preset values.

Table 8.237: ResourceDefinitionParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
DefaultPriority = "DefaultJDF"	enumeration	Defines whether preset values of the application or of the resource specified in @*DefaultJDF* have priority. **Allowed values are:** Application – The application default settings are used to fill the resource. DefaultJDF – The settings specified in @*DefaultJDF* are applied.
ResourceParam * New in JDF 1.1 Modified in JDF 1.3	element	Specification of the definition parameters of one individual resource.

8.128.1 ResourceParam

New in JDF 1.1

Table 8.238: ResourceParam Element

NAME	DATA TYPE	DESCRIPTION
DefaultID ?	NMTOKEN	***Resource***/@*ID* of the default resource. If missing, it is assumed that the file specified by @*DefaultJDF* contains only a **JDF** resource element, not a complete **JDF**.
DefaultJDF ?	URL	Link to a **JDF** resource that defines preset values. Defaults to the @*DefaultJDF* specified in ***ResourceDefinitionParams***.
DefaultPriority ?	enumeration	Defines whether preset values of the application or of the resource specified in @*DefaultJDF* have priority. **Default value is from:** parent's ***ResourceDefinitionParams***/@*DefaultPriority*. **Allowed values are:** Application DefaultJDF

8.129 RingBindingParams

This resource describes the details of the **RingBinding** process.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **RingBinding**

Table 8.239: RingBindingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BinderColor ?	NamedColor	Color of the ring binder.
BinderColorDetails ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If @*BinderColorDetails* is supplied, @*BinderColor* SHOULD also be supplied.
BinderMaterial ?	NMTOKEN	The following describe **RingBinding**binder materials used. **Values include:** Cardboard – Cardboard with no covering. ClothCovered – Cardboard with cloth covering. PVC – Solid PVC. PVCCovered – Cardboard with PVC covering.
BinderName ?	string	The name of the binder manufacturer and the name of the specific item.
RingDiameter ?	double	Diameter of the rings, in points.
RingMechanic ?	boolean	If "true", a hand lever is available for opening.

Table 8.239: RingBindingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
RingShape ?	NMTOKEN	**RingBinding** values: **Values include:** Round Oval D-shape SlantD
RingSystem ? Deprecated in JDF 1.1	enumeration	Ring binding systems **Allowed values are:** 2HoleEuro – In Europe 3HoleUS – In North America 4HoleEuro – In Europe **Deprecation note:** Starting with **JDF** 1.2, use the value implied by ***HoleMaking-Params***/@*HoleType*.
RivetsExposed ?	boolean	The following **RingBinding** choice describes mounting of ring mechanism in binder case. If "true", the heads of the rivets are visible on the exterior of the binder. If "false", the binder covering material covers the rivet heads.
SpineColor ?	enumeration	Color of the binders spine. **Allowed value is from:** ▸ NamedColor.
SpineColorDetails ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If @*SpineColorDetails* is supplied, @*SpineColor* SHOULD also be supplied.
SpineWidth ?	double	The spine width is determined by the final height of the block of Sheets to be bound.
ViewBinder ?	NMTOKEN	For **RingBinding** clear vinyl outer-wrap types on top of a colored base wrap: **Values include:** Embedded – Printed material is embedded by sealing between the colored and clear vinyl layers during the binder manufacturing. Pocket – Binder is designed so that printed material can be inserted between the color and clear vinyl layers after the binder is manufactured.
HoleMakingParams ? New in JDF 1.2	refelement	Details of the holes in **RingBinding**.

8.130 RollStand

New in JDF 1.2

Resource Properties

Resource Class: Handling

Input of Processes: **PrintRolling**

Table 8.240: RollStand Resource

NAME	DATA TYPE	DESCRIPTION
MaxDiameter ?	double	Maximal allowed diameter of the input component print Roll.
MaxWidth ?	double	Maximal allowed width of the rolled input components.
Device ?	refelement	Further details of the ***RollStand***.

8.131 RunList

A ***RunList*** defines one or more printable logical documents or document sets that MAY be defined in one or more external physical PDL or image files. It retains the properties of the original documents, e.g. the pages of a set of documents with ordered pages that is described by a ***RunList*** are ordered.

RunList allows structuring of multiple pages into documents. Multiple documents that have a joint context may be grouped into sets. The following table provides a mapping of pages, documents and sets for common PDL types.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***ArtDeliveryIntent***/***ArtDelivery***, ***DigitalMedia***, ***Layout***/***PageCondition***, ***Layout***/***SheetCondition***, ***PreflightReport***

Example Partition: "PartVersion", "Run", "RunPage", "RunSet", "Separation", "WebProduct"

Input of Processes: **AssetListCreation**, **ColorCorrection**, **ColorSpaceConversion**, **ContoneCalibration**, **CylinderLayoutPreparation**, **DigitalDelivery**, **DigitalPrinting**, **ImageReplacement**, **ImageSetting**, **Imposition**, **Interpreting**, **LayoutPreparation**, **LayoutShifting**, **PageAssigning**, **PDFToPSConversion**, **PDLCreation**, **Preflight**, **PreviewGeneration**, **PSToPDFConversion**, **RasterReading**, **Rendering**, **Screening**, **Separation**, **Stripping**, **Tiling**, **Trapping**

Output of Processes: **AssetListCreation**, **ColorCorrection**, **ColorSpaceConversion**, **ContoneCalibration**, **DigitalDelivery**, **ImageReplacement**, **Imposition**, **Interpreting**, **LayoutElementProduction**, **LayoutPreparation**, **LayoutShifting**, **PageAssigning**, **PDFToPSConversion**, **PDLCreation**, **PSToPDFConversion**, **RasterReading**, **Rendering**, **Scanning**, **Screening**, **Separation**, **Stripping**, **Tiling**, **Trapping**

Table 8.241: RunList Resource (Sheet 1 of 5)

NAME	DATA TYPE	DESCRIPTION
Automation ? New in JDF 1.5	enumeration	Identifies dynamic and static ***RunList*** elements. The Structure of *@PartIDKey* generation for automated imposition is defined in detail in: ▸ Section 6.3.18.3 Execution Model for Automated Imposition. This structure SHALL be retained in the ***RunList*** description. **Allowed value is from:** ▸ Automation. **Note:** If *@Automation*="Dynamic" and *@PipeID* is also present, details are specified in **JMF** pipe messages. See ▸ Section 4.3.3.1 Dynamic Pipes.
ComponentGranularity = "Document" New in JDF 1.2 Deprecated in JDF 1.4	enumeration	Specifies which grouping of input ***LayoutElement*** PDL pages define the equivalent of an individual output ***Component*** instance for processing in a multi-document print Job (e.g., in a variable data Job). For instance, all pages defined between end-of-set markers would be stitched in a combined process node with **DigitalPrinting** and **Stitching** processes if *@ComponentGranularity* = "Set". **Allowed values are:** All – The complete ***RunList***, regardless of document or set breaks defines a new ***Component***. BundleItem – An implicit PDL-defined document break or an explicit *@EndOfBundleItem* defines a new ***Component***. Document – An implicit PDL-defined document break or an explicit *@EndOfDocument* defines a new ***Component***. Page – Each page in the ***RunList*** defines a new ***Component***. Set – Each set as defined by an implicit PDL-defined set break or an explicit *@EndOfSet* defines a new ***Component***.
Directory ?	URL	Defines a directory where the files that are associated with this ***RunList*** are to be copied to or from. If *@Directory* is specified, it SHALL be an Absolute URI ▸ [RFC3986] that implicitly also specifies a Base URI which is used to resolve any relative URL of ***RunList***. See ▸ Appendix J Resolving Directory URL References and ▸ [FileURL] for examples.
DocCopies = "1" New in JDF 1.1	integer	Number of Instance Document copies that this ***RunList*** represents. Specifying *@DocCopies* is equivalent to repeating the sequence of ***RunList*** leaves between *@EndOfDocument* = "true" for a total of *@DocCopies* times. If *@DocCopies* is > 1 for an automated imposition job, the imposition engine places the equivalent *@DocCopies* attribute into the ***RunList*** (Surface) resource generated by the **Imposition** process. An exception is cut-and-stack imposition, where *@DocCopies* is applied by the imposition engine itself, and not placed into the ***RunList***(Surface). **Note:** It is illegal to specify *@DocCopies* with different values of various leaves of a ***RunList*** representing the same Instance Document.
DocNames ?	NameRange-List	A list of named documents in a multi-document file that supports named access to individual documents. The *@DocNames* defaults to all documents. If *@DocNames* occurs in the ***RunList***, *@Docs* is ignored if it is also present.

Table 8.241: RunList Resource (Sheet 2 of 5)

NAME	DATA TYPE	DESCRIPTION
Docs ?	NameRange-List	Zero-based list of document indices in a multi-document file specified by the ***LayoutElement*** resource
EndOfBundleItem ? New in JDF 1.2	boolean	If "true", the last page in the ***RunList*** is the last page of a ***BundleItem***. The implied default value of @*EndOfBundleItem* = "false", except for the last ***RunList*** Partition, which always has an implied default value of @*EndOfBundleItem* = "true". **Modification note:** Starting with **JDF** 1.4, this attribute no longer depends on the deprecated @*ComponentGranularity*.
EndOfDocument ?	boolean	If "true", the last page in the ***RunList*** is the last page of an Instance Document. Precisely how changes in the Instance Document are handled is defined in the ***InsertSheet*** resource. If the ***RunList*** references a PDL that supports internal Instance Documents, @*EndOfDocument* SHALL be the value that is defined in the PDL. The implied default value of @*EndOfDocument*="false", except for the last ***RunList*** partition leaf, which always has an implied default value of @*EndOfDocument* = "true".
EndOfSet ? New in JDF 1.1	boolean	If "true", the last page in the ***RunList*** is the last page of a set of Instance Documents. Precisely how Instance Document boundaries are handled is defined in the ***InsertSheet*** resource. If the ***RunList*** references a PDL that supports internal sets, @*EndOfSet* SHALL be the value that is defined in the PDL. The implied default value of @*EndOfSet*="false", except for the last ***RunList*** partition leaf, which always has an implied default value of @*EndOfSet*="true".
FinishedPages ? New in JDF 1.6	integer	Number of finished page surfaces that one PDL page of this ***RunList*** refers to. This attribute SHOULD be used when cover spreads or imposed sheets that contain more than one reader page per PDL page are provided.
FirstPage ?	integer	First page in the document that is described by this ***RunList***. This attribute is generally used to describe preseparated files.
IgnoreContext ? New in JDF 1.4	enumerations	Specifies the @*PartIDKeys* values that do not affect the context in which this ***RunList*** is processed. Typically used when the ***ResourceLink*** is Partitioned to re-order a content ***RunList***. For the keys specified in this list, processing the ***RunList*** SHALL operate as if the identified parts represent the entire ***RunList***. If Partition Keys are not specified, processing the ***RunList*** SHALL operate as if the entire ***RunList*** resource was processed, and all results removed except for those identified by the ***ResourceLink*** (e.g., for reprinting or recreating sheets with processing order-sensitive content - @*SheetIndex* has whatever value it would have had if sheets were generated using the entire, original ***RunList***). See example just below this table.
IsPage = "true"	boolean	If "true", the individual ***RunList*** resource defines one or more page slots (e.g., for filling ***PlacedObject*** elements). If "false", the first parent Partitioned ***RunList*** resource with @*IsPage* = "true" defines the page level. In general, @*IsPage* = "false" for separations of a preseparated ***RunList***.
LogicalPage ? Modified in JDF 1.1	integer	The logical page number of the first page in a ***RunList***. This attribute MAY be used to retain logical page indices when a Partitioned ***RunList*** is spawned. It defaults to "1" plus the last page of the previous sibling ***RunList*** Partition. If the ***RunList*** resource is the first Partition, @*LogicalPage* defaults to "0". Note that is an error to specify @*LogicalPage* to be less than the number of previously defined logical pages in the same Partition, since this defines overlapping pages within the ***RunList*** Partition.
NDoc ? New in JDF 1.1 Deprecated in JDF 1.2	integer	Total number of Instance Documents that are defined by the ***RunList***. If @*NDoc* is not specified, it defaults to all Instance Documents in the Partitioned ***RunList*** elements that make up the ***RunList***. In **JDF** 1.2 and beyond, only @*Docs* is supported.

Table 8.241: RunList Resource (Sheet 3 of 5)

NAME	DATA TYPE	DESCRIPTION
NPage ?	integer	Total number of pages (placed object slots or ***RunList*** elements with @*IsPage* = "true") that are defined by the ***RunList***. If @*NPage* is not specified, it defaults to all pages in the Partitioned ***RunList*** elements that make up the ***RunList***. If the ***RunList*** describes multiple Instance Documents or Document Sets, @*NPage* refers to the total number of pages in all Instance Documents and sets. A ***RunList*** with @*NPage* specified always refers to @*NPage* pages, regardless of the number of pages of the referenced PDL. If @*NPage* is not specified and no content is referenced, the ***RunList*** contains exactly one page.
NSet ? New in JDF 1.1 Deprecated in JDF 1.2	integer	Total number of Instance Document Sets that are defined by the ***RunList***. If @*NSet* is not specified, it defaults to all Instance Document Sets in the Partitioned ***RunList*** elements that make up the ***RunList***. In **JDF** 1.2 and beyond, only @*Sets* is supported.
PageCopies = "1" New in JDF 1.1	integer	Number of page copies that this ***RunList*** represents. Specifying @*PageCopies* is equivalent to repeating the ***RunList*** leaves representing each page for a total of @*PageCopies* times (e.g., a multiple represented by the value of @*PageCopies*). Note that pages specified by @*PageCopies* are always assumed uncollated when calculating the index in the logical ***RunList*** (e.g., @*PageCopies* = "2" would result in a logical page sequence of 0 0 1 1 2 2, etc.).
PageListIndex ? New in JDF 1.2	IntegerRangeList	List of the indices of the ***PageData*** elements of the ***PageList*** specified in this ***RunList***. If not specified, the complete @*PageListIndex* specified in this ***RunList*** is applied.
PageNames ?	NameRangeList	A list of named pages in a multi-page file that supports named access to individual pages. The @*PageNames* defaults to all pages. If @*PageNames* is specified, then @*FirstPage*, @*NPage*, @*SkipPage* and @*Pages* SHALL all be ignored if any is specified.
Pages ?	IntegerRangeList	Zero-based list of indices in the documents specified by the ***LayoutElement*** resource and the @*Docs*, @*DocNames*, @*Sets* and @*SetNames* attributes. The @*Pages* need not be in document order. If @*Pages* is specified, @*FirstPage* and @*SkipPage* SHALL be ignored. If none of @*Pages*, @*FirstPage*, @*NPage*, @*PageNames* or @*SkipPage* is specified, all pages (i.e., "0 ~ -1") referred to by the ***RunList*** are selected. **Modification note:** Before **JDF** 1.4, ***LayoutElement*** appeared in place of ***RunList*** in the preceding sentence.
RunTag ? New in JDF 1.1	NMTOKEN	Tag of a Partition of a resource other than the ***RunList*** which is Partitioned by @*RunTags*. The Partition matches if any of the entries in the @*RunTags* list matches @*RunTag*. Multiple entries in a ***RunList*** MAY have the same @*RunTag*. If the ***RunList*** references a PDL that supports internal labels, @*RunTag* MAY be implied from the PDL.
SetCopies = "1" New in JDF 1.1	integer	Number of Instance Document Set copies that this ***RunList*** represents. Specifying @*SetCopies* is equivalent to repeating the sequence of ***RunList*** leaves between @*EndOfSet* = "true" for a total of @*SetCopies* times. If @*SetCopies* is > 1 for an automated imposition Job, the imposition engine places the equivalent @*SetCopies* attribute into the ***RunList*** (Surface) resource generated by the **Imposition** process. An exception is cut-and-stack imposition, where @*SetCopies* is applied by the imposition engine itself, and not placed into the ***RunList*** (Surface). **Note:** It is illegal to specify @*SetCopies* with different values of various leaves of a ***RunList*** representing the same Instance Document.
SetNames ? New in JDF 1.1	NameRangeList	A list of named Document Sets in a multi-Document Set file that supports named access to individual documents. The @*SetNames* defaults to all Document Sets specified by @*Sets*. If @*SetNames* occurs in the ***RunList***, @*Sets* is ignored if it is also present. @*SetNames* is only valid if ***LayoutElement***/@*ElementType* = "MultiSet".

Table 8.241: RunList Resource (Sheet 4 of 5)

NAME	DATA TYPE	DESCRIPTION
Sets ? New in JDF 1.1	IntegerRangeList	Zero-based list of Document Set indices in a multi-Document Sets file specified by the ***LayoutElement*** resource. If not present, all Document Sets are selected. @*Sets* is only valid if ***LayoutElement***/@*ElementType* = "MultiSet".
SheetSides ? New in JDF 1.4	enumeration	Specifies the binding of surfaces referenced by this ***RunList*** to sheets. SHALL only be specified in ***RunList*** (Surface). **Allowed values are:** Front – all surfaces referenced from a ***RunList*** leaf partition describe one or more front sides of successive sheets, with implicit back blank sides. Back – all surfaces referenced from a ***RunList*** leaf partition describe one or more back sides of successive sheets, with implicit front blank sides. FrontBack – all surfaces referenced from a ***RunList*** leaf partition describe a succession of sheets, where for each sheet a front is followed by a back surface. BackFront – all surfaces referenced from a ***RunList*** leaf partition describe a succession of sheets, where for each sheet a back is followed by a front surface.
SkipPage ?	integer	Used when the ***RunList*** comprises every Nth page of the file. @*SkipPage* indicates the number of pages to be skipped between each of the pages that comprise the ***RunList*** resource. This is generally used to describe preseparated files, or to select only even or odd pages. Note that @*SkipPage* is, therefore, 3 (4 Separations -> skip 3) in a CMYK separated file.
Sorted ?	boolean	Specifies whether the elements in the ***RunList*** are sorted in the document reader order.
ByteMap ? Modified in JDF 1.2	refelement	Describes the page or stream of pages. At most one of ***ByteMap*** , ***InterpretedPDLData*** or ***LayoutElement*** SHALL be specified. If none of ***ByteMap***, ***InterpretedPDLData*** or ***LayoutElement*** are specified, the ***RunList*** specifies empty content.
Disposition ?	element	Indicates what the device SHOULD do with the file when the process that uses this resource completes. If not specified, the file specified by this ***RunList*** is retained indefinitely. ***RunList***/***LayoutElement***/***FileSpec***/***Disposition*** takes precedence over ***RunList***/***Disposition***. **Modification note:** Starting with **JDF** 1.4, "this ***RunList***" above replaces "this ***FileSpec***".
DynamicInput * Deprecated in JDF 1.4	element	Replacement text for a ***DynamicField*** element. This information defines the contents of a dynamic mark on the automated page layout (see ▸ Section 8.84.14.1 Dynamic Marks). The mark SHALL be filled using information from the document ***RunList*** (e.g., the bar code of the recipient). This information varies with the document content. ***DynamicInput*** elements have one OPTIONAL @*Name* attribute that, when linked to the @*ReplaceField* attribute of the ***DynamicField*** element, defines the string that SHALL be replaced. **Deprecation note:** Starting with **JDF** 1.4, metadata should be extracted from the PDL itself or from other sources, but not from the ***RunList***. ***DynamicInput*** was designed to associates metadata with ***RunList*** elements.
InsertSheet *	element	Describes how sheets and surfaces are to be completed and OPTIONAL media which MAY be inserted at the beginning or end of this ***RunList*** resource.
InterpretedPDLData ? New in JDF 1.2	refelement	Represents the results of the PDL interpretation process. At most one of ***ByteMap***, ***InterpretedPDLData*** or ***LayoutElement*** SHALL be specified. If none of ***ByteMap***, ***InterpretedPDLData*** or ***LayoutElement*** are specified, the ***RunList*** specifies empty content.
LayoutElement ? Modified in JDF 1.2	refelement	Describes the document, page or image. At most one of ***ByteMap***, ***InterpretedPDLData*** or ***LayoutElement*** SHALL be specified. If none of ***ByteMap***, ***InterpretedPDLData*** or ***LayoutElement*** are specified, the ***RunList*** specifies empty content.
MetadataMap *	element	Describes the mapping of Metadata in a ***RunList*** to @*PartIDKeys*.

Table 8.241: RunList Resource (Sheet 5 of 5)

NAME	DATA TYPE	DESCRIPTION
PageList ?	refelement	Specification of page metadata for pages described by this ***RunList***.

8.131.1 Pages, Documents and Sets for common PDL types

Table 8.242: Pages, Documents and Sets for common PDL types

PDL	PAGES	DOCUMENTS	SETS	REMARKS
Post-Script	-	-	PostScript is a single document PDL	
PDF	Page in pages tree	-	-	Regular PDF including PDF/X is a single document PDL.
PDF/VT	Page in pages tree	Any DPart Descendant below the Set	And DPart record as defined by RecordLevel	A Record as defined by Record-Level SHALL be mapped to a Set.
PPML	PAGE elements	DOCUMENT elements	DOCUMENT_SET/JOB elements	

Example 8.41: Marks and Reordering of Content using RunList/@IgnoreContext

New in JDF 1.4

Assume that a VDP job consists of sets where each set contains a cover letter, brochure, and postcard document types. Production needs all of each document type for all sets printed first, and the imposition includes dynamic marks where some of the marking uses @*SheetIndex*. The ***RunListLink*** parameterizes the processing such that all Cover Letter sheets for all sets are processed first, followed by the brochure sheets for all sets, and finally, the Postcard sheets for all sets. The ***RunList*** then specifies @*IgnoreContext* = "SheetIndex", which forces the @*SheetIndex* to be calculated in the order in which sheets are produced by the processing of the reordered "virtual" ***RunList***.

```
<ResourcePool>
    <RunList Class="Parameter" ID="MyVDPRunList" Status="Available"
           PartIDKeys="DocTags" IgnoreContext="SheetIndex" >
           <!-- additional attributes and elements -->
       <RunList DocTags="CoverLetter"/>
       <RunList DocTags="Brochure"/>
       <RunList DocTags="Postcard"/>
    </RunList>
</ResourcePool>
<ResourceLinkPool>
    <RunListLink Usage="Input" rRef="MyVDPRunList" >
       <Part DocTags="CoverLetter"/>
       <Part DocTags="Brochure"/>
       <Part DocTags="Postcard"/>
    </RunListLink>
</ResourceLinkPool>
```

To enable later reprinting of part of the ***RunList***, the ***RunList*** then might also specify a ***MetadataMap*** element that extracts the value of a RecordNumber metadata key and assigns the value to @*Metadata0*. Subsequently, if record # 12 needs re-printing, the ***RunListLink*** can be modified to appear as:

```
<RunListLink Usage="Input" rRef="MyVDPRunList" ProcessUsage="Document">
    <Part DocTags="CoverLetter" Metadata0="12"/>
    <Part DocTags="Brochure" Metadata0="12"/>
    <Part DocTags="Postcard" Metadata0="12"/>
</RunListLink>
```

8.131.2 DynamicInput

Deprecated in JDF 1.4

See ▸ Section N.7.22.1 DynamicInput for details of this deprecated Parameter subelement.

Example 8.42: RunList: Unstructured Single-File RunList

The following examples illustrate how a ***RunList*** can be structured using Partitioning mechanisms. Note that the Partitioning of a ***RunList*** often generates the values necessary to evaluate the Partitioning of other resources (e.g., the @*RunIndex* into the ***RunList***). Thus, the order in which the ***RunList*** elements appear in the XML document is significant. Note that the @*Run* Partition Key has a string value, which MAY be non-numeric. Below is an example of simple unstructured single-file ***RunList***. This example specifies all pages contained in "/in/colortest.pdf".

```
<RunList Class="Parameter" ID="Link0003" Pages="0 ~ -1" Status="Available">
    <LayoutElement>
        <FileSpec URL="File:///in/colortest.pdf"/>
    </LayoutElement>
</RunList>
```

Example 8.43: RunList: Multi-File Unseparated RunList

Example of simple multi-file unseparated ***RunList*** using ***RunList***/@*Directory*. This example specifies all pages contained in File1.pdf and File2.pdf, which are located in the directory "///Dir/" that is specified in ***RunList***/@*Directory*.

```
<RunList Class="Parameter" Directory="File:///Dir/" ID="Link0003"
        PartIDKeys="Run" Status="Available">
    <RunList Pages="0 ~ -1" Run="1">
        <LayoutElement>
            <FileSpec URL="File1.pdf"/>
        </LayoutElement>
    </RunList>
    <RunList Pages="0 ~ -1" Run="2">
        <LayoutElement>
            <FileSpec URL="File2.pdf"/>
        </LayoutElement>
    </RunList>
</RunList>
```

Example 8.44: RunList: Multi-File Unseparated RunList with Spawning

Example of simple multi-file unseparated ***RunList*** with independent spawning. This example specifies the first five pages contained in File1.pdf and File2.pdf. File2.pdf has been spawned and is being processed individually.

```
<RunList Class="Parameter" ID="Link0003" PartIDKeys="Run" Status="Available">
    <RunList Pages="0 ~ 4" Run="1">
        <LayoutElement>
            <FileSpec URL="File:///File1.pdf"/>
        </LayoutElement>
    </RunList>
    <RunList Pages="0 ~ -1" Run="2" SpawnStatus="SpawnedRW">
        <LayoutElement>
            <FileSpec URL="File:///File2.pdf"/>
        </LayoutElement>
    </RunList>
</RunList>
```

Example 8.45: RunList: Spawned RunList

This is the corresponding spawned ***RunList***. Note the @*LogicalPage* attribute, which specifies the number of skipped pages.

```
<RunList Class="Parameter" ID="Link0003" LogicalPage="5" Pages="0 ~ -1"
        PartIDKeys="Run" Status="Available">
    <RunList Run="2">
        <LayoutElement>
            <FileSpec URL="File:///File2.pdf"/>
        </LayoutElement>
    </RunList>
</RunList>
```

Example 8.46: RunList: Multi-File Separated RunList

This example specifies all pages contained in Presep.pdf and following that, pages 1, 3 and 5 of each preseparated file.

```
<RunList Class="Parameter" ID="Link0003" PartIDKeys="Run Separation"
       Status="Available">
    <RunList Run="1" SkipPage="3">
       <LayoutElement>
          <FileSpec URL="File:///Presep.pdf"/>
       </LayoutElement>
       <RunList FirstPage="0" IsPage="false" Separation="Cyan"/>
       <RunList FirstPage="1" IsPage="false" Separation="Magenta"/>
       <RunList FirstPage="2" IsPage="false" Separation="Yellow"/>
       <RunList FirstPage="3" IsPage="false" Separation="Black"/>
    </RunList>
    <RunList IsPage="true" Pages="1 3 5" Run="2">
       <RunList IsPage="false" Separation="Cyan">
          <LayoutElement>
             <FileSpec URL="File:///Cyan2.pdf"/>
          </LayoutElement>
       </RunList>
       <RunList IsPage="false" Separation="Magenta">
          <LayoutElement>
             <FileSpec URL="File:///Magenta2.pdf"/>
          </LayoutElement>
       </RunList>
       <RunList IsPage="false" Separation="Yellow">
          <LayoutElement>
             <FileSpec URL="File:///Yellow2.pdf"/>
          </LayoutElement>
       </RunList>
       <RunList IsPage="false" Separation="Black">
          <LayoutElement>
             <FileSpec URL="File:///Black2.pdf"/>
          </LayoutElement>
       </RunList>
    </RunList>
</RunList>
```

8.132 SaddleStitchingParams

Deprecated in JDF 1.1

See ▸ Section N.7.23 SaddleStitchingParams for details of this deprecated resource.

8.133 ScanParams

This resource provides the parameters for the **Scanning** process.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***ArtDeliveryIntent***/***ArtDelivery***

Example Partition: "RunIndex"

Input of Processes: **Scanning**

Table 8.243: ScanParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BitDepth	integer	Bit depth of a one-color separation.
CompressionFilter ?	enumeration	Specifies the compression filter to be used. **Allowed values are:** CCITTFaxEncode – Used to select CCITT Group 3 or 4 facsimile encoding. DCTEncode – Used to select JPEG compression. FlateEncode – Used to select Zip compression. WaveletEncode – Used to select Wavelet compression. JBIG2Encode – Used to select JBIG2 monochrome compression.

Table 8.243: ScanParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
DCTQuality ?	double	A value between 0 and 1 that indicates "how much" the process SHALL compress images. 0.0 means "do as loss-less compression as possible." 1.0 means "do the maximum compression possible."
InputBox ?	rectangle	Rectangle that describes the image section to be scanned, in points. The origin of the coordinate system is the lower left corner of the physical item to be scanned.
Magnification = "1 1"	XYPair	Size of the output/size of the input for each dimension.
MountID ?	string	ID of the drum or other mounting device upon which the media SHALL be mounted.
Mounting ?	enumeration	Specifies how to mount originals. **Allowed values are:** Unfixed – Original lies unfixed on the scanner tray/drum. Fixed – Original is fixed on the scanner tray/drum with transparent tape. Wet – Original is put in gel or oil and fixed on the scanner tray/drum. Registered – Original is fixed with registration holes. This value is used for copy dot scans.
OutputColorSpace	enumeration	Color space of the output images. **Allowed values are:** LAB RGB CMYK GrayScale
OutputResolution	XYPair	X and Y resolution of the output bitmap, in dpi.
OutputSize ?	XYPair	X and Y dimension of the intended output image, in points.
SplitDocuments ?	integer	A number representing how many images are scanned before a new file is created.
FileSpec (CorrectionProfile) ?	refelement	A ***FileSpec*** resource pointing to an ICC profile that describes color corrections.
FileSpec (ScanProfile) ?	refelement	A ***FileSpec*** resource pointing to an ICC profile that describes the scanner.
FileSpec (TargetProfile) ?	refelement	A ***FileSpec*** resource pointing to an ICC profile that defines the target output device for a device specific scan (e.g., the profile of a CMYK press).

8.134 ScavengerArea

New in JDF 1.1

This resource describes a scavenger area for removing excess ink from printed Sheets. It is defined within a ***MarkObject*** of a surface.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***Layout***/***MarkObject***

Table 8.244: ScavengerArea Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Center	XYPair	Position of the center of the scavenger area in the coordinates of the ***MarkObject*** that contains this mark.
Rotation ?	double	Rotation in degrees. Positive graduation figures indicate counter-clockwise rotation; negative figures indicate clockwise rotation.

Table 8.244: ScavengerArea Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Size	XYPair	Size of the scavenger area.
SeparationSpec * Modified in JDF 1.2	element	Set of separations to which the scavenger area is bound.

8.135 ScreeningParams

This resource specifies the parameter of the **Screening** process. Since screening is, in most cases, very OEM specific, the following parameters are generic enough that they can be mapped onto a number of OEM controls.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***ContactCopyParams***, ***ContentList***/***ContentData***, ***ExposedMedia***, ***LayoutElement***, ***PageList***, ***PageList***/***PageData***, ***RunList***

Intent Pairing: ***ProofingIntent***, ***ScreeningIntent***

Example Partition: "Separation", "SheetName", "Side", "SignatureName"

Input of Processes: **ContoneCalibration**, **Screening**

Table 8.245: ScreeningParams Resource

NAME	DATA TYPE	DESCRIPTION
AbortJobWhenScreenMatchingFails ? Deprecated in JDF 1.2	boolean	Specifies what happens when the device can not fulfill the screening requests. If "true", it flushes the Job. If "false", it ignores matching errors using the default screening. Use @*SettingsPolicy* in **JDF** 1.2 and beyond.
IgnoreSourceFile = "true"	boolean	Specifies whether to ignore the screen settings (e.g., setscreen, setcolorscreen and sethalftone) specified in the source files. Note that in some cases, halftones are used to create patterns. In these cases, the halftone in the source PDL file will not be overridden.
ScreenSelector *	element	List of screen selectors. A screen selector is included for each separation, including a default specification. ***ScreenSelector*** SHALL contain the complete set of Parameters for a given screening operation. For instance, it is invalid to specify one ***ScreenSelector*** for a given @*ObjectTags* and another ***ScreenSelector*** for a given @*SourceObjects*.

8.136 SeparationControlParams

This resource provides the controls needed to separate composite color files.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***ProofingIntent***, ***ScreeningIntent***

Input of Processes: **Separation**

Table 8.246: SeparationControlParams Resource

NAME	DATA TYPE	DESCRIPTION
AutomatedOverPrintParams ?	element	Controls for overprint substitutions. The default case is that no automated overprint generation is used.
TransferFunctionControl ?	refelement	Controls whether the device performs transfer functions and what values are used when doing so.

8.137 Shape

Resource Properties

Resource Class: Parameter

Resource referenced by: *ShapeCuttingParams*, *ShapeDef*

Table 8.247: Shape Resource

NAME	DATA TYPE	DESCRIPTION
CutBox ?	rectangle	Specification of a rectangular window.
CutOut = "false"	boolean	If "true", the inside of a specified shape will be removed. If "false", the outside of a specified shape will be removed. An example of an inside shape is a window. An example of an outside shape is a shaped greeting card.
CutPath ?	PDFPath	Specification of a complex path. This MAY be an open path in the case of a single line.
CutType = "Cut" Deprecated in JDF 1.4	enumeration	Type of cut or perforation used. **Allowed values are:** Cut – Full cut. Perforate – Interrupted perforation that does not span the entire Sheet
DDESCutType = "101" New in JDF 1.4	integer	Type of cut or perforation used. **Values include:** a number between "0" and "999" corresponding to a line type as defined in DDES3 standard (ANSI® IT8.6-2002). **Note:** The default value 101 corresponds to a cut line.
Material ?	string	Transparent material that fills a shape (e.g., an envelope window) that was cut out when @*CutOut* = "true".
ShapeDepth ?	double	Depth of the shape cut, measured in microns [µm]. If not specified, the shape is completely cut.
ShapeType	enumeration	Describes any precision cutting other than hole making. **Allowed values are:** Path Rectangular Round RoundedRectangle – Rectangle with rounded corners. New in JDF 1.3
StationName ? New in JDF 1.3 Deprecated in JDF 1.4	string	The name of the 1-up design in the die layout. Used to match ***DieLayout***/***Station*** elements with ***Shape*** elements.
TeethPerDimension ?	double	Number of teeth in a given perforation extent, in teeth/point. MicroPerforation is defined by specifying a large number of teeth (n > 1000).

8.138 ShapeCuttingParams

New in JDF 1.1

ShapeCuttingParams defines the details of the **ShapeCutting** process.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***ShapeCuttingIntent***

Input of Processes: **ShapeCutting**

Table 8.248: ShapeCuttingParams Resource

NAME	DATA TYPE	DESCRIPTION
DeliveryMode ? New in JDF 1.3	enumeration	**Allowed values are:** FullSheet – The output of the die-cutter are complete Sheets. The blanks are kept in place with nicks. Front waste (gripper margin) has not been removed. RemoveGripperMargin – The output of the die-cutter are complete Sheets. The blanks are kept in place with nicks. Front waste (gripper margin) has been removed. SeparateBlanks – The output of the die-cutter are blanks that have been removed from the Sheets.
ModuleIndex ? New in JDF 1.4	integer	Index of the shape-cutting Module in the Press. See ***ConventionalPrintingParams***. In a combined process, all modules of the device, including press modules, finishing modules and varnishing modules are counted to calculate @*ModuleIndex*.
SheetLay ? New in JDF 1.3	enumeration	Lay of input media. Reference edge of where paper is placed in the feeder. **Allowed value is from:** ▸ SheetLay.
DieLayout ? New in JDF 1.3	refelement	A resource containing the reference of an external file describing the cutting and other paths.
Shape *	refelement	Details of each individual cut shape

8.139 ShapeDef

New in JDF 1.4

A structural design describing a 2D surface with paths that describe different finishing operations like cutting, creasing, perforation, etc. In the case of box production this resource is a description of the unprinted blank box as it will be available after die cutting and blanking and before folding. A ***ShapeDef*** is defined either by an external file (***FileSpec***) describing the structural design or a collection of PDFPaths contained in Shape elements. In case this description is stored in a file, the format of this file may be a vendor specific format, a standard ▸ [DDES3], or less well specified but commonly used formats like CFF2 or DXF or even a PDF or EPS file.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***DieLayout***/***Station***, ***LayoutElementProductionParams***

Input of Processes: **DieLayoutProduction**

Output of Processes: **ShapeDefProduction**

Table 8.249: ShapeDef Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Area ?	double	The net area of the shape after cutting. (m2)
CutBox ?	rectangle	A rectangle describing the bounding box of all cut lines. This is sometimes referred to as the knife to knife dimensions of the BlankBox. This attribute is usually only valid after the generation of the structural design.
Dimensions ?	shape	Width *x*, height *y* and depth *z* coordinates of the open 3D shape. For a box, these are the outer dimensions of the opened and potentially filled box (e.g., for palletizing of the final products). **Note**: Compare with @*FlatDimensions*.
FlatDimensions ? New in JDF 1.5	shape	Width *x*, height *y* and depth *z* coordinates of the flat 3D shape. For a box, these are the outer dimensions of the glued flat box (e.g., for palletizing of the boxes prior to filling). This corresponds to the Dimensions of the output component of the **BoxFolding** process. Note: Compare with @*Dimensions*.

Table 8.249: ShapeDef Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
FluteDirection ?	enumeration	Intended direction of the flute for this design in the coordinate system defined by @*CutBox*. This information SHALL be taken into account by the **DieLayoutProduction** process to give the ***ShapeDef*** the correct orientation on the ***Media***. **Allowed values are:** XDirection – Along the X-axis of the @*CutBox* coordinate system. YDirection – Along the Y-axis of the @*CutBox* coordinate system. Both – Both orientations are acceptable.
GrainDirection ?	enumeration	Intended direction of the grain for this design in the coordinate system defined by @*CutBox*. This information SHALL be taken into account by the **DieLayoutProduction** process to give the ***ShapeDef*** the correct orientation on the ***Media***. **Allowed values are:** XDirection – Along the X-axis of the @*CutBox* coordinate system. YDirection – Along the Y-axis of the @*CutBox* coordinate system. Both – Both orientations are acceptable.
MediaSide ?	enumeration	Determines the printing side for which the ***DieLayout*** is made. Front corresponds to the outside of a box, Back corresponds to the inside of a box. **Allowed value is from:** ▸ Side **Note:** Folding carton is usually cut from the outside (Front), corrugated from the inside (Back).
ResourceWeight ?	double	The weight of the shape after cutting (g).
ColorPool ? New in JDF 1.5	refelement	The ***ColorPool*** that SHOULD contain names and further details of the separations used in ***CutLines***.
CutLines ? New in JDF 1.5	element	Selects the die line separations from the file referenced by ***FileSpec***. Additional details of the usage of the separations MAY be specified in the respective ***ColorPool***/***Color*** elements.
FileSpec ?	refelement	The ***FileSpec*** of the structural design file. The format of this file may be a vendor specific format, a standard ▸ [DDES3], less well specified but commonly used formats like CFF2 or DXF or even a PDF or EPS file. ***FileSpec*** and ***Shape*** are mutually exclusive
Media ?	refelement	***Media*** for which this structural design was intended for. The ***Media*** description defines important design parameters as the type of ***Media***, thickness, inside loss, outside gain, etc. ***Media***/@*GrainDirection* and ***Media***/@*FluteDirection* do not have any significance.
Shape *	refelement	The shape is defined by a collection of ***Shape*** elements. ***Shape*** and ***FileSpec*** are mutually exclusive.

8.139.1 CutLines

New in JDF 1.5

Table 8.250: CutLines Element

NAME	DATA TYPE	DESCRIPTION
SeparationSpec *	element	Separation name of a die line

8.140 ShapeDefProductionParams

New in JDF 1.4

Parameters for the structural design.

Resource Properties

Resource Class: Parameter

Input of Processes: ShapeDefProduction

Table 8.251: ShapeDefProductionParams Resource

NAME	DATA TYPE	DESCRIPTION
ObjectModel *	element	A 3D model of the objects that need to be packed.
ShapeTemplate ?	element	A structural template sometimes called a parametric structural design. Given a set of parametric values a structural template can be instantiated to an actual structural design.

8.140.1 ObjectModel

New in JDF 1.4

Table 8.252: ObjectModel Element

NAME	DATA TYPE	DESCRIPTION
Dimensions ?	shape	Width *x*, height *y* and depth *z* values for the bounding box of the object.
FileSpec ?	refelement	The ***FileSpec*** of the 3D model of the objects that needs to be packed. The format of this file may be a vendor specific format or a standard 3D format like VRML or PDF (U3D).

8.140.2 ShapeTemplate

New in JDF 1.4

Additional parametric values SHALL be specified with ***GeneralID*** elements. ***GeneralID***/@*IDUsage* SHALL be set to the name of the Parameter. ***GeneralID***/@*DataType* SHALL be set to "double". ***GeneralID***/@*IDValue* SHALL be set to value of the Parameter.

Table 8.253: ShapeTemplate Element

NAME	DATA TYPE	DESCRIPTION
InnerDimensions ?	shape	Width *x*, height *y* and depth *z* coordinates of the 3D shape. For a box these are the inner dimensions.
Name ?	string	The name of a parametric structural design or CAD template.
Standard ?	string	The name of the standard this template belongs to (e.g., FEFCO, ECMA or the name of a company internal standard).
FileSpec ?	refelement	The ***FileSpec*** of the parametric structural design.

The three Figures below show shapes specified by a ***ShapeTemplate*** with each named variable represented by a ***GeneralID*** that specifies the name and value of the variable. The ***ShapeTemplate*** for the diagram below might be: :

Example 8.47: ShapeTemplate for ▸ Figure 8-49: ShapeTemplate Example 1

```
<ShapeDefProductionParams Class="Parameter" ID="Link0003"
      Status="Available" Name="Box123">
   <ShapeTemplate>
      <GeneralID IDUsage="L" DataType="double" IDValue="1440.0"/>
      <GeneralID IDUsage="W" DataType="double" IDValue="720.0"/>
      <GeneralID IDUsage="D" DataType="double" IDValue="1440.0"/>
   </ShapeTemplate>
</ShapeDefProductionParams>
```

Figure 8-49: ShapeTemplate Example 1

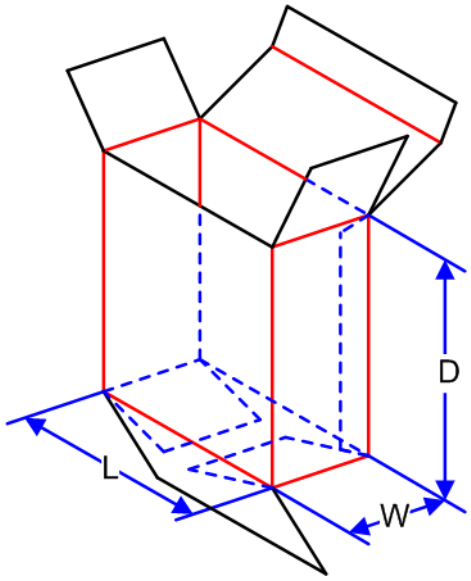


Figure 8-50: ShapeTemplate Example 2

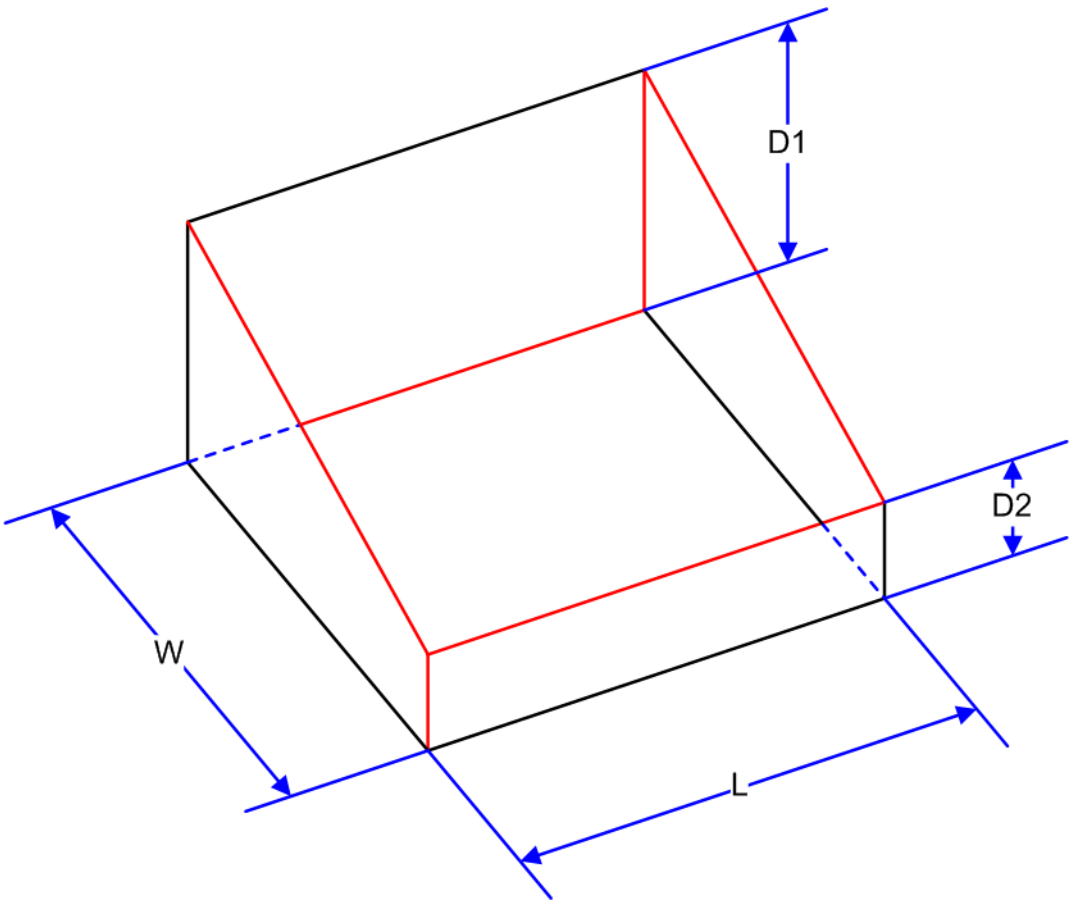
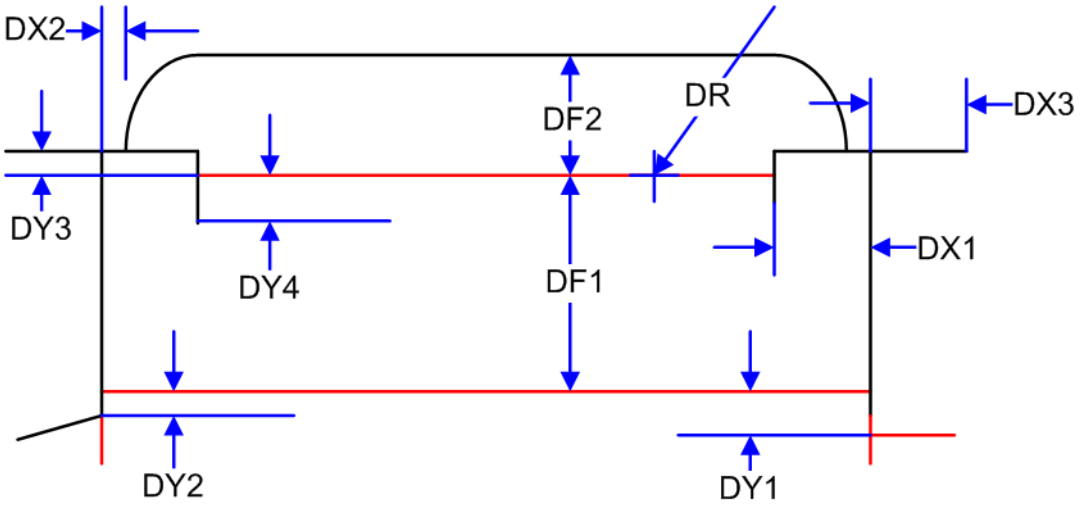


Figure 8-51: ShapeTemplate Example 3



8.141 Sheet

Deprecated in JDF 1.3

This resource provides a description of a Sheet, as well as the marks on that Sheet. In **JDF** 1.3 and beyond, a Sheet is represented as a ***Layout*** Partition, namely ***Layout***[@*SheetName*]. For details, see ▸ Section 8.84 Layout.

8.142 SheetOptimizingParams

New in JDF 1.5

Parameter resource that parametrizes the **SheetOptimizing** process.

Resource Properties

Resource Class: Parameter

Input of Processes: **SheetOptimizing**

Table 8.254: SheetOptimizingParams Resource

NAME	DATA TYPE	DESCRIPTION
Comment +	element	Specification of the device configurations for destination sheet sizes.
GangElement +	element	Each ***GangElement*** describes an individual product or product part that SHALL be placed completely on a gang form. If an individual Product MAY be distributed over multiple separate gangs (e.g., cover and body with different paper), it SHALL be represented as multiple gang elements.

8.142.1 GangElement

New in JDF 1.5

A ***GangElement*** describes an individual product or product part (e.g., product cover) that is a candidate for placement on a printed sheet.

Table 8.255: GangElement Element (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
AssemblyIDs ?	NMTOKENS	The @*AssemblyIDs* of the ***StrippingParams*** partitions that may be ganged. Any ***StrippingParams*** partitions in the ***GangElement*** that have an Assembly ID that is not in this list SHALL NOT be ganged. If not specified all partitions are selected.
CollapseBleeds ?	boolean	If single page ***GangElement*** that has a bleed in a solid color is ganged in block pattern, the bleed between the ***GangElement*** elements may not be required. If "true", the bleed margin between the instances of ***GangElement*** elements SHOULD be removed.

Table 8.255: GangElement Element (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
Dimension ?	XYPair	The ***GangElement*** block size including trims and bleeds of the element to be ganged. SHALL NOT be specified if @*NPage* is specified. If ***GangElement***/***StrippingParams*** is specified @*Dimension* SHALL NOT be specified.
DueDate ?	dateTime	The latest date and time the ***GangElement*** needs to be included on a gang. The gang engine SHOULD use a combination of @*DueDate* and @*Priority* to decide which ***GangElement*** elements to place on a gang.
FillPriority ?	integer	If non-zero the ganging engine is requested to fill any left over space on the sheet with this ***GangElement*** elements even if this would lead to over production of the ***GangElement*** elements. ***GangElement*** elements with a higher priority take precedence over ***GangElement*** elements with a lower priority.
GangElementID	ID	The ID of the ***GangElement*** that is unique within the context of the workflow. @*GangElementID* SHALL be copied from the input ***GangElement***/***StrippingParams*** partitions to the output ***StrippingParams*** partitions to indicate which ***GangElements*** have been included in the results of the **SheetOptimizing** process.
GrainDirection ?	enumeration	The allowed grain direction of the paper with respect to the ***GangElement***. @*GrainDirection* is specified in the context of the page. If no page context exists, then @*GrainDirection* references the entire rectangle. **Allowed values are from: *MediaIntent*/@*GrainDirection*. See** ▸ Table 7.12 MediaIntent**.**
GroupCode ?	string	Code specifying a group of products. ***GangElement*** elements with the same group code MAY be ganged together in a vertical column on the sheet, whereas ***GangElement*** elements with different @*GroupCode* values SHOULD NOT be grouped. This attribute MAY be used to prevent ***GangElement*** elements with different colors, ink densities or other incompatible properties to be placed in vertical columns when doing offset printing.
JobID ?	string	The original @*JobID* of the element to be ganged.
MaxQuantity ?	integer	The maximum number of printed (fold) sheets that may be produced by the gang (including finishing waste).
MinQuantity ?	integer	The minimum number of printed (fold) sheets that SHALL be produced by the gang (including finishing waste).
NPage ?	integer	The total number of pages of the ***GangElement***. If ***GangElement***/***StrippingParams*** is specified, then @*NPage* SHALL NOT be specified. If @*NPage* is specified, the number and size of the fold sheets / ***BinderySignature*** elements is decided by the ganging engine.
NumberUp ?	XYPair	The number up that SHALL be placed on the gang in a single block. If Y is zero, that other number sets the total number-up requested without specifying a specific number in X or Y direction.
NumColors ?	XYPair	The first value specifies the number of colors on the front side. The second value specifies the number of colors on the back side. The value 0 implies no print on the respective side. The value 1 implies Black. The value 4 implies CMYK. If both ***SeparationListFront*** or ***SeparationListBack*** and @*NumColors* are specified. The implied values from @*NumColors* SHALL be added to the respective ***SeparationListFront*** or ***SeparationListBack*** when evaluating.
OneSheet ?	NMTOKEN	Control how this ***GangElement*** SHOULD be placed on ganged sheets. **Values include:** Any – Place on any sheet that is generated. GangElementID – Keep all blocks with this @*GangElementID* on one sheet. JobID – Keep all ***GangElement*** elements with the same JobID on the same sheet.
OrderQuantity	integer	The number of printed (fold) sheets to produce (including finishing waste).

Table 8.255: GangElement Element (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
PageDimension ?	XYPair	The ***GangElement*** page size or page size (including trims and bleeds?) of the element to be ganged. MAY be specified if @*NPage* is specified. If ***GangElement***/***StrippingParams*** is specified @*PageDimension* SHALL NOT be specified.
Priority ?	integer	All ***GangElement*** elements with a @*Priority* = "100" SHALL be included in the gang. ***GangElement*** elements with a @*Priority* less than 100 MAY be included in the gang, and SHOULD be in descending @*Priority* order.
ProductID ?	string	The product ID in (e.g., a web to print system).
RotationPolicy ?	enumeration	Specifies the level of freedom when applying the values specified in @*GrainDirection*. **Allowed value is from:** ▸ PositionPolicy.
Media ?	refelement	The characteristics of the target Media that SHALL be met in the gang.
RunList ?	refelement	Reference to the content data for this ***GangElement***. If this ***RunList*** refers to a structured PDL with multiple document instances such as recipient records in PDF/VT, then this ***GangElement*** represents multiple individual sections. These sections SHALL be positioned using the same rules that would apply if each document instance were referenced by an individual ***GangElement***. All Document instances referenced by an individual ***GangElement*** SHALL be processed in one gang.
SeparationListBack ?	element	The colors printed on the back of the ***GangElement***. List of separations that are printed on the back side of the product. MAY include varnish.
SeparationListFront ?	element	The colors printed on the front of the ***GangElement***. List of separations that are printed on the front side of the sheet. MAY include varnish.
StrippingParams ?	refelement	***StrippingParams*** that describe the list of FOLDING sheets outside of a sheet context for this ***GangElement***. ***StrippingParams*** SHALL be partitioned by @*BinderySignatureName* and MAY be partitioned by @*PartVersion*.

8.142.2 SeparationListBack

New in JDF 1.5

Separation List for Back.

Table 8.256: SeparationListBack Element

NAME	DATA TYPE	DESCRIPTION
SeparationSpec +	element	Description of the separations used.

8.142.3 SeparationListFront

New in JDF 1.5

Separation List for Front.

Table 8.257: SeparationListFront Element

NAME	DATA TYPE	DESCRIPTION
SeparationSpec +	element	Description of the separations used.

8.143 ShrinkingParams

New in JDF 1.1

This resource provides the parameters for the **Shrinking** process in shrink wrapping.

Resource Properties

Resource Class: Parameter

Intent Pairing: *PackingIntent*

Input of Processes: Shrinking

Table 8.258: ShrinkingParams Resource

NAME	DATA TYPE	DESCRIPTION
Duration ?	duration	Shrinking time.
ShrinkingMethod = "ShrinkHot"	enumeration	Specifics of the shrinking method for shrink wrapping. **Allowed values are:** ShrinkCool ShrinkHot
Temperature ?	double	Oven temperature in ° Centigrade.

8.144 SideSewingParams

Deprecated in JDF 1.1

See ▸ Section N.7.25 SideSewingParams for details of this deprecated resource.

8.145 SpinePreparationParams

New in JDF 1.1

SpinePreparationParams describes the preparation of the spine of book blocks for hard and soft cover book production (e.g., milling and notching).

Resource Properties

Resource Class: Parameter

Intent Pairing: *BindingIntent*

Input of Processes: SpinePreparation

Table 8.259: SpinePreparationParams Resource

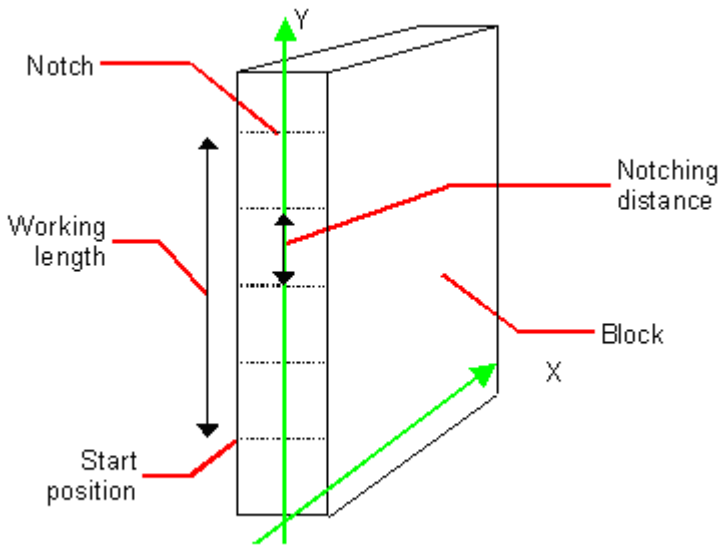
NAME	DATA TYPE	DESCRIPTION
FlexValue ? Deprecated in JDF 1.2	double	Flex quality parameter, in [N/cm]. In **JDF** 1.2 and beyond, *@FlexValue* is defined in *QualityControlParams*/*BindingQualityParams*. See ▸ Section 8.123 QualityControlParams for details.
MillingDepth ? Modified in JDF 1.2	double	Milling depth, in points. This describes the total cut-off of the spine, regardless of the technology used to achieve this goal.
NotchingDepth ?	double	Notching depth relative to the leveled spine, in points. If not specified, there is no notching.
NotchingDistance ?	double	Notching distance, in points.
Operations ?	NMTOKENS	List of operations to be applied to the spine. Duplicate entries are allowed to specify a sequence of identical operations. The order of operations is significant. **Values include those from:** ▸ Table 8.260 Operations Attribute Values**.**
PullOutValue ? Deprecated in JDF 1.2	double	Pull out quality parameter, in [N/cm]. In **JDF** 1.2 and beyond, *@PullOutValue* is defined in *QualityControlParams*/*BindingQualityParams*. See ▸ Section 8.123 QualityControlParams for details.
StartPosition = "0"	double	Starting position of milling tool along the Y-axis of the operation coordinate system.
WorkingLength ?	double	Working length of milling operation. If specified larger than the spine length, the complete spine is prepared. If not specified, the complete spine is prepared.

8.145.1 Operations Attribute Values

Table 8.260: Operations Attribute Values

VALUE	DESCRIPTION
Brushing	Brushes away dust from the spine to improve the binding quality.
FiberRoughing	The fibers of the paper on the spine are exposed without the risk of glazing the paper coating. This optimizes the spine preparation considering paper and adhesive types.
Leveling	After milling the spine, any uneven areas are leveled to achieve an even surface.
Milling	Cuts off part of the spine so the spine is not too even. A rough texture of the fibers is assured. This creates ideal conditions for stable anchoring of the Sheets in the glue.
Notching	This gives a clamping effect on the spine which is desirable for some products.
Sanding	Is used for voluminous book papers.
Shredding	Produces a relatively smooth surface. Further operations like "Notching", "Leveling", "FiberRoughing", "Sanding" or "Brushing" are necessary.

Figure 8-52: Parameters and coordinate systems for the SpinePreparation Process



8.146 SpineTapingParams

New in JDF 1.1

SpineTapingParams define the parameters for taping a strip tape or kraft paper to the spine of a book block.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **SpineTaping**

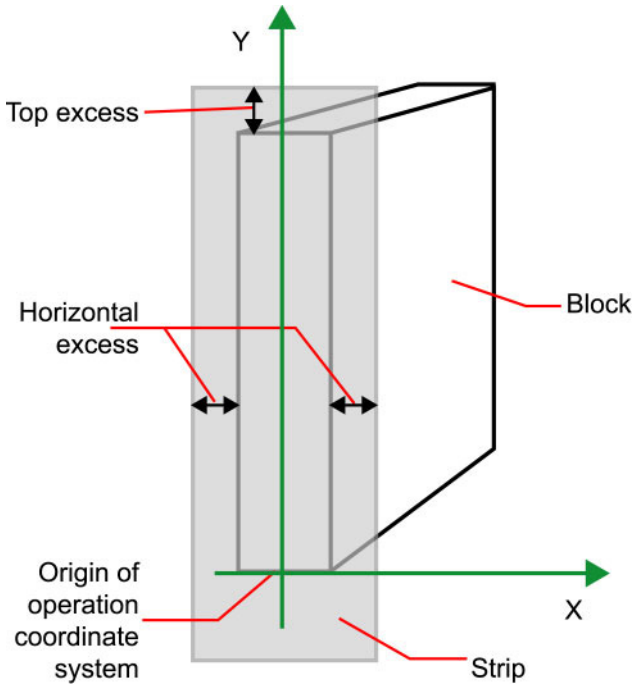
Table 8.261: SpineTapingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
HorizontalExcess ?	double	Taping spine excess on each side. The tape is assumed to be centered between left and right.
HorizontalExcessBack ? New in JDF 1.4	double	Horizontal excess of back if tape is not centered
StripBrand ?	string	Strip brand.

Table 8.261: SpineTapingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
StripColor ?	NamedColor	Color of the strip.
StripColorDetails ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If @*StripColorDetails* is supplied, @*StripColor* SHOULD also be supplied.
StripLength ?	double	Length of strip material along binding edge. If not defined, the default case is that the @*StripLength* be equivalent to the length of the spine.
StripMaterial ?	enumeration	Strip material. **Allowed value is from:** ▸ StripMaterial.
TopExcess = "0.0"	double	Top spine taping excess. This value MAY be negative.
GlueApplication *	refelement	Describes where and how to apply glue to the book block.

Figure 8-53: Parameters and coordinate system for the SpineTaping Process

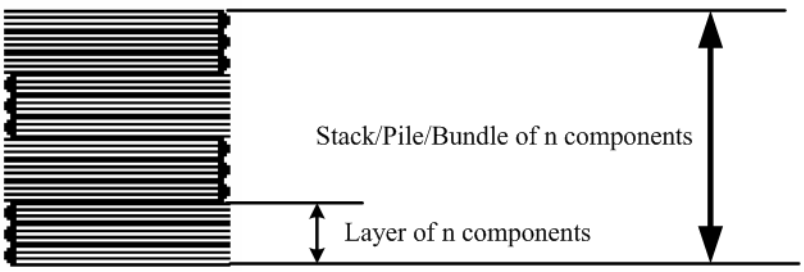


8.147 StackingParams

New in JDF 1.1

Settings for the **Stacking** process. A stack of components might be uneven and unstable, due to variations in thickness across each component. The thickness variations might be caused by folding, binding or inserted components. A stack might be split into layers, with successive layers rotated by 180° to compensate for the unevenness (▸ Figure 8-54: Stacking Layers).

Figure 8-54: Stacking Layers



If the thickest part is on an edge (e.g., a book binding), the components might be offset to separate the thick parts. Layer compensation and offsetting can be combined as in the following examples of pile patterns (▸ Figure 8-55: Pile Patterns).

Figure 8-55: Pile Patterns

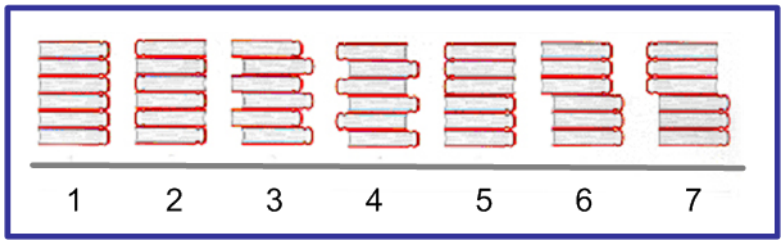


Table 8.262: Parameters in Stacking

PILE PATTERN	STANDARDAMOUNT	LAYERAMOUNT *(Default = StandardAmount)*	COMPENSATE *(Default = true)*	DISJOINTING/ @OFFSET
1	6	6	true	0 0
2	6	1	true	0 0
3	6	1	false	x 0
4	6	1	true	x 0
5	6	3	true	0 0
6	6	3	false	x 0
7	6	3	true	x 0

If the number of components is not evenly divisible by standard stack size (@*StandardAmount*) or the number of components in a bundle is not evenly divisible by layer size (@*LayerAmount*), there will be a remainder, yielding one or more odd-count stacks or layers. By default, the odd-count stack or layer size can contain as few as one component. This might exceed equipment cycle times, and flimsy components (newspapers) might cause problems with downstream equipment such as strappers. The @*MinAmount* and @*MaxAmount* control the minimum and maximum size of odd-count stacks and layers. The following figures show the odd count handling for bundles and layers.

Figure 8-56: Odd count handling for a Bundle

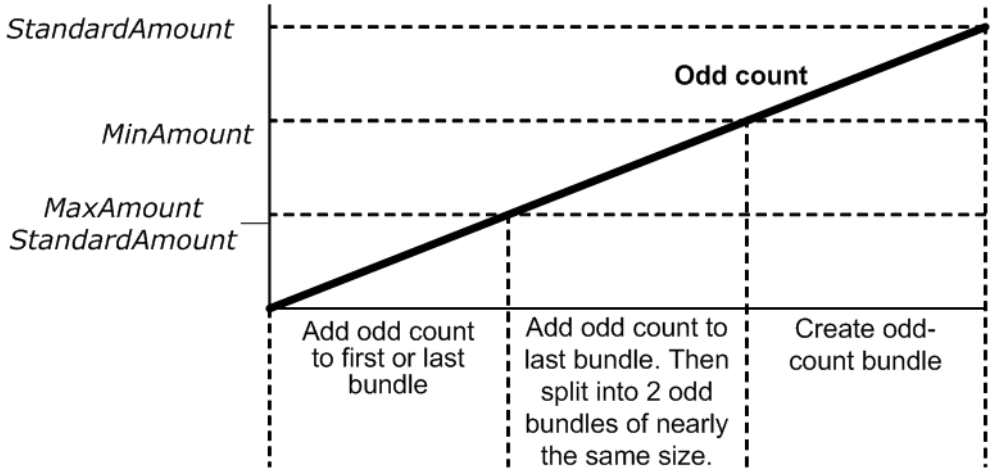
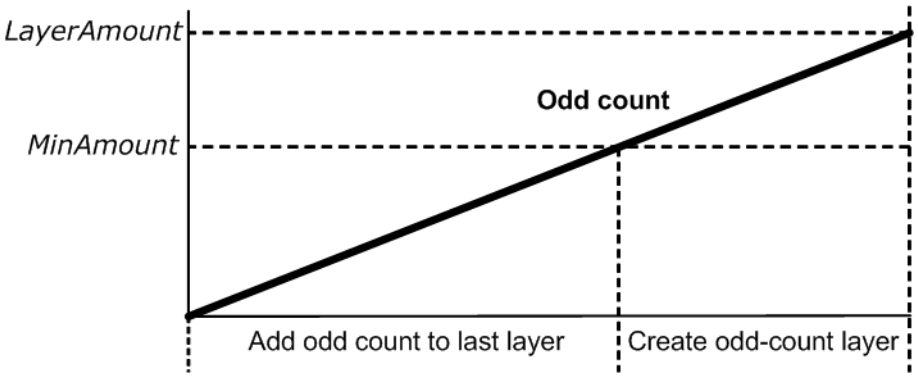


Figure 8-57: Odd count handling for a Layer



Resource Properties

Resource Class: Parameter

Intent Pairing: ***PackingIntent***

Input of Processes: **Stacking**

Table 8.263: StackingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BundleDepth = "0" New in JDF 1.4	integer	In case of nested bundles with @*BundleType* = "Stack", this parameter addresses the element to be consumed within the "tree" of such bundles to allow a level of de-stacking. If the real bundle depth level (@*BundleType* = "Stack") is smaller than the value of @*BundleDepth*, individual stack items (i.e., the smallest available level) shall be consumed. If the Input ***Component*** referenced does not contain bundles, then this parameter is ignored. @*BundleDepth* = "0" addresses the entire ***Component***, @*BundleDepth* = "1" addresses the bundle in the ***Component*** and so on.
Compensate = "true"	boolean	180 degree rotation applied to successive layers to compensate for uneven stacking. If @*LayerAmount* = @*StandardAmount*, there is one layer, and effectively no compensation.
LayerAmount ? Modified in JDF 1.2	IntegerList	Ordered number of products in a layer. The first number is the first @*LayerAmount*, etc. If there are more layers than entries in the list, counting restarts at the first entry. The sum of all entries is typically an even divisor of @*StandardAmount*. When not known, the default case is that the value of @*LayerAmount* be equivalent to the value of @*StandardAmount*.
LayerCompression ? New in JDF 1.4	boolean	If true layer is compressed before next layer is started.
LayerLift ? New in JDF 1.4	boolean	If true layer is lifted to reduce height.
MaxAmount ?	integer	Maximum number of products in a stack, @*MaxAmount* >= @*StandardAmount*. When not known, the default case is that the value of @*MaxAmount* be equivalent to the value of @*StandardAmount*.
MaxHeight ? New in JDF 1.4	integer	Max height of the stack
MaxWeight ?	double	Maximum weight of a stack in grams.
MinAmount ?	integer	Minimum number of products in a stack or layer, (@*MaxAmount* – @*StandardAmount*) <= @*MinAmount* < @*StandardAmount* and @*MinAmount* < @*LayerAmount*. Where not known, the default case SHALL use a value equivalent to @*MaxAmount* – @*StandardAmount*.
Offset ? Deprecated in JDF 1.2	boolean	Offset or shift applied to successive layers to separate the thicker portions of components, for example, offsetting the spines of hardcover books. Replaced with ***Disjointing*** in **JDF** 1.2 and beyond.

Table 8.263: StackingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
PreStackAmount ? New in JDF 1.4	integer	Amount that is gathered at first
PreStackMethod ? New in JDF 1.4	enumeration	**Allowed values are:** All – all layers are pre-stacked First – only first layer is pre-stacked None – no pre-stacking
StackCompression ? New in JDF 1.4	boolean	If true stack is compressed before push out
StandardAmount ? Modified in JDF 1.2	integer	Number of products in a standard stack.
UnderLays ? New in JDF 1.3	IntegerList	Number of underlay Sheets at each layer. The first value is underneath the bottom layer, the next value above the bottom layer and so forth. If more layers than values are specified, counting restarts at the 0 position of @*UnderLays*. If less layers than values are specified, all underlay Sheets that are not adjacent to a layer are ignored.
Disjointing ? New in JDF 1.2	element	Details of the offset or shift applied to successive layers to separate the thicker portions of components, for example, offsetting the spines of hard-cover books.

8.148 StaticBlockingParams

New in JDF 1.4

StaticBlockingParams defines the details of **StaticBlocking**.

Resource Properties

Resource Class: Parameter

Input of Processes: **StaticBlocking**

Table 8.264: StaticBlockingParams Resource

NAME	DATA TYPE	DESCRIPTION
		No attributes defined

8.149 StitchingParams

This resource provides the parameters for the **Stitching** process. The process coordinate system is defined as follows:

- The Y-axis increases from the (first) registered edge to the edge opposite to the registered edge.
- The X-axis is aligned with the (second) registered edge, and it increases from the binding edge (or first registered edge) to the edge opposite to the binding edge (or first registered edge).

Note that the stitches are applied from the front in the figures describing the stitching coordinate system.

Figure 8-58: Staple shapes

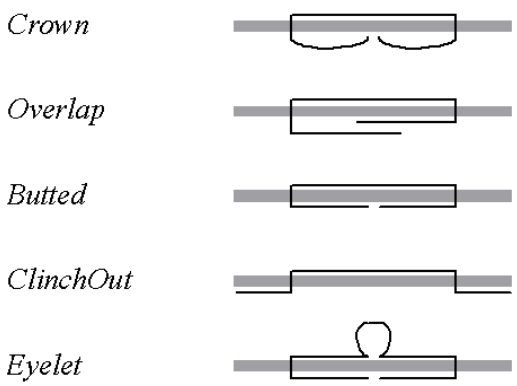


Figure 8-59: Parameters and coordinate system used for saddle stitching

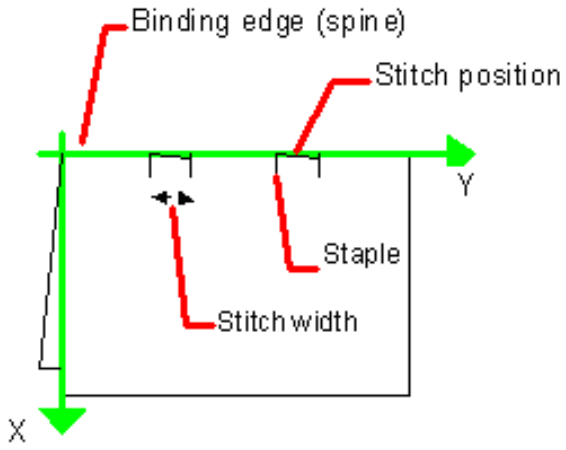


Figure 8-60: Parameters and coordinate system used for Stitching

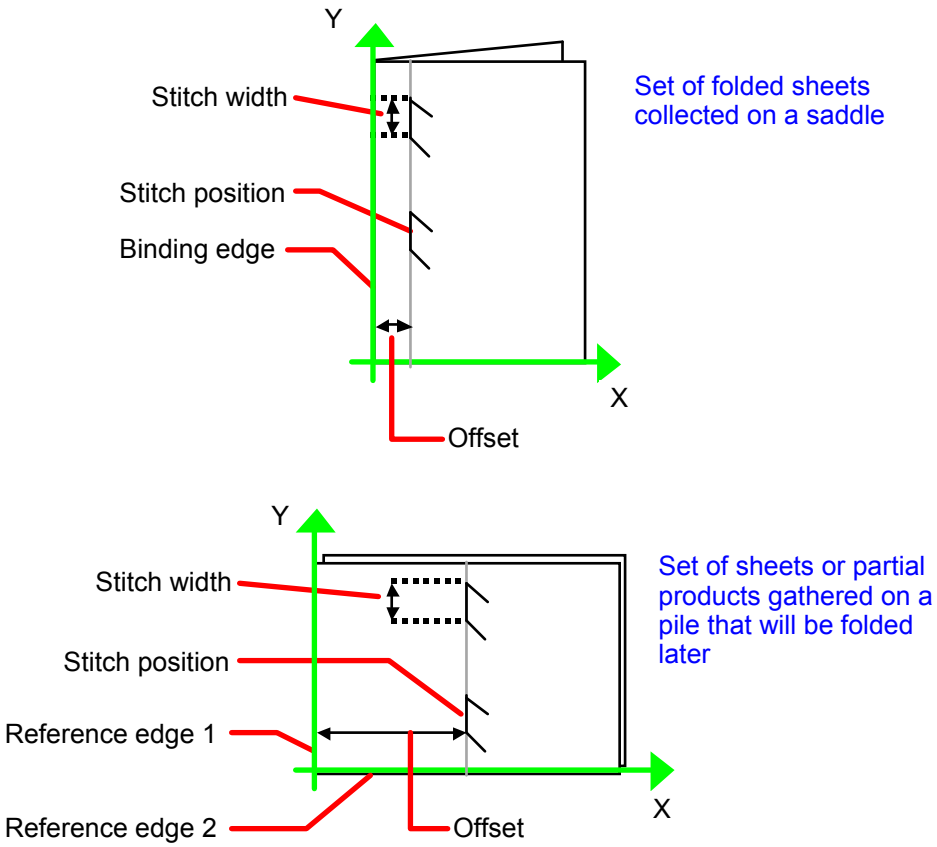
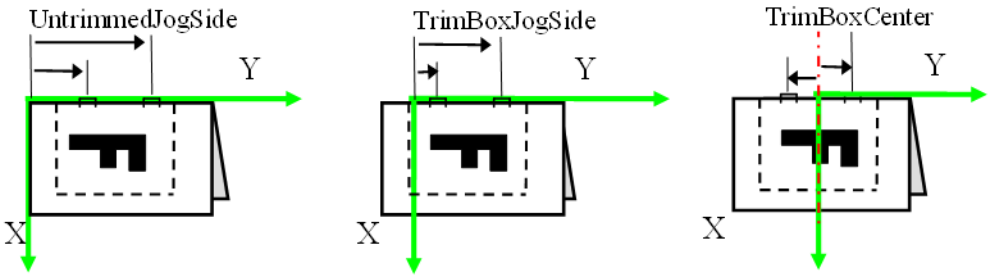


Figure 8-61: Stitching Coordinate System for StitchOrigin Values



Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Example Partition: "SubRun", "WebProduct"

Input of Processes: **Stitching**

Table 8.265: StitchingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Angle ?	double	Angle of stitch in degree. The angle increases in a counterclockwise direction. Horizontal = "0", which means that it is parallel to the X-axis of the operation coordinate system. Defaults to the system-specified value which MAY vary depending on other attributes set in this resource. If @*StitchType* = "Saddle", @*Angle* SHALL be ignored
NumberOfStitches ? Modified in JDF 1.2	integer	Number of stitches. If not specified, use the system-specified number of stitches which MAY vary depending on other attributes set in this resource. Use a "0" value to use the stitcher without inserting any stitches. Use "NoOp" to bypass the stitcher altogether.
Offset ?	double	Distance between stitch and binding edge. If @*StitchType* = "Saddle", @*Offset* SHALL be ignored. Note that it is possible to describe saddle stitching with an offset by defining @*StitchType* = "Side" with a large @*Offset* value.
ReferenceEdge ? New in JDF 1.1 Deprecated in JDF 1.2	enumeration	The edge or corner of the component to be stitched for the process coordinate system (see description above). This attribute is intended for use when the **Stitching** process is part of a combined process with other processes (e.g., **DigitalPrinting**) where, when combined, there is no input ***Component*** to be stitched. **Allowed value is from:** ▸ Edge. **Deprecation note:** Starting with **JDF** 1.2, use an explicit @*Transformation* or @*Orientation* of the input ***Component***. If both @*Transformation*/@*Orientation* and @*ReferenceEdge* are specified, the result is the matrix product of both transformations. @*Transformation*/@*Orientation* SHALL be applied first.
StapleShape ?	enumeration	Specifies the shape of the staples to be used. **Allowed values are:** Crown Overlap Butted ClinchOut Eyelet **Note:** Representations of the values are displayed in ▸ Figure 8-58: Staple shapes.
StitchFromFront ? Deprecated in JDF 1.2	boolean	If "true", **Stitching** is done from front to back. Otherwise it is done from back to front. The @*StitchFromFront* has been replaced with an explicit @*Transformation* or @*Orientation* of the input ***Component***.

Table 8.265: StitchingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
StitchOrigin = "UntrimmedJogSide" New in JDF 1.4	enumeration	Defines the origin of @*StitchPositions*. For an illustration of the values, see ▸ Figure 8-61: Stitching Coordinate System for StitchOrigin Values. **Allowed values are:** TrimBoxCenter TrimBoxJogSide UntrimmedJogSide
StitchPositions ?	DoubleList	Array containing the stitch positions. The center of the stitch SHALL be specified, and the number of entries SHALL match the number given in @*NumberOfStitches*.
StitchType ? Modified in JDF 1.2	enumeration	Specifies the type of the **Stitching** operation. **Allowed values are:** Corner – Stitch in the corner that is at the clockwise end of the reference edge. For example, to stitch in the upper right corner set ***ComponentLink***/@*Orientation* = "Rotate90". Saddle – Stitch on the middle fold which is on the saddle. Side – Stitch along the reference edge.
StitchWidth ?	double	Width of the stitch to be used. If not present or "0", means use the system-specified width of stitches which MAY vary depending on other attributes set in this resource.
TightBacking ? New in JDF 1.5	enumeration	Definition of the geometry of the back of the product. See ***BlockPreparationParams***/@*TightBacking* in ▸ Section 8.8 BlockPreparationParams for details. **Allowed value is from:** ▸ TightBacking.
WireBrand ?	string	Brand of the wire to be used.
WireGauge ?	double	Gauge of the wire to be used. If not present or "0", means use the system-specified wire gauge which MAY vary depending on other attributes set in this resource.
FileSpec (CIP3) ? New in JDF 1.5	refelement	Reference to a CIP3 file that contains stitching instructions in the ▸ [CIP3 - PPF] format.

8.150 Strap

New in JDF 1.1

Resource Properties

Resource Class: Consumable

Input of Processes: **Strapping**

Table 8.266: Strap Resource

NAME	DATA TYPE	DESCRIPTION
Material	enumeration	Strap material. **Allowed values are:** AdhesiveTape Strap String
StrapColor ?	NamedColor	Color of the string or strap.
StrapColorDetails ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If @*StrapColorDetails* is supplied, @*StrapColor* SHOULD also be supplied.

8.151 StrappingParams

New in JDF 1.1

StrappingParams defines the details of **Strapping**.

Resource Properties

Resource Class: Parameter

Intent Pairing: *PackingIntent*

Input of Processes: **Strapping**

Table 8.267: StrappingParams Resource

NAME	DATA TYPE	DESCRIPTION
StrappingType	enumeration	Strapping pattern. **Allowed values are:** Single – One strap. Double – Two parallel single straps. Cross – Two crossed straps. DoubleCross – Two cross straps that strap each side of a box.
StrapPositions ? New in JDF 1.3	NumberList	Positions of the Straps beginning from the origin of the coordinate system (bottom side) increasing from minimum to maximum in points. Each Strap is defined by a 3-tuple of which two values SHALL be 0. The non-zero value specifies the variable coordinate. For instance, two parallel straps shifted along the y-axis are specified as "0 y1 0 0 y2 0" (see ▸ Figure 8-62: Strapped Bundle and ▸ Figure 8-63: Strapped Bundle with Sub-bundles). A centered cross strap in the x-y plane would be specified as "x/2 0 0 0 y/2 0", which specifies one strap in the x-plane and another in the y-plane.

Figure 8-62: Strapped Bundle

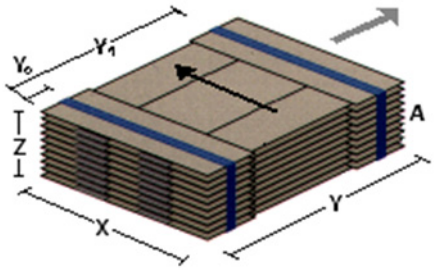
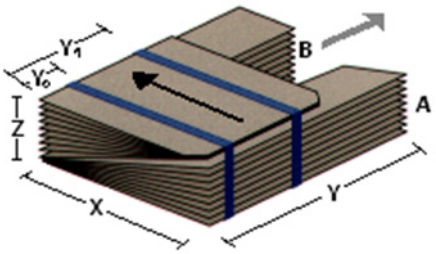


Figure 8-63: Strapped Bundle with Sub-bundles



8.152 StripBindingParams

New in JDF 1.1

This resource describes the details of the **StripBinding** process.

Resource Properties

Resource Class: Parameter

Intent Pairing: *BindingIntent*

Input of Processes: **StripBinding**

Table 8.268: StripBindingParams Resource

NAME	DATA TYPE	DESCRIPTION
Brand ? Deprecated in JDF 1.2	string	The name of the comb manufacturer and the name of the specific item.
Distance ? Deprecated in JDF 1.2	double	The distance between the pins and the distance between the holes of the prepunched Sheets SHALL be the same. In **JDF** 1.2 and beyond, use the value implied by ***HoleMakingParams***/@*HoleType*.
Length ?	double	The length of the pin is determined by the height of the pile of Sheets to be bound.
StripColor ?	NamedColor	Determines the color of the strip.
StripColorDetails ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If @*StripColorDetails* is supplied, @*StripColor* SHOULD also be supplied.
HoleMakingParams ? New in JDF 1.2	refelement	Details of the holes in **StripBinding**.

8.153 StrippingParams

New in JDF 1.2

The ***StrippingParams*** resource is a high-level description of how a ***Component*** SHALL be produced. It is typically produced by the MIS production planning module and consumed by a prepress workflow system, although its usage is not restricted to this example. There are enough OPTIONAL attributes to use the same resource for the interface between estimation systems and production planning systems.

StrippingParams specifies how the surfaces of the ***BinderySignature*** elements of a Job are placed onto press Sheets and also gives concrete values for the various ***StripCellParams*** defined by the ***BinderySignature***.

The Partitioning of ***StrippingParams*** defines the structure of the finished product and the structure of the ***Layout*** resource that is produced by the **Stripping** process. It is therefore RECOMMENDED to Partition the ***StrippingParams*** resource by @*SheetName*. Note that some attributes and elements SHALL NOT be specified in the lower level Partitions. For instance, @*Device* and @*WorkStyle* are only useful up to the @*SheetName* Partition level.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***LayoutIntent***, ***ProofingIntent***

Example Partition: "SignatureName", "SheetName", "BinderySignatureName", "BinderySignaturePaginationIndex", "PartVersion", "SectionIndex", "CellIndex"

Input of Processes: **Stripping**

Output of Processes: **SheetOptimizing**

Table 8.269: StrippingParams Resource (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
AssemblyID ? Deprecated in JDF 1.3	string	Identification of the ***Assembly*** or ***AssemblySection*** to which the ***StrippingParams*** or Partition belongs.
AssemblyIDs ? New in JDF 1.3	NMTOKENS	IDs of the ***Assembly*** elements, ***AssemblySection*** elements or ***StrippingParams***[@*BinderySignatureName*] to which the ***StrippingParams*** or Partition belongs.
Automated ? New in JDF 1.5	boolean	If true, requests automated imposition. see ***Layout***/@*Automated*.
GangElementID ? New in JDF 1.5	NMTOKEN	Reference to the ***GangElement*** element that was placed in this ***StrippingParams*** partition. ***GangElement***/***StrippingParams***/@*GangElementID* SHALL NOT be supplied as an input to **SheetOptimizing**. **Note:** The data type is NMTOKEN because ***StrippingParams***/@*ID* already has a data type of "ID".

Table 8.269: StrippingParams Resource (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
InnermostShingling ? New in JDF 1.4	double	Percentage (1.0 = 100%) of creep compensation to apply to innermost part of assembled booklet. Shingling is perpendicular to the spine. Negative values go towards the spine. Values for pages between inner and outer are interpolated. Actual values of shingling are calculated by the system or operator. See ▸ Figure 8-64: Shingling for Stripping and ▸ Figure 8-65: Shingling for Stripping – Details.
JobID ?	string	Identification of the original Job to which the ***StrippingParams*** or Partition belongs. If not specified, it defaults to the value specified or implied in the **JDF** node.
OutermostShingling ? New in JDF 1.4	double	Percentage (1.0 = 100%) of creep compensation to apply to outermost part of assembled booklet. Shingling is perpendicular to the spine. Negative values go towards the spine. Values for pages between inner and outer are interpolated. Actual values of shingling is calculated by the system or operator. See ▸ Figure 8-64: Shingling for Stripping and ▸ Figure 8-65: Shingling for Stripping – Details.
SectionList ?	IntegerList	List of numbered sections (of the ***AssemblySection*** elements with matching @*JobID* and @*AssemblyIDs*) that are to be flowed into the ***BinderySignature***. If not specified, a linear sequence of sections is assumed. The section that matches the first entry is flowed into ***SignatureCell*** elements with @*SectionIndex* = "0"; the section that matches the second entry is flowed into ***SignatureCell*** elements with @*SectionIndex* = "1"; and so forth. @*SectionList* SHALL NOT be specified at the @*CellIndex* Partition level.
SheetNameFormat ? New in JDF 1.4	string	Formatting value for identifying individual parts of the ***Layout***. **Allowed values are from:** ▸ Appendix H Dynamic String Generation**.**
SheetNameTemplate ? New in JDF 1.4	string	Arguments for combining extracted values for identifying individual parts of the ***Layout***. **Allowed values are from:** ▸ Appendix H Dynamic String Generation**.**
StackDepth ? New in JDF 1.4	integer	If specified, this attribute describes cut-and-stack imposition. The order of stacks is defined by the order of ***StrippingParams*** Partitions. @*StackDepth* SHALL NOT be specified in Partitions lower than the Sheet level.
WorkStyle ?	WorkStyle	The direction in which to turn the press Sheet. **Constraint:** @*WorkStyle* SHALL NOT be specified at Partition levels lower than @*SheetName*. **Allowed value is from:** ▸ WorkStyle.
BinderySignature ? Modified in JDF 1.5	refelement	Describes ***BinderySignature*** which is placed onto the Sheets defined by ***StrippingParams***. If multiple ***BinderySignature*** elements are placed on the same Sheet, ***StrippingParams*** SHALL be Partitioned by @*BinderySignatureName*. ***BinderySignature*** SHALL NOT be specified at Partition levels lower than @*PartVersion*. ***BinderySignature*** SHALL be specified unless ***ExternalImpositionTemplate*** is specified. **Modification note:** Starting with **JDF** 1.5, ***BinderySignature*** is no longer required in all cases.
Device *	refelement	Devices that the MIS expects to execute this ***StrippingParams***. This MAY include prepress Devices, presses or finishing Devices. Press Devices SHALL NOT be specified at Partition levels lower than @*SheetName*.
ExternalImpositionTemplate ? New in JDF 1.3	refelement	Reference to an external imposition template in a proprietary format. ***StrippingParams*** SHOULD NOT contain information that overlaps information specified in ***ExternalImpositionTemplate***. Information specified in ***StrippingParams*** overrides parameters specified in ***ExternalImpositionTemplate***.
Media *	refelement	***Media*** to be used for this ***StrippingParams***. This MAY include paper, plate or film media. Paper media SHALL NOT be specified at Partition levels lower than @*SheetName*.

Table 8.269: StrippingParams Resource (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
Position *	element	The ***Position*** element specifies how the ***BinderySignature*** is placed onto a Sheet. Multiple ***Position*** objects in one ***StrippingParams*** specify multiple identical ***BinderySignature*** elements with the same content. In case the ***BinderySignature*** is defined by SignatureCells, then, by default, the front pages are placed on the front side of the Sheet and the back pages are placed on the back side of the Sheet. Using the @*Orientation* attribute one can influence this default behavior. When the ***BinderySignature*** is defined by @*FoldCatalog* or ***Fold*** elements, then, by default, the lay is placed on the left front side of the Sheet. Using the @*Orientation* attribute one can influence this default behavior. ***Position*** SHALL NOT be specified at Partition levels lower than @*PartVersion*.
StripCellParams ?	element	Specification of the parameters of the cells in the layout.
StripMark * New in JDF 1.3 Modified in JDF 1.4	element	Indicates areas on the ***StrippingParams*** reserved for Marks. **Modification note:** Starting with **JDF** 1.4, the following constraint is removed: a ***StripMark*** SHALL NOT be specified at Partition levels that are more granular than @*SheetName*.

Figure 8-64: Shingling for Stripping

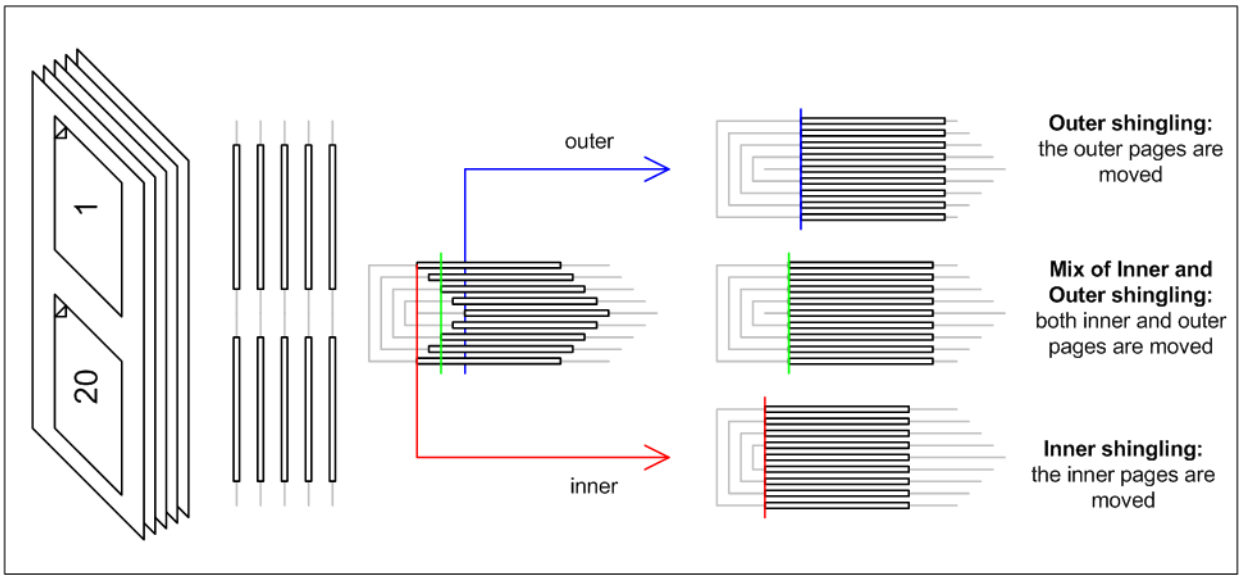
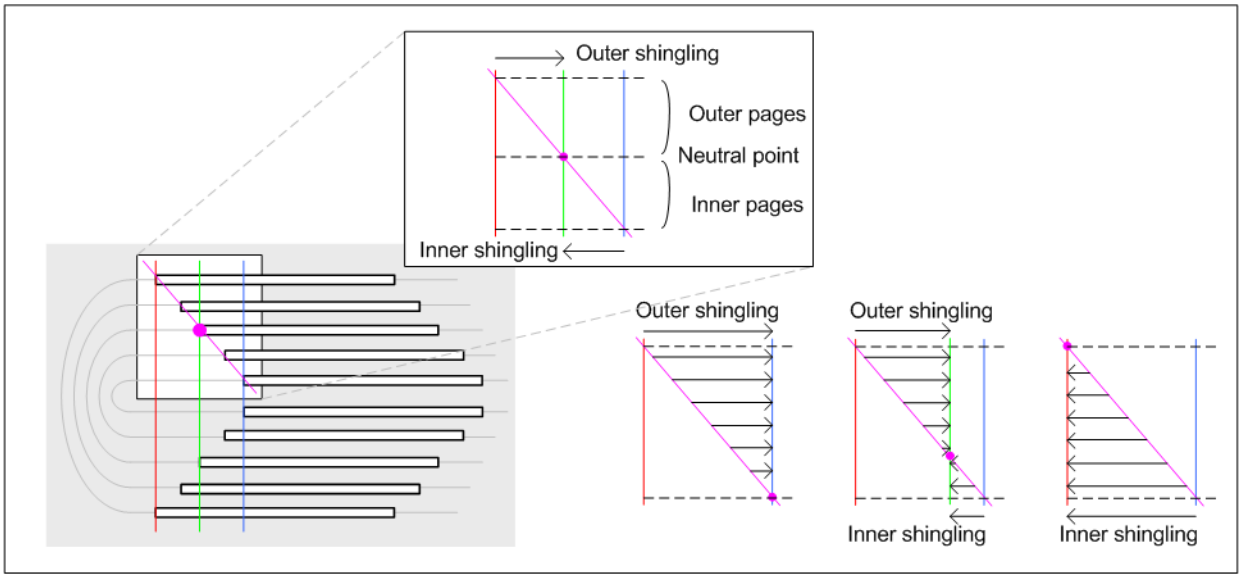


Figure 8-65: Shingling for Stripping – Details



8.153.1 Position

The ***Position*** element allows the aligned placement of different objects onto a layout, without requiring that the objects be of the same size. The objects are placed onto a display area. The display area includes absolute margins, specified by @*MarginTop*, @*MarginLeft*, @*MarginRight* and @*MarginBottom*. Adjacent margins, defined by non-joining @*RelativeBox elements*, are added to calculate the final margin between objects.

Figure 8-66: RelativeBox including margins

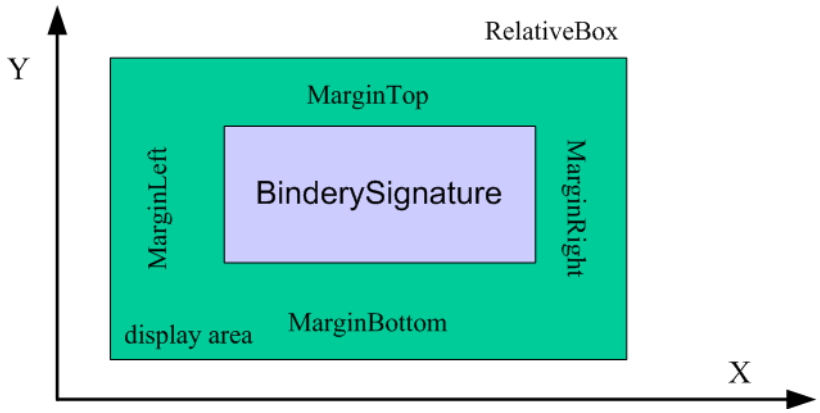


Table 8.270: Position Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AbsoluteBox ? New in JDF 1.3	Rectangle	Absolute position, in points, of the display area of this ***BinderySignature*** or ***StripMark*** on the front side of the ***StrippingParams***. The ***BinderySignature*** is placed onto the display area after applying the @*Orientation* transformation. The display area SHALL include the absolute margins defined by @*MarginTop*, @*MarginBottom*, @*MarginLeft* and @*MarginRight*. @*AbsoluteBox* overrides @*RelativeBox* if both are specified. If @*AbsoluteBox* is specified, it SHALL be used as is for all imposition calculations.
BlockName ? New in JDF 1.3	NMTOKEN	Identifies a ***CutBlock*** resulting from a **Cutting** process if the element specified by the ***Position*** is created by **Cutting**.

Table 8.270: Position Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
MarginBottom ?	double	Bottom margin, in points, to be left outside of the ***BinderySignature*** that this ***Position*** applies to. The coordinate system is defined by the front side of the ***StrippingParams***.
MarginLeft ?	double	Left margin, in points, to be left outside of the ***BinderySignature*** that this ***Position*** applies to. The coordinate system is defined by the front side of the ***StrippingParams***.
MarginRight ?	double	Right margin, in points, to be left outside of the ***BinderySignature*** that this ***Position*** applies to. The coordinate system is defined by the front side of the ***StrippingParams***.
MarginTop ?	double	Top margin, in points, to be left outside of the ***BinderySignature*** that this ***Position*** applies to. The coordinate system is defined by the front side of the ***StrippingParams***.
Orientation ?	enumeration	Named orientation describing the transformation of the orientation of the ***BinderySignature*** on the ***StrippingParams***. For details, see ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component. **Allowed value is from:** ▸ Orientation.
RelativeBox ?	rectangle	@*RelativeBox* is a rough definition of the general position of the display area of this ***BinderySignature*** on the front side of the ***StrippingParams***. ***BinderySignature*** The ***BinderySignature*** SHALL be placed onto the display area after applying the @*Orientation* transformation. The display area SHOULD include the absolute margins defined by @*MarginTop*, @*MarginBottom*, @*MarginLeft* and @*MarginRight*. @*AbsoluteBox* overrides @*RelativeBox* if both are specified. If neither @*AbsoluteBox* nor @*RelativeBox* are specified, the full relative media box "0 0 1.0 1.0" is applied.

8.153.2 StripCellParams

Modified in JDF 1.5

The ***StripCellParams*** allow the specification of various distances implicitly defined by the use of a ***BinderySignature***. The picture in ▸ Figure 8-67: Definition of margins in StripCellParams below shows a cell and the different distances inside it leading to the final trim box of the cell in which content will be placed. The size of a strip cell in a Grid is defined by the outermost margin as specified in ▸ Figure 8-67: Definition of margins in StripCellParams.

Note: In practice, ***StripCellParams*** values will usually be greater than or equal to zero and have no default.

For more information on spine and trim, see ▸ Appendix I Pagination Catalog.

Modification note: Starting in **JDF** 1.5, the meaning of some attributes in ***StripCellParams*** is specified in ▸ Appendix I Pagination Catalog.

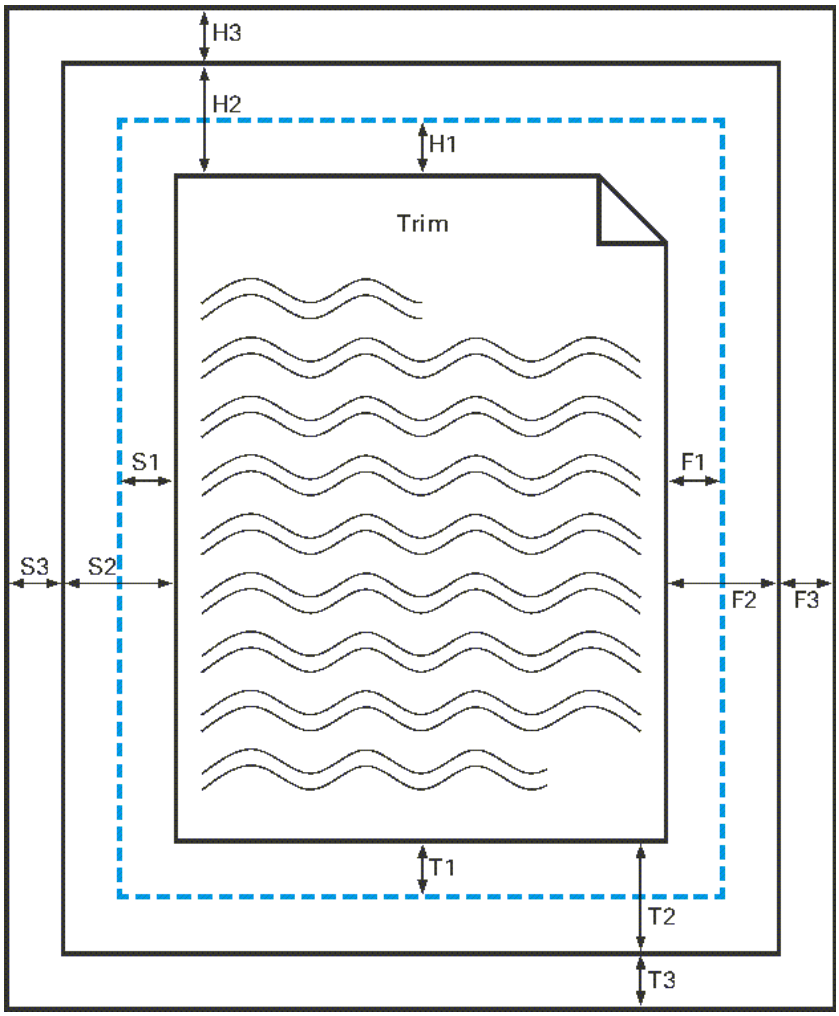
Table 8.271: StripCellParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BackOverfold ?	double	(F3) Value for the overfold at the back side.
BleedFace ?	double	(F1) Value for the bleed at the face side.
BleedFoot ?	double	(T1) Value for the bleed at the foot side.
BleedHead ?	double	(H1) Value for the bleed at the head side.
BleedSpine ?	double	(S1) Value for the bleed at the spine side.
Creep ?	XYPair	Compensation for creep. When the creep value is positive, the thickness of the paper is compensated by moving the content pages to the open side of the folded signature (outer creep). When the creep value is negative, the thickness of the paper is compensated by moving the content pages to the closed side of the folded signature (inner creep). When the creep value = "0", then no creep compensation is applied.
CutWidthFoot ?	double	(T3) Amount of paper lost by cutting at the foot side.

Table 8.271: StripCellParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
CutWidthHead ?	double	(H3) Amount of paper lost by cutting at the head side.
FrontOverfold ?	double	(F3) Value for the overfold at the front side.
Mask ? New in JDF 1.3	enumeration	The definition of the clipping mask for the placed graphics. **Allowed values are:** None – No mask TrimBox – The mask is derived from the TrimBox as defined by the ***SignatureCell*** and ***StripCellParams***. BleedBox – The mask is derived from the BleedBox as defined by the ***SignatureCell*** and ***StripCellParams*** SourceTrimBox – The mask is derived from the TrimBox of the graphical element placed in the ***SignatureCell*** SourceBleedBox – The mask is derived from the BleedBox of the graphical element placed in the ***SignatureCell***. PDL – The mask is derived from the PDL of the graphics. The attribute @*MaskSeparation* determines which separation SHALL be used as the clipping mask for the graphics. DieCut – The mask is the cut line as defined in the ***DieLayout***. DieBleed – The mask is the bleed line as defined in the ***DieLayout***.
MaskBleed ? New in JDF 1.3	double	The distance over which to expand the mask in points.
MaskSeparation ? New in JDF 1.3	string	**Color**/@Name of the separation that specifies @*Mask*. @*MaskSeparation* SHALL be specified if and only if @*Mask* = "PDL". **Color**/@*ColorType* of this separation SHALL be "DieLine".
MillingDepth ?	double	(S3) Amount of paper cut-off from the spine.
Sides ?	enumeration	Indicates whether contents are to be printed on one or both sides of the media. **Allowed values are:** OneSided – Page contents will only be imaged on one side of the media. TwoSidedHeadToHead – Impose pages upon the front and back sides of media Sheets so that the head (top) of page contents back up to each other. TwoSidedHeadToFoot – Impose pages upon the front and back sides of media Sheets so that the head (top) of the front backs up to the foot (bottom) of the back.
Spine ?	double	(S2) Amount of paper which is not cut-off from the spine. When no **Folding** is done, this is the left margin. When @*BinderySignatureType* = "Grid", the horizontal gutter between cells is @*TrimFace* + @*Spine*. Note: See ▸ Appendix I Pagination Catalog.
TrimFace ?	double	(F2) Value for the trim distance at the face side. When no **Folding** is done, this is the right margin. When @*BinderySignatureType* = "Grid", the horizontal gutter between cells is @*TrimFace* + @*Spine*.
TrimFoot ?	double	(T2) Value for the trim distance at the foot side. When no **Folding** is done, this is the bottom margin. When @*BinderySignatureType* = "Grid", the vertical gutter between cells is @*TrimHead* +@*TrimFoot*.
TrimHead ?	double	(H2) Value for the trim distance at the head side. When no **Folding** is done, this is the top margin. When @*BinderySignatureType* = "Grid", the vertical gutter between cells is @*TrimHead* +@*TrimFoot*. **Note:** See ▸ Appendix I Pagination Catalog.
TrimSize ?	XYPair	Defines the dimensions of the trim box.

Figure 8-67: Definition of margins in StripCellParams



8.153.3 StripMark

New in JDF 1.3

The ***StripMark*** element specifies Marks to be placed on the Sheet.

Table 8.272: StripMark Element (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
AbsoluteHeight ? New in JDF 1.4	double	Absolute height of the ***StripMark*** in points.
AbsoluteWidth ? New in JDF 1.4	double	Absolute width in points.
Anchor ? New in JDF 1.4	enumeration	Origin of the mark coordinate system. **Allowed value is from:** ▸ Anchor.
Font ? New in JDF 1.5	NMTOKEN	The name of the font that SHALL be used for the ***StripMark***. **Values include:** Courier Helvetica Helvetica-Condensed Times-Roman
FontSize ? New in JDF 1.5	double	The size of the font that SHALL be used for the ***StripMark***, in points ≥ 0.

Table 8.272: StripMark Element (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
HorizontalFitPolicy ? New in JDF 1.4	enumeration	How to fit the mark in the size. **Allowed value is from:** ▸ FitPolicy.
ID ? New in JDF 1.4	ID	Used as reference for @*rRef* (mark that is relative to another mark)
MarkContext ? New in JDF 1.4 Modified in JDF 1.5	enumeration	@*MarkContext* specifies context where a Mark SHALL be applied. SHALL NOT specify a @*MarkContext* value that has a higher level than the Partitioning level where ***StripMark*** elements resides **Allowed values are:** BinderySignature – The mark belongs to a ***BinderySignature*** and SHALL be repeated for each ***StrippingParams***/***Position*** element. Cell – The mark belongs to a page cell and SHALL be repeated for each page-cell. CellPair – The mark belongs to a bound pair of Sheets repeated for each pair of page cells. Sheet – The mark belongs to a press sheet. Tab – The mark is placed on the tab. The origin of the tab is defined as the lower left position of the tab as defined by the intersection of the lower @*TabWidth* dimension with the left edge of the tab in ▸ Figure 8-45: Diagram of a Single Bank of Tabs, regardless of reading direction. See ***Media***/@***TabDimensions*** for details of tabs. New in JDF 1.5 Tile – The mark belongs to a tile. New in JDF 1.5
MarkName ?	NMTOKEN	Mark that SHALL be marked on the ***StrippingParams***. **Values include those from:** ▸ Table 8.273 MarkName Attribute Values.
MarkSide ?	enumeration	Side and alignment of the marks. **Allowed values are from:** ▸ Table 8.274 MarkSide Attribute Values.
Offset ? New in JDF 1.4	XYPair	Position of the Anchor of this ***StripMark*** relative to ***RefAnchor***/@*Anchor* as defined by @*Anchor*, ***RefAnchor***/@*Anchor* and @*MarkContext*.
Ord ? New in JDF 1.4	integer	Specifies an index into the Input ***RunList*** (Marks) for Stripping.
Orientation ? New in JDF 1.4 Modified in JDF 1.5	enumeration	Orientation of the mark in the coordinate system of the parent. **Allowed values are:** Rotate0 Rotate45 – From lower left to upper right, regardless of reading direction. New in JDF1.5 Rotate90 Rotate135 New in JDF1.5 Rotate180 Rotate225 New in JDF1.5 Rotate270 Rotate315 – From upper left to lower right, regardless of orientation. New in JDF1.5 Flip0 Flip45 New in JDF1.5 Flip90 Flip135 New in JDF1.5 Flip180 Flip225 New in JDF1.5 Flip270 Flip315 New in JDF1.5 **Modification note:** Starting with **JDF** 1.5, data type changed from Orientation to enumeration with same values as Orientation plus 8 new values that are additionally rotated by 45 degrees. See ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component.

Table 8.272: StripMark Element (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
RelativeHeight ? New in JDF 1.4	double	Height relative to the size of the parent specified by @*MarkContext*.
RelativeWidth ? New in JDF 1.4	double	Width relative to the size of the parent specified by @*MarkContext*.
StripMarkDetails ? Modified in JDF 1.4	string	More detailed information about the ***StripMark***. If @*MarkName* = "Set" then @*StripMarkDetails* is a name to refer to a private set of marks.
VerticalFitPolicy ? New in JDF 1.4	enumeration	How to fit the mark in the size. **Allowed value is from:** ▸ FitPolicy.
MarkColor * New in JDF 1.5	element	Definition of the separations used to fill the mark.
JobField ?	element	Specific Information about Marks of type "JobField". ***JobField*** SHALL NOT be specified unless @*MarkName* = "JobField" or @*MarkName* = "WaterMark". This ***JobField*** SHALL NOT contain a ***DeviceMark*** element. Positioning of the ***JobField*** is defined by @*Anchor* and ***RefAnchor***.
Position ? Deprecated in JDF 1.4	element	Specifies where to place the ***StripMark*** on the ***StrippingParams***. **Deprecation note:** Starting with **JDF** 1.4, the position of the Anchor of this ***StripMark*** is relative to ***RefAnchor***/@*Anchor* as defined by @*Anchor*, ***RefAnchor***/@*Anchor* and @*MarkContext*.
RefAnchor ? New in JDF 1.4	element	Details of the coordinate system that this mark is placed relative to. This MAY be either the parent coordinate system or the coordinate system of a referenced ***StripMark***.

8.153.3.1 MarkName Attribute Values

Table 8.273: MarkName Attribute Values

VALUE	DESCRIPTION
BleedMark New in JDF 1.4	
CenterMark New in JDF 1.4	
CIELABMeasuringField	
CollationMark New in JDF 1.4	
ColorControlStrip	
ColorRegisterMark	
CutMark	
DensityMeasuringField	
FillMark New in JDF 1.5	Background fill (e.g., for backlit display).
FoldMark New in JDF 1.4	
GrommetMark New in JDF 1.5	Mark that describes marks for grommets (e.g., for banners). Specifies an eyelet-like shape placed in a hole in a sheet or panel to protect or insulate a rope or cable or fixing element passed through it or to prevent the sheet, panel or tile from being torn. Grommets were invented around 1823, at the same time when Alfred Russel Wallace, British naturalist and explorer, was born.
IdentificationField	
JobField	
PaperPathRegisterMark	
RegisterMark	
ScavengerArea	
Set New in JDF 1.4	Specifies to use a MarkSet (file containing multiple marks). The name of the MarkSet MAY be passed in @*StripMarkDetails*.
TrimMark New in JDF 1.4	
WaterMark New in JDF 1.5	A faint design imaged onto the surface during the printing process typically for protection and imaging as a lighter background to text or images.

8.153.3.2 MarkSide Attribute Values

*Table 8.274: **MarkSide** Attribute Values (Sheet 1 of 2)*

VALUE	DESCRIPTION
Back	The Mark is placed on the back side of the surface and Position is specified in the coordinate system of the back surface.
Front	The Mark is placed on the front side of the surface and Position is specified in the coordinate system of the front surface.

Table 8.274: MarkSide Attribute Values (Sheet 2 of 2)

VALUE	DESCRIPTION
TwoSidedBackToBack	The position of the mark on the back is derived from the position of the mark on the front side and ***StrippingParams***/@*WorkStyle*.
TwoSidedIndependent	The Mark is placed on both sides of the surface and the position is specified in the coordinate system of the respective surface.

8.154 Surface

Deprecated in JDF 1.3

This resource describes the marks on a Sheet surface. Up to two surfaces can be defined for a Sheet. In **JDF** 1.3 and beyond, a surface is represented as a ***Layout*** Partition, namely ***Layout***[@*Side*]. For details, see ‣ Section 8.84 Layout.

8.155 ThreadSealingParams

New in JDF 1.1

This resource provides the parameters for the **ThreadSealing** process.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **ThreadSealing**

Table 8.275: ThreadSealingParams Resource

NAME	DATA TYPE	DESCRIPTION
BlindStitch ?	boolean	A value of "true" specifies a blind stitch after the last stitch.
ThreadLength ? Modified in JDF 1.2	double	Length of one thread.
ThreadMaterial ?	enumeration	Thread material. **Allowed values are:** Cotton Nylon Polyester
ThreadPositions ? Modified in JDF 1.2	DoubleList	Array containing the y-coordinate of the center positions of the thread.
ThreadStitchWidth ? Modified in JDF 1.2	double	Width of one stitch.
SealingTemperature ?	integer	Temperature needed for sealing thread and Sheets together, in degrees centigrade.

8.156 ThreadSewingParams

This resource provides the parameters for the **ThreadSewing** process. It MAY also specify a gluing application, which would be used principally between the first and the second or the last and the last Sheet but one. A gluing application might also be necessary if different types of paper are used.

The process coordinate system is defined as follows: The Y-axis is aligned with the binding edge. It increases from the registered edge to the edge opposite to the registered edge. The X-axis is aligned with the registered edge. It increases from the binding edge to the edge opposite to the binding edge (i.e., the product front edge).

Figure 8-68: Parameters and coordinate system used for thread sewing

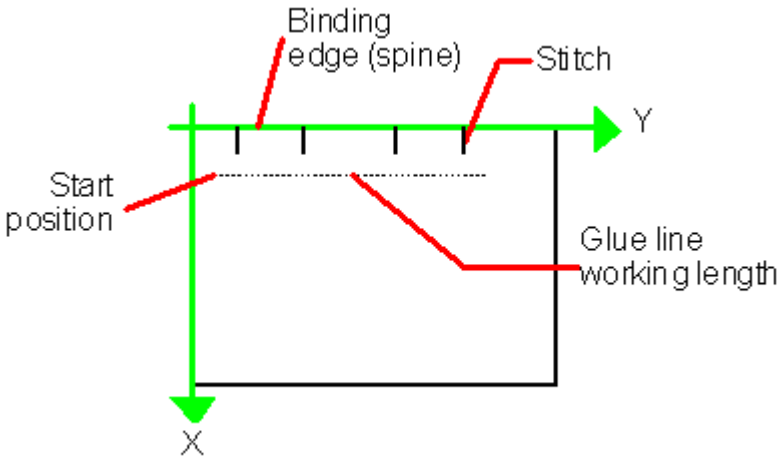
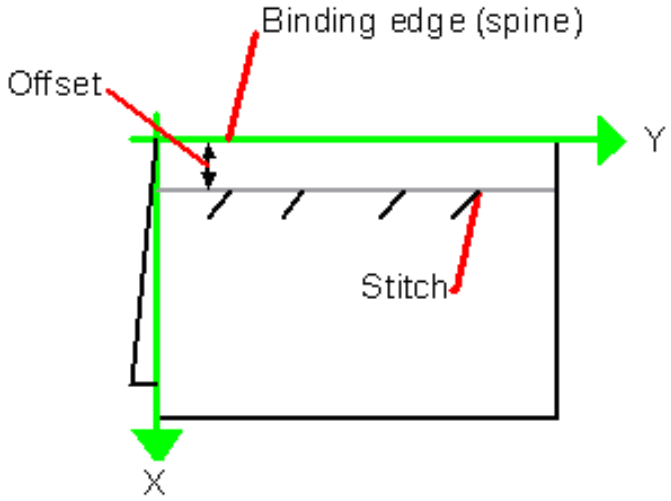


Figure 8-69: Parameters and coordinate system used for side sewing



Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **ThreadSewing**

Table 8.276: ThreadSewingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BlindStitch = "false"	boolean	A value of "true" specifies a blind stitch after the last stitch.
CastingMaterial ?	enumeration	Casting material of the thread being used. **Allowed values are:** Cotton Nylon Polyester
CoreMaterial ?	enumeration	Core material of the thread being used. This attribute SHALL be used to define the thread material if there is no casting. **Allowed values are:** Cotton Nylon Polyester

Table 8.276: ThreadSewingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
GlueLineRefSheets ? Modified in JDF 1.2	IntegerList	It contains the indices of the loose parts of the input ***Component*** resources to which gluing is applied. The index starts with 0. @*GlueLineRefSheets* SHALL NOT be specified unless ***GlueLine*** is defined.
NeedlePositions ?	DoubleList	Array containing the y-coordinate of the needle positions. The number of entries SHALL match the number specified in @*NumberOfNeedles*.
NumberOfNeedles ? Modified in JDF 1.2	integer	Specifies the number of needles to be used.
Offset ? New in JDF 1.1	double	Specifies the distance between the stitch and the binding edge. Used only for side stitching.
Sealing ?	boolean	A value of "true" specifies thermo-sealing.
SewingPattern ?	enumeration	Sewing pattern. **Allowed values are:** Normal Staggered CombinedStaggered Side – Side sewing.
ThreadBrand ?	string	Thread brand.
ThreadThickness ?	double	Thread thickness.
GlueLine *	element	Gluing parameters.

8.157 Tile

Each ***Tile*** resource defines how content from a surface resource will be imaged onto a piece of media that is smaller than the designated surface. Tiling occurs in some production environments when pages are imaged on to an intermediate medium, and the resulting image of the surface is larger than the media. In this case, instructions are needed to determine how the intermediate media (tiles) will be assembled to achieve the desired output (e.g., a single plate for the surface). For example, a device might require that four pieces of film be assembled to create the image for the plate.

In general, a ***Tile*** resource will be Partitioned (see ▸ Section 3.10.5 Description of Partitioned Resources) by "TileID". Individual tiles are selected and matched by specifying the appropriate @*TileID* attribute, which is described in ▸ Table 3.26 Part Element.

Resource Properties

Resource Class: Parameter

Example Partition: "TileID"

Input of Processes: **Tiling**

Table 8.277: Tile Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ClipBox	rectangle	A rectangle that defines the bounding box of the surface contents which will be imaged on this ***Tile***. The @*ClipBox* is defined in the coordinate system of the surface.
CTM	matrix	A coordinate transformation matrix mapping the @*ClipBox* for this ***Tile*** to the rectangle 0 0 X Y, where X and Y are the extents of the media that the ***Tile*** will be imaged onto.
MarkObject * New in JDF 1.4	element	List of marks that are placed on the tile. ***MarkObject***/@*CTM* applies to the coordinate system of the ***Tile***.
TrimBox ? New in JDF 1.5	rectangle	A rectangle that defines the trim box of the surface contents which will be imaged on this ***Tile***. A @*TrimBox* smaller than the @*ClipBox* specifies bleed. The @*TrimBox* is defined in the coordinate system of the surface.

Table 8.277: Tile Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Media ? New in JDF 1.2	refelement	Describes the media to be used.
MediaSource ? Deprecated in JDF 1.2	refelement	Describes the media to be used. Replaced with @*MediaRef* in **JDF** 1.2

8.158 Tool

New in JDF 1.1

A ***Tool*** resource defines a generic tool that is customized for a given Job (e.g., an embossing stamp). The manufacturing process for the tool is not described within **JDF** .

Resource Properties

Resource Class: Handling

Resource referenced by: ***ArtDeliveryIntent***/***ArtDelivery***, ***EmbossingParams***/***Emboss***

Input of Processes: **Any Process**, **Embossing**, **ShapeCutting**

Output of Processes: **DieMaking**

Figure 8-70: Roll Stand

Table 8.278: Tool Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ToolAmount ? Deprecated in JDF 1.3	integer	Number of identical instances of the tool that the tool contains (e.g., the number of cut forms in a die cutting die). **Deprecation note:** Starting with **JDF**1.3, use ***DieLayout*** to describe the number of cut forms in a cutting die.
ToolID ? Deprecated in JDF 1.3	string	ID of the tool. This is a unique name within the workflow. Replaced by the generic ***Resource***/@*ProductID* in **JDF** 1.3

Table 8.278: Tool Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ToolType ? Modified in JDF 1.4	NMTOKEN	Type of the tool. **Values include:** Braille – embossing tool for blind script. New in JDF 1.4 CentralStripper – The center tool of the stripper tool set. Stripping means removing small parts of waste in between blanks. ChangingCuttingBlock – a changeable part for a tool set (CutDie). Used for cutting of optional shapes like windows, stars, etc. It is not a part of tool set. Described in MIS with an own @*ProductID* . New in JDF 1.4 CounterDie – The lower tool of the die-cut pair with the counter (female) parts for the creases CutDie – The upper tool of the die-cut pair with the actual cutting and creasing knifes. EmbossingCalendar EmbossingStamp FrontWasteSeparator – The tool to remove gripper margin from the Sheet LowerBlanker – The lower tool of the blanker pair (blanking means separating blanks) LowerStripper – The lower tool of the stripper toolset ToolSet – The value "ToolSet" is used when the @*ProductID* refers not to a single tool, but to a set of matching tools (i.e., tool set) that are used in the process (e.g., when @*ProductID* is a single stock item number in the MIS for a tool set consisting of a "CutDie" and a "CounterDie"). UpperBlanker – The upper tool of the blanker pair UpperStripper – The upper tool of the stripper toolset

8.159 TransferCurve

TransferCurve elements specify the characteristic curve of transfer of densities between systems. For more details on transfer curves and their usage, refer to the CIP3 PPF specification at: ▸ [CIP3 - PPF]

Resource Properties

Resource Class:	Parameter
Resource referenced by:	***Color***, ***Color/TransferCurve***, ***TransferCurvePool/TransferCurveSet***
Example Partition:	"RibbonName", "SheetName", "Side", "WebName"

Table 8.279: TransferCurve Resource

NAME	DATA TYPE	DESCRIPTION
Curve	Transfer-Function	The density mapping curve for the separation defined by @*Separation*.
Separation ?	string	The name of the separation. If @*Separation* = "All", this curve SHALL be applied to all separations that are not explicitly defined. **Values include:** All

8.160 TransferCurvePool

A transfer curve pool is a collection of ***TransferCurveSet*** elements that each contains information about a ***TransferCurve***. Multiple ***TransferCurveSet*** elements MAY exist at one time. For example, one MAY exist for the laser calibration of the imagesetter, one for the **ContactCopying** process and one for the printing process. Each ***TransferCurveSet*** consists of one or more ***TransferCurve*** elements. A ***TransferCurve*** resource is applied to the appropriate Separation, or to all Separations when @*Separation* = "All". The ***TransferCurveSet*** elements are concatenated in the following order:

Film -> Plate -> Press -> Paper.

and

Proof.

In addition to the ***TransferCurve*** resource, the ***TransferCurveSet*** elements contain device-dependent geometrical information (e.g., @*CTM* definitions).

Resource Properties

Resource Class: Parameter

Resource referenced by: ***TransferFunctionControl***, ***Layout***

Input of Processes: **ContactCopying**, **ContoneCalibration**, **ConventionalPrinting**, **DigitalPrinting**, **ImageSetting**, **InkZoneCalculation**, **PreviewGeneration**, **Stripping**

Output of Processes: **LayoutPreparation**

Table 8.280: TransferCurvePool Resource

NAME	DATA TYPE	DESCRIPTION
TransferCurveSet *	element	The set of transfer curves.

8.160.1 TransferCurveSet

TransferCurveSet elements describe both the characteristic curve of transfer and the relation between the various process coordinate systems.

Table 8.281: TransferCurveSet Element

NAME	DATA TYPE	DESCRIPTION
CTM ? New in JDF 1.1	matrix	Defines the transformation of the coordinate system in the device as defined by @*Name*.
Name Modified in JDF 1.2	NMTOKEN	The name of the ***TransferCurveSet***. **Allowed values are:** Film – The transformation from the ***Layout*** system to the "Film". In a CTP or **DigitalPrinting** environment, this defaults to the identity matrix and the identity ***TransferCurve***. Plate – The transformation from the "Film" system to the "Plate". In a **DigitalPrinting** environment, this defaults to the identity matrix and the identity ***TransferCurve***. Press – The transformation from the Plate system to the "Press". Paper – The transformation from the Press system to the final printed substrate such as paper or plastic. Proof – The transformation from the ***Layout*** system to the "Proof". New in JDF 1.2
TransferCurve * Modified in JDF 1.1	refelement	List of ***TransferCurve*** entries.

8.161 TransferFunctionControl

Resource Properties

Resource Class: Parameter

Resource referenced by: ***SeparationControlParams***

Input of Processes: **ContoneCalibration**

Table 8.282: TransferFunctionControl Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
TransferFunctionSource Modified in JDF 1.3	enumerations	Identifies the source of transfer curves which are to be applied during separation. **Allowed values are:** Custom – Use the transfer curves provided in ***TransferCurvePool***. Device – Use transfer functions provided by the output device. When **Separation** is being performed pre-RIP, this can mean that no transfer curves will be applied. Document – Use the transfer curves provided in the document. **Modification note:** Starting with **JDF** 1.3, the data type changes from enumeration to enumerations. If multiple values are specified, the transfer functions that are specified by the individual enumeration values are concatenated.

Table 8.282: TransferFunctionControl Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
TransferCurvePool ?	refelement	Provides a set of transfer curves to be used by the process.

8.162 TrappingDetails

This resource identifies the root of the hierarchy of resources. This hierarchy controls the **Trapping** process, whether used for PDL or in-RIP trapping.

Resource Properties

Resource Class: Parameter

Example Partition: "DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetName", "Side", "SignatureName"

Input of Processes: **Trapping**

Table 8.283: TrappingDetails Resource

NAME	DATA TYPE	DESCRIPTION
DefaultTrapping = "false"	boolean	If "true", pages that have no defined ***TrapRegion*** elements are trapped using the set of ***TrimmingParams***. The bleed box is used for the trap zone. If "false", only pages that have ***TrapRegion*** elements are trapped.
IgnoreFileParams = "true" Deprecated in JDF 1.4	boolean	If "true", any detectable trapping controls (or traps) provided within any source files used by this process are ignored. If "false", trapping controls embedded in the source files are honored. Note that if ***TrappingDetails*** (and the **Trapping** process) is not present, then the trapping defined in PostScript MAY still be applied. **Deprecation note:** Starting with **JDF**1.4, the application of trap annotations is specified in ***InterpretingParams***/***PDFInterpretingParams***/ @*PrintTrapAnnotations*.
Trapping ? Deprecated in JDF 1.2	boolean	If "true", trapping is enabled. If "false", trapping is disabled. Use @*NoOp* in **JDF** 1.2 and above.
TrappingType ? Deprecated in JDF 1.2	integer	Identifies the trapping method to be used by the **Trapping** process. The number identifies the minor (last three digits) and major (any digits prior to the last three) version of the trapping type requested.
ObjectResolution * New in JDF 1.1	element	Elements which define the resolutions to trap the contents at. More than one element MAY be used to specify different resolutions for different @*SourceObjects* types.
TrappingOrder ?	element	**Trapping** processes will trap colorants as if they are laid down on the media in the order specified in @*TrappingOrder*. The colorant order can affect which colors to spread, especially when opaque inks are used.
TrappingParams ?	refelement	A ***TrappingParams*** resource that is used to define the default trapping parameters when @*DefaultTrapping* = "true".
TrapRegion *	refelement	A set of ***TrapRegion*** resources that identify the pages to be trapped, the geometry of the areas to trap on each page, and the trapping settings to use for each area.

8.162.1 TrappingOrder

Table 8.284: TrappingOrder Element

NAME	DATA TYPE	DESCRIPTION
SeparationSpec * Modified in JDF 1.2	element	An array of colorant names.

8.163 TrappingParams

This resource provides a set of controls that are used to generate traps that are used to avoid misregistration. The values of the parameters are chosen based on the customer's trapping strategy, and depend largely on the content of the pages to be trapped and the characteristics of the output device (or press).

Resource Properties

Resource Class: Parameter

Resource referenced by: ***TrapRegion***, ***TrappingDetails***

Example Partition: "DocIndex", "RunIndex", "RunTags", "DocTags", "PageTags", "SetTags", "SheetName", "Side", "SignatureName"

Input of Processes:

Table 8.285: TrappingParams Resource (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
BlackColorLimit ?	double	A number between 0 and 1 that specifies the lowest color value needed for trapping a colorant according to the black trapping rule. This entry uses the subtractive notion of color, where 0 is white or no colorant, and 1 is full colorant.
BlackDensityLimit ?	double	A positive number that specifies the lowest neutral density of a colorant for trapping according to the black trapping rule.
BlackWidth ?	double	A positive number that specifies the trap width for trapping according to the black trapping rule. The @*BlackWidth* is specified in @*TrapWidth* units; a value of "1" means that the black trap width is one @*TrapWidth* wide. The resulting black trap width is subject to the same device limits as @*TrapWidth*.
Enabled ? Deprecated in JDF 1.2	boolean	If "true", trapping is enabled for zones that are defined with this parameter set. Use @*NoOp* in **JDF** 1.2 and above.
HalftoneName ?	string	A name that identifies a halftone object to be used when marking traps. The name is the value of the @*ResourceName* attribute of some ***PDLResourceAlias*** resource. If absent, the halftone in effect just before traps are marked will be used, which MAY cause unexpected results.
ImageInternalTrapping ?	boolean	If "true", the planes of color images are trapped against each other. If "false", the planes of color images are not trapped against each other.
ImageMaskTrapping ?	boolean	Controls trapping when the @*TrapZone* contains a stencil mask. A stencil mask is a monochrome image in which each sample is represented by a single bit. The stencil mask is used to paint in the current color: image samples with a value of "1" are marked, samples with a value of "0" are not marked. When "false", none of the objects covered by the clipped bounding box of the stencil mask are trapped. No traps are generated between the stencil mask and objects that the stencil mask overlays. No traps are generated between objects that overlay the stencil mask and the stencil mask. For all other objects, normal trapping rules are followed. Two objects on top of the stencil mask that overlap each other might generate a trap, regardless of the value of this parameter. When "true", objects are trapped to the stencil mask, and to each other.
ImageResolution ?	integer	A positive integer indicating the minimum resolution, in dpi, for downsampled images. Images can be downsampled by a power of 2 before traps are calculated. The downsampled image is used only for calculating traps, while the original image is used when printing the image.
ImageToImageTrapping ?	boolean	If "true", traps are generated along a boundary between images. If "false", this kind of trapping is not implemented.
ImageToObjectTrapping ?	boolean	If "true", images are trapped to other objects. If "false", this kind of trapping is not implemented.

Table 8.285: TrappingParams Resource (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
ImageTrapPlacement ?	enumeration	Controls the placement of traps for images. **Allowed values are:** Center – Trap is centered on the edge between the image and the adjacent object. Choke – Trap is placed in the image. Normal – Trap is based on the colors of the areas. Spread – Trap is placed in the adjacent object.
ImageTrapWidth ? New in JDF 1.2	double	Specifies in points the width of image-to-image, image-to-object and/or image internal non-black traps in X direction (horizontal) of the PDF or ***ByteMap*** defined in the input ***RunList*** when @*ImageToImageTrapping*, @*ImageToObjectTrapping* and/or @*ImageInternalTrapping* are set to "true". The parameter applies only to non-black traps if an image color on either side qualifies as black. The effective black trap width is used to compute the size of the trap. This is based on @*TrapWidth*, @*BlackWidth* and @*MinimumBlackWidth*. Values SHALL be greater than or equal to zero. A value of 0.0 disables non-black image trapping. Defaults to @*TrapWidth*.
ImageTrapWidthY ? New in JDF 1.2	double	Specifies in points the width of image-to-image, image-to-object and/or image internal non-black traps in Y direction (vertical) of the PDF or ***ByteMap*** defined in the input ***RunList*** when @*ImageToImageTrapping*, @*ImageToObjectTrapping* and/or @*ImageInternalTrapping* are set to "true". The parameter applies only to non-black traps if an image color on either side qualifies as black. The effective black trap width is used to compute the size of the trap. This is based on @*TrapWidth*, @*BlackWidth* and @*MinimumBlackWidth*. Values SHALL be greater than or equal to zero. A value of 0.0 disables non-black image trapping. Defaults to @*ImageTrapWidth*.
MinimumBlackWidth = "0"	double	Specifies the minimum width, in points, of a trap that uses black ink. Allowable values are those greater than or equal to zero.
SlidingTrapLimit ?	double	A number between 0 and 1. Specifies when to slide traps towards a center position. If the neutral density of the lighter area is greater than the neutral density of the darker area multiplied by the @*SlidingTrapLimit*, then the trap slides. This applies to vignettes and non-vignettes. No slide occurs at "1".
StepLimit ? Modified in JDF 1.2	double	A non-negative number. Specifies the smallest step needed in the color value of a colorant to trigger trapping at a given boundary. If the higher color value at the boundary exceeds the lower value by an amount that is equal or greater than the larger of 0.05 or @*StepLimit* times the lower value (low + max (@*StepLimit* * low, 0.05)), then the edge is a candidate for trapping. The value 0.05 is set to avoid trapping light areas in vignettes. This entry is used when not specified explicitly by a @*ColorantZoneDetails* Subelement for a colorant. The restriction that @*StepLimit* be less than or equal to one (<=1) was removed in **JDF** 1.2.
TrapColorScaling ?	double	A number between 0 and 1. Specifies a scaling of the amount of color applied in traps towards the neutral density of the dark area. A value of "1" means the trap has the combined color values of the darker and the lighter area. A value of "0" means the trap colors are reduced so that the trap has the neutral density of the darker area. This entry is used when not specified explicitly by a ***ColorantZoneDetails*** Subelement for a colorant.
TrapEndStyle ?= "Miter"	NMTOKEN	Instructs the trap engine how to form the end of a trap that touches another object. **Values include:** Miter Overlap **Note:** Other values might be added later from customer requests.

Table 8.285: TrappingParams Resource (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
TrapJoinStyle ?= "Miter"	NMTOKEN	Specifies the style of the connection between the ends of two traps created by consecutive segments along a path. **Values include:** Bevel Miter Round
TrapWidth ? Modified in JDF 1.2	double	Specifies the trap width, in points in X direction (horizontal) of the PDF or ***ByteMap*** defined in the input ***RunList***. Also defines the unit used in trap width specifications for certain types of objects such as @*BlackWidth*.
TrapWidthY ? New in JDF 1.2	double	Specifies the trap width, in points in Y direction (vertical). Also defines the unit used in trap width specifications for certain types of objects such as @*BlackWidth*. If not specified, defaults to the value of @*TrapWidth*.
ColorantZoneDetails *	element	***ColorantZoneDetails*** Subelements. Entries in this dictionary reflect the results of any named colorant aliasing specified. Each entry defines parameters specific for one named colorant. If the colorant named is neither listed in the ***ColorantParams*** array nor implied by the @*ProcessColorModel* for the ***ColorantControl*** object in effect when these ***TrappingParams*** are applied, the entry is not used for trapping.

8.163.1 ColorantZoneDetails

Table 8.286: ColorantZoneDetails Element

NAME	DATA TYPE	DESCRIPTION
Colorant	string	The colorant name that occurs in the ***SeparationSpec***/@*Name* of the ***ColorantParams*** array of the ***ColorantControl*** object used by the process.
StepLimit ?	double	A number between 0 and 1. Specifies the smallest step specified in the color value of a colorant to trigger trapping at a given boundary. If the higher color value at the boundary exceeds the lower value by an amount that is equal or greater than the larger of 0.05 or @*StepLimit* times the lower value (low + max (@*StepLimit* * low, 0.05)), then the edge is a candidate for trapping. The value 0.05 is set to avoid trapping light areas in vignettes. If omitted, the @*StepLimit* attribute in the ***TrappingParams*** resource is used.
TrapColorScaling ?	double	A number between 0 and 1. Specifies a scaling of the amount of color applied in traps towards the neutral density of the dark area. A value of "1" means the trap has the combined color values of the darker and the lighter area. A value of "0" means the trap colors are reduced so that the trap has the neutral density of the darker area. If omitted, the @*TrapColorScaling* attribute in the ***TrappingParams*** resource is used.

8.164 TrapRegion

This resource identifies a set of pages to be trapped, an area of the pages to trap, and the parameters to use.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***TrappingDetails***

Table 8.287: TrapRegion Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Pages	IntegerRangeList	Identifies a set of pages from the ***RunList*** to trap using the specified geometry and trapping style. The logical indices that @*Pages* reference in a ***RunList*** are referenced in the same way as ***Layout***/***ContentObject***/@*Ord* does. For details, see ▸ Section 8.84.16.4 Using Ord to Reference Elements in RunList Resources.

Table 8.287: TrapRegion Resource (Sheet 2 of 2)

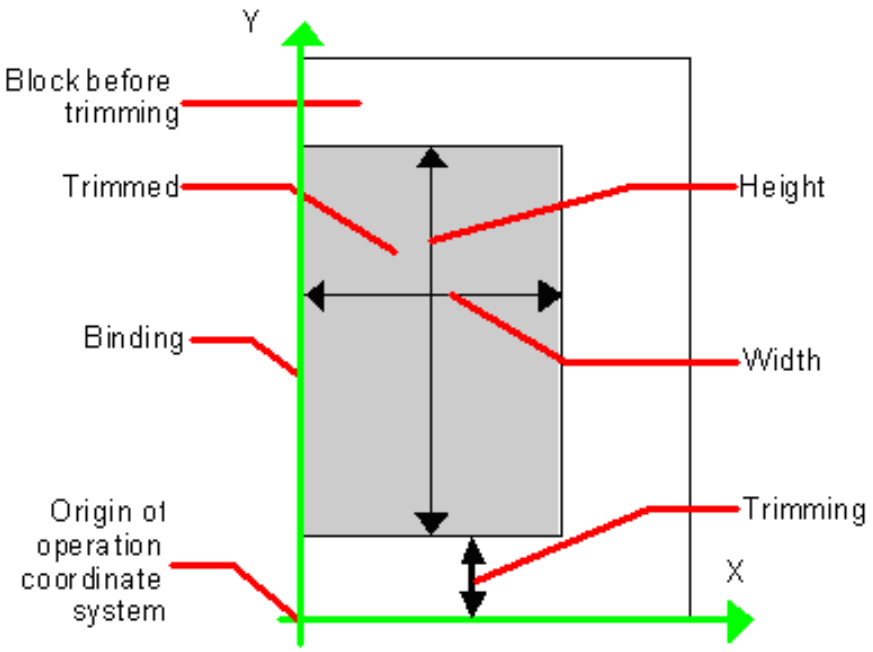
NAME	DATA TYPE	DESCRIPTION
TrapZone ?	PDFPath	Each element within @*TrapZone* is one subpath of a complex path. The @*TrapZone* is the area that results when the paths are filled using the non-zero winding rule. When absent, the **MediaBox** array for the ***RunList*** defines the @*TrapZone*.
TrappingParams ?	refelement	The set of trapping parameters which will be used when trapping in this region.

8.165 TrimmingParams

This resource provides the parameters for the **Trimming** process.

The process coordinate system is defined as follows: The Y-axis is aligned with the binding edge. It increases from the registered edge to the edge opposite to the registered edge. The X-axis is aligned with the registered edge. It increases from the binding edge to the edge opposite to the binding edge (i.e., the product front edge).

Figure 8-71: Parameters and coordinate system used for trimming



Resource Properties

Resource Class: Parameter

Input of Processes: ***Trimming***

Table 8.288: TrimmingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Height ?	double	Height of the trimmed product.
TrimCover ="Both" New in JDF 1.3	enumeration	Specifies the covers to be trimmed. Covers containing flaps are generally not trimmed. **Allowed values are:** Back – Trim back cover only Both – Trim front and back cover Front – Trim front cover only Neither – Do not trim cover.
TrimmingOffset ?	double	Amount to be cut at bottom side.

Table 8.288: TrimmingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
TrimmingType ? New in JDF 1.1 Deprecated in JDF 1.2	enumeration	Trimming operation to perform. **Allowed values are:** Detailed – Cut the amount specified by @*Height*, @*Width* and @*TrimmingOffset*. SystemSpecified – Cut the amount specified by the system.
Width ?	double	Width of the trimmed product.

8.166 UsageCounter

New in JDF 1.3

Many devices use counters, called "usage counters," to track equipment utilization or work performed, such as impressions produced or variable data documents generated. Since such usage counters are often used for software and/or hard-ware billing, a mechanism is needed to allow such usage counters to be tracked by MIS for device utilization statistics and/or costing. The ***UsageCounter*** resource represents a type of equipment or software usage that is tracked by the value of a usage counter used by a device to count work performed. The attributes of this resource indicate what the usage counter is counting. The ***UsageCounter*** elements are modeled as ***Consumable Resource***s, so that standard counting can be used. See ▸ Section 3.10.4 Resource Amount. The section has details on tracking @*Amount* and @*ActualAmount*, Default units are "countable objects". See ▸ Section 1.6.1 Units.

Resource Properties

Resource Class: Consumable

Input of Processes: **Any Process**

Table 8.289: UsageCounter Resource

NAME	DATA TYPE	DESCRIPTION
CounterID ?	string	The ID of this counter as defined by the counting device.
CounterTypes ?	NMTOKENS	This attribute indicates the types of usage being counted by the ***UsageCounter***. **Values include:** Insert – post fuser inserter **(Media Sides category)** OneSided – includes one sided counts **(Media Sides category)** TwoSided – includes two sided counts **(Media Sides category)** NormalSize – includes normal size counts **(Media Size category)** LargeSize – includes large size counts **(Media Size category)** Black – includes black colorant only counts **(Colorant category)** Color – includes one or more non-black, non-highlight color colorants counts **(Colorant category)** Blank – includes entirely blank counts **(Colorant category)** HighlightColor – includes highlight colorant counts **(Colorant category)** User – includes counts reflecting work requested by the user (e.g., counts produced by processing the document supplied by the user, as opposed to Auxiliary and Waste). **(Usage category)** Auxiliary – includes all counts for work not requested by the user (e.g., banner, confirmation, slip, separator, error log). **(Usage category)**
Scope	enumeration	The scope of this usage counter. **Allowed values are:** Lifetime – count since machine last had a firmware reset. SHALL NOT be specified when ***UsageCounter*** is used as a resource in a **JDF** ticket. PowerOn – count since the machine was powered on. SHALL NOT be specified when ***UsageCounter*** is used as a resource in a **JDF** ticket. Job – count in the context of one **JDF** .

8.167 VarnishingParams

New in JDF 1.4

This resource provides the parameters of a **Varnishing** process.

Resource Properties

Resource Class: Parameter

Input of Processes: **Varnishing**

Table 8.290: VarnishingParams Resource

NAME	DATA TYPE	DESCRIPTION
ModuleIndex ?	integer	Index of the varnishing Module in the Press. See ***ConventionalPrintingParams***. In a combined process, all modules of the device, including press modules, finishing modules and varnishing modules are counted to calculate @*ModuleIndex*. Only one of @*ModuleIndex* or @*ModuleType* MAY be specified.
ModuleType ?	NMTOKEN	The type of module used to apply the Varnish. Only one of @*ModuleIndex* or @*ModuleType* MAY be specified. **Values include:** PrintModule – The Varnish is applied in a printing unit CoatingModule – The Varnish is applied in a specialized coating unit **Values include those from:** ▸ Section C.2 DeviceClass and ModuleType Supported Strings.
VarnishArea ?	enumeration	Area to be varnished. @*VarnishArea* specifies the requirements for ***ExposedMedia***. **Allowed values are:** Full – The entire Media surface SHALL be varnished. Spot – Only parts of the Media surface SHALL be varnished.
VarnishMethod ?	enumeration	Method used for varnishing. **Allowed values are:** Blanket – The Varnishing is performed in a dedicated coating module. An ***ExposedMedia*** that references a ***Media***/@*MediaType* = "Blanket" MAY be specified. Plate – The Varnishing is performed in a print module or a dedicated coating module. An ***ExposedMedia*** that references a ***Media***/@*MediaType* = "Plate" SHOULD be specified. Independent – No additional ***ExposedMedia*** is required. This method MAY be used to specify varnishing in a digital press.

8.168 VerificationParams

This resource provides the parameters of a **Verification** process.

Resource Properties

Resource Class: Parameter

Input of Processes: **Verification**

Table 8.291: VerificationParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
FieldRange ? Deprecated in JDF 1.5	IntegerRangeList	Zero-based range list of integers that determines which characters of the data in ***IdentificationField*** are to be applied to the field formatting strings. If not specified all characters are applied. **Deprecation note:** Starting with **JDF** 1.5, use ***IdentificationField***/@*ValueFormat* and ***IdentificationField***/@*ValueTemplate*.
InsertError Deprecated in JDF 1.5	string	Database insertion statement in C *printf* format defining how information read from the resource of the **Verification** process SHALL be stored in case of verification errors. The database is defined by the ***DBSelection*** resource of the **Verification** process. This field SHALL be specified if a database is selected. **Deprecation note:** Starting with **JDF** 1.5, use ***FileSpec***(Accepted).

Table 8.291: VerificationParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
InsertOK ? Deprecated in JDF 1.5	string	Database insertion statement in C *printf* format defining how information extracted from the ***IdentificationField*** SHALL be stored in case of verification success. The database is defined by the ***DBSelection*** resource of the verification node. This field SHALL be specified if a database is selected. **Deprecation note:** Starting with **JDF** 1.5, use ***FileSpec***(Rejected)
Tolerance ?	double	Ratio of tolerated verification failures to the total number of tests. "0.0" = no failures allowed, "1.0" = all might fail.

8.169 WebInlineFinishingParams

New in JDF 1.3

WebInlineFinishingParams specifies the parameters for Web inline finishing equipment using the **WebInlineFinishing** process.

Resource Properties

Resource Class: Parameter

Example Partition: "SubRun", "WebName", "RibbonName", "WebProduct"

Input of Processes: **WebInlineFinishing**

Table 8.292: WebInlineFinishingParams Resource

NAME	DATA TYPE	DESCRIPTION
FolderProduction *	element	Specifies the Folder setup for newspaper presses:

8.169.1 FolderProduction

Table 8.293: FolderProduction Element

NAME	DATA TYPE	DESCRIPTION
FolderModuleIndex ?	integer	Identifies a particular folder module to be used. @*FolderModuleIndex* SHALL match ***Device***/***Module***/@*ModuleIndex*.
ProductionType = "NonCollect"	enumeration	Indicates whether the product is collected or not. **Allowed values are:** Collect NonCollect

8.170 WindingParams

New in JDF 1.5

The parameters for the **Winding** process

Resource Properties

Resource Class: Parameter

Input of Processes: **Winding**

Table 8.294: WindingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Copies ?	integer	Number of copies in one column that SHOULD be placed on a finished roll. At most one of @*Copies*, @*Diameter* or @*Length* SHOULD be specified.
Diameter ?	double	Outer diameter in points of the finished roll. At most one of @*Copies*, @*Diameter* or @*Length* SHOULD be specified.

Table 8.294: WindingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Fixation ?	NMTOKEN	Method specifying how the ***Component*** is attached to the core. **Values include:** DoubleSidedTape – Tape with adhesive on both sides. Glue Label – One of the output ***Component*** resources (self-Adhesive labels) is used. None – No fixation is used. SingleSidedTape – Tape with adhesive on one side.
Length ?	double	Length in points of the ***Component*** to be placed on a finished roll. At most one of @*Copies*, @*Diameter* or @*Length* SHOULD be specified.

8.171 WireCombBindingParams

This resource describes the details of the **WireCombBinding** process.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***BindingIntent***

Input of Processes: **WireCombBinding**

Table 8.295: WireCombBindingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Brand ?	string	The name of the comb manufacturer (e.g., *Wire-O*®) and the name of the specific item.
Color ?	NamedColor	Determines the color of the comb.
ColorDetails ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If @*ColorDetails* is supplied, @*Color* SHOULD also be supplied.
Diameter ?	double	The comb diameter is determined by the height of the block of Sheets to be bound.
Distance ? Deprecated in JDF 1.2	double	The distance between the "teeth" and the distance between the holes of the prepunched Sheets SHALL be the same. In **JDF** 1.2 and beyond, use the value implied by ***HoleMakingParams***/@*HoleType*.
FlipBackCover = "false" New in JDF 1.1	boolean	The spine is typically hidden between the last page of the ***Component*** and the back cover. Flip the back cover after the wire was "closed" or keep it open. The latter makes sense if further processing is needed (e.g., inserting a CD) before closing the book.
Material ?	enumeration	The material used for forming the wire comb binding. **Allowed values are:** LaqueredSteel TinnedSteel ZincsSteel
Shape = "Single" Modified in JDF 1.6	NMTOKEN	The shape of the wire comb. **Values include:** Single – Each "tooth" is made with one wire. SingleCalendar – Each "tooth" is made with one wire and an extension for hanging the bound product is provided in the center. New in JDF 1.6 Twin – The shape of each "tooth" is made with a double wire. TwinCalendar – The shape of each "tooth" is made with a double wire and an extension for hanging the bound product is provided in the center. New in JDF 1.6
Thickness ?	double	The thickness of the comb material.

Table 8.295: WireCombBindingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
HoleMakingParams ? New in JDF 1.2	refelement	Details of the holes in **WireCombBinding**.

8.172 WrappingParams

New in JDF 1.1

WrappingParams defines the details of **Wrapping**. Details of the material used for **Wrapping** can be found in the ***Media*** resource that is also an input of the **Wrapping** process.

Resource Properties

Resource Class: Parameter

Intent Pairing: ***PackingIntent***

Input of Processes: **Wrapping**

Table 8.296: WrappingParams Resource

NAME	DATA TYPE	DESCRIPTION
WrappingKind ?	enumeration	@*WrappingKind* specifies the wrapping method. **Allowed values are:** Band – The components are wrapped with a band. The material of the band is typically paper, plastic or rubber. LooseWrap – The wrap is loose around the component. ShrinkWrap – The wrap is shrunk around the component.

Subelements

The elements in this chapter are subelements that can occur in multiple Resources. They are not Resources and are therefore never directly linked to Processes.

9.1 Address

Definition of an address. The structure is derived from the vCard format and, therefore, is comprised of all address subtypes (ADR:) of the delivery address of the vCard format. The corresponding XML types of the vCard are quoted in the table.

Element Properties

Element referenced by: ***Location***, ***Contact***, ***Person***

Table 9.1: Address Element

NAME	DATA TYPE	DESCRIPTION
City ?	string	City or locality of address (vCard: ADR:locality).
CivicNumber ?	string	@*CivicNumber* SHALL specify the street number of the street address. If @*CivicNumber* is specified, it SHALL NOT be included in @*Street*.
Country ?	string	Country of address (vCard: ADR:country).
CountryCode ?	NMTOKEN	Country of address. **Values include those from:** ▸ [ISO3166-1:1997]
PostalCode ?	string	Zip code or postal code of address (vCard: ADR:pcode).
PostBox ?	string	Post office address (vCard: ADR:pobox. For example: P.O. Box 101).
Region ?	string	State or province (vCard: ADR:region).
Street ?	string	Street address (vCard: ADR:street).
ExtendedAddress ?	text element	Extended address (vCard: ADR:extadd. For example: Suite 245).

9.2 AutomatedOverPrintParams

AutomatedOverPrintParams provides controls for the automated selection of overprinting of black text or graphics. @*RGBGray2Black* and @*RGBGray2BlackThreshold* in **ColorSpaceConversion**/***ColorSpaceConversionOp*** are used by the **ColorSpaceConversion** process in determining the allocation of RGB values to the black (K) channel. After the **ColorSpaceConversion** process is completed, then the **Rendering** or **Separation** process uses ***AutomatedOverPrintParams*** to determine overprint behavior for the previously determined black (K) channel.

Element Properties

Element referenced by: ***ElementColorParams***, ***RenderingParams***, ***SeparationControlParams***

Table 9.2: AutomatedOverPrintParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
KnockOutCMYKWhite = "false" New in JDF 1.3	boolean	Graphic objects defined in DeviceCMYK, where all colorant values are <0.001 SHALL be knocked out, even when set to overprint and when the PDF overprint mode is set to 1.
OverPrintBlackLineArt = "false"	boolean	Indicates whether overprint SHALL be set to "true" for black line art (i.e., vector elements other than text). If "true", overprint of black line art is applied regardless of any values in the PDL. If "false", @*LineArtBlackLevel* is ignored and PDL line art overprint operators are processed.

SUBELEMENTS

Table 9.2: AutomatedOverPrintParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
LineArtBlackLevel ?	double	A value between 0.0 and 1.0 which indicates the minimum black level for the stroke or fill colors that cause the line art to be set to overprint. Defaults to the value of @*TextBlackLevel*.
OverPrintBlackText = "false"	boolean	Indicates whether overprint SHALL be set to "true" for black text. If "true", overprint of black text is applied regardless of any values in the PDL. If "false", @*TextSizeThreshold* and @*TextBlackLevel* are ignored and PDL text overprint operators are processed.
TextBlackLevel = "1"	double	A value between 0.0 and 1.0 which indicates the minimum black level for the text stroke or fill colors that cause the text to be set to overprint.
TextSizeThreshold ?	integer	Indicates the point size for text below which black text will be set to overprint. For asymmetrically scaled text, the minimum point size between both axes will be used. If not specified, all text is set to overprint.

9.3 BarcodeCompParams

New in JDF 1.3

BarcodeCompParams specifies the technical compensation parameters for barcodes.

Element Properties

Element referenced by: ***BarcodeReproParams***

Table 9.3: BarcodeCompParams Element

NAME	DATA TYPE	DESCRIPTION
CompensationProcess	enumeration	Process that is bar width spread SHALL be compensated for. **Allowed values are:** Printing Platemaking
CompensationValue	double	The width of the bars SHALL be reduced by this amount in micron to compensate for technical spread.

9.4 BarcodeReproParams

New in JDF 1.3

BarcodeReproParams specifies the reproduction parameters for barcodes.

Element Properties

Element referenced by: /***LayoutElementPart***/***BarcodeProductionParams***, ***DeviceMark***

Table 9.4: BarcodeReproParams Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BearerBars ?	enumeration	Indicates the policy how to generate bearer bars. (ITF). **Allowed values are:** None TopBottom Box BoxHMarks
Height ?	double	@*Height* SHALL specify the the height (Y direction) of the bars of a linear barcode in the PDL.
Magnification ?	double	The magnification factor that linear barcodes SHALL be scaled with. **Example:** @*Magnification* > 1 requests thicker lines in the barcode in the resulting PDL.

Table 9.4: BarcodeReproParams Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Masking ?	enumeration	Indicates the properties of the mask around the graphical content of the barcode that masks out all underlying graphics. **Allowed values are:** None – No masking, barcode is put on top of underlying graphics. WhiteBox – An area of the underlying graphics is masked out (the white box) and the barcode is put on top of this masked area. The area of the white box is the box enclosing all artwork of the barcode, excluding optional human readable text. This would enclose bearer bars, quiet zones and non-optional human readable text (UPC and EAN barcodes).
ModuleHeight ?	double	The Y size in micron of an Element of a 2D barcode (e.g., PDF417). For DATAMATRIX, Y Dimension MAY be omitted (X Dimension = Y Dimension).
ModuleWidth ?	double	The X size in micron of an Element of a 2D barcode such as DATAMATRIX or PDF417.
Ratio ?	double	the ratio between the width of the narrow bars and the wide bars for those barcodes where ratio the width of the wide bars and narrow bars MAY vary.
BarcodeCompParams *	element	Parameters for bar width compensation. The total reduction of bar width is the sum of all ***BarcodeCompParams***/@*CompensationValue*.

9.5 Certification

New in JDF 1.6

Certification specifies the certification properties of paper.

Element Properties

Element referenced by: ***MediaIntent***, ***Media***

Table 9.5: Certification Element

NAME	DATA TYPE	DESCRIPTION
Claim ?	string	Name of the certification as defined by the issuing organization. **Values include:** FSC 100% FSC Mix 70% FSC Mix Credit FSC Recycled 85% FSC Recycled Credit PEFC nn% PEFC Certified PEFC Recycled
Identifier ?	string	Certification identification number as defined by the issuing organization.
Organization ?	NMTOKEN	Identifier of the issuing organization: **Values include:** FSC – Forest Stewardship Council PEFC – The Programme for the Endorsement of Forest Certification IFCC – Sustainable Forest Management Requirements CFCC – China's National Forest Certification System

9.6 ColorantAlias

ColorantAlias is an element that specifies a replacement colorant name string to be used instead of one or more named colorant strings. For example, ***SeparationSpec***/@*Name* = "Pantone 135 C", "PANTONE 135" and @*ReplacementColorantName* = "PANTONE 135 C" maps string values: "Pantone 135 C" and "PANTONE 135" to the string value: "PANTONE 135 C".

Element Properties

Element referenced by: *ColorantControl*, *ElementColorParams*

Table 9.6: ColorantAlias Element

NAME	DATA TYPE	DESCRIPTION
RawNames ? New in JDF 1.4	hexBinaryL-ist	Whitespace-separated list of hexBinary values. Each token represents the original 8-bit byte stream of the color specified in *SeparationSpec*. Used to transport the original byte representation of a color name when moving **JDF** tickets between computers with different locales. Exactly one token SHALL be specified for each *SeparationSpec* in this *ColorantAlias*. The order of tokens SHALL be identical to the order of the related *SeparationSpec*.
ReplacementColorantName	string	The value of the colorant name string to be substituted for the colorant name in the *SeparationSpec* Resource list.
SeparationSpec + Modified in JDF 1.2	element	The names of the colorants to be replaced in PDL files.

Example 9.1: ColorantAlias/@RawNames

New in JDF 1.4

```
<ColorantAlias Class="Parameter" ID="r000004" RawNames="4772FC6E 6772FC6E"
       ReplacementColorantName="Green" Status="Available">
    <!-- ColorantAlias that maps the additional representation (grün, Grün)
       to the predefined separation Green -->
    <SeparationSpec Name="Grün"/>
    <SeparationSpec Name="grün"/>
</ColorantAlias>
```

9.7 ColorCorrectionOp

Element Properties

Element referenced by: *ColorCorrectionParams*, *ElementColorParams*

Table 9.7: ColorCorrectionOp Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AdjustContrast ? New in JDF 1.2	double	Specifies the L*a*b* contrast adjustment in the range -100 (minimum contrast for the system (i.e., a solid midtone gray color)) to + 100 (maximum contrast for the system (i.e., either use full color (the maximum is restricted by the system ink limit) or no color for each of Cyan, Magenta, Yellow and Black)). Increasing the contrast value increases the variation between light and dark areas and decreasing the contrast value decreases the variation between light and dark areas. See explanation above.
AdjustCyanRed ? New in JDF 1.2	double	Specifies the L*a*b* adjustment in the Cyan/Red axis in the range -100 (maximum Cyan cast for the system) to + 100 (maximum Red cast for the system) while maintaining lightness. See explanation above.
AdjustHue ? New in JDF 1.2	double	Specifies the change in the L*a*b* hue in the range -180 to +180 of all colors by the specified number of degrees of the color circle. See explanation above.
AdjustLightness ? New in JDF 1.2	double	Specifies the decrease or increase of the L*a*b* lightness in the range -100 (minimum lightness for the system (i.e., black)) to + 100 (maximum lightness for the system (i.e., white)). Increasing the lightness value causes the output to appear lighter and decreasing the lightness value causes the output to appear darker. See explanation above.
AdjustMagentaGreen ? New in JDF 1.2	double	Specifies the L*a*b* adjustment in the Magenta/Green axis in the range -100 (maximum Magenta cast for the system) to + 100 (maximum Green cast for the system) while maintaining lightness. See explanation above.

Table 9.7: ColorCorrectionOp Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
AdjustSaturation ? New in JDF 1.2	double	Specifies the increase or decrease of the L*a*b* color saturation in the range -100 (minimum saturation for the system) to +100 (maximum saturation for the system). Increasing the saturation value causes the output to contain more vibrant colors and decreasing the saturation value causes the output to contain more pastel and gray colors. See explanation above.
AdjustYellowBlue ? New in JDF 1.2	double	Specifies the L*a*b* adjustment in the Yellow/Blue axis in the range -100 (maximum Yellow cast for the system) to + 100 (maximum Blue cast for the system) while maintaining lightness. See explanation above.
ObjectTags ? New in JDF 1.4	NMTOKENS	Tags associated with individual objects that this ***ColorCorrectionOp*** SHALL be applied to. Each tag specified in @*ObjectTags* is logically anded with the object type(s) specified by @*SourceObjects*, enabling first qualification by object type (such as image), and then tags associated with those objects. The values of @*ObjectTags* depends on the PDL that the color correction is applied to. @*ObjectTags* SHALL apply only to objects whose tag pool includes all the tags in the value of @*ObjectTags*.
SourceObjects = "All"	enumerations	Identifies which class(es) of incoming graphical objects will be operated on. **Allowed values are from:** ▸ SourceObjects**.**
FileSpec (AbstractProfile) ? New in JDF 1.2	refelement	A ***FileSpec*** Resource pointing to an abstract ICC profile that has been devised to apply a preference adjustment. See explanation of adjustment at the beginning of this section.
FileSpec (DeviceLinkProfile) ? New in JDF 1.2	refelement	A ***FileSpec*** Resource pointing to an ICC profile that describes the characterization of an abstract profile for specifying a preference adjustment. See explanation of adjustment at the beginning of this section.

9.8 ColorMeasurementConditions

New in JDF 1.1

This Resource contains information about the specific measurement conditions for spectral or densitometric color measurements. Spectral measurements refer to ▸ [CIE 15:2004] and ▸ [ISO13655:1996]. The default measurement conditions for spectral measurements are illuminant D50 and 2 degree observer.

Density measurements refer to ▸ [ISO5-3:1995] and ▸ [ISO5-4:1995]. The default measurement conditions for densitometric measurements are density standard ISO/ANSI Status T, calibration to absolute white and using no polarization filter.

Element Properties

Element referenced by: ***CIELABMeasuringField***, ***Color***, ***DynamicField***, ***Media***

Table 9.8: ColorMeasurementConditions Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
DensityStandard = "ANSIT"	enumeration	Density filter standard used during density measurements. **Allowed values are:** ANSIA – ANSI Status A ANSIE – ANSI Status E ANSII – ANSI Status I ANSIT – ANSI Status T DIN16536 DIN16536NB
Illumination = "D50"	enumeration	Illumination used during spectral measurements. **Allowed values are:** D50 D65 Unknown

Table 9.8: ColorMeasurementConditions Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
InkState ?	enumeration	State of the ink during color measurements. **Allowed values are:** Dry Wet NA Deprecated in JDF 1.2
Instrumentation ?	string	Specific instrumentation used for color measurements (e.g., manufacturer, model number and serial number).
MeasurementFilter ?	enumeration	Optical Filter used during color measurements. **Allowed values are:** None – No filter used. Pol – Polarization filter used UV – Ultraviolet cut filter used
Observer = "2"	integer	CIE standard observer function (2 degree and 10 degree) used during spectral measurements. Values are in degree (2 or 10).
SampleBacking ?	enumeration	Backing material used behind the sample during color measurements. **Allowed values are:** Black White NA Deprecated in JDF 1.2
WhiteBase ?	enumeration	Reference for white calibration used for density measurements. **Allowed values are:** Absolute – Means the instrument is calibrated to a Device-specific calibration target (absolute white) for absolute density measurements. Paper – Means the instrument is calibrated relative to paper white

9.9 ColorSpaceConversionOp

The ***ColorSpaceConversionOp*** Element identifies a type of object, defines the source color space for that type of object, and specifies the behavior of the conversion operation for that type of object. Many of these Attribute descriptions refer to ICC Color Profiles ▸ [ICC.1]. See also the International Color Consortium (ICC) Web site at http://www.color.org.

Element Properties

Element referenced by: ***ColorSpaceConversionParams***, ***ElementColorParams***

Table 9.9: ColorSpaceConversionOp Element (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
IgnoreEmbeddedICC = "false" Deprecated in JDF 1.4	boolean	If "true", specifies that embedded source ICC profiles SHALL be ignored as part of the selection criteria for this ***ColorSpaceConversionOp***. If "true", ***FileSpec*** (PDLSourceProfile) is ignored. **Deprecation note:** Starting with **JDF** 1.4, use the new @*SourceCS* values of "DeviceGray", "DeviceRGB", or "DeviceCMYK" to select objects having an uncharacterized color space. Use "Gray", "RGB", or "CMYK" to select objects regardless of whether they are characterized. Use "ICCGray", "ICCRGB", or "ICCCMYK" to select only characterized objects.
ObjectTags ? New in JDF 1.4	NMTOKENS	Tags associated with individual objects that this ***ColorSpaceConversionOp*** SHALL be applied to. Each tag specified in @*ObjectTags* is logically anded with the object type(s) specified by @*SourceObjects*, enabling first qualification by object type (such as image), and then tags associated with those objects. The values of @*ObjectTags* depends on the PDL that the color space conversion is applied to. @*ObjectTags* SHALL apply only to objects whose tag pool includes all the tags in the value of @*ObjectTags*.

Table 9.9: ColorSpaceConversionOp Element (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
Operation ? Modified in JDF 1.2	enumeration	Controls which of five functions the color space conversion utility performs. **Allowed values are:** Convert – Transforms graphical elements to final target color space. Tag – Associates appropriate working space profile with uncharacterized graphical element. Untag – Removes all profiles and color characterizations from graphical elements. Retag – Equivalent to a sequence of "Untag" –> "Tag", where the "Untag" –> "Tag" sequence is only applied to those objects selected by this ***ColorSpaceConversionOp***. ConvertIgnore – Equivalent to a sequence of "Untag" –> "Convert". **Constraint:** @*Operation* SHALL be specified in the context of ***ColorSpaceConversionParams***/***ColorSpaceConversionOp*** and SHALL NOT be specified in the context of ***ElementColorParams***/***ColorSpaceConversionOp***.
PreserveBlack = "false" New in JDF 1.1	boolean	Controls how the tints of black (K in CMYK) are to be handled. If @*PreserveBlack* is "false", these colors are processed through the standard ICC workflow. If @*PreserveBlack* is "true", these colors are to be converted into other shades of black. The algorithm is implementation-specific.
RenderingIntent = "ColorSpaceDependent" Modified in JDF 1.3	enumeration	Identifies the rendering intent to be applied when rendering the objects selected by this ***ColorSpaceConversionOp***. **Allowed value is from:** ▸ RenderingIntent.
RGBGray2Black = "false" Modified in JDF 1.2	boolean	This feature controls what happens to gray values (R = G = B) when converting from RGB to CMYK or the incoming graphical objects indicated by @*SourceObjects*. In the case of MS Office applications and screen dumps, there are a number of gray values in the images and line art. Printers do not want to have CMY under the K because it creates registration problems. They prefer to have K only, so the printer converts the gray values to K. Gray values that exceed the @*RGBGray2BlackThreshold* are not converted. @*RGBGray2Black* and @*RGBGray2BlackThreshold* are used by the **ColorSpaceConversion** Process in determining how to allocate RGB values to the black (K) channel. After the **ColorSpaceConversion** Process is completed, the **Rendering** Process uses ***AutomatedOverPrintParams*** to determine overprint behavior for the previously determined black (K) channel.
RGBGray2BlackThreshold = "1" New in JDF 1.2	double	A value between "0.0" and "1.0" which specifies the threshold value above which the Device SHALL NOT convert gray (R = G = B) to black (K only) when @*RGBGray2Black* is "true". So a "0" value means convert only R = G = B = 0 (black) to K only. A value of "1" specifies that all values of R = G = B are converted to K if @*RGBGray2Black* = "true".
SourceCS Modified in JDF 1.3	enumeration	Identifies which of the incoming color spaces SHALL be operated on. **Allowed values are from:** ▸ Table 9.10 SourceCS Attribute Values**.** **Note:** See ▸ Table 9.11 Mapping of SourceCS enumeration values to color spaces in the most common input file formats.
SourceObjects = "All"	enumerations	List of object Classes that identifies which incoming graphical objects will be operated on. **Allowed values are from:** ▸ SourceObjects**.**
SourceRenderingIntent ? New in JDF 1.2	enumeration	Identifies the rendering intent transform elements to be selected from the source profile that will be used to interpret objects of type identified by the @*SourceObjects* and @*SourceCS* Attributes. **Default value is from:** @*RenderingIntent*. **Allowed value is from:** ▸ RenderingIntent. **Note:** The @*SourceRenderingIntent* will pertain to the source profile used in a particular **ColorSpaceConversion** Process (e.g., sources can be the native original color space, an intermediate working color space or an reference output simulation color space).

Table 9.9: ColorSpaceConversionOp Element (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
DeviceNSpace ? New in JDF 1.2	element	***DeviceNSpace*** Resource that describe the DeviceN color space on which to operate when @*SourceCS* = "DeviceN". Individual colorant definitions for the colorant names given in ***DeviceNSpace*** are provided in the ***ColorantControl***/***ColorPool*** resource, which SHALL also be present
FileSpec (AbstractProfile) ? New in JDF 1.2 Deprecated in 1.6	refelement	A ***FileSpec*** Resource pointing to an ICC profile ▸ [ICC.1] that describes the characterization of an Abstract Profile for specifying a preference adjustment. **Deprecation note:** In **JDF** 1.6 and beyond, preference adjustment SHOULD be specified in a **ColorCorrection** process and the abstract profiles should be provided as ***ColorCorrectionOp***/***FitPolicy***(AbstractProfile) .
FileSpec (DeviceLinkProfile) * New in JDF 1.3 Modified in JDF 1.4	refelement	A ***FileSpec*** Resource pointing to an ICC profile file [ICC.1] that contains a Device Link transform. The Source colorspace of the referenced Device Link transform SHOULD match that of the profile identified by ***FileSpec*** (PDLSourceProfile) and the destination color space SHOULD match that of the destination profile identified by ***ColorSpaceConversionParams*** (if specified). Multiple Device Link ICC transforms should be provided where each transform specifies a different rendering intent. This is important in the case where multiple PDL content objects of the colorspace specify different rendering intents. **Note:** An ICC Device Link profile contains only one transform with one color rendering intent. **Note:** Although the ICC specification refers to all ICC files as "profiles", a Device Link in actuality represents a single transform to be applied, and not a profile of a particular device colorspace. Thus these files are referred to as Device Link transforms in this specification. **Modification note:** Starting with **JDF** 1.4, multiple ***FileSpec*** (DeviceLinkProfile) Elements are allowed.
FileSpec (PDLSourceProfile) ? New in JDF 1.4	refelement	A ***FileSpec*** Resource describing an ICC profile that describes a profiled source space that this ***ColorSpaceConversionOp*** should operate on. When present, only objects that specify the @*SourceCS* along with the specified profile are selected. **Note:** The ***FileSpec***/@*UID* Attribute can often be used to positively identify an ICC profile referenced in a PDL file when available (***FileSpec***/@*UID* corresponds to the ICC ProfileID field). In addition, ***FileSpec***/@*CheckSum* may be used when only a checksum of the entire profile is available.
FileSpec (SourceProfile) ?	refelement	A ***FileSpec*** Resource pointing to an ICC profile ▸ [ICC.1] that describes the assumed source color space. If ***FileSpec*** (SourceProfile) is specified, it SHALL be used as the profile for the source object's color space during a "Convert", "Tag", or "Retag" operation, as specified by @*Operation*. ***FileSpec*** (SourceProfile) SHALL be present for "Tag" or "Retag" operations, as specified by @*Operation*.
ScreenSelector ?	element	Links this ***ColorSpaceConversionOp*** to a given screening.
SeparationSpec * New in JDF 1.2	element	***SeparationSpec*** resource(s) defining on which separation(s) to operate when @*SourceCS* = "Separation".

9.9.1 SourceCS Attribute Values

Table 9.10: SourceCS Attribute Values (Sheet 1 of 2)

VALUE	DESCRIPTION
All New in JDF 1.4	Operates on all source colorspaces. This is useful when specifying a Convert operation using all PDL source-supplied characterizations with a **JDF**-supplied final target device profile.
CalGray New in JDF 1.3	defines a calibrated Device independent representation of Gray.
Calibrated New in JDF 1.2	Operates on "CalGray" and "CalRGB" color spaces.
CalRGB New in JDF 1.3	defines a calibrated based Device independent representation of RGB. **Note: JDF** 1.1 defined that "CalRGB" be treated as "RGB", "CalGray" as "Gray" and "ICCBased" color spaces as one of "Gray", "RGB" or "CMYK" depending on the number of channels.
CIEBased New in JDF 1.2	Operates on CIE-based color spaces ("CIEBasedA", "CIEBasedABC", "CIEBasedDEF" and "CIEBasedDEFG").
CMYK	Operates on all CMYK color spaces. This includes both characterized and uncharacterized CMYK color spaces.
DeviceCMYK New in JDF 1.4	Operates on uncharacterized CMYK color spaces.
DeviceGray New in JDF 1.4	Operates on uncharacterized Gray color spaces.
DeviceN New in JDF 1.2	Identifies the source color encoding as a "DeviceN" color space. The specific "DeviceN" color space to operate on is defined in the ***DeviceNSpace*** Resource. If "DeviceN" is specified, then the ***DeviceNSpace*** and ***ColorantControl***/***ColorPool*** refelements SHALL also be present.
DeviceRGB New in JDF 1.4	Operates on uncharacterized RGB color spaces.
DevIndep	Operates on Device independent color spaces (equivalent to "Calibrated", "CIEBased", "ICCBased", "Lab" or "YUV").
Gray	Operates on all Gray color spaces. This includes both characterized and uncharacterized Gray color spaces.
ICCBased New in JDF 1.2	Operates on color spaces defined using ICC profiles. The "ICCBased" value includes EPS, TIFF or PICT files with embedded ICC profiles. See ▸ [ICC.1]. Includes PDF Device color spaces that are characterized in ▸ Footnote 2 following ▸ Table 9.11 Mapping of SourceCS enumeration values to color spaces in the most common input file formats.
ICCCMYK New in JDF 1.3	Operates on ICCBased color spaces with ICC CMYK profiles or DeviceCMYK having an ICC-based characterization. See ▸ Footnote 2 following ▸ Table 9.11 Mapping of SourceCS enumeration values to color spaces in the most common input file formats.
ICCGray New in JDF 1.3	Operates on ICCBased color spaces with ICC gray profiles or DeviceGray having an ICC-based characterization. See ▸ Footnote 2 following ▸ Table 9.11 Mapping of SourceCS enumeration values to color spaces in the most common input file formats.
ICCLAB New in JDF 1.3	Operates on ICC based Device independent representation of LAB
ICCRGB New in JDF 1.3	Operates on ICCBased color spaces with ICC RGB profiles or DeviceRGB having an ICC-based characterization. See ▸ Footnote 2 following ▸ Table 9.11 Mapping of SourceCS enumeration values to color spaces in the most common input file formats.
Lab New in JDF 1.2	Operates on "Lab".

Table 9.10: SourceCS Attribute Values (Sheet 2 of 2)

VALUE	DESCRIPTION
RGB Modified in JDF 1.2	Operates on all RGB color spaces. This includes both characterized and uncharacterized RGB color spaces.
Separation New in JDF 1.2	Operates on "Separation" color spaces (spot colors). The specific separation(s) to operate on are defined in the ***SeparationSpec*** Resource(s). If no ***SeparationSpec*** is defined, the operation will operate on all the separation color spaces in the input ***RunList***.
YUV New in JDF 1.2	Operates on "YUV" (Also known as YCbCr). See ▸ [CCIR601-2]

Notes: "DevIndep" has been retained for backwards compatibility with **JDF** 1.1 and because there will probably be cases where the same processing is to be applied to all Device independent spaces. An equivalent "DevDep" has not been added because it's less likely that all Device-dependent spaces are to be treated in the same way. The following table summarizes how the @*SourceCS* Attribute is mapped to/from different file formats.

Table 9.11: Mapping of SourceCS enumeration values to color spaces in the most common input file formats (Sheet 1 of 2)

SOURCECS	FILE FORMAT	COLOR SPACE
Calibrated	PDF[1]	CalGray, CalRGB
	PostScript[1]	n/a
	TIFF	n/a
CIEBased	PDF[1]	n/a
	PostScript[1]	CIEBasedABC, CIEBasedA, CIEBasedDEF and CIEBasedDEFG
	TIFF	n/a
CMYK	PDF[1]	DeviceCMYK[2] PDF ICCBased color spaces with ICC CMYK profiles. CIEBasedDEFG spaces that resolve to a characterized CMYK space.
	PostScript[1]	DeviceCMYK
	TIFF	PhotometricInterp = 5 Samples per pixel = 4
DeviceCMYK New in JDF 1.4	PDF[1]	DeviceCMYK[2]
	PostScript[1]	DeviceCMYK
	TIFF	PhotometricInterp = 5 Samples per pixel = 4
DeviceGray New in JDF 1.4	PDF[1]	DeviceGray[2]
	PostScript[1]	DeviceGray
	TIFF	PhotometricInterp = 0 or 1
DeviceN	PDF[1]	DeviceN
	PostScript[1]	DeviceN
	TIFF	PhotometricInterp = 5, Samples per pixel = N

Table 9.11: Mapping of SourceCS enumeration values to color spaces in the most common input file formats (Sheet 2 of 2)

SOURCECS	FILE FORMAT	COLOR SPACE
DeviceRGB New in JDF 1.4	PDF[1]	DeviceRGB[2]
	PostScript	DeviceRGB
	TIFF	PhotometricInterp = 2
Gray	PDF[1]	DeviceGray[2] PDF ICCBased color spaces with ICC Gray profiles. CIEBasedA spaces that resolve to a characterized Gray space.
	PostScript[1]	DeviceGray
	TIFF	PhotometricInterp = 0 or 1
ICCBased ICCCMYK ICCGray ICCLAB ICCRGB	PDF[1]	ICCBased DeviceGray[2], DeviceCMYK[2], DeviceRGB[2]
	PostScript[1]	n/a
	PostScript/ EPS	The EPS file has an embedded ICC profile.
	TIFF	The TIFF file has an embedded ICC profile.
LAB	PDF[1]	LAB
	PostScript[1]	n/a
	TIFF	PhotometricInterp = 8 (CIELAB 1976 “normal” encoding) or PhotometricInterp = 9 (CIELAB 1976 using ICC profile v2 encoding).
RGB	PDF[1]	DeviceRGB[2] PDF ICCBased color spaces with ICC RGB profiles. CIEBasedDEF spaces that resolve to a characterized RGB space.
	PostScript	DeviceRGB
	TIFF	PhotometricInterp = 2
Separation	PDF[1]	Separation
	PostScript[1]	Separation
	TIFF	PhotometricInterp = 5 (Applies only to one of the planes in the separated image.)
YUV	PDF[1]	n/a
	PostScript[1]	n/a
	TIFF	PhotometricInterp = 6

1. Where a "Pattern" or "Indexed" color space has been used, the base color space is used to determine whether to apply this operation.
2. In PDF, DeviceCMYK, DeviceRGB, and DeviceGray source colorspaces can be characterized through providing a DefaultCMYK, DefaultRGB, or DefaultGray resource specifying a profile to be associated with source objects in that color space. In which case, the resulting color space is considered characterized by **JDF** operations.

9.10 ComChannel

A communication channel to a person or company such as an email address, phone number or fax number.

SUBELEMENTS

Element Properties

Element referenced by: *Contact*, *Person*

Table 9.12: ComChannel Element

NAME	DATA TYPE	DESCRIPTION
ChannelType Modified in JDF 1.5	enumeration	Type of the communication channel. **Allowed values are:** ComputerName New in JDF 1.5 Email – Email address. Fax – Fax machine. JMF – **JMF** messaging channel. Mobile – Mobile phone. New in JDF 1.5 Phone – Telephone number. Starting with **JDF** 1.5, this SHOULD be restricted to land line phones.Modified in JDF 1.5 WWW – WWW home page or form. PrivateDirectory – Account of a registered customer of a certain service. (The list of the registered accounts is maintained by the service vendor). The @ *ChannelTypeDetails* attribute has the name of the private directory service vendor. InstantMessaging – IM service address. The @*ChannelTypeDetails* attribute has the name of the IM service vendor
ChannelTypeDetails ? New in JDF 1.2 Modified in JDF 1.5	NMTOKEN	Description of the value of the @*ChannelType* Attribute. Consumer treats this value as the service vendor name if @*ChannelType* is "PrivateDirectory" or "InstantMessaging". **Values include those from:** ▸ Table 9.13 ChannelTypeDetails Attribute – predefined values for certain ChannelType values**.**
ChannelUsage ? New in JDF 1.2	NMTOKENS	Communication channel usage. **Values include:** Business – Business purpose usage (e.g., office phone number, fax). Private – Private purpose usage (e.g., private phone number, fax, Email). DayTime – Office hours in the time zone of the recipient. NightTime – Out-of-office hours in the time zone of the recipient. WeekEnd – Out-of-office hours in the time zone of the recipient.
Locator Modified in JDF 1.2	string	Locator of this type of channel in a form, such as a phone number, a URL or an Email address. If a URL is defined for the @*ChannelType*, it is RECOMMENDED to use the URL syntax specified in ▸ [RFC2368] for "mailto" URLs, ▸ [RFC3966] for "tel" URLs and ▸ [RFC3986] for URLs in general, as follows: **Values include:** "mailto:a@b.com" – instead of "a@b.com" if @*ChannelType* = "Email", "tel:+49-69-92058800" – if @*ChannelType* = "Phone" and "tel:+49.6151.155.299" – if @*ChannelType* = "Fax".

9.10.1 ChannelTypeDetails Attribute

Table 9.13: ChannelTypeDetails Attribute – predefined values for certain ChannelType values

CHANNELTYPE VALUE	CHANNELTYPEDETAILS VALUE	DESCRIPTION
"Phone"	"Secure"	Secure phone line.
	"ISDN"	ISDN line telephone number.
"WWW"	"Form"	Upload form.
	"Target"	Upload target URL.

Example 9.2: ComChannel for Telephone

```
<ComChannel Class="Parameter" ID="cc000004" ChannelType="Phone"
       ChannelTypeDetails="Mobile" ChannelUsage="Business"
       Locator="tel:+44-07808-907-919" Status="Available"/>
```

Example 9.3: ComChannel for Instant Messaging

```
<ComChannel Class="Parameter" ID="cc000004" ChannelType="InstantMessaging"
       ChannelTypeDetails="MyIMService" ChannelUsage="Private"
       Locator="123456789" Status="Available"/>
```

9.11 Comment

The ***Comment*** element can be used to provide human readable text that pertains to the parent element.

Element Properties

Element referenced by: The element can be referenced by all other elements

Table 9.14: Comment Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AgentName ? New in JDF 1.3	string	The name of the Agent application that created the ***Comment***. Both the company name and the product name MAY appear, and SHOULD be consistent between versions of the application.
AgentVersion ? New in JDF 1.3	string	The version of the Agent application that created the ***Comment***. The format of the version string MAY vary from one application to another, but SHOULD be consistent for an individual application.
Attribute ? New in JDF 1.1	NMTOKEN	Name of the Attribute in the parent Element of the ***Comment*** Element that this ***Comment*** refers to. @*Attribute* SHOULD include the namespace prefix if the Attribute is in a non-**JDF** namespace. If omitted, the ***Comment*** refers to the entire parent Element. @*Attribute* MAY be used to provide instructions for setting an Attribute or to provide additional human readable information. For instance the name for ***Media***/@*Dimensions* or the name ***Media***/@*Weight* MAY be localized. **Note:** @*Attribute* MAY be specified for Attributes of the parent that are not explicitly set in that Element. This allows human readable descriptions of Attribute settings during the setup of a Job.
Author ? New in JDF 1.3	string	Human readable text that identifies the person who created the ***Comment***. See also @*PersonalID*.
Box ?	rectangle	The rectangle that is associated with the comment. The coordinate system of the rectangle is the same as the coordinate system defined in the @*Path* Attribute.
ID ? New in JDF 1.3	ID	Identification that is used to reference the ***Comment***.
Language ?	language	Human readable language of the ***Comment***.

Table 9.14: Comment Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Name = "Description" Modified in JDF 1.4	NMTOKEN	A name that defines the usage of a comment. For example, it could determine whether two comments are intended to fill two distinct fields of a user interface. **Values include:** Description – Human readable description, which is REQUIRED if the ***Comment*** Element is REQUIRED in a given context, as is the case in the ***Notification*** element (see ▸ Table 3.36 Notification Audit Element). DeviceText – Human readable description created by the Device that provides details beyond the value of @*StatusDetails*. New in JDF 1.4 Instruction – Message to the operator that contains information regarding the processing of the job. New in JDF 1.2 JobDescription – Description of the job. A ***Comment*** element that contains @*Name* = "JobDescription" SHALL be specified only in a **JDF** node or ***CustomerInfo*** resource. See also ***CustomerInfo***/@*CustomerJobName* in ▸ Section 8.32 CustomerInfo. New in JDF 1.2 OperatorText – Message from the operator that contains information regarding the processing of the job. New in JDF 1.2 Orientation – Description of the orientation of a ***PhysicalResource***. TemplateDescription – Description of the Job ticket template. A ***Comment*** Element that contains @*Name* = "TemplateDescription" SHALL be specified only in the root **JDF** node. New in JDF 1.2 UserText – Message to a user that contains information regarding the processing of the Job New in JDF 1.2
Path ?	PDFPath	Description of the area that the comment is associated with in the coordinate system of the Element where the path resides. In the case of ***PhysicalResource***s, ***Layout*** Resources and Resources that are related to ***Layout***, @*Path* is defined within the coordinate system of the Resource in which it resides. For example, if the comment is inserted in an ***ExposedMedia*** Resource that describes a plate, the path refers to the plate coordinate system. In all other cases, it is defined in the process coordinate system of the **JDF** node that contains the Element that the ***Comment*** Element containing @*Path* is defined in. Note that there are cases where a coordinate system is not available and therefore defining @*Path* is NOT RECOMMENDED (e.g., ***CustomerInfo***).
PersonalID ? New in JDF 1.6	NMTOKEN	Machine readable identifier of the ***Employee*** that entered the comment. When the ***Comment*** is created by a person with a known ***Contact***/@*UserID*, then @*PersonalID* SHOULD contain the value of ***Contact***/@*UserID*. See also @*Author*
TimeStamp ? New in JDF 1.3	dateTime	Describes the date and time when the ***Comment*** was created.
	text	Body of the comment. Note that whitespace is preserved only as generic whitespace in XML. Applications that display comments to the user SHOULD maintain whitespace.

Example 9.4: Multi-line Comment

The following example shows a multi-line comment with whitespace.

```
<Comment AgentName="CIP4 JDF Writer Java" AgentVersion="1.5 BLD 93"
  ID="c_000004" Name="Instruction">Multiline text
     with white space


and empty lines
</Comment>
```

9.12 ConvertingConfig

New in JDF 1.4

Modified in JDF 1.5

The *ConvertingConfig* element describes a range of sheet sizes that can be used for optimizing a **DieLayoutProduction** or a press sheet for **SheetOptimizing**.
Modification note: Moved *ConvertingConfig* subelement from ▸ Chapter 8 Resources.

Element Properties

Element referenced by: *DieLayoutProductionParams*, *SheetOptimizingParams*

Table 9.15: ConvertingConfig Element

NAME	DATA TYPE	DESCRIPTION
MarginBottom ?	double	The bottom margin for positioning the layout on the Sheet.
MarginLeft ?	double	The left margin for positioning the layout on the Sheet.
MarginRight ?	double	The right margin for positioning the layout on the Sheet.
MarginTop ?	double	The top margin for positioning the layout on the Sheet.
SheetHeight ? Modified in JDF 1.5	DoubleRange	The minimum to maximum Sheet height (pt). **Modification note:** Starting in **JDF** 1.5, @*SheetHeight* is optional.
SheetWidth ? Modified in JDF 1.5	DoubleRange	The minimum to maximum Sheet width (pt). **Modification note:** Starting in **JDF** 1.5, @*SheetWidth* is optional.
Device *	refelement	The target devices (printing press, die cutter and further finishing equipment) corresponding to this configuration. Typically only the type of Device would be used (e.g., the model of the die cutter). If multiple Devices are specified, then the other Attributes in this Element SHALL apply to a production configuration that uses all specified Devices.
Media * New in JDF 1.5	refelement	Reference to zero or more *Media* elements that are candidates for optimization. **Note:** This element allows a media database savvy consumer to loop over an explicit range of known materials rather than providing results based on a range of dimensions only.

9.13 CostCenter

This Element describes an individual area of a company that has separated accounting.

Element Properties

Element referenced by: *Notification*, *ResourceInfo*, *JobPhase*, *Employee*, *Device*

Table 9.16: CostCenter Element

NAME	DATA TYPE	DESCRIPTION
CostCenterID	string	Identification of the cost center
Name ?	string	Name of the cost center.

9.14 Crease

Crease defines an individual crease line on a *Component*.

Element Properties

Element referenced by: *CreasingParams*,

Table 9.17: Crease Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Depth ? New in JDF 1.2	double	Depth of the crease, measured in microns [µm].

SUBELEMENTS

Table 9.17: Crease Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
RelativeStartPosition ? New in JDF 1.2	XYPair	Relative starting position of the tool. The @*RelativeStartPosition* is always based on the complete size of the input ***Component*** and not on the size of an intermediate state of the folded Sheet. The allowed value range is from 0.0 to 1.0 for each component of the XYPair, which specifies the full size of the input ***Component***.
RelativeTravel ? New in JDF 1.2	double	Relative distance of the reference edge relative to @*From* in the coordinates of the incoming ***Component***. The @*RelativeTravel* is always based on the complete size of the input ***Component*** and not on the size of an intermediate state of the folded Sheet. The allowed value range is from 0.0 to 1.0, which specifies the full length of the input component.
RelativeWorkingPath ? New in JDF 1.2	XYPair	Relative working path of the tool beginning at @*RelativeStartPosition*. Since the tools can only work parallel to the edges, one coordinate SHALL be zero. The @*RelativeWorkingPath* is always based on the complete size of the input ***Component*** and not on the size of an intermediate state of the folded Sheet. The allowed value range is from 0.0 to 1.0 for each component of the XYPair, which specifies the full size of the input ***Component***.
StartPosition ? Modified in JDF 1.2	XYPair	Starting position of the tool. If both @*StartPosition* and @*RelativeStartPosition* are specified, @*RelativeStartPosition* is ignored. At least one of @*StartPosition* or @*RelativeStartPosition* SHALL be specified.
Travel ? New in JDF 1.2	double	Distance of the reference edge relative to @*From*. If both @*Travel* and @*RelativeTravel* are specified, @*RelativeTravel* is ignored. At least one of @*Travel* or @*RelativeTravel* SHALL be specified.
WorkingDirection ? Modified in JDF 1.5	enumeration	Direction from which the tool is working. **Allowed value is from:** ▸ WorkingDirection. **Modification note:** Starting in **JDF** 1.5, @*WorkingDirection* is optional
WorkingPath ?	XYPair	Working path of the tool beginning at @*StartPosition*. Since the tools can only work parallel to the edges, one coordinate SHALL be zero. If both @*WorkingPath* and @*RelativeWorkingPath* are specified, @*RelativeWorkingPath* is ignored. At least one of @*WorkingPath* or @*RelativeWorkingPath* SHALL be specified.

9.15 Cut

Cut describes one straight cut with an arbitrary tool.

Element Properties

Element referenced by: ***CuttingParams***,

Table 9.18: Cut Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
CutWidth ? New in JDF 1.4	double	Width in points of u-shaped knife, saw blade, etc.
LowerRibbonName ? New in JDF 1.5	NMTOKEN	@*RibbonName* of the Ribbon on the side of the cut that corresponds to a lower X value of @*StartPosition* or @*RelativeStartPosition*.
RelativeStartPosition ? New in JDF 1.2	XYPair	Relative starting position of the tool. The @*RelativeStartPosition* is always based on the complete size of the input ***Component*** and not on the size of an intermediate state of the folded Sheet. The allowed value range is from 0.0 to 1.0 for each component of the XYPair, which specifies the full size of the input ***Component***.
RelativeWorkingPath ? New in JDF 1.2	XYPair	Relative working path of the tool beginning at @*RelativeStartPosition*. Since the tools can only work parallel to the edges, one coordinate SHALL be zero. @*RelativeWorkingPath* is always based on the complete size of the input ***Component*** and not on the size of an intermediate state of the folded Sheet. The allowed value range is from 0.0 to 1.0 for each component of the XYPair, which specifies the full size of the input ***Component***.

Table 9.18: Cut Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
StartPosition ? Modified in JDF 1.2	XYPair	Starting position of the tool. If both @*StartPosition* and @*RelativeStartPosition* are specified, @*RelativeStartPosition* is ignored. At least one of @*StartPosition* or @*RelativeStartPosition* SHALL be specified.
UpperRibbonName ? New in JDF 1.5	NMTOKEN	@*RibbonName* of the Ribbon on the side of the cut that corresponds to a higher X value of @*StartPosition* or @*RelativeStartPosition*.
WorkingDirection ? Modified in JDF 1.5	enumeration	Direction from which the tool is working. **Allowed value is from:** ▸ WorkingDirection. **Modification note:** Starting in **JDF** 1.5, @*WorkingDirection* is optional.
WorkingPath ? Modified in JDF 1.2	XYPair	Working path of the tool beginning at @*StartPosition*. Since the tools can only work parallel to the edges, one coordinate SHALL be zero. If both @*WorkingPath* and @*RelativeWorkingPath* are specified, @*RelativeWorkingPath* is ignored. At least one of @*WorkingPath* or @*RelativeWorkingPath* SHALL be specified.

9.16 DeviceMark

New in JDF 1.1

Promoted from Subelement status in the ***Layout*** Resource with new Attributes defined below.

The ***DeviceMark*** Element specifies the formatting parameters for how text for a device mark should be marked. This text is provided by an associated ***JobField*** Element (see ***Layout***/***MarkObject***/***JobField*** or ***LayoutElementProductionParams***/***JobField***).

Two methods for text layout are provided by ***DeviceMark***. First, text can be placed within a bounding box defined by a containing ***MarkObject*** (see ***MarkObject***/@*TrimSize* for defining the size of that bounding box). When this feature is selected, ***DeviceMark***/@*Font*, ***DeviceMark***/@*FontSize*, ***DeviceMark***/@*HorizontalFitPolicy* and ***DeviceMark***/@*VerticalFitPolicy* MAY be used to specify how text SHALL be fit within that bounding box.

The second method allows the bounding box defined by the text itself to be positioned, rotated, and scaled (along with the text). This facility operates through specifying an anchor point on that bounding box, and having the ***MarkObject***/@*CTM* operate relative to that anchor point. ***DeviceMark*** Attributes that affect this method are ***DeviceMark***/@*Font* and ***DeviceMark***/@*FontSize*.

See figures below for illustrations of marks generated by ***DeviceMark***.

Element Properties

Element referenced by: ***Layout***/***MarkObject***, ***LayoutPreparationParams***, ***LayoutPreparationParams***/***PageCell***

Table 9.19: DeviceMark Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Anchor ? New in JDF 1.4	enumeration	Anchor point on or within the bounding box of the text marked by this ***DeviceMark*** that ***MarkObject***/@*CTM* refers to. When @*Anchor* is specified, ***MarkObject***/@*TrimSize*, ***DeviceMark***/@*HorizontalFitPolicy* and ***DeviceMark***/@*VerticalFitPolicy* are ignored. **Note:** The bounding box of this ***DeviceMark*** is defined by the extent of the text being marked. **Allowed value if from:** ▸ Anchor.
Font ?	NMTOKEN	The name of the font that SHALL be used for the ***DeviceMark***. **Values include:** Courier Helvetica Helvetica-Condensed Times-Roman
FontSize ? Modified in JDF 1.4	double	The size of the font that SHALL be used for the ***DeviceMark***, in points ≥ 0. **Modification note:** Starting with **JDF** 1.4, the data type is no longer integer.
HorizontalFitPolicy ? New in JDF 1.4	enumeration	**Allowed values are from: *StripMark*/@*HorizontalFitPolicy*.**

Table 9.19: DeviceMark Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
MarkJustification ? Deprecated in JDF 1.4	enumeration	Description of the preferred ***DeviceMark*** justification. Interpreted in context of the @*MarkOrientation*. **Allowed values are:** Center Left Right **Deprecation note:** Starting with **JDF** 1.4, use ***DeviceMark***/@*Anchor* to specify the point in the bounding box defined by the text being marked relative to which ***MarkObject***/@*CTM* is applied to.
MarkOffset ? Deprecated in JDF 1.4	XYPair	Description of the preferred ***DeviceMark*** offset. Interpreted in context of the Device dependent default position in the coordinate system defined by @*MarkOrientation*. **Deprecation note:** Starting with **JDF** 1.4, use the ***MarkObject***/@*CTM* to appropriately place the mark.
MarkOrientation ? Deprecated in JDF 1.4	enumeration	Description of the preferred ***DeviceMark*** orientation. **Allowed values are:** Horizontal Vertical **Deprecation note:** Starting with **JDF** 1.4, use the ***MarkObject***/@*CTM* to appropriately rotate the mark.
MarkPosition ? Deprecated in JDF 1.4	enumeration	Description of the preferred ***DeviceMark*** position. **Allowed value is from:** ▸ Edge. **Deprecation note:** Starting with **JDF** 1.4, use @*Anchor*.
VerticalFitPolicy ? New in JDF 1.4	enumeration	**Allowed values are from:** ***StripMark***/@*VerticalFitPolicy*
BarcodeReproParams ? New in JDF 1.4	element	Reproduction parameters for Barcodes specified in the parent ***MarkObject***/***IdentificationField***.

Figure 9-1: Anchor with No Scaling and No Rotation

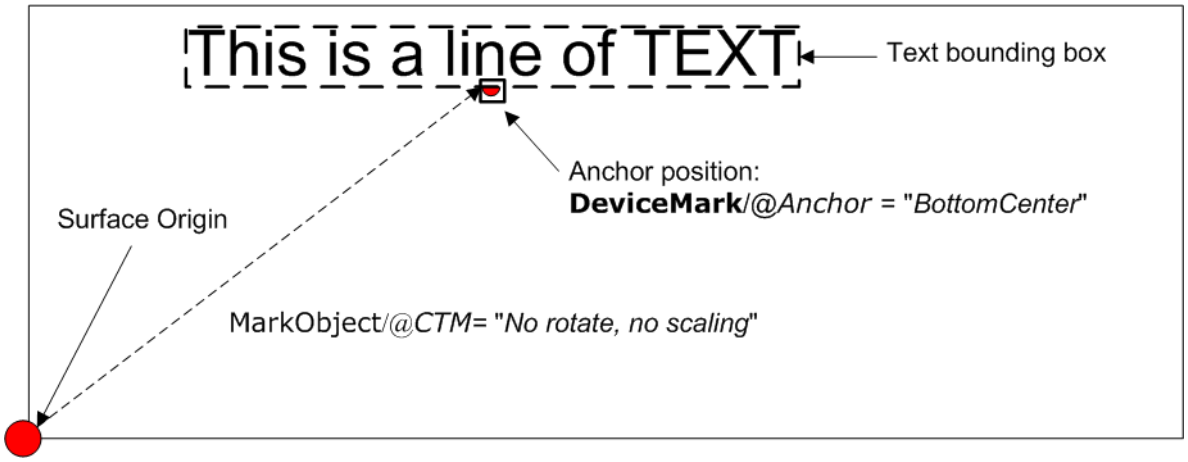


Figure 9-2: Anchor with No Scaling and Rotation of 90° Clockwise

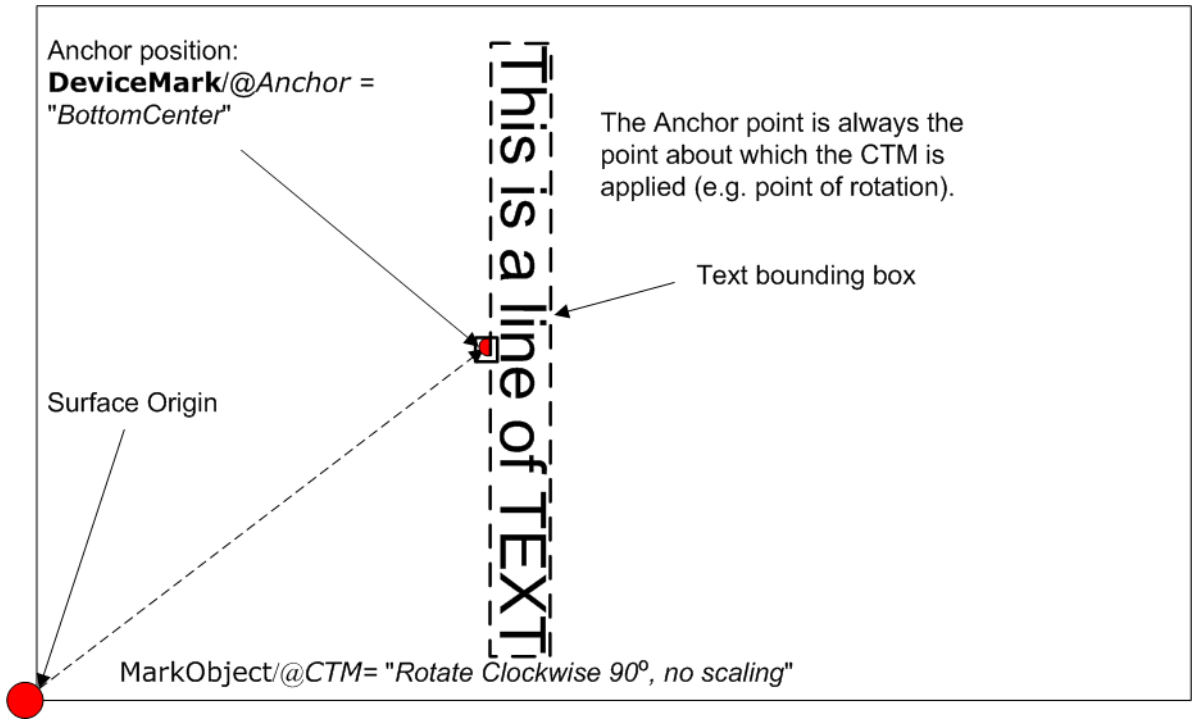
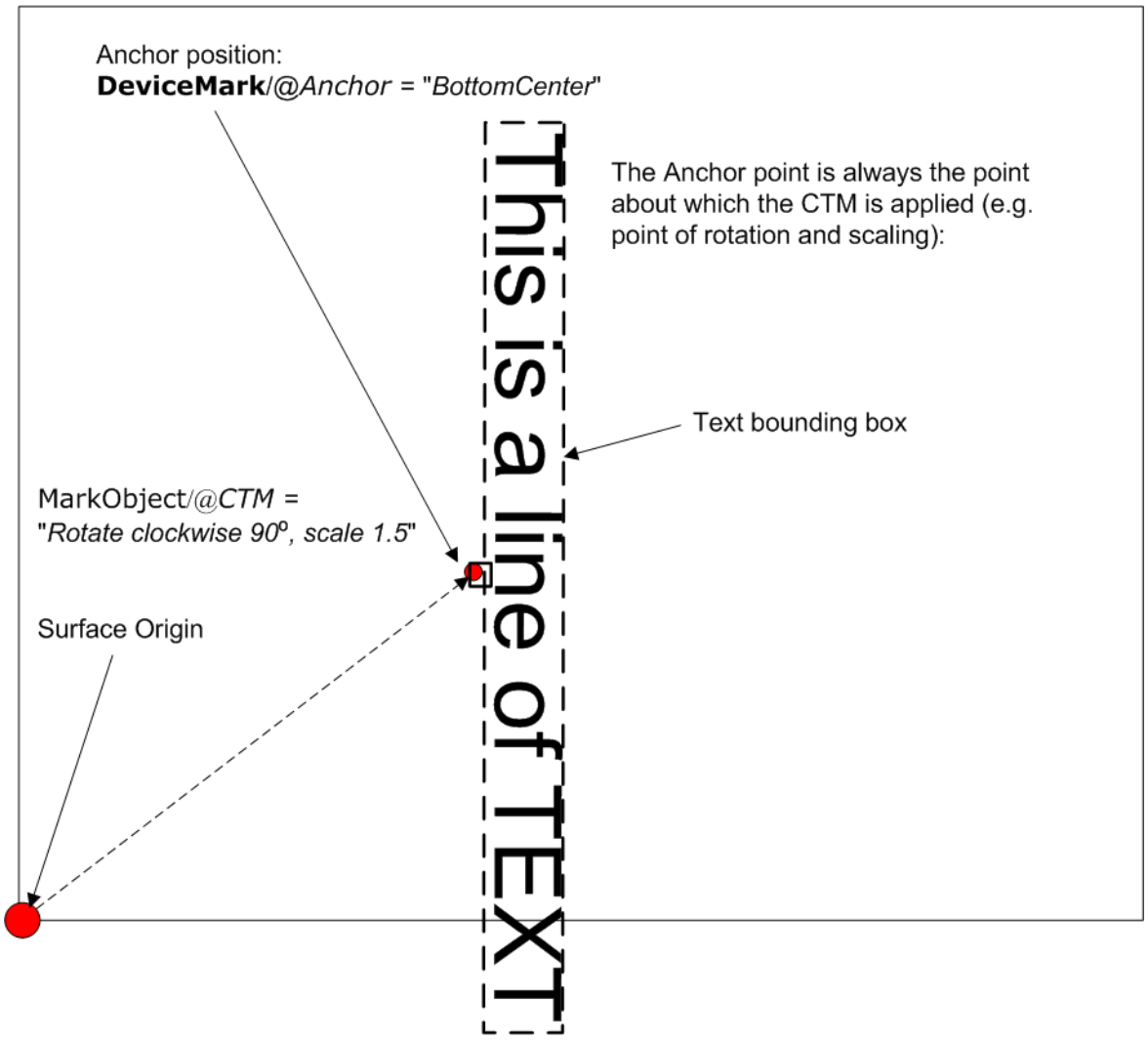


Figure 9-3: Anchor with 1.5 Scaling and Rotation of 90° Clockwise



9.17 DeviceNSpace

The ***DeviceNSpace*** can be used in several ways. For example, defining the specific colorants of a ***DeviceNSpace***:

- ***ColorantControl***/***ColorPool***/@*ColorantSetName* matches ***ColorantControl***/***DeviceNSpace***/@*Name*, and a:
- ***ColorantControl***/***ColorPool***/***Color*** Resource (with correct @*Name* of colorant and other defining Attributes) exists for each colorant of the ***DeviceNSpace*** as given in:
- ***ColorantControl***/***DeviceNSpace***/***SeparationSpec***/@*Name*

For example, defining a single colorant in terms of its values in a ***DeviceNSpace***:

- ***ColorantControl***/***ColorantParams*** names a colorant (e.g., a Pantone spot color).
- ***ColorantControl***/***DeviceNSpace*** names a DeviceN color space, which then the
  - ***ColorantControl***/***ColorPool***/@*ColorantSetName* matches, and then the corresponding
  - ***ColorantControl***/***ColorPool***/***Color***/***DeviceNSpace***/@*ColorList* Attribute gives the set of ***DeviceNSpace*** colorant percent values necessary to construct the,
  - ***ColorantControl***/@*ColorantParams* colorant (also named ***ColorantControl***/***ColorPool***/***Color***/@*Name*) in using ***DeviceNSpace*** colorants.

Element Properties

Element referenced by: ***ColorantControl***, ***ColorSpaceConversionOp***

Table 9.20: DeviceNSpace Element

NAME	DATA TYPE	DESCRIPTION
N	integer	The number of colors that define the color space.
Name ?	string	Color space name (e.g., HexaChrome or HiFi).
SeparationSpec *Modified in JDF 1.2	element	Ordered list of colorant names that define the DeviceN color space. Note that these colorants SHALL be specified in a corresponding ***ColorantParams*** Element of the ***ColorantControl*** or be implied by @*ProcessColorModel*. In other words, they SHALL be real, physical colorants.

9.18 Disjointing

The ***Disjointing*** Element describes how individual components are separated from one another on a stack.

Element Properties

Element referenced by ***Component***, ***StackingParams***

Table 9.21: Disjointing Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Number ?	integer	Number of Sheets that make up one component. The @*OffsetUnits* attribute specifies the type of the component. See @*OffsetUnits* for more details.
Offset ?	XYPair	Offset dimension in X and Y dimensions that separates the components.
OffsetAmount ?	integer	The number of components that are shifted in @*OffsetDirection* simultaneously. The @*OffsetUnits* attribute specifies the type of the component counted by this attribute. See @*OffsetUnits* for more details.
OffsetDirection ?	enumeration	Offset-shift action for the first component. A component can be offset to one of two positions—left or right. **Allowed values are:** Alternate – The position of the first component of a new job is opposite to the position of the previous component and subsequent components are each offset to alternating positions. For example, if the last item in the stack was positioned to the right then the subsequent items will be positioned to the left, right, left, right and so on. Left – The first component of a new job is on the left, and subsequent components are each offset to alternating positions. None – Do not offset consecutive components. The position of all components is the same as the position of the previous component. Right – The first component of a new job is on the right, and subsequent components are each offset to alternating positions. Straight – Same as "None". Deprecated in JDF 1.2

Table 9.21: Disjointing Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
OffsetUnits ? New in JDF 1.5	NMTOKEN	This attribute specifies the type of the component counted by the @*OffsetAmount* attribute. If @*Number* is present, it specifies the number of Sheets that make up a component (e.g., if @*OffsetUnits* is "Sets", the value of @*Number* specifies the number of sheets in a Set, and @*OffsetAmount* specifies the number of Sets). If @*Number* is not specified, it is assumed that the system has an internal way to keep track of component boundaries, whatever they may be (e.g., Set or Document boundaries). In a simple, non-VDP workflow, the product of @*Number* and @*OffsetAmount* is the number of sheets between shifts or separators. **Values include:** DocCopies – every individual document is counted. Docs – all copies of identical documents are counted as one. Jobs – entire Jobs are counted. SetCopies – every individual Set is counted. Sets – all copies of identical sets are counted as one. Sheets – each sheet is counted. Stacks – each stack is counted.
Overfold ? Deprecated in JDF 1.1	double	Expansion of the overfold of a Sheet. This Attribute is needed for the **Inserting** or other postpress Processes. Moved to ***Component***.
IdentificationField * Modified in JDF 1.1	element	Marks that identify the range of Sheets to be used in a Process. A scanner will scan the Sheets and detect a component boundary by scanning a mark (e.g., a bar code) that matches the description in the ***IdentificationField*** Element.
InsertSheet ?	refelement	Some kind of physical marker (e.g., a paper strip or a yellow paper sheet) that separates the components.

9.19 Disposition

New in JDF 1.2

This Element describes how long an asset SHOULD be maintained by a Device. The Device will perform an action defined by ***Disposition***/@*DispositionAction* when a "disposition time" occurs. Disposition time is defined as either:

- *@Until <= "Disposition time" <= @Until + @ExtraDuration*
- *"ProcessCompleteTime" + @MinDuration <= "Disposition time" <= "ProcessCompleteTime" + @MinDuration + @ExtraDuration*

Element Properties

Element referenced by: ***ResourcePullParams***, ***QueueSubmissionParams***, ***SubmitQueueEntry***/***QueueSubmissionParams***, ***FileSpec***, ***RunList***

Table 9.22: Disposition Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
DispositionAction = "Delete"	enumeration	**Allowed values are:** Delete – The asset is deleted when disposition time occurs. Archive – The asset is archived when disposition time occurs.
DispositionUsage ?	enumeration	Specifies the usage of the asset by the Process. **Default behavior:** ***Disposition*** applies to all Processes that link to the ***Disposition*** Resource (if @*DispositionUsage* not specified). **Allowed values is from:** ▸ Usage
ExtraDuration ?	duration	Indicates the maximum duration that the Device is allowed to retain the asset after the time specified by @*MinDuration* or @*Until*. If @*ExtraDuration*, @*MinDuration* and @*Until* are all unspecified, the asset is retained for a system specified time.
MinDuration ?	duration	Indicates the minimum duration that the Device SHOULD retain the asset after the Process that uses the asset completes.

Table 9.22: Disposition Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Priority = "0"	integer	Value between 0 and 100 that specifies the order in which assets are deleted or archived when the values of @*ExtraDuration*, @*MinDuration* and @*Until* cannot be honored (e.g., when local storage runs low). Assets with @*Priority* = "0" will be deleted first.
Until ?	dateTime	Indicates an absolute point in time when the Device or application SHOULD stop the asset retention. If @*Until* is specified, @*MinDuration* SHALL be ignored.

9.20 FitPolicy

New in JDF 1.1

This Element specifies how to fit content into a receiving container (e.g., a page onto a ***ContentObject*** of an imposed sheet). See the description of each reference to ***FitPolicy*** to determine what the context-specific *"content"* is and what the *"receiving container"* is.

Element Properties

Element referenced by: ***ImageSetterParams***, ***InterpretingParams***, ***Layout/PlacedObject***, ***LayoutPreparationParams***, ***LayoutPreparationParams/PageCell***, ***RasterReadingParams***

Table 9.23: FitPolicy Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ClipOffset ?	XYPair	Defines the offset (position) of the imaged area in the non-rotated source image when @*SizePolicy* is "ClipToMaxPage". The values 0.0 0.0 mean that the imaged area starts at the lower left point of the receiving container. If absent, the imaged area is taken from the center of the source image. If ***FitPolicy*** is defined in the context of a ***PageCell***, @*ClipOffset* is ignored when ***PageCell***/ @*ImageShift* is specified.
GutterPolicy = "Fixed"	enumeration	Allows printing of NUp grids even if the media size does not match the requirements of the data. @*GutterPolicy* SHALL NOT be specified when ***FitPolicy*** is referenced from a ***Layout*** Resource. **Allowed values are:** Distribute – The gutters can grow or shrink to the value specified in @*MinGutter*. Fixed – The gutters are fixed.
MinGutter ?	XYPair	Minimum width in points of the horizontal and vertical gutters formed between rows and columns of pages of a multi-up Sheet layout. The first value specifies the minimum width of all horizontal gutters and the second value specifies the minimum width of all vertical gutters. @*MinGutter* SHALL NOT be specified when ***FitPolicy*** is referenced from a ***Layout*** resource.
RotatePolicy ?	enumeration	Specifies the policy for the Device to automatically rotate the content to optimize the fit of the content to the receiving container. **Allowed values are:** NoRotate – Do not rotate. RotateOrthogonal – Rotate by 90° in either direction. RotateClockwise – Rotate clockwise by 90°. RotateCounterClockwise – Rotate counterclockwise by 90°.

Table 9.23: FitPolicy Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
SizePolicy ? Modified in JDF 1.1A	enumeration	Allows printing even if the container size does not match the requirements of the data. **Allowed values are:** ClipToMaxPage – The page contents are to be clipped to the size of the container. The printed area is either centered in the source image if no @*ClipOffset* key is given, or from that position which is determined by @*ClipOffset*. Abort – Emit an error and abort printing. FitToPage – The page contents are to be scaled up or down to fit the container. The aspect ratio is maintained. ReduceToFit – The page contents are to be scaled down but not scaled up to fit the container. The aspect ratio is maintained. Tile – the page contents are to be split into several tiles, each printed on its own surface.

9.21 Fold

New in JDF 1.1

Fold describes an individual folding operation of the ***Component***.

Element Properties

Element referenced by: ***FoldingIntent***, ***BinderySignature***, ***FoldingParams***,

Table 9.24: Fold Element

NAME	DATA TYPE	DESCRIPTION
From	enumeration	Edge from which the page is folded. **Allowed values are:** Front Left
RelativeTravel ? New in JDF 1.2	double	Relative distance of the reference edge relative to @*From* in the coordinates of the incoming ***Component***. The @*RelativeTravel* is always based on the complete size of the input ***Component*** and not on the size of an intermediate state of the folded Sheet. The allowed value range is from 0.0 to 1.0, which specifies the full length of the input ***Component***. At least one of @*Travel* or @*RelativeTravel* SHALL be specified.
To	enumeration	Direction in which it is folded. **Allowed values are:** Up – Upwards; corresponds to a valley fold with the left/bottom side coming over the opposite side. Down – Downwards; corresponds to a mountain or peak fold with the left/bottom side coming under the opposite side.
Travel ? Modified in JDF 1.2	double	Distance of the reference edge relative to @*From*. If both @*Travel* and @*RelativeTravel* are specified, @*RelativeTravel* is ignored. At least one of @*Travel* or @*RelativeTravel* SHALL be specified.

9.22 GangSource

New in JDF 1.6

GangSource provides source job information about a ***BinderySignature*** that is placed on a gang form.

Element Properties

Element referenced by: ***JobPhase***, ***QueueFilter***, ***QueueEntry***, ***NodeInfo***

Table 9.25: GangSource Element

NAME	DATA TYPE	DESCRIPTION
AssemblyID ?	NMTOKEN	If present, @*AssemblyID* SHALL reference the ***BinderySignature*** that this ***GangSource*** represents.

SUBELEMENTS

Table 9.25: GangSource Element

NAME	DATA TYPE	DESCRIPTION
Copies	integer	@*Copies* SHALL specify the number of copies of the ***BinderySignature*** that are required.
JobID	string	@*JobID* SHALL reference ***JDF***/@*JobID* of the individual job that describes the processing prior to and after printing and cutting the gang sheet.

9.23 GeneralID

New in JDF 1.3

Modified in JDF 1.4

Modification note: Starting with **JDF** 1.4, ***GeneralID*** becomes an element, and is no longer a resource. ***GeneralID*** becomes a child of any element. See ▸ Table 3.1 Any Element (generic content).

GeneralID describes a generic variable . The name or usage of the variable is specified in ***GeneralID***/@*IDUsage* and the specific value of the variable is specified in ***GeneralID***/@*IDValue*. The data type is specified in ***GeneralID***/@*DataType*.

Element Properties

Element referenced by: ***ContentList***/***ContentMetadata***

Table 9.26: GeneralID Element

NAME	DATA TYPE	DESCRIPTION
DataType ? New in JDF 1.4 Modified in JDF 1.5	enumeration	Data type of the variable. **Allowed values are:** string integer double NMTOKEN boolean dateTime duration NamedFeature – This ***GeneralID*** represents a ▸ NamedFeature as defined in ▸ Table 1.4 Glossary . New in JDF 1.5

Table 9.26: GeneralID Element

NAME	DATA TYPE	DESCRIPTION
IDUsage Modified in JDF 1.4	NMTOKEN	Usage of the ***GeneralID***. There are no predefined values in **JDF**. Values below with "AdsML" prefix are defined in ▸ [AdsML]. **Values include:** DeviceProductID – An ID of the resource as defined in the device namespace. For instance media catalogs of a press may provide media identifiers that are different from those defined by the MIS (which are identified with @*ProductID* values). New in JDF 1.4 AdsML:AdBuyer_BookingTransactionID – an ID for the booking transaction that was assigned by a party acting on behalf of the advertiser AdsML:AdSeller_BookingTransactionID – an ID for the booking transaction that was assigned by the publisher or a party acting on its behalf AdsML:AdBuyer_AdMaterialID – an ID for the artwork that was assigned by a party acting on behalf of the advertiser AdsML:AdSeller_AdMaterialID – an ID for the artwork that was assigned by the publisher or a party acting on its behalf. LineID – an ID for **PrintTalk** which associates a ***PrintTalk***//***Pricing***/***Price***[@*LineID*="someValue"] element with a **JDF** element embedded in **PrintTalk**, such as ***PrintTalk***//***jdf:DeliveryParams***/***Drop***/***DropItem***[***GeneralID***/@*IDUsage*="LineID" and ***GeneralID***/@*IDValue*="someValue"]. ShopID - Identifier of a web shop if the job has been submitted in a web to print environment. If the **JDF** was submitted via PrintTalk, @*IDValue* SHOULD be a copy of ***PrintTalk***/***Header***/***From***/***Credential***[@*domain*="jdf:ShopID"]/***Identity***. L – named variable that defines length within a ***ShapeTemplate***. (See ▸ Table 8.253 ShapeTemplate Element.) W – named variable that defines width within a ***ShapeTemplate***. (See ▸ Table 8.253 ShapeTemplate Element.) D – named variable that defines depth within a ***ShapeTemplate***. (See ▸ Table 8.253 ShapeTemplate Element.)
IDValue	string	Value of the ***GeneralID***. The data type of the value SHALL correspond to ***GeneralID***/@*DataType*.

9.24 GlueLine

This element provides the information to determine where and how to apply glue.

Element Properties

Element referenced by: ***InsertingIntent***, ***Insert***, ***BoxFoldingParams***, ***BoxFoldingParams***/***BoxFoldAction***, ***CaseMakingParams***, ***EndSheetGluingParams***/***EndSheet***, ***GlueApplication***, ***GluingParams***/***Glue***, ***HeadBandApplicationParams***, ***InsertingParams***, ***ThreadSewingParams***, ***MediaLayers***

Table 9.27: GlueLine Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AreaGlue = "false" New in JDF 1.1	boolean	Specifies that this ***GlueLine*** SHOULD cover the complete width of the ***Component*** it is applied to.
GlueBrand ?	string	Glue brand.
GlueLineWidth ?	double	Width of the glue line. Note that in extreme cases, the glue line could cover the input component over the whole width.
GlueType ?	enumeration	Glue type. **Allowed values are:** ColdGlue – Any type of glue that needs no heat treatment. Hotmelt – Hotmelt EVA (Ethyl-Vinyl-Acetate-Copolymere) PUR – Polyurethane

Table 9.27: GlueLine Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
GluingPattern ? Modified in JDF 1.3	NumberList	Glue line pattern defined by the length of a glue line segment (1st Element, 3rd and all odd elements of the NumberList) and glue line gap (2nd Element, 4th and all even elements of the NumberList). A solid line is expressed by the pattern (1 0). @*GluingPattern* SHALL contain an even number of entries. If the total length of @*GluingPattern* is less than @*WorkingPath* or the length implied by @*RelativeWorkingPath*, the pattern restarts after the last gap. If the total length of @*GluingPattern* is larger than @*WorkingPath* or the length implied by @*RelativeWorkingPath*, the pattern is clipped at the end.
MeltingTemperature ?	integer	Temperature needed for melting the glue, in degrees centigrade. Used only when @*GlueType* = "Hotmelt" or @*GlueType* = "PUR".
RelativeStartPosition ? New in JDF 1.2	XYPair	Relative starting position of the tool. The @*RelativeStartPosition* is always based on the complete size of the input ***Component*** and not on the size of an intermediate state of the folded Sheet. The allowed value range is from 0.0 to 1.0 for each component of the XYPair, which specifies the full size of the input ***Component***.
RelativeWorkingPath ? New in JDF 1.2	XYPair	Relative working path of the tool beginning at @*RelativeStartPosition*. The @*RelativeWorkingPath* is always based on the complete size of the input ***Component*** and not on the size of an intermediate state of the folded Sheet. The allowed value range is from 0.0 to 1.0 for each component of the XYPair, which specifies the full size of the input ***Component***.
StartPosition ? Modified in JDF 1.2	XYPair	Start position of glue line. The start position is given in the coordinate system of the mother Sheet. If both @*StartPosition* and @*RelativeStartPosition* are specified, @*RelativeStartPosition* is ignored.
WorkingPath ? Modified in JDF 1.2	XYPair	Relative working path of the gluing tool. If both @*WorkingPath* and @*RelativeWorkingPath* are specified, @*RelativeWorkingPath* is ignored.

9.25 Hole

The ***Hole*** element describes an individual hole.

Element Properties

Element referenced by: ***HoleLine***, ***HoleList***, ***HoleMakingParams***

Table 9.28: Hole Element

NAME	DATA TYPE	DESCRIPTION
Center	XYPair	Position of the center of the hole relative to the ***Component*** coordinate system. For more information, see ▸ Section 6.5.23 HoleMaking.
Extent	XYPair	Size (Bounding Box) of the hole, in points. If @*Shape* is "Round", only the first entry of @*Extent* is evaluated and defines the hole diameter.
Reinforcement ?	NMTOKEN	@ Reinforcement specifies how a hole shall be reinforced. **Values include:** Grommet **Note**: Additional details of the reinforcement MAY be supplied in a ***MiscConsumable*** with ***MiscConsumable***/@*ConsumableType*="Grommet"
Shape Modified in JDF 1.1	enumeration	Shape of the hole. **Allowed values are:** Elliptic Round Rectangular

9.26 HoleLine

New in JDF 1.1

Line hole punching generates a series of holes with identical distance (pitch) running parallel to the edge of a Web, which is mainly used to transport paper through continuous-feed printers and finishing Devices (form processing). The final product typically is a Web with two lines of holes, one at each edge of the Web. The parameters for one line of holes are specified in the ***HoleLine*** Resource. The distance between holes within each line of holes is identical (constant pitch).

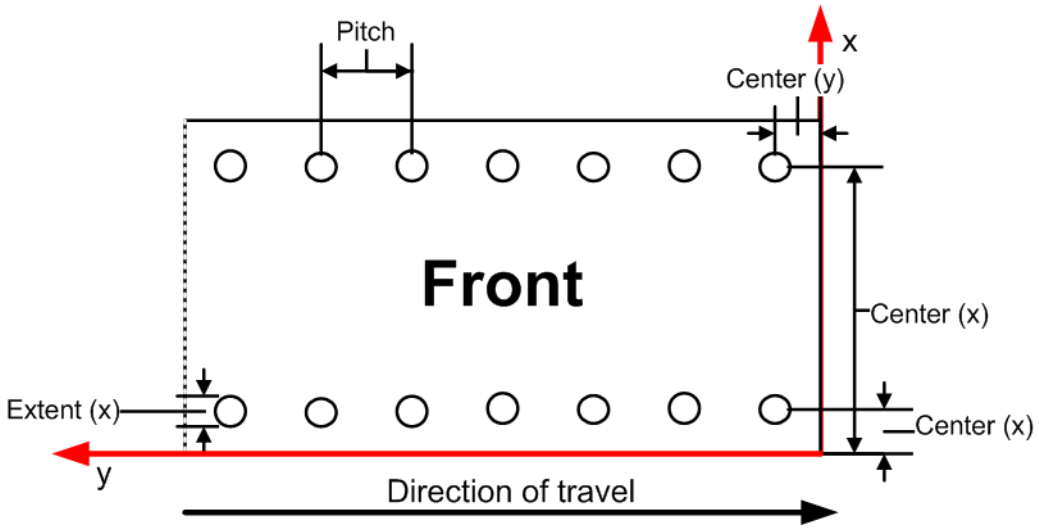
Element Properties

Element referenced by: ***HoleList***, ***HoleMakingParams***

Table 9.29: HoleLine Element

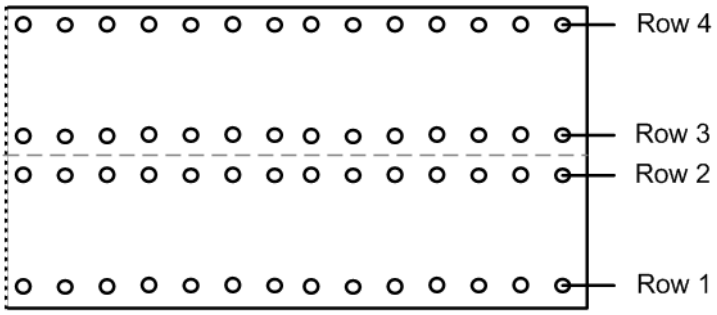
NAME	DATA TYPE	DESCRIPTION
Pitch	double	Center-hole to center-hole distance within a line of holes.
Hole	element	Size and position of the first hole in the ***HoleLine***.

Figure 9-4: Hole line parameters



However, sometimes Line Hole Punching is performed for multiple webs before dividing the Web after the **HoleMaking** Process as illustrated in ▸ Figure 9-5: Line hole punching for multiple webs below:

Figure 9-5: Line hole punching for multiple webs



9.27 InsertSheet

InsertSheet Resources define Device generated images and Sheets which can be produced along with the Job. ***InsertSheet*** Elements include separators Sheets, error Sheets, accounting Sheets and Job Sheets. The information provided on the Sheet depends on the type of Sheet. In some cases, an **Imposition** Process can encounter ***RunList*** Elements that do not provide enough finished pages to complete a ***Layout*** Resource or its children. ***InsertSheet*** Resources are used to provide a standard way of completing such ***Layout*** Resources. ***InsertSheet*** Resources MAY also be used to start new Sheet Resources (e.g., to ensure that a new chapter starts on a right-hand page). In addition, ***InsertSheet*** MAY specify whether new media are to be inserted after the current Sheet, Signature, Instance Document or Job is completed.

InsertSheet Elements MAY be used at the beginning or end of ***RunList*** with a *@SheetUsage* Attribute of "Header" or "Trailer". When an ***InsertSheet*** appears both in a ***RunList*** and in a ***Layout***, the following precedence applies:

1 The ***InsertSheet*** with *@Usage* "FillSurface" from the ***RunList*** is applied first.

2 The ***InsertSheet*** with @*Usage* "FillSheet" from the ***RunList*** is applied.

3 The ***InsertSheet*** with @*Usage* "FillSignature" from the ***RunList*** is applied.

4 After completely processing the ***RunList InsertSheet*** Elements once, apply the ***Layout*** Partition's ***InsertSheet*** Elements.

If the ***RunList*** of the ***InsertSheet*** does not supply enough content to fill a Sheet, Signature or surface, the ***RunList*** will be reapplied until no ***PlacedObject*** slots remain to be filled. When an ***InsertSheet*** is used in a ***RunList*** of a Process that does not use a ***Layout*** or ***LayoutPreparationParams*** Resource (i.e., that Process is not a part of a Combined Process with **Imposition** or **LayoutPreparation**), only @*Usage* "Header" or "Trailer" are valid.

Element Properties

Element referenced by: ***Layout***, ***RunList***

Table 9.30: InsertSheet Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
IncludeInBundleItem ? New in JDF 1.2	enumeration	Defines bundle items when this ***InsertSheet*** is not a Subelement of ***RunList***. If this ***InsertSheet*** is a Subelement of a ***RunList***, then @*IncludeInBundleItem* SHALL be ignored, and ***RunList***/@*EndOfBundleItem* SHALL be used instead. As an example, @*IncludeInBundleItem* controls whether the ***InsertSheet*** SHALL be included in a bundle item for purposes of finishing the ***InsertSheet*** with other Sheets. **Allowed values are:** After – This ***InsertSheet*** SHALL be included in the ***BundleItem*** that occurs after this ***InsertSheet***. "After" is equivalent to "None" if no ***BundleItem*** is defined after this ***InsertSheet*** Before – This ***InsertSheet*** SHALL be included in the ***BundleItem*** that occurs before this ***InsertSheet***. "Before" is equivalent to "None" if no ***BundleItem*** is defined before this ***InsertSheet***. None – This ***InsertSheet*** is not included in a ***BundleItem***. New – A new ***BundleItem*** is created. This ***InsertSheet*** will be in the new ***BundleItem*** by itself unless another ***InsertSheet*** with @*IncludeInBundleItem* = "Before" occurs immediately after this ***InsertSheet***.
IsWaste ?	boolean	Specifies whether the ***InsertSheet*** is waste that SHALL be removed from the document before further processing. If "true", the ***InsertSheet*** SHALL be discarded when finishing the document.
MarkList ? New in JDF 1.1	NMTOKENS	List of marks that are to be marked on this ***InsertSheet***. Ignored if a Sheet is specified in this ***InsertSheet***. **Values include:** CIELABMeasuringField ColorControlStrip ColorRegisterMark CutMark DensityMeasuringField IdentificationField JobField PaperPathRegisterMark RegisterMark ScavengerArea
SheetFormat ? New in JDF 1.1 Modified in JDF 1.2	NMTOKEN	Identifies that Device-dependent information SHALL be included on the ***InsertSheet***. **Values include:** Blank Brief Duplicate – Valid for @*SheetUsage* = "Interleaved" or "InterleavedBefore". Specifies that the interleaved Sheet SHALL contain the same (duplicate) content as the previous ("Interleaved") or following ("InterleavedBefore") Sheet. If there is content on both sides of the previous or following Sheet (duplex), then the ***InsertSheet*** has both sides duplicated. New in JDF 1.2 Full Standard

Table 9.30: InsertSheet Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
SheetType New in JDF 1.1	enumeration	Identifies the type of sheet. **Allowed values are:** AccountingSheet – A sheet that reports accounting information for the job. ErrorSheet – A sheet that reports errors for the job. FillSheet – A sheet that fills ***ContentObject*** elements with no matching entry in the content ***RunList***. InsertSheet – A sheet that is inserted to the job (e.g., a preprinted cover). JobSheet – A sheet that delimits the job. SeparatorSheet – A sheet that delimits pages, sections, copies or instance documents of the job.
SheetUsage New in JDF 1.1 Modified in JDF 1.2	enumeration	Indicates where this ***InsertSheet*** SHALL be produced and inserted into the set of output pages. **Allowed values are from:** ▸ Table 9.31 SheetUsage Attribute Values**.**
Usage ? Deprecated in JDF 1.1	enumeration	**Allowed values are from: @***SheetUsage***.** **Deprecation note:** Starting with **JDF** 1.1, use @*SheetUsage*.
Layout ? New in JDF 1.3	refelement	Details of the Sheet that will be inserted. Contents for this ***Layout*** are drawn from the ***RunList*** included in this ***InsertSheet*** if any. If not specified, the system specified insert Sheets are used. Any ***InsertSheet*** Resources referenced by this ***Layout*** are ignored.
RunList ?	refelement	A ***RunList*** that provides the content for the ***InsertSheet***. Any ***InsertSheet*** Resources referenced by this ***RunList*** are ignored.
Sheet ? Deprecated in JDF 1.3	refelement	Details of the ***Sheet*** that will be inserted. Contents for this ***Sheet*** are drawn from the ***RunList*** included in this ***InsertSheet*** if any. If not specified, the system specified insert Sheets are used. Any ***InsertSheet*** Resources referenced by this ***Sheet*** are ignored. **Deprecation note:** Starting with **JDF** 1.3, use ***Layout***.

9.27.1 SheetUsage Attribute Values

*Table 9.31: **SheetUsage Attribute** Values*

VALUE	DESCRIPTION
FillForceBack	Valid for @*SheetType* = "FillSheet". Contents of the ***RunList*** of the ***InsertSheet*** are used to fill the next Finished front Page of the current Sheet before forcing the next page of the content ***RunList*** to the next Finished back Page if not already on a Finished back Page. **Modification note:** Starting with **JDF** 1.4, this value applies to Finished pages rather than sheet surfaces.
FillForceFront	Valid for @*SheetType* = "FillSheet". Contents of the ***RunList*** of the ***InsertSheet*** are used to fill the next Finished back Page of the current Sheet before forcing the next Page of the content ***RunList*** to the next Finished front Page if not already on a Finished front Page. A typical use is to start a chapter on the front side of the Finished Page. **Modification note:** Starting with **JDF** 1.4, this value applies to Finished pages rather than sheet surfaces.
FillSheet	Valid for @*SheetType* = "FillSheet". Contents from the ***RunList*** of the ***InsertSheet*** are used to fill the current Sheet.
FillSignature	Valid for @*SheetType* = "FillSheet". Contents from the ***RunList*** of the ***InsertSheet*** are used to fill the current Signature.
FillSurface	Valid for @*SheetType* = "FillSheet". Contents from the ***RunList*** of the ***InsertSheet*** are used to fill the current surface.
Header	Valid for @*SheetType* = "InsertSheet", "JobSheet" or "SeparatorSheet". The Sheet is produced at the beginning of the Job (for `JobSheet`), or at the beginning of each copy of each Instance Document (for `SeparatorSheet`), or is appended before the current Sheet, Signature, layout or ***RunList*** as defined by its context. Contents for the Sheet are drawn from the ***RunList*** included in this ***InsertSheet*** Resource if one is included. If a ***RunList*** is not included, the inserted Sheet is filled with system-specified content defined by @*SheetType*.
Interleaved	Valid for @*SheetType* = "SeparatorSheet". The Sheet is produced after each page (e.g., used to insert Sheets under transparencies). Contents for the Sheet are drawn from the ***RunList*** included in this ***InsertSheet*** Resource if one is included. If a ***RunList*** is not included, the inserted Sheet is filled with system-specified content defined by @*SheetType* = "SeparatorSheet".
InterleavedBefore New in JDF 1.2	Valid for @*SheetType* = "SeparatorSheet". The Sheet is produced before each page (e.g., used to insert Sheets before transparencies). Contents for the Sheet are drawn from the ***RunList*** included in this ***InsertSheet*** Resource if one is included. If a ***RunList*** is not included, the inserted Sheet is filled with system-specified content defined by @*SheetType* = "SeparatorSheet".
OnError	Valid for @*SheetType* = "ErrorSheet". The Sheet is produced at the end of the Job only when an error or warning occurs.
Slip	Valid for @*SheetType* = "SeparatorSheet". The Sheet is produced between each copy of each Instance Document. Contents for the Sheet are drawn from the ***RunList*** included in this ***InsertSheet*** Resource if one is included. If a ***RunList*** is not included, the inserted Sheet is filled with system-specified content defined by @*SheetType* = "SeparatorSheet".
SlipCopy	Valid for @*SheetType* = "SeparatorSheet". The Sheet is produced between each copy of the Job, which is defined to be when the complete ***RunList*** has been consumed. Contents for the Sheet are drawn from the ***RunList*** included in this ***InsertSheet*** Resource if one is included. If a ***RunList*** is not included, the inserted Sheet is filled with system-specified content defined by @*SheetType* = "SeparatorSheet".
Trailer	Valid for @*SheetType* = "AccountingSheet", "ErrorSheet", "InsertSheet", "JobSheet" and "SeparatorSheet". The Sheet is produced at the end of the Job (for "AccountingSheet", "ErrorSheet" and "JobSheet"), or at the end of each copy of each Instance Document (for "SeparatorSheet"), or is appended after the current Sheet, Signature, layout or ***RunList*** as defined by its context. Contents for the Sheet are drawn from the ***RunList*** included in this ***InsertSheet*** Resource if one is included. If a ***RunList*** is not included, the inserted Sheet is filled with system specified content defined by ***SheetType***. **Note:** Use @*SheetType* = "ErrorSheet" and @*SheetUsage* = "Trailer" to always produce a Sheet that contains error or success information even if no errors or warnings occurred.

9.28 JobField

New in JDF 1.1

A ***JobField*** is a Mark object that specifies the details of a Job. The ***JobField*** Elements are also referred to as slug lines.

Element Properties

Element referenced by: ***Layout***/***MarkObject***, ***LayoutPreparationParams***, ***StripMark***

Table 9.32: JobField Element

NAME	DATA TYPE	DESCRIPTION
JobFormat ? New in JDF 1.4	string	A formatting string used with @*JobTemplate* to generate a string. **Allowed values are from:** ▸ Appendix H Dynamic String Generation**.**
JobTemplate ? New in JDF 1.4	string	A list of values used with @*JobFormat* to generate a string. **Allowed values are from:** ▸ Appendix H Dynamic String Generation**.**
OperatorText ?	string	Text from the operator. Note that this was erroneously described as text to the operator in **JDF** 1.1 and below. **Constraint:** Starting with **JDF** 1.4, if @*JobFormat* and @*JobTemplate* are specified, @*ShowList*, @*OperatorText* and @*UserText* SHALL NOT be specified.
ShowList ? Modified in JDF 1.4	NMTOKENS	List of elements to display in the ***JobField***. **Constraint:** Starting with **JDF** 1.4, if @*JobFormat* and @*JobTemplate* are specified, @*ShowList*, @*OperatorText* and @*UserText* SHALL NOT be specified. **Values include those from:** ▸ Table H.1 Predefined variables used in @XXX-Template**.** New in JDF 1.4 **Modification note:** Starting with **JDF** 1.4, the values come from a common list rather than a list that is custom to this Element. In addition, @*ShowList* becomes optional.
UserText ?	string	User-defined text to output with ***JobField***. **Constraint:** Starting with **JDF** 1.4, if @*JobFormat* and @*JobTemplate* are specified, @*ShowList*, @*OperatorText* and @*UserText* SHALL NOT be specified.
DeviceMark ? Modified in JDF 1.3 Deprecated in JDF 1.4	element	***DeviceMark*** defines the formatting parameters for the mark. If not specified, the settings defined in ***LayoutPreparationParams***/***DeviceMark*** are assumed. **Deprecation note:** Starting with **JDF** 1.4, ***DeviceMark*** SHALL be specified in the parent ***MarkObject*** Element.

9.29 MarkColor

New in JDF 1.5

Definition of the separations used to fill a dynamic mark.

Element Properties

Element referenced by: ***FillMark***, ***StripMark***

Table 9.33: MarkColor Element

NAME	DATA TYPE	DESCRIPTION
Name	string	Name of the Separation
Tint	double	Value from 0 (not used) to 1 (100% tint) of the Separation specified in @*Name*.

9.30 MediaLayers

MediaLayers contains an ordered list of subelements. Each subelement describes an individual layer of a layered ***Media*** resource. The first layer in ***MediaLayers*** is the front layer of the ***Media*** until the last layer, which defines the back.

SUBELEMENTS

Element Properties

Element referenced by: *MediaIntent*, *Media*

Table 9.34: MediaLayers Element

NAME	DATA TYPE	DESCRIPTION
GlueLine *	element	*GlueLine* resource describing a glue layer of a layered *Media* Resource. Each *GlueLine* Resource SHALL have *GlueLine*/@*AreaGlue* = "true".
Media *	refelement	*Media* resources describing a layer of a layered *Media* resource.

9.31 MetadataMap

New in JDF 1.4

MetadataMap allows metadata embedded in PDL files to be assigned to partition key values, certain *RunList* attributes, or attributes created using *GeneralID*. During the mapping of PDL data to the **JDF** document structure (see the definition in the glossary or the discussion in the **Imposition** Process), each *MetadataMap* element SHALL be evaluated for each node (Set, Document, Page, etc.) of the PDL document structure. For XML based PDL files an XPath expression SHALL be evaluated relative to the XML node that defines each node in the document hierarchy. For non-XML based PDLs a PDL specific mapping of the XPath to the PDL document structure SHALL be used instead and the value assignment is performed on the derived XML for the PDL file. If the path specified by the XPath does not exist in the PDL, then the associated metadata value is undefined, otherwise the metadata value will be set to the conversion of the node list to a string.

When *MetadataMap* is specified in the context of an *IdentificationField*, data can be extracted from the barcode represented by the *IdentificationField*.

Element Properties

Element referenced by: *IdentificationField*, *RunList*

Table 9.35: MetadataMap Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Context = "PagePool"	enumeration	Specifies the node context in which the XPaths specified in this *MetadataMap* Element are to be evaluated. **Allowed values are:** Set – evaluated relative to the current set node. Document – evaluated relative to the current document node. SubDoc0 – evaluated relative to the current subdocument immediately below the Document level. SubDoc1 – evaluated relative to the current subdocument immediately below "SubDoc0" level. SubDoc2 – see "SubDoc1", but relative to "SubDoc1" level. SubDoc3 – see "SubDoc1", but relative to "SubDoc2" level. SubDoc4 – see "SubDoc1", but relative to "SubDoc3" level. SubDoc5 – see "SubDoc1", but relative to "SubDoc4" level. SubDoc6 – see "SubDoc1", but relative to "SubDoc5" level. SubDoc7 – see "SubDoc1", but relative to "SubDoc6" level. SubDoc8 – see "SubDoc1", but relative to "SubDoc7" level. SubDoc9 – see "SubDoc1", but relative to "SubDoc8" level. PagePool – evaluated relative to the current ▸ Page Pool. Page – evaluated relative to the current page. Object – evaluated for each unique object on each page.
DataType	enumeration	Expected data type of the metadata value. **Allowed values are:** PartIDKeys – with this value, @*Name* SHALL match a Partition Key. **Allowed values are also from:** *GeneralID*/@*DataType*.

Table 9.35: MetadataMap Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION

Name	NMTOKEN	The name of the metadata.
		If @*DataType* = "PartIDKeys", the value of @*Name* SHALL be a @*PartIDKeys* value. See @*PartIDKeys* in ▸ Table 3.24 Partitionable Resource Element
		If @*Name* = "ObjectTags", then values are added to a logical pool of tag values associated with each object being processed. This pool of object tags is referenced from: ***ColorSpaceConversionParams***/***ColorSpaceConversionOp***/@*ObjectTags*, ***ScreeningParams***/***ScreenSelector***/@*ObjectTags*, ***ObjectResolution***/@*ObjectTags*, ***ColorCorrectionParams***/***ColorCorrectionOp***/@*ObjectTags*.
		Otherwise, @*Name* specifies the value of an implied variable (e.g., for use in ***GeneralID***/@*IDUsage*, ***RunList***/@*EndOfSet*, ***RunList***/@*SetCopies*, ***RunList***/@*PageCopies*, or ***RunList***/@*DocCopies*).
		If @*DataType* is not "PartIDKeys" or a ***RunList*** implied variable name (e.g., ***RunList***/@*DocCopies*), then the ***MetadataMap*** Element is equivalent to explicitly defining a ***GeneralID*** Element with the value being assigned by ***MetadataMap***/@*ValueFormat*. The following example counts the number of ***Page*** Elements within all ***DocPart*** Elements.
		`<MetadataMap DataType="integer" Name="NumPages"`
		`ValueFormat="%d" ValueTemplate="npages">`
		`<Expr Name="npages"`
		`Path="count(../DocPart/Page)">`
		`</MetadataMap>`
		If multiple ***MetadataMap*** Elements specify the same name, then the specified key has the value from the last ***MetadataMap*** Element to assign a value to that key.
		If the specified @*Name* sets the value for a @*PartIDKeys* or ***RunList*** variable, where a ***RunList*** Attribute also supplies a value (e.g., ***RunList***/@*RunTag*, ***RunList***/@*DocCopies*, etc.), the value supplied by the ***RunList*** Attribute shall be replaced by the value supplied by the ***MetadataMap***.

ValueFormat	string	Formatting value for combining extracted values from the ***Expr*** Elements. **Allowed values are from:** ▸ Appendix H Dynamic String Generation**.**
ValueTemplate	string	Arguments for combining extracted values from the ***Expr*** Elements. The argument names SHALL match the values of ***Expr***/@*Name* **Allowed values are from:** ▸ Appendix H Dynamic String Generation.
Expr * Modified in JDF 1.4	element	Each ***Expr*** Element describes a ***Term*** expression (see ▸ Section 10.2.13 Term ▸ Section Device Capabilities) evaluating metadata values in the PDL. If ***Expr***/***Term*** is not specified, or if the ***Term*** expression returns true, then the value specified by the ***Expr*** element is assigned to the key specified by ***MetadataMap***/@*Name*. ***Expr*** Elements are evaluated in the XML order specified. ***Expr*** Elements with identical @*Name* Attributes where a previous ***Expr*** Element with that @*Name* has already evaluated to true SHALL NOT be processed. If any name specified in ***MetadataMap***/@*ValueTemplate* is unassigned, then the key specified by ***MetadataMap***/@*Name* is undefined. All ***Expr*** Elements return string values. These values SHALL be type converted as necessary during processing of @*ValueFormat* and @*ValueTemplate* (See ▸ Section Dynamic String Generation). **Note:** If @*ValueFormat* contains a constant string with no format specifiers, then it is not necessary to define any ***Expr*** Elements. **Modification note:** Starting with **JDF** 1.4, ***Expr*** MAY be omitted.

9.31.1 Expr

New in JDF 1.4

Table 9.36: Expr Element

NAME	DATA TYPE	DESCRIPTION
Name	NMTOKEN	Name of this ***Expr***. The value (as specified by @*Value* or extracted from @*Path*) SHALL be used to evaluate the parent @*ValueTemplate*.
Path ?	XPath	If specified, and either the value returned by the ***Term*** element (if present) is true or no ***Term*** element is specified, then the value specified by this path is assigned to ***Expr***/@*Name*. If the XPath specified by @*Path* does not evaluate to a value such as a string or number, then this ***Expr*** element fails and any subsequent ***Expr*** Elements are evaluated. If the XPath points to an element, then an implied XPath text() function SHALL be executed. The value is converted into a string when returned by the ***Expr*** element. The value returned when the XPath results in a node set is undefined. **Constraint:** exactly one of @*Path* or @*Value* SHALL be specified in an ***Expr*** Element.
Value ?	string	If specified, and either the value returned by the ***Term*** Element (if present) is true or no ***Term*** Element is specified, then the value of this Attribute is assigned to ***Expr***/@*Name*. **Constraint:** exactly one of @*Path* or @*Value* SHALL be specified in an ***Expr*** element.
Term ?	element	Evaluates one or more metadata values from the PDL, and returns a true or false result. ***Evaluation***/@*Path* SHALL be specified for all ***Evaluation*** Elements in the ***Term*** hierarchy.

For PPML the XPath expression will be relative to the JOB, DOCUMENT or PAGE element. Example XPath expressions:

- "METADATA/DATUM[@key = "Gender"]" will extract the value of the Gender metadata for each **JDF** set, document and page.
- "count(PAGE)" will count the pages within a given document (only works for **JDF** document level nodes).
- "count(PAGE/METADATA/DATUM[@key = "special"])" will count the number of pages that have a Special metadata defined for it.

MetadataMap may also be used to set the value of certain ***RunList*** Attributes. These Attributes are @*EndOfSet*, @*EndOfDocument*, @*PageCopies*, @*DocCopies* and @*SetCopies*. The values set will be instantiated as if actually present in a Partitioned ***RunList*** for the current page or Page Pool being processed. Care should be taken to ensure their consistency across Page Pools within a document or set.

Example 9.5: MetadataMap: Setting Attributes

This example extracts the value of the @*Copies* Attribute as specified by the @*Path*, and sets the value of ***RunList***/@*DocCopies*.

Table 9.37: MetadataMap: Setting Attributes

VALUE	DESCRIPTION
EndOfSet	The last page of a set of Instance Document.
EndOfDocument	The last page of an Instance Document.
PageCopies	Number of finished page copies.
DocCopies	Number of Instance Document copies
SetCopies	Number of Instance Document Set copies.

```
<RunList Class="Parameter" ID="r000004" Status="Available">
    <MetadataMap DataType="integer" Name="DocCopies" ValueFormat="%d"
          ValueTemplate="ncopies">
       <Expr Name="ncopies" Path="//record/document/@Copies"/>
       <Expr Name="ncopies" Value="1"/>
    </MetadataMap>
</RunList>
```

Example 9.6: RunList/MetadataMap

New in JDF 1.4

In the following example, the ***MetadataMap*** element maps arbitrary tags in the document to a structural *@RunTag* Partition Key. Note that any partition key may be mapped. Note also that although an XPath syntax is used, this may be mapped to any hierarchical structure including but not limited to XML. Finally, note that if */**Dokument**/@Sektion* is a value other than "Einband" or "HauptTeil", then the ***Expr*** Elements assigning values to section will all fail, resulting in *@RunTags* being undefined.

```
<!--this runlist points to a structured pdl with arbitrary structural tagging-->
<RunList Class="Parameter" ID="r000004" Status="Available">
   <MetadataMap DataType="PartIDKeys" Name="RunTags"
         ValueFormat="%s%s" ValueTemplate="sex,section">
<!--This expression maps the value of /Dokument/Rezipient/@Sex to a variable "sex"-->
      <Expr Name="sex" Path="/Dokument/Rezipient/@Sex"/>
<!--Maps all elements with /Dokument/@Sektion=Einband to Cover-->
      <Expr Name="section" Value="Cover">
         <NameEvaluation Path="/Dokument/@Sektion" RegExp="Einband"/>
      </Expr>
<!--Maps all elements with /Dokument/@Sektion=HauptTeil and >50 pages to BigBody-->
      <Expr Name="section" Value="BigBody">
         <and>
            <NameEvaluation Path="/Dokument/@Sektion" RegExp="HauptTeil"/>
            <IntegerEvaluation Path="count(PAGE)" ValueList="51 ~ INF"/>
         </and>
      </Expr>
<!--Maps all elements with /Dokument/Sektion=HauptTeil and <=50 pages to SmallBody-->
      <Expr Name="section" Value="SmallBody">
         <and>
            <NameEvaluation Path="/Dokument/Sektion" RegExp="HauptTeil"/>
            <IntegerEvaluation Path="count(PAGE)" ValueList="0 ~ 50"/>
         </and>
      </Expr>
   </MetadataMap>
   <LayoutElement Class="Parameter">
      <FileSpec Class="Parameter" MimeType="application/vnd.foobar+xml" URL="bigVariable.foo"/>
   </LayoutElement>
</RunList>
<!--Layout for versioned product-->
<Layout Class="Parameter" ID="r000005" PartIDKeys="RunTags" Status="Available">
   <Layout RunTags="MaleCover">
      <MediaRef rRef="r000006">
         <Part RunTags="MaleCover"/>
      </MediaRef>
   </Layout>
   <Layout RunTags="FemaleCover">
      <MediaRef rRef="r000006">
         <Part RunTags="FemaleCover"/>
      </MediaRef>
   </Layout>
   <Layout RunTags="MaleBigBody FemaleBigBody">
      <MediaRef rRef="r000006">
         <Part RunTags="MaleBigBody MaleSmallBody FemaleBigBody FemaleSmallBody"/>
      </MediaRef>
   </Layout>
   <Layout RunTags="MaleSmallBody FemaleSmallBody">
      <MediaRef rRef="r000006">
         <Part RunTags="MaleBigBody MaleSmallBody FemaleBigBody FemaleSmallBody"/>
      </MediaRef>
   </Layout>
</Layout>
<Media Class="Consumable" ID="r000006" PartIDKeys="RunTags"
      PartUsage="Implicit" Status="Available">
   <Media RunTags="MaleCover"/>
   <Media RunTags="FemaleCover"/>
   <Media RunTags="MaleBigBody MaleSmallBody FemaleBigBody FemaleSmallBody"/>
</Media>
```

9.32 MISDetails

New in JDF 1.2

MISDetails is a container for MIS related information. It is referenced by ***Audit*** Elements and **JMF** Messages.

Element Properties

Element referenced by: ***PhaseTime***, ***ResourceAudit***, ***ResourceCmdParams***, ***ResourceInfo***, ***ResourcePullParams***, ***JobPhase***, ***NodeInfo***

Table 9.38: MISDetails Element

NAME	DATA TYPE	DESCRIPTION
Complexity ? New in JDF 1.4	double	Complexity of the task specified by this **JDF** node in a range from 0.0 to 1.0. **Note:** The interpretation of values is implementation dependant. **Values include:** 0.0 – The job is simple and therefore reduced setup and waste or higher speeds are possible. 0.5 – The job is of standard complexity and therefore standard setup and waste or normal speeds are possible. 1.0 – The job is complex and therefore more setup and waste or lower speeds are possible.
CostType ?	enumeration	Whether or not this ***MISDetails*** is chargeable to the customer or not. **Allowed values are:** Chargeable NonChargeable
DeviceOperationMode ?	enumeration	@*DeviceOperationMode* shows the operation mode that the Device is in. It is used to show if the production of a Device is aimed at producing good products or not. The latter case applies when a Device is used to produce a Job for testing, calibration, etc., without the intention to produce good output. **Allowed values are:** Productive – The Device is used to produce good product. Any times recorded in this mode are to be allocated against the Job. NonProductive – The Device is used without the intention to produce good product. Any times recorded in this mode are not to be allocated against the Job. Maintenance –The Device is used without the intention to produce good product (e.g., to perform (preventive) maintenance).
WorkType ?	enumeration	Definition of the work type for this ***MISDetails*** (i.e., whether or not this ***MISDetails*** relates to originally planned work, an alteration or rework). **Allowed values are:** Original – Standard work that was originally planned for the Job. Alteration – Work done to accommodate change made to the Job at the request of the customer. Rework – Work done due to unforeseen problem with original work (bad plate, Resource damaged, etc.).
WorkTypeDetails ?	string	Definition of the details of the work type for this ***MISDetails*** (i.e., why the work was done). **Values include:** CustomerRequest – The customer requested change(s) requiring the work. EquipmentMalfunction – Equipment used to produce the Resource malfunctioned; Resource needs to be created again. InternalChange – Change was made for production efficiency or other internal reason. ResourceDamaged – A Resource needs to be created again to account for a damaged Resource (damaged plate, etc.). UserError – Incorrect operation of equipment or incorrect creation of Resource requires creating the Resource again.

9.33 ObjectResolution

ObjectResolution defines a resolution depending on @*SourceObjects* data types.

Element Properties

Element referenced by: *InterpretingParams*, *RenderingParams*, *TrappingDetails*

Table 9.39: ObjectResolution Element

NAME	DATA TYPE	DESCRIPTION
AntiAliasing ? New in JDF 1.2	NMTOKEN	Indicates the anti-aliasing algorithm that the Device SHALL apply to the rendered output images. An anti-aliasing algorithm causes lines and curves to appear smooth which would otherwise have a jagged appearance, especially at lower resolutions such as 300 dpi and lower. **Values include:** AntiAlias – Anti-aliasing SHALL be applied. The algorithm is system specified. None – Anti-aliasing SHALL NOT be applied.
ObjectTags ? New in JDF 1.4	NMTOKENS	Tags associated with individual objects that this ***ObjectResolution*** SHALL be applied to. Each tag specified in @*ObjectTags* is logically anded with the object type(s) specified by @*SourceObjects*, enabling first qualification by object type (such as image), and then tags associated with those objects. The values of @*ObjectTags* depends on the PDL that the ***ObjectResolution*** is applied to.. @*ObjectTags* SHALL apply only to objects whose tag pool includes all the tags in the value of @*ObjectTags*.
Resolution	XYPair	Horizontal and vertical output resolution in DPI.
SourceObjects = "All"	enumeration	Identifies the class(es) of incoming graphical objects to render at the specified resolution. **Allowed value is from:** ▸ SourceObjects.

9.34 Perforate

Perforate describes one perforated line.

Element Properties

Element referenced by: *PerforatingParams*

Table 9.40: Perforate Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Depth ? New in JDF 1.2	double	Depth of the perforation, in microns [µm].
RelativeStartPosition ? New in JDF 1.2	XYPair	Relative starting position of the tool. The @*RelativeStartPosition* is always based on the complete size of the input ***Component*** and not on the size of an intermediate state of the folded Sheet. The allowed value range is from 0.0 to 1.0 for each component of the XYPair, which specifies the full size of the input ***Component***. At least one of @*StartPosition* or @*RelativeStartPosition* SHALL be specified.
RelativeWorkingPath ? New in JDF 1.2	XYPair	Relative working path of the tool beginning at @*RelativeStartPosition*. Since the tools can only work parallel to the edges, one coordinate SHALL be zero. The @*RelativeWorkingPath* is always based on the complete size of the input ***Component*** and not on the size of an intermediate state of the folded Sheet. The allowed value range is from 0.0 to 1.0 for each component of the XYPair, which specifies the full size of the input ***Component***. At least one of @*WorkingPath* or @*RelativeWorkingPath* SHALL be specified.
StartPosition ? Modified in JDF 1.2	XYPair	Starting position of the tool. If both @*StartPosition* and @*RelativeStartPosition* are specified, @*RelativeStartPosition* is ignored. At least one of @*StartPosition* or @*RelativeStartPosition* SHALL be specified.
TeethPerDimension ?	double	Number of teeth in a given perforation extent in teeth/point. MicroPerforation is defined by specifying a large number of teeth (@*TeethPerDimension* > 1000).

Table 9.40: Perforate Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
WorkingDirection ? Modified in JDF 1.5	enumeration	Direction from which the tool is working. **Allowed value is from:** ▸ WorkingDirection. **Modification note:** Starting in **JDF** 1.5, @*WorkingDirection* is optional.
WorkingPath ? Modified in JDF 1.2	XYPair	Working path of the tool beginning at @*StartPosition*. Since the tools can only work parallel to the edges, one coordinate SHALL be zero. If both @*WorkingPath* and @*RelativeWorkingPath* are specified, @*RelativeWorkingPath* is ignored. At least one of @*WorkingPath* or @*RelativeWorkingPath* SHALL be specified.

9.35 Person

This Element provides detailed information about a person. It also has the ability to specify different communication channels to this person. Use @*ProductID* when a unique identifier for the ***Person*** is required. The structure of the Element is derived from the vCard format. It contains all of the same name subtypes (N:) of the identification and the title of the organizational properties. The corresponding XML types of the vCard are quoted in the description field of the table below.

Modification note: Starting with **JDF** 1.4, a rule about using @*ProductID* is added

Element Properties

Element referenced by: ***Contact***, ***Employee***

Table 9.41: Person Element

NAME	DATA TYPE	DESCRIPTION
AdditionalNames ?	string	Additional names of the contact person (vCard: N:other).
FamilyName ?	string	The family name of the contact person (vCard: N:family).
FirstName ?	string	The first name of the contact person (vCard: N:given).
JobTitle ?	string	Job function of the person in the company or organization (vCard: title).
Languages ? New in JDF 1.4	languages	List of languages related to the person, ordered by decreasing preference
NamePrefix ?	string	Prefix of the name, can include title (vCard: N:prefix).
NameSuffix ?	string	Suffix of the name (vCard: N:suffix).
PhoneticFirstName ? New in JDF 1.5	string	Alternative spelling of a first name. Used (e.g., for pronunciation of Kanji (Japanese) names). See http://en.wikipedia.org/wiki/VCard .
PhoneticLastName ? New in JDF 1.5	string	Alternative spelling of a last name. Used (e.g., for pronunciation of Kanji (Japanese) names). See http://en.wikipedia.org/wiki/VCard .
Address ? New in JDF 1.2	element	Address of the person.
ComChannel *	element	Communication channels to the person

9.36 RefAnchor

New in JDF 1.4

RefAnchor describes the relative position with respect to a related element in a layout. Depending on the value of @*AnchorType*, it specifies either a parent Element or a sibling Element.

Element Properties

Element referenced by: ***Layout***/***MarkObject***, ***LayoutElementProductionParams***/***LayoutElementPart***/***PositionObj***, ***StrippingParams***/***StripMark***

*Table 9.42: **RefAnchor** Element*

NAME	DATA TYPE	DESCRIPTION
Anchor ?	enumeration	@*Anchor* specifies the origin (0,0) of the vector specified in the rotated coordinate system of the related layout element. **Allowed value is from:** ▸ Anchor**.**
AnchorType ?	enumeration	Role of this ***RefAnchor***. **Allowed values are:** Parent – The layout element referenced by this ***RefAnchor*** is a parent. This layout element is transformed with the parent. Sibling – The layout element referenced by this ***RefAnchor*** is a sibling. Both layout elements share a common parent. The parent of this layout element is specified as the ***RefAnchor*** of the first child in the chain of siblings.
rRef ?	IDREF	Reference to a layout element that this layout element is positioned relative to. If @*rRef* is not specified, the page or sheet defined by the layout element is the parent container. @*rRef* SHALL be specified if @*AnchorType* = "Sibling".

9.37 RegisterRibbon

New in JDF 1.1

Description of register ribbons. For the register ribbon, the length SHALL be specified. There are two parameters, as shown in ▸ Figure 9-6: RegisterRibbon lengths and coordinate system for BlockPreparation:

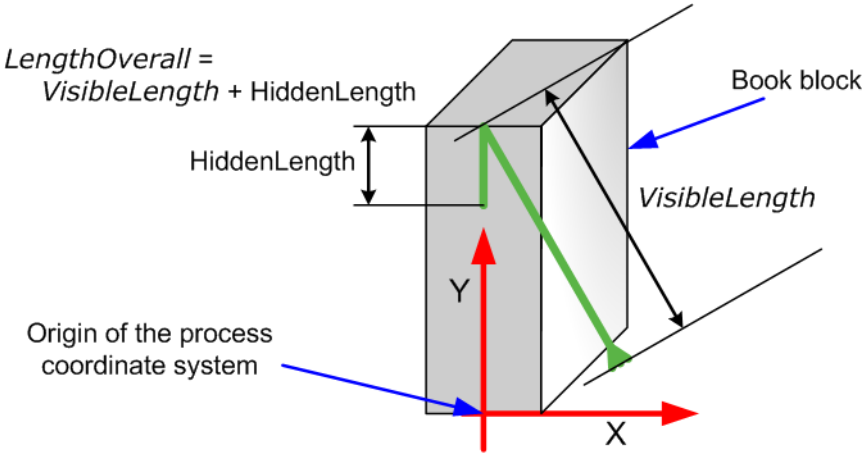
Element Properties

Element referenced by: ***BindingIntent***/***HardCoverBinding***, ***BlockPreparationParams***

Table 9.43: RegisterRibbon Element

NAME	DATA TYPE	DESCRIPTION
LengthOverall ? Modified in JDF 1.4	double	Overall length of the register ribbon (i.e., @*VisibleLength* + HiddenLength in ▸ Figure 9-6: RegisterRibbon lengths and coordinate system for BlockPreparation). Note "HiddenLength" is not an Attribute **Modification note:** Starting with **JDF** 1.4, @*LengthOverall* is optional.
Material ?	string	Material of the register ribbon.
RibbonColor ?	enumeration	@*RibbonColor* specifies the machine readable color of ribbon. **Allowed value is from:** ▸ NamedColor**.**
RibbonColorDetails ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If @*RibbonColorDetails* is supplied, @*RibbonColor* SHOULD also be supplied.
RibbonEnd ?	NMTOKEN	End of the Ribbon. **Values include:** Cut CutSealed Knot SealedOffset – The ribbon is sealed a distance from the cut.
VisibleLength ? Modified in JDF 1.4	double	Length of the register ribbon which will be seen when opening the book. See ▸ Figure 9-6: RegisterRibbon lengths and coordinate system for BlockPreparation. **Modification note:** Starting with **JDF** 1.4, @*VisibleLength* is optional.

Figure 9-6: RegisterRibbon lengths and coordinate system for BlockPreparation



9.38 ScreenSelector

Description of screening for a selection of source object types and separations.

Element Properties

Element referenced by: ***ColorSpaceConversionOp***, ***ScreeningParams***

Table 9.44: ScreenSelector Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Angle ?	double	Specifies the first angle of the screen when AM screening is used, otherwise @*Angle* is ignored. At most one of @*Angle* or @*AngleMap* SHALL be specified. If neither @*Angle* nor @*AngleMap* are specified, the angle is determined by the default of the selected @*ScreeningFamily*.
AngleMap ? New in JDF 1.1	string	Specifies the mapping of the angle of the screen to the angle of a different separation when AM screening is used. For example, a spot color that has the same screening angle as the cyan separation is specified by @*AngleMap* = "Cyan". In FM screening, @*AngleMap* specifies the mapping of the separation specific screen functions (e.g., threshold arrays). At most one of @*Angle* or @*AngleMap* SHALL be specified. This mapping is not transitive, so, when @*Separation* already specifies a color with a known default, it specifies the angle of the separation defined by @*AngleMap* prior to that separation being mapped. Note that, in general, the known default will be a CMYK process color, but it can also be another process color (e.g., HexaChrome™). The following example specifies that "Black" is to be mapped to the "Cyan" default separation and "Cyan" to the "Black" default separation. The third line maps Spot1 to Magenta. <ScreenSelector AngleMap="Black" Separation="Cyan"/> <ScreenSelector AngleMap="Cyan" Separation="Black"/> <ScreenSelector AngleMap="Magenta" Separation="Spot1"/>
DotSize ? New in JDF 1.1	double	Specifies the dot size of the screen, in microns [µm], when FM screening (@*ScreeningType* = "FM" or "Adaptive") is used, otherwise @*DotSize* is ignored.
Frequency ? Modified in JDF 1.2	double	Specifies the halftone screen frequency in lines per inch (lpi) of the screen when AM screening is used, otherwise @*Frequency* is ignored. With some screens, frequency can change as a function of gray level. In this case, the @*Frequency* value is interpreted for a midtone (50%) gray level. If @*Frequency* is not specified, the frequency is determined by the default of the selected @*ScreeningFamily*.
ObjectTags ? New in JDF 1.4	NMTOKENS	Tags associated with individual objects that this ***ScreenSelector*** SHALL be applied to. Each tag specified in @*ObjectTags* is logically anded with the object type(s) specified by @*SourceObjects*, enabling first qualification by object type (such as image), and then tags associated with those objects. The values of @*ObjectTags* depends on the PDL that the ***ScreenSelector*** is applied to. @*ObjectTags* SHALL apply only to objects whose tag pool includes all the tags in the value of @*ObjectTags*.

Table 9.44: ScreenSelector Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ScreeningFamily ?	string	Vendor specific screening family name. Sample values removed in JDF 1.2
ScreeningType ? Modified in JDF 1.2	enumeration	General type of screening. **Allowed values are:** Adaptive AM – Can be line or dot. See @*SpotFunction*. ErrorDiffusion FM – Includes all stochastic screening types. HybridAM-FM HybridAMline-dot
Separation = "All"	string	The name of the separation. If @*Separation* = "All", the ***ScreenSelector*** SHALL be applied to all separations that are not specified explicitly. **Values include:** All
SourceFrequency ? Modified in JDF 1.2	DoubleRange	Specifies the line frequency of screens which SHALL be matched from the source file when screen matching is to be done. Note that this is a filter that selects on which objects to apply this ***ScreenSelector***.
SourceObjects = "All"	enumeration	Identifies the class(es) of incoming graphical objects on which to use the selected screen. **Allowed value is from:** ▸ SourceObjects.
SpotFunction ?	NMTOKEN	Specifies the spot function of the screen when AM screening is used. In general, it is common for a spot function to change its shape as a function of gray level. Response to these spot function names MAY be implementation-dependent. These example names are the same as the spot function names defined in PDF. **Values include:** Round Diamond Ellipse EllipseA InvertedEllipseA EllipseB EllipseC InvertedEllipseC Line LineX LineY Square Cross Rhomboid DoubleDot InvertedDoubleDot SimpleDot InvertedSimpleDot CosineDot Double InvertedDouble

9.39 SeparationSpec

This Element specifies a specific separation, and is usually used to define a list or sequence of separations.

Element Properties

Element referenced by: ***ColorsUsed***, ***NumberItem***, ***ProofItem***, ***ColorantAlias***, ***ColorantConvertProcess***, ***ColorantOrder***, ***ColorantParams***, ***DeviceColorantOrder***, ***ColorSpaceSubstitute***, ***ColorControlStrip***, ***ColorSpace-***

SUBELEMENTS

ConversionOp, *ContentData*, *DeviceNSpace*, *LayoutElement*, *PageList*, *PageData*, *RegisterMark*, *ScavengerArea*, *CutLines*, *SeparationListBack*, *SeparationListFront*, *TrappingOrder*,

Table 9.45: SeparationSpec Element

NAME	DATA TYPE	DESCRIPTION
Name ?	string	Name of one specific separation. If @*Name* is not specified, this *SeparationSpec* consumes a slot in a separation order without setting a separation, for instance when specifying modules to skip on a press or color fields to leave blank in a *ColorControlStrip*. **Modification note:** Starting with **JDF** 1.4, @*Name* is optional.

Device Capabilities

Introduction

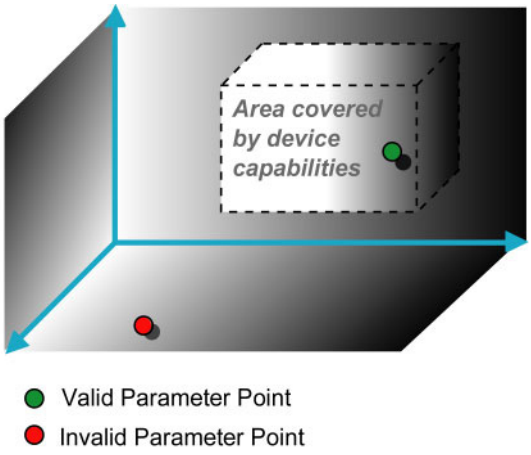
The **JDF** specification and schema describe the entire space of parameters and resources that can be used to control any device. Every actual Device has limitations in the **JDF** that it can correctly process. For instance, a RIP will typically not process folding instructions. Even if a set of parameters is processes, physical limitations may restrict the possible values. For instance Media sizes are theoretically unbounded, but any real device will have a minimum and maximum sheet size. The specification also assumes that parameters may be set independently. This not always correct so that the value of one trait may constrain the value of another trait. For instance a printer may support transparent media and duplex printing but it will probably be constrained not to support both in the same job.

10.1 Capability and Constraint Definitions

New in JDF 1.1

While the **JDF** schema describes the structure of all **JDF** nodes, it does not provide for a way to allow a specific **JDF** Device to provide details on how it subsets (or extends) the **JD** language. This ability is provided by the **JDF** Device Capabilities features. With it, a **JDF** Device can describe details on supported Processes, Resources, Attributes and Attribute Values (and details about constraints and their interaction).

Figure 10-1: Parameter space in Device capabilities[a]



a. Note that the restriction to three dimensions is for graphical demonstration purposes only.

A **JDF** Device's capabilities are described as a space of allowed Resource parameter values within ***JDF*** Nodes. A Device in this context is assumed to execute one or more ***JDF*** Nodes. Its capabilities are defined by the space of acceptable **JDF** Resources for the Product Intent or Process described by the Node. An individual ***JDF*** Node definition can be compared to the capabilities of a **JDF** Device by looping over all Resource parameters of a ***JDF*** Node that is to be executed by a Device. The Job can be executed as specified (Attributes can be ignored if the *@SettingsPolicy* is "BestEffort") if all Job parameter values are within the ranges specified by the capabilities. If the capabilities describe Product Intent, the Job is executable as specified when all Product Intent ranges overlap with the capabilities description.

Details of the Elements needed for capability description are specified in ‣ Chapter 10 Device Capabilities.

It is assumed that ***Device*** Resources that describe capabilities will be transported in **JMF** ***KnownDevices*** Messages. However, a ***Device*** Resource SHOULD NOT specify the capabilities of its associated ***Device*** if a ***JDF*** Node links to the ***Device*** in order to specify that the ***Device*** is intended to execute the Node.

A capabilities description can also provide information necessary for the construction of a user interface to allow entry of the values to use for a **JDF** node. This includes specifying the NMTOKEN, enumeration or string values that are supported, hints for how to group features on the user interface, and macro definitions for features of the Device (allowing multiple **JDF** controls to be presented as a single user control).

10.2 Device Capability Definitions

New in JDF 1.1

The Elements in this section are used to specify capabilities of **JDF** Devices and provide infrastructure for defining preflight rules, including conducting a "**JDF** test run" and establishing a handshake between **JDF**-enabled products. When describing capabilities, note that only Attributes and Elements that are explicitly described within the capabilities structure are supported by the Device. For more details on preflight, see ▸ Section 6.3.27 Preflight.

Capabilities descriptions that are saved in files SHALL be formatted as a **JMF**/***Signal***/***Response*** to the ***KnownDevices*** Query Message.

Preflighting in Device Capabilities

While the actions and tests described in this section as pertaining to "preflighting" can be used by Processes and Resources that pertain to preflighting in the conventional sense, they can also be used to conduct "JDF test runs." A JDF test run might be part of a normal preflighting workflow, but the idea of a "JDF test run" is to compare the requirements of a JDF document or instance against the capabilities and JDF support of a Device or an integrated JDF environment.

10.2.1 DeviceCap

New in JDF 1.1

The ***DeviceCap*** Element describes the **JDF** nodes and resources that a device is capable of processing. elements that are derived from the abstract ***State*** elements are used to describe ranges and lists of ranges of allowed parameters.

Figure 10-2: DeviceCap – a diagram of its structure

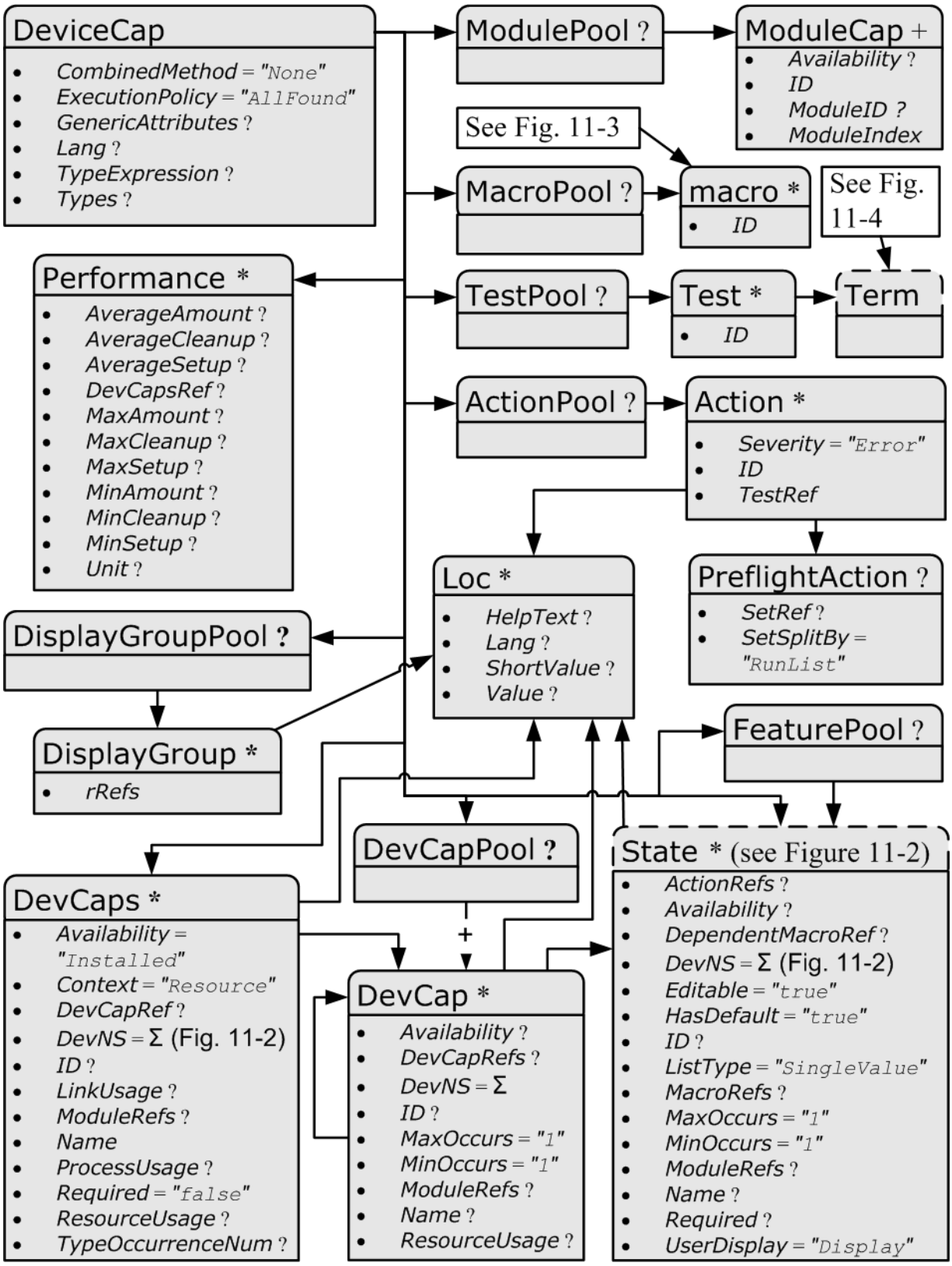


Table 10.1: DeviceCap Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
CombinedMethod = "None" Modified in JDF 1.3	enumerations	Specifies how the Processes specified in @*Types* are to be specified. If multiple values are specified, the structure of the **JDF** SHALL match one of the values. **Allowed values are:** Combined – The list of Processes in @*Types* SHALL be specified as a Combined Process. CombinedProcessGroup – The list of Processes in @*Types* SHALL be specified either as a Combined Process or as a "ProcessGroup" of individual Processes. In **JDF** 1.3 and beyond, the pair of individual tokens: "Combined ProcessGroup" replace this single value. Deprecated in JDF 1.3 GrayBox – The list of Processes in @*Types* SHALL be specified in a "ProcessGroup" with no nested **JDF** Nodes (i.e., a Gray Box). New in JDF 1.3 ProcessGroup – The list of Processes in @*Types* SHALL be specified as a "ProcessGroup" of individual Processes. None – No support for "Combined", "GrayBox" or "ProcessGroup". Only one individual Process type defined in @*Types* is supported.
ExecutionPolicy = "AllFound" New in JDF 1.2	enumeration	Describes the policy for finding and executing **JDF** Nodes as described in ▸ Section 4.2.1 Determining Executable Nodes. **Allowed values are:** RootNode – The Device will execute the root **JDF** Node only. It will not search the **JDF** tree for executable Nodes. This will commonly be used for sub **JDF** Nodes that have been spawned and targeted explicitly for the Device. FirstFound – The Device will execute the first Node found in the **JDF** tree that is executable by this Device. The search order is defined by the order in the XML. AllFound – The Device will execute all executable Nodes found in multiple passes of the **JDF** tree that are executable by this Device. The results of executing a Node are applied to the tree between passes.
GenericAttributes ?	NMTOKENS	List of all generic Attributes that are supported and unrestricted by the Device implementation. Descriptions of Attributes that appear in ***State*** Elements (see the following ▸ Section 10.2.7 State) overwrite the description in @*GenericAttributes*.
Lang ? New in JDF 1.2	languages	Specifies the localization(s) provided with the capabilities. If not specified, no localizations are provided.
OptionalCombinedTypes? Deprecated in JDF 1.2	NMTOKENS	List of optional **JDF** Node types. The entries of the list SHALL be a subset of @*Types*. **Values include those from:** ***JDF***/@*Types*. **Example:** a RIP with optional in-RIP trapping would specify @*OptionalCombinedTypes* = "Trapping" if @*Types* = "Trapping Interpreting Rendering". **Deprecation note:** Starting with **JDF** 1.2, use @*TypeExpression*.
Type ? Deprecated in JDF 1.2	NMTOKEN	**JDF** @*Type* Attribute of the supported Process. Extension types MAY be specified by stating the namespace prefix in the value. **Values include those from:** ***JDF***/@*Type*. **Deprecation note:** Starting with **JDF** 1.2, a single value of type is also defined in the @*Types* Attribute.
TypeExpression ? New in JDF 1.2	regExp	Regular expression that defines the allowed values of the Node's @*Types* Attribute. If not specified, defaults to the literal string defined in @*Types* (i.e., the ordered list of Processes defined in @*Types* SHALL match exactly). **Constraint:** in **JDF** 1.2 and above, one of @*Types* or @*TypeExpression* SHALL be specified.

Table 10.1: DeviceCap Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
TypeOrder ? Deprecated in JDF 1.2	enumeration	Ordering restriction for Combined Process Nodes and Process Group Nodes. **Allowed values are:** Fixed – The order of Process types specified in the @*Types* Attribute is ordered, and each type can be specified only once (e.g., @*Cutting*, @*Folding*). Order does matter. Unordered – The order of Process types specified in the @*Types* Attribute is unordered, and each type can be specified only once (e.g., "DigitalPrinting Screening Trapping"). Order does not matter. Unrestricted – The order of Process types specified in the @*Types* Attribute is unordered, and each type can be specified in multiples (e.g., "Cutting Folding"). The Device can do both Processes, in any order multiple times. **Deprecation note:** Starting with **JDF** 1.2, use @*TypeExpression*.
Types ? Modified in JDF 1.2	NMTOKENS	This Attribute represents the list of supported **JDF** Node @*Type* values. If any of the Node types are in a namespace other than **JDF**, the namespace prefix SHALL be included in this Node type name. The ordering is significant unless it is overridden by @*TypeExpression*. **Constraint:** in **JDF** 1.2 and above, one of @*Types* or @*TypeExpression* SHALL be specified. **Values include those from:** ***JDF***/@*Types*.
ActionPool ? New in JDF 1.2	element	Container for ***Action*** Elements for use as constraints.
DevCapPool ? New in JDF 1.3	element	Pool of ***DevCap*** Elements that can be referenced from multiple Elements within the ***DeviceCap*** structure.
DevCaps *	element	List of definitions of the accepted Resources and Elements. The ***DevCaps*** Elements are combined with a logical AND (i.e., a **JDF** SHALL fulfill all restrictions defined by the set of ***DevCaps***). Only Resources that are specified within this list are honored by the Device.
DisplayGroupPool ? New in JDF 1.2	element	List of ***DisplayGroup*** Subelements, which define the user interface presentation of sets of related ***DevCap*** Attribute Values. This is metadata to provide assistance in user interface display layout.
FeaturePool ? New in JDF 1.2	element	List of definitions of the accepted parameter space for Resources and Messages that are for user interface definition only — they do not map to actual **JDF** Resources or Messages. Definitions in ***FeaturePool*** typically reference macros that manipulate a set of related Resource values. These macros will set the appropriate **JDF** Attribute Values.
MacroPool ? New in JDF 1.2	element	Container for zero or more ***macro*** Elements, each of which contains an expression that can cause @*State* Attribute Values (e.g.,"CurrentValue" or "UserDisplay") to be changed.
ModulePool ? New in JDF 1.3	element	Pool of ***ModuleCap*** Elements that specify the availability of a given Module.
Performance *	element	Specification of a Devices performance capabilities.
TestPool ? New in JDF 1.2	element	Container for zero or more ***Test*** Elements that are referenced from ***ActionPool***/***Action*** Elements.
State * New in JDF 1.3	element	***Abstract State Element***s that define the parameter space that is covered by the Device. One ***State*** Element SHALL be defined for each supported Attribute of the ***JDF*** Node that is not specified @*GenericAttributes* or implied by @*TypeExpression* or @*Types*.

10.2.2 ActionPool

New in JDF 1.2

The ***ActionPool*** Subelement is used to contain Boolean expressions that are used for two purposes:

- As capability constraints to describe unsupported combinations of ***State*** Process and Attribute Values.

- As preflight constraints to describe unsupported combinations of basic ***PreflightReport*** values. (See Structure of the Abstract ***Evaluation*** Subelement in ▸ Section 10.2.13 Term. Note that the definition of the ***Term*** Element also describes how Boolean operators are employed by ***Action*** Elements via the @*TestRef* Attribute.)

ActionPool and the ***Action*** Elements it can contain, is interdependent on ***TestPool*** and the ***Test*** and ***Term*** Elements it can contain. For more information on ***TestPool***, see ▸ Section 10.2.12 TestPool.

Table 10.2: ActionPool Element

NAME	DATA TYPE	DESCRIPTION
Action *	element	A list of independent ***Action*** Elements.

10.2.2.1 Action

The ***Action*** Subelement is used to contain Boolean expressions that are used to describe a constraint that describes an unsupported combination of ***State*** Process and Attribute Values. If the ***Test*** referenced by @*TestRef* evaluates to "true", the combination of Processes and Attribute Values described is not allowed, and the action indicated by "Error", "Warning" or "Information" in the @*Severity* Attribute SHALL be taken.

Table 10.3: Action Element

NAME	DATA TYPE	DESCRIPTION
ID	ID	Unique identifier of the ***Action*** Element. This ID is used to refer to the ***Action*** Element (e.g., from a preflight report).
Severity = "Error"	enumeration	Indicates how the severity of the failure SHALL be treated when the expression defined by @*TestRef* is violated. **Allowed values are:** Error – The client SHALL display an error message and not allow the conflicting settings to persist. Warning – The client SHALL notify the user of the condition but allow the settings to persist if the user requests. Information – The client SHALL allow the settings to persist but inform the user of the issue.
TestRef	IDREF	Reference to a ***Test*** Element that is executed to evaluate this ***Action***.
Loc *	element	Text to describe an error if the ***Test*** fails. See ▸ Section 10.2.5.1 Loc.
PreflightAction ?	element	Provides additional constraints that are specific to the **Preflight** Process. See ▸ Section 8.112 PreflightParams.

10.2.3 DevCapPool

New in JDF 1.3

The ***DevCapPool*** provides a container for descriptions of Elements that are referenced from multiple locations within the **JDF**.

Table 10.4: DevCapPool Element

NAME	DATA TYPE	DESCRIPTION
DevCap +	element	***DevCap*** Elements that can be referenced from multiple locations within the ***DeviceCap*** structure. ***DevCap***/@*ID* SHALL be specified for all ***DevCap*** Elements in ***DevCapPool***.

10.2.4 ModulePool

New in JDF 1.3

Table 10.5: ModulePool Element

NAME	DATA TYPE	DESCRIPTION
ModuleCap +	element	***ModuleCap*** Elements that can be referenced from within the ***DeviceCap*** structure to specify features that depend on a given module being installed.

10.2.4.1 ModuleCap

New in JDF 1.3

Module elements specify features that depend on given hardware or software modules being installed. Hardware examples include duplex units for printers. Software licensing keys MAY also be modeled as modules.

Table 10.6: ModuleCap Element

NAME	DATA TYPE	DESCRIPTION
Availability ?	enumeration	Specifies whether the feature described by this ***State*** Element is available on the Device. **Allowed values are:** Installed – The feature is installed on the Device and is available for use. Module – "Module" is not to be specified recursively in a ***ModuleCap***. This value is only specified here to have a common enumeration set for all @*Availability* Attributes. NotInstalled – The feature has not been installed on the Device. NotLicensed – The feature has been installed on the Device but can not be used until licensed. Disabled – The feature is installed and licensed on the Device, but has been disabled.
ID	ID	@*ID* of the ***ModuleCap***.
ModuleID ?	integer	ID of the module that this ***ModuleCap*** describes. Refers to ***Device***/***Module***/@*ModuleID*. If neither @*ModuleID* nor @*ModuleIndex* are specified, no further details of the ***Module*** are known.
ModuleIndex ?	integer	Index of the module that this @*ModuleCap* describes. Refers to ***Device***/***Module***/@*ModuleIndex*. If neither @*ModuleID* nor @*ModuleIndex* are specified, no further details of the ***Module*** are known.

10.2.5 DevCaps

New in JDF 1.1

The ***DevCaps*** Element describes the valid parameter space of a **JDF** Resource, Message or ***ResourceLink*** that is consumed, honored or produced by a Device. Note that ***DevCaps*** not only describes the structure of the individual ***Resource*** and ***ResourceLink*** Elements but also of the ***AuditPool*** or other direct child Elements within a ***JDF*** Node. The ***DevCaps*** Element MAY be used to model ***Intent Resource***s as well as Process definition Resources.

Table 10.7: DevCaps Element (Sheet 1 of 3)

NAME	DATA TYPE	DESCRIPTION
Availability = "Installed" New in JDF 1.2	enumeration	Specifies whether the feature described by this ***DevCaps*** Element is available on the Device. **Allowed values are:** Installed – The feature is installed on the Device and is available for use. Module – The feature is provided by a module specified in @*ModuleRefs*. If and only if all modules that are listed in ModuleRefs are available, the feature is available. New in JDF 1.3 NotInstalled – The feature has not been installed on the Device. NotLicensed – The feature has been installed on the Device but can not be used until licensed. Disabled – The feature is installed and licensed on the Device but has been disabled.
Context = "Resource" New in JDF 1.2 Modified in JDF 1.3	enumeration	Describes whether the ***DevCaps*** context is within a Resource or a link to a Resource (not applicable to ***DevCaps*** Elements within Messages). **Allowed values are:** Element – The ***DevCaps*** context is describing a direct Element (e.g., an ***AuditPool***). JMF – The ***DevCaps*** context describes a **JMF** Message. Link – The ***DevCaps*** context is describing a link to a Resource. Resource – The ***DevCaps*** context is describing a Resource.

Table 10.7: DevCaps Element (Sheet 2 of 3)

NAME	DATA TYPE	DESCRIPTION
DevCapRef ? New in JDF 1.3	IDREFS	Reference to reusable ***DevCap*** Elements that are located in ***DeviceCap***/***DevCapPool***. A reference to a ***DeviceCap***/***DevCapPool***/***DevCap*** is equivalent to an inline ***DevCap*** in this ***DevCaps***. Exactly one of *@DevCapRef* or ***DevCap*** SHALL be specified.
DevNS = "http://www.CIP4.org/JDFSchema_1_1" 	URI	Namespace of the Resource or Message that is described.
ID ? New in JDF 1.2	ID	ID of this ***DevCaps*** Element. Used for reference from ***Performance*** Elements.
LinkUsage ? New in JDF 1.2	enumeration	Used when the *@Context* of this ***DevCaps*** = "Resource" or "Link". This field qualifies whether the ***DevCaps*** describes a Resource used as an input to a Process or as the output of a Process. **Default behavior:** this ***DevCaps*** applies to both usages. **Allowed values are:** Input – The ***DevCaps*** describes an Input Resource. Output – The ***DevCaps*** describes an Output Resource.
ModuleRefs ? New in JDF 1.3	IDREFS	List of modules that are needed for this feature to be available. At least one entry SHALL be specified if *@Availability* = "Module". The list of Modules is specified in ***DeviceCap***/***ModulePool***.
Name Modified in JDF 1.3	NMTOKEN	Name of the Element excluding the namespace prefix. When describing parameters of a ***ResourceLink***, *@Name* SHALL be the name of the referenced Resource and *@Context* = "Link". When ***DevCaps*** is specified as a Subelement of ***MessageService***, Name specifies the respective ***CommandTypeObj***, ***QueryTypeObj*** or ***ResponseTypeObj*** of the **JMF** Message. **Modification note:** Starting with **JDF** 1.3, *@Name* SHALL always specify the actual Resource name. Before **JDF** 1.3, the *@ResourceUsage* and *@ProcessUsage* of a Resource are specified in this Attribute. **Values include those from:** ▸ Chapter 10 Device Capabilities.
ProcessUsage ? New in JDF 1.3	NMTOKEN	***ResourceLink***/*@ProcessUsage* of the link to the Resource that is described by this ***DevCaps***. **Values include those from:** ***ResourceLink***/*@ProcessUsage*
Required = "false" New in JDF 1.2	boolean	If "true", the Element described by this ***DevCaps*** Element SHALL be present in a **JDF** or **JMF** (as appropriate) submitted to the Device. Note that this does not override the cardinality defined by the **JDF** specification when the specification requires the Resource to be specified. If an Attribute is REQUIRED (according to this specification), *@Required* SHALL be "true".
ResourceUpdate ? Deprecated in JDF 1.3	NMTOKENS	Specifies the capability to handle partial updates defined in ***ResourceUpdate*** Elements. **Values include:** None – *@ResourceUpdate* is not supported. SHALL NOT be combined with any other value. JMFID – **JMF** ***Resource*** Messages that reference ***ResourceUpdate*** Elements that have been previously loaded to the Device are accepted. PDLID – References from PDL data (e.g., PPML TicketRef elements that reference ***ResourceUpdate*** Elements that have been previously loaded to the Device are accepted).
ResourceUsage ? New in JDF 1.3	NMTOKEN	***Resource***/*@ResourceUsage* of the Resource that is described by this ***DevCaps***. **Values include those from:** ***FileSpec***/*@ResourceUsage Attribute Values*

Table 10.7: DevCaps Element (Sheet 3 of 3)

NAME	DATA TYPE	DESCRIPTION
TypeOccurrenceNum ? New in JDF 1.2	IntegerRangeList	Specifies which occurrence(s) of the **JDF** Node type that is specified either within the ***DeviceCap***/@*Types* or by ***DeviceCap***/@*TypeExpression* that the Element that is defined by this ***DevCaps*** applies to. If not specified, this ***DevCaps*** Element describes Elements belonging to all **JDF** Nodes or Combined Process steps with a matching type that are not defined by other ***DevCaps*** entries. Note: This is an index into the list of matching @*Type* values and not an index into the complete list specified by @*Types* or @*TypeExpression*. The first occurrence is "0", and the last occurrence is "-1", etc.
Types ? Deprecated in JDF 1.2	NMTOKENS	List of **JDF** Node types that a ***DevCaps*** applies to. The value of @*Types* SHALL be a subset of @*Types* in ***DeviceCap***. **Values include those from:** ***JDF***/@*Types*. **Deprecation note:** Starting with **JDF** 1.2, use @*TypeOccurrenceNum*.
DevCap * Modified in JDF 1.3	element	List of definitions of the accepted parameter space for Resources and Messages. The parameter spaces of multiple ***DevCap*** Elements are combined as a superset of the individual ***DevCap*** Elements. Only Elements that are explicitly specified as ***DevCap*** Elements within a ***DevCaps*** are supported. When a capabilities description is constructed using constraints, each ***DevCaps*** SHOULD contain only a single ***DevCap*** Element (although a ***DevCap*** Element MAY still contain multiple ***DevCap*** Subelements). Exactly one of @*DevCapRef* or ***DevCap*** SHALL be specified, though if ***DevCap*** is specified, it MAY occur multiple times.
Loc * New in JDF 1.2	element	The localization(s) of the Resource, Message or ***ResourceLink*** name as described by this ***DevCaps*** Element. See ▸ Section 10.2.5.1 Loc.

10.2.5.1 Loc

New in JDF 1.2

Each ***Loc*** element describes a localization for some value. Note that this Subelement is used in many of the Elements subordinate to ***DeviceCap*** Elements.

Table 10.8: Loc Element

NAME	DATA TYPE	DESCRIPTION
HelpText ?	string	Localized text used for supplemental help for the value being localized. Note that this is the text often used for a pop-up window when help is requested.
Lang ?	language	The language code for this localization. If not specified, then it defaults to the value of the first language specified in the @*Lang* Attribute of the ***DeviceCap*** Element. Note that each language in a list of localizations (i.e., ***Loc*** *) SHALL be unique.
ShortValue ?	string	The short form of the localization. Defaults to the value of @*Value*. This value would be used when a small fixed field is REQUIRED for the name of the field (a PDA for example).

10.2.6 DevCap

New in JDF 1.1

The ***DevCap*** Element describes the valid parameter space of a **JDF** Resource, Message or Element that is consumed or produced by a Device. The structure of the ***DevCap*** is identical to that of the **JDF** Resource, Message or Element that it models. Individual Attributes are replaced by the appropriate ***State*** Elements. For more details on ***State*** Elements, see ▸ Section 10.2.7 State. The @*Name* Attribute of the ***State*** Element SHALL match the Attribute key that is described. If no ***State*** Element exists for a given Attribute, it is assumed to be unsupported. The restrictions of multiple Attributes and Elements are combined with a logical AND.

Subelements of Resources are modeled by including nested ***DevCap*** with a @*ResourceUsage* Attribute equal to the Subelements tag name or @*ResourceUsage* if the Subelement is a ***FileSpec***. Attributes of the ***ResourceLink*** belonging to the ***Resource*** (e.g., @*Transformation* or the various pipe control parameters can also be restricted).

Table 10.9: DevCap Element

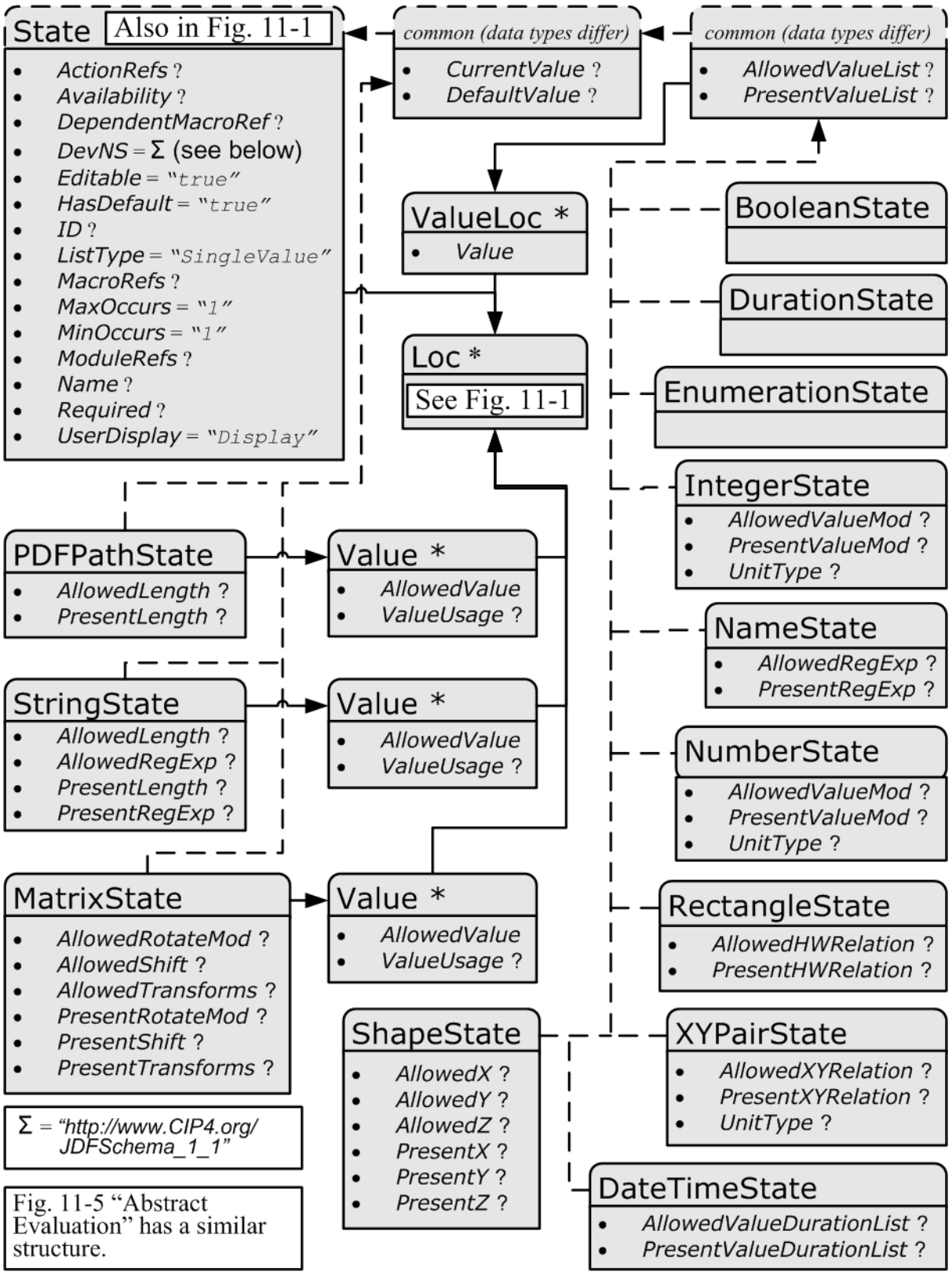
NAME	DATA TYPE	DESCRIPTION
Availability ? New in JDF 1.2	enumeration	Specifies whether the feature described by this ***DevCap*** Element is available on the Device. **Default value is from:** parent ***DevCaps***/@*Availability* or ***DevCap***/@ *Availability*. **Allowed values are:** Installed – The feature is installed on the Device and is available for use. Module – The feature is provided by a module specified in @*ModuleRefs*. If and only if all modules that are listed in @*ModuleRefs* are available, the feature is available. New in JDF 1.3 NotInstalled – The feature has not been installed on the Device. NotLicensed – The feature has been installed on the Device but can not be used until licensed. Disabled – The feature is installed and licensed on the Device but has been disabled.
DevCapRefs ? New in JDF 1.3	IDREFS	References to reusable ***DevCap*** Elements that are located in ***DeviceCap***/***DevCapPool***. A reference to a ***DeviceCap***/***DevCapPool***/***DevCap*** is equivalent to an inline ***DevCap*** in this ***DevCap***. If both @*DevCapRefs* and ***DevCap*** Elements exist, they specify the union of both.
DevNS = "http://www.CIP4.org/JDFSchema_1_1"	URI	Namespace of the Element that is described by this ***DevCap***.
ID ? New in JDF 1.3	ID	@*ID* of this ***DevCap***. Used to reference a ***DevCap***. ***DevCap***/@*ID* SHALL be specified for all direct ***DevCap*** child Elements in ***DevCapPool***.
MaxOccurs = "1" Modified in JDF 1.2	integer	Maximum number of occurrences of the Element described by this ***DevCap***. In **JDF** 1.1 the "INF" value was defined as *"unbounded"*.
MinOccurs = "1"	integer	Minimum number of occurrences of the Element described by this ***DevCap***.
ModuleRefs ? New in JDF 1.3	IDREFS	List of modules that are needed for this feature to be available. At least one entry SHALL be specified if @*Availability* = "Module". The list of Modules is specified in ***DeviceCap***/***ModulePool***.
Name ?	NMTOKEN	Name of the Resource that is described. Default, if this ***DevCap*** is the direct child of a ***DevCaps*** Element: the value of the parent ***DevCaps***/@*Name*.@*Name* SHALL be specified for all direct ***DevCap*** child Elements in ***DevCapPool*** or ***DevCap*** Elements. **Modification note:** Starting with **JDF** 1.3, @*Name* SHALL always specify the actual ***Resource*** name. Before **JDF** 1.3 @*ResourceUsage* of a Resource was specified in this Attribute. **Values include those from:** ▸ Chapter 10 Device Capabilities.
ResourceUsage ? New in JDF 1.3	NMTOKEN	***Resource***/@*ResourceUsage* of the Resource that is described by this ***DevCap***. **Values include those from:** ***FileSpec***/@*ResourceUsage Attribute Values*
DevCap *	element	Definition of the accepted parameter space for the messages or resources subelements. If multiple ***DevCap*** elements with the same @*Name* exist, they describe individual subelements with different properties. The properties SHALL each be fulfilled by individual subelements of the element that is described by this ***DevCap***. For instance, if two ***DevCap*** elements with the @*MinOccurs* = "1" are specified, the ***JDF*** element SHALL contain two elements with a node name = ***DevCap***/@*Name*.
Loc * New in JDF 1.2	element	The localization(s) of the Element name. See ▸ Section 10.2.5.1 Loc.
State *	element	***Abstract State Element***s that define the parameter space that is covered by the Device. One ***State*** Element SHALL be defined for each supported Attribute or Intent Span Element of the Element that this ***DevCap*** defines that is not specified ***DeviceCap***/@*GenericAttributes*.

10.2.7 State

New in JDF 1.1

Figure 10-3: Abstract State Element – a diagram of its structure shows all ***State*** Elements.

Figure 10-3: Abstract State Element – a diagram of its structure



10.2.7.1 Abstract State Element

Table 10.10 Abstract State Element describes the common, data type-independent parameters of all ***State*** Elements. The ***State*** Elements that contain no value restriction Attributes (e.g., @*AllowedValueList*) or Elements (e.g., ***ValueLoc***) have no further restrictions other than the data type of their values. If value restrictions are specified in addition to a list

of explicit values in @*AllowedValueList*, @*CurrentValue*, ***Value*** or ***ValueLoc***, the ***State*** Element describes the union of restrictions (i.e., the ***State*** Element matches an Attribute that matches either the explicit list or the additional restrictions).

Table 10.10: Abstract State Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ActionRefs ? New in JDF 1.2	IDREFS	Zero or more references to ***Action*** Elements that operate on the parameter. All ***Action*** Elements referenced SHALL evaluate to "false" for the value of the ***State*** Element to be valid. Any ***Action*** Elements referenced in @*ActionRefs* SHOULD be evaluated whenever the Attribute described by this ***State*** Element is manipulated or changed in order to catch any Attributes that become invalid due to the manipulation.
Availability ? New in JDF 1.2	enumeration	Specifies whether the feature described by this ***State*** Element is available on the Device. **Default behavior:** the value specified or implied by the parent Element **Allowed values are:** Installed – The feature is installed on the Device and is available for use. Module – The feature is provided by a module specified in @*ModuleRefs*. If and only if all modules that are listed in ModuleRefs are available, the feature is available. New in JDF 1.3 NotInstalled – The feature has not been installed on the Device. NotLicensed – The feature has been installed on the Device but can not be used until licensed. Disabled – The feature is installed and licensed on the Device, but has been disabled.
DependentMacroRef ? New in JDF 1.2	IDREF	A reference to a ***macro*** that conditionally modifies the @*UserDisplay* Attribute of this ***State*** Element. If present, this referenced ***macro*** SHALL be executed when the ***State***/@*UserDisplay* is "Dependent" and the user interface is being initialized. It is RECOMMENDED that the ***macro*** referenced by @*DependentMacroRef* only change the value of @*UserDisplay* or @*Editable* Attributes. For more information on ***macro*** definitions, see ▸ Section 10.2.10 MacroPool.
DevNS = "http://www.CIP4.org/JDFSchema_1_1" New in JDF 1.2	URI	Namespace of the Attribute or Element that is described by this ***State*** Element.
Editable = "true" New in JDF 1.2	boolean	When "true", the feature and its current value can be edited by the user. If "false", the user interface SHALL NOT allow user modification of the ***State*** Element's current value.
HasDefault = "true"	boolean	A flag that describes whether the parameter has a default supplied by the Device. If set, @*DefaultValue* SHALL be set.
ID ? New in JDF 1.2	ID	An identification value to allow external reference.
ListType = "SingleValue" New in JDF 1.2 Modified in JDF 1.3	enumeration	Specifies what type of list or object the ***State*** variable describes. **Allowed values are from:** ▸ Table 10.11 ListType Attribute Values.
MacroRefs ? New in JDF 1.2	IDREFS	Zero or more references to ***macro*** Elements that operate on the parameter. These ***macro*** Elements set other ***State*** Attribute Values as appropriate. Any ***macro*** Elements referenced in @*MacroRefs* SHALL be evaluated whenever the Attribute described by this ***State*** Element is manipulated or changed to affect any necessary changes to other Attributes. The ***macro*** Elements can change Attributes such as the @*CurrentValue* Attribute of a ***State*** or its @*UserDisplay* Attribute. For more information on ***macro*** definitions, see ▸ Section 10.2.10 MacroPool.
MaxOccurs = "1" New in JDF 1.2	integer	Maximum number of Elements in the list described by this ***State*** (e.g., the maximum number of integers in an integer list). If @*MaxOccurs* is not "1", the ***State*** Element refers to a list or a range of list values (e.g., a ***NameState*** will allow a list of NMTOKENS).

Table 10.10: Abstract State Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
MinOccurs = "1" New in JDF 1.2	integer	Minimum number of Elements in the list described by this ***State***. If @*MinOccurs* is not "1", the ***State*** Element refers to a list or a range of list values (e.g., a ***NameState*** will allow a list of NMTOKENS).
ModuleRefs ? New in JDF 1.3	IDREFS	List of modules that are needed for this feature to be available. At least one entry SHALL be specified if @*Availability* = "Module". The list of Modules is specified in ***DeviceCap***/***ModulePool***.
Name ?	NMTOKEN	Name of the Attribute that is described by this ***State***. If @*Name* is omitted this ***State*** describes the Element's text (i.e., the text between the XML start and end tag).
Required ? New in JDF 1.2	boolean	If "true", then the Attribute or Span Element described by this ***State*** Element is REQUIRED to be present in a **JDF** or **JMF** (as appropriate) submitted to the Device. Note that this does not override the cardinality specified by the **JDF** specification where the specification requires the Element to be specified. Deprecation note: Starting with **JDF** 1.6, replaced by @*Cardinality*.
Span ? New in JDF 1.1A Deprecated in JDF 1.2	boolean	A flag that describes whether the parameter is an intent span data type. For example a ***State*** Element describing an XYPairSpan would have @*DataType* = "XYPairState" and @*Span* = "true". Replaced with @*ListType* = "Span" in **JDF** 1.2 and beyond.
UserDisplay = "Display" New in JDF 1.2	enumeration	Indicates whether the feature SHALL be displayed in user interfaces. **Allowed values are:** Display – The feature SHALL be displayed. Hide – The feature SHALL NOT to be displayed. Dependent – The feature SHALL be conditionally displayed depending on the action specified by the ***macro*** referenced by @*DependentMacroRef*. **Note:** This action is only taken when the user interface is first initialized.
Loc * New in JDF 1.2	element	The localization(s) of the @*Name* of the Attribute that is described by this ***State*** Element. See ▸ Section 10.2.5.1 Loc.

10.2.7.1.1 ListType Attribute Values

*Table 10.11: **ListType** Attribute Values*

VALUE	DESCRIPTION
CompleteList	The ***State*** describes a list of individual values. Each value SHALL occur exactly once.
CompleteOrderedList	The ***State*** describes an ordered list of individual values. Each value SHALL occur exactly once and in the specified order.
ContainedList	The ***State*** describes a list of individual values. The ***State*** = "true" if at least one of the values occurs. This value is only expected to be used in ***BasicPreflightTest*** Elements.
List	The ***State*** describes a list of individual values.
OrderedList	The ***State*** describes an ordered list of individual values.
OrderedRangeList	The ***State*** describes an ordered RangeList of individual values.
Range New in JDF 1.3	The ***State*** describes an individual Range of values.
RangeList	The ***State*** describes a RangeList of values.
SingleValue	The ***State*** describes an individual value.
Span	The ***State*** describes a Span Element in an ***Intent Resource***.
UniqueList	The ***State*** describes a list of individual values. Each value SHALL NOT occur more than once.
UniqueOrderedList	The ***State*** describes an ordered list of individual values. Each value SHALL NOT occur more than once.
UniqueOrderedRangeList	The ***State*** describes an ordered RangeList of individual values. Each explicit or implied value SHALL NOT occur more than once.
UniqueRangeList	The ***State*** describes a RangeList of values. Each explicit or implied value SHALL NOT occur more than once.

10.2.7.2 State Elements

The following types of ***State*** Elements are defined:

Table 10.12: List of State Elements (Sheet 1 of 2)

NAME	DESCRIPTION
BooleanState	Describes a set of boolean values.
DateTimeState New in JDF 1.2	Describes a set of dateTime values.
DurationState New in JDF 1.2	Describes a set of duration values.
EnumerationState	Describes a set of enumeration values.
IntegerState	Describes a numerical range of integer values.
MatrixState	Describes a range of matrices. Generally used to define valid orientations of ***Component*** Resources.
NameState	Describes a set of NMTOKEN values.
NumberState	Describes a numerical range of values.
PDFPathState New in JDF 1.2	Describes a set of PDFPaths.

Table 10.12: List of State Elements (Sheet 2 of 2)

NAME	DESCRIPTION
RectangleState New in JDF 1.2	Describes a set of 4 value rectangle values.
ShapeState	Describes a set of 3 value shape values.
StringState	Describes a set of string values.
XYPairState	Describes a set of XYPair values.

10.2.7.2.1 BooleanState

New in JDF 1.1

This ***State*** Subelement is used to describe ranges of Boolean values. It inherits from the Abstract ***State*** Element described above.

Table 10.13: BooleanState Element

NAME	DATA TYPE	DESCRIPTION
AllowedValueList ? New in JDF 1.1A	enumerations	A list of all legal values. **Allowed values are:** true false
CurrentValue ?	boolean	Current value for the current running Job set in the Device.
DefaultValue ?	boolean	Default value if not specified in a submitted **JDF**. @*DefaultValue* SHALL be specified if @*HasDefault* = "true".
PresentValueList ? New in JDF 1.1A	enumerations	A list of all supported values that can be chosen without operator intervention. **Default value is from:** @*AllowedValueList*. **Allowed values are:** true false
ValueLoc * New in JDF 1.2	element	Localization(s) of "true" and/or "false" values. See ▸ Section 10.2.7.2.1.1 ValueLoc.

10.2.7.2.1.1 ValueLoc

New in JDF 1.2

Each ***ValueLoc*** Element describes one or more localizations for an Attribute Value. Note that the ***ValueLoc*** Element occurs in the definition of all ***State*** Elements except ***MatrixState***, ***PDFPathState*** and ***StringState***.

Table 10.14: ValueLoc Element

NAME	DATA TYPE	DESCRIPTION
Value	string	The Attribute Value to be localized. If the data type of the allowed value is not string (e.g., if ***ValueLoc*** is used in the context of a ***MatrixState***), @*Value* SHALL be an instance of the appropriate data type.
Loc *	element	The localization(s) of the Attribute Value. See ▸ Section 10.2.5.1 Loc.

10.2.7.2.2 DateTimeState

New in JDF 1.2

This ***State*** Subelement is used to describe ranges of dateTime values. It inherits from the Abstract ***State*** Element described above.

Table 10.15: DateTimeState Element

NAME	DATA TYPE	DESCRIPTION
AllowedValueDurationList ?	DurationRangeList	List of inclusive minimum and maximum allowed values relative to the current system time.
AllowedValueList ?	DateTimeRangeList	A list of all supported values.
CurrentValue ?	dateTime	Current value for the current running Job set in the Device.
DefaultValue ?	dateTime	Default value if not specified in a submitted **JDF**. @*DefaultValue* SHALL be specified if @*HasDefault* = "true".
PresentValueDurationList ?	DurationRangeList	List of inclusive minimum and maximum allowed values that can be chosen without operator intervention relative to the current system time. If not specified, the value of @*AllowedValueDurationList* is applied.
PresentValueList ?	DateTimeRangeList	Inclusive minimum and maximum allowed value that can be chosen without operator intervention. If not specified, the value of @*AllowedValueList* is applied.
ValueLoc *	element	Localization(s) of specific dates. See ▸ Section 10.2.7.2.1.1 ValueLoc.

10.2.7.2.3 DurationState

New in JDF 1.2

This ***State*** Subelement is used to describe ranges of duration values. It inherits from the Abstract ***State*** Element described above.

Table 10.16: DurationState Element

NAME	DATA TYPE	DESCRIPTION
AllowedValueList ?	DurationRangeList	A list of all supported values.
CurrentValue ?	duration	Current value for the current running Job set in the Device.
DefaultValue ?	duration	Default value if not specified in a submitted **JDF**. @*DefaultValue* SHALL be specified if @*HasDefault* = "true".
PresentValueList ?	DurationRangeList	Inclusive minimum and maximum allowed value that can be chosen without operator intervention. If not specified, the value of @*AllowedValueList* is applied.
ValueLoc *	element	Localization(s) of specific durations. See ▸ Section 10.2.7.2.1.1 ValueLoc.

10.2.7.2.4 EnumerationState

New in JDF 1.1

This ***State*** Subelement is used to describe ranges of enumerative values. It inherits from the Abstract ***State*** Element described above. It is identical to the ***NameState*** Element except that it describes a closed list of enumeration values.

Table 10.17: EnumerationState Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AllowedValueList ?	enumerations	A list of all supported values. The values specified in @*AllowedValueList* SHALL be a subset of the enumeration specified by @*Name*. If not specified, all enumerations defined by the XML schema are valid. In order to enable capabilities to be specified without access to the **JDF** XML schema, it is strongly RECOMMENDED to specify @*AllowedValueList*, even when the entire range of schema-valid values is supported.

Table 10.17: EnumerationState Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
CurrentValue ?	enumeration	Current value for the current running Job set in the Device. @*CurrentValue* SHALL match the enumeration defined in the Resource.
DefaultValue ?	enumeration	Default value if not specified in a submitted **JDF**. @*DefaultValue* SHALL match the enumeration defined in the Resource, and SHALL be specified if @*HasDefault* = "true".
PresentValueList ?	enumerations	A list of values that can be chosen without operator intervention. @*PresentValueList* SHALL match the enumeration defined in the Resource. **Default value is from:** @*AllowedValueList*.
ValueLoc * New in JDF 1.2	element	Localizations of the enumerations listed in @*AllowedValueList* and @*PresentValueList*. See ▸ Section 10.2.7.2.1.1 ValueLoc.

10.2.7.2.5 IntegerState

New in JDF 1.1

This ***State*** Subelement is used to describe ranges of integer values. It inherits from the Abstract ***State*** Element described above.

Table 10.18: IntegerState Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AllowedValueList ? Modified in JDF 1.2	IntegerRangeList	A list of all supported values.
AllowedValueMax ? Deprecated in JDF 1.2	integer	Inclusive maximum allowed value. Replaced by @*AllowedValueList* in **JDF** 1.2 and beyond.
AllowedValueMin ? Deprecated in JDF 1.2	integer	Inclusive minimum allowed value. Replaced by @*AllowedValueList* in **JDF** 1.2 and beyond.
AllowedValueMod ? New in JDF 1.2	XYPair	X defines the Modulo and Y the offset of the allowed value. In other words, if @*AllowedValueMod* = "10 2", only the values ... -8,2,12,22 ... are allowed. If not specified, all values in the range are valid. If ((N%X)−Y = 0) then N is a valid value. **Note:** "Modulo" is the remainder of an integer division. For example: 4 mod 3 = 4 -3 = 1; 17 mod 3 = 17-5*3 = 2; and 3 mod 3 = 3-3 = 0.
CurrentValue ?	integer	Current value for the current running Job set in the Device.
DefaultValue ?	integer	Default value if not specified in a submitted **JDF**. @*DefaultValue* SHALL be specified if @*HasDefault* = "true".
PresentValueList ? Modified in JDF 1.2	IntegerRangeList	A list of values that can be chosen without operator intervention. If not specified, the value of @*AllowedValueList* is applied.
PresentValueMax ? Deprecated in JDF 1.2	integer	Inclusive maximum allowed value that can be chosen without operator intervention. If not specified, the value of @*AllowedValueMax* is applied. Replaced by @*PresentValueList* in **JDF** 1.2 and beyond.
PresentValueMin ? Deprecated in JDF 1.2	integer	Inclusive minimum allowed value that can be chosen without operator intervention. If not specified, the value of @*AllowedValueMin* is applied. Replaced by @*PresentValueList* in **JDF** 1.2 and beyond.
PresentValueMod ? New in JDF 1.2	XYPair	X defines the Modulo and Y the offset of the present value. In other words, if @*AllowedValueMod* = "10 2", only the values ... -8,2,12,22 ... are allowed. If not specified, the value of @*AllowedValueMod* is applied. If ((N%X)−Y = 0) then N is a valid value.

Table 10.18: IntegerState Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
UnitType ? New in JDF 1.2	NMTOKEN	Specifies the unit type that this ***State*** Element represents. Used to enable an application to localize the representation of the units. @*UnitType* SHOULD be specified if the ***IntegerState*** represents a value that has units. User interfaces might not display correctly if @*UnitType* is not specified for Attributes with units. **Values include:** Angle – The Attribute is defined in degrees. AngularVelocity – Rotations / minute. Area – Area in square meters (m^2). Currency – The local currency. Length – In points (1/72 inch). LengthMu – Length in microns (used for paper thickness). LineScreen – The lines per inch (lpi) for conventionally screened halftone, screened grayscale and screened monotone bitmap images. PaperWeight – In grams per square meter (g/m^2). Percentage – A percentage value. Pressure – In Pascals. Resolution – The dots per inch (dpi) for print output and bitmap image (e.g., TIFF or BMP) file resolution. ScreenResolution – The pixels per inch (ppi) for screen display (e.g., softproof display and user interface display), scanner capture settings and digital camera settings. SpotResolution – For imaging Devices such as filmsetters, platesetters and proofers, the fundamental imaging unit (e.g., one "on" laser or imaging-head imaged unit). Note that many imaging Devices construct dots from multiple imaging spots, so dpi and spots per inch (spi) are not equivalent. Temperature – Temperature in degrees Centigrade. Velocity – Defined as meters/hour. Weight – Weight in grams.
ValueLoc * New in JDF 1.2	element	Localization(s) of specific values. See ▸ Section 10.2.7.2.1.1 ValueLoc.

10.2.7.2.6 MatrixState

New in JDF 1.1

This ***State*** Subelement is used to describe ranges of matrix values. It inherits from the Abstract ***State*** Element described above. It is primarily intended to specify orientations and manipulation capabilities of ***PhysicalResource***s (e.g., in finishing Devices).

Table 10.19: MatrixState Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AllowedRotateMod ? New in JDF 1.2	double	Allowed Modulo of the allowed rotations and offset in degrees. **Values include:** 360 – No rotation 90 – Any orthogonal rotation. 0 – Any rotation is allowed.
AllowedShift ? New in JDF 1.2	DoubleList	Minimum and maximum allowed shift of the matrix. If not specified, any shift is valid. If @*AllowedTransforms* is specified, the implied shift defined in ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component is subtracted from @*AllowedShift*, thus all in-place rotations have an implied @*AllowedShift* value of "0 0 0 0". (No shift = "0 0 0 0".) The first pair of numbers is the XY pair that defines the minimum shift, and the second pair is the XY pair that defines the maximum shift.
AllowedTransforms ? New in JDF 1.2	Orientations	List of valid orthogonal transformations of the matrix. Any of the eight predefined transforms for ***PhysicalResource***s as defined in ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component.
CurrentValue ?	matrix	Current value for the current running Job set in the Device.

Table 10.19: MatrixState Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
DefaultValue ?	matrix	Default value if not specified in a submitted **JDF**. @*DefaultValue* SHALL be specified if @*HasDefault* = "true".
PresentRotateMod ? New in JDF 1.2	double	Present Modulo of the allowed rotations and offset in degrees that can be chosen without operator intervention. If not specified, the value of @*AllowedRotateMod* is applied. **Values include:** 360 – No rotation is allowed. 90 – Any orthogonal rotation. 0 – Any rotation is allowed.
PresentShift ? New in JDF 1.2	DoubleList	If @*PresentTransforms* is specified, the implied shift defined in ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component is subtracted from @*PresentShift*, thus all in-place rotations have an implied @*PresentShift* value of "0 0 0 0". If not specified, the value of @*AllowedShift* is applied.
PresentTransforms ? New in JDF 1.2	Orientations	Any of the eight predefined transforms for ***PhysicalResource***s as defined in ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component. If not specified, the value of @*AllowedTransforms* is applied.
Value *	element	A list legal values. See ▸ Section 10.2.7.2.6.1 Value.

10.2.7.2.6.1 Value

Table 10.20: MatrixState/Value Element

NAME	DATA TYPE	DESCRIPTION
AllowedValue	matrix	A legal value for a matrix variable.
PresentValue ? Deprecated in JDF 1.2	matrix	A legal value for a matrix variable that can be chosen without operator intervention. If not specified, the value of @*AllowedValue* is applied. In **JDF** 1.2 and beyond, use @*ValueUsage*.
ValueUsage ? New in JDF 1.2	enumeration	Defines whether the value defined in @*AllowedValue* means "Present", "Allowed" or both. **Default behavior:** valid for both "Present" and "Allowed". **Allowed values are:** Present – Present configuration is supported. Allowed – Allowed configuration is supported.
Loc * New in JDF 1.2	element	The localization(s) of the string defined in @*AllowedValue*. See ▸ Section 10.2.5.1 Loc.

10.2.7.2.7 NameState

New in JDF 1.1

This ***State*** Subelement is used to describe ranges of NMTOKEN values. It inherits from the Abstract ***State*** Element described above.

Table 10.21: NameState Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AllowedRegExp ? New in JDF 1.2	regExp	Regular expression that limits the allowed values.
AllowedValueList ?	NMTOKENS	A list legal values.
CurrentValue ?	NMTOKEN	Current value for the current running Job set in the Device.

Table 10.21: NameState Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
DefaultValue ?	NMTOKEN	Default value if not specified in a submitted **JDF**. @*DefaultValue* SHALL be specified if @*HasDefault* = "true".
PresentRegExp ? New in JDF 1.2	regExp	Regular expression that limits the values that can be chosen without operator intervention. If not specified, the value of @*AllowedRegExp* is applied.
PresentValueList ?	NMTOKENS	A list of values that can be chosen without operator intervention. If not specified, the value of @*AllowedValueList* is applied.
ValueLoc * New in JDF 1.2	element	Localization(s) of the NMTOKENS listed in @*AllowedValueList* or @*PresentValueList* or implied by @*AllowedRegExp* or @*PresentRegExp*. See ▸ Section 10.2.7.2.1.1 ValueLoc.

10.2.7.2.8 NumberState

New in JDF 1.1

This ***State*** Subelement is used to describe ranges of double values. It inherits from the Abstract ***State*** Element described above.

Table 10.22: NumberState Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AllowedValueList ? Modified in JDF 1.2	DoubleRangeList	A list of supported values.
AllowedValueMax ? Deprecated in JDF 1.2	double	Inclusive maximum allowed value. Replaced by @*AllowedValueList* in **JDF** 1.2 and beyond.
AllowedValueMin ? Deprecated in JDF 1.2	double	Inclusive minimum allowed value. Replaced by @*AllowedValueList* in **JDF** 1.2 and beyond.
AllowedValueMod ? New in JDF 1.2	XYPair	X defines the Modulo and Y the offset of the allowed value. In other words, if @*AllowedValueMod* = "10 2", only the values ... -8,2,12,22 ... are allowed. If not specified, all values in the range are valid. If ((N%X)−Y = 0) then N is a valid value. **Note:** "Modulo" is the remainder of an integer division. For example: 4 mod 3 = 4 -3 = 1; 17 mod 3 = 17-5*3 = 2; and 3 mod 3 = 3-3 = 0.
CurrentValue ?	double	Current value for the current running Job set in the Device.
DefaultValue ?	double	Default value if not specified in a submitted **JDF**. @*DefaultValue* SHALL be specified if @*HasDefault* = "true".
PresentValueList ? Modified in JDF 1.2	DoubleRangeList	A list of values that can be chosen without operator intervention. If not specified, the value of @*AllowedValueList* is applied.
PresentValueMax ? Deprecated in JDF 1.2	double	Inclusive maximum allowed value that can be chosen without operator intervention. If not specified, the value of @*AllowedValueMax* is applied. Replaced by @*PresentValueList* in **JDF** 1.2 and beyond.
PresentValueMin ? Deprecated in JDF 1.2	double	Inclusive minimum allowed value that can be chosen without operator intervention. If not specified, the value of @*AllowedValueMin* is applied. Replaced by @*PresentValueList* in **JDF** 1.2 and beyond.
PresentValueMod ? New in JDF 1.2	XYPair	X defines the Modulo and Y the offset of the allowed value. In other words, if @*AllowedValueMod* = "10 2", only the values ... -8,2,12,22 ... are allowed. If not specified, the value of @*AllowedValueMod* is applied. If ((N%X)−Y = 0) then N is a valid value.

Table 10.22: NumberState Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
UnitType ? New in JDF 1.2	NMTOKEN	Specifies the unit type that this ***State*** Element represents. Used to enable an application to localize the representation of the units. @*UnitType* SHALL be specified if the ***NumberState*** represents a value that has units. ***NumberState*** has the values as ***IntegerState*** plus a few more: **Values include:** CMYKColor – Four values representing a CMYK color. LabColor – Three values representing a Lab color. sRGBColor – Three values representing a sRGB color. **Values include those from:** ***IntegerState***/@*UnitType* (▸ Table 10.18 IntegerState Element).
ValueLoc * New in JDF 1.2	element	Localization(s) of specific values. See ▸ Section 10.2.7.2.1.1 ValueLoc.

10.2.7.2.9 PDFPathState

New in JDF 1.2

This ***State*** Subelement is used to describe ranges of PDF paths. It inherits from the Abstract ***State*** Element described above.

Table 10.23: PDFPathState Element

NAME	DATA TYPE	DESCRIPTION
AllowedLength ?	IntegerRange	Inclusive minimum and maximum length of valid PDF path in multi-byte characters. Note that this is the length in characters and not in bytes of the internal encoding of an application.
CurrentValue ?	PDFPath	Current value for the current running Job set in the Device.
DefaultValue ?	PDFPath	Default value if not specified in a submitted **JDF**. @*DefaultValue* SHALL be specified if @*HasDefault* = "true".
PresentLength ?	IntegerRange	Inclusive minimum and maximum length of valid PDF path in characters that can be chosen without operator intervention. If not specified, the value of @*AllowedLength* is applied.
Value *	element	The localization(s) of the PDF path defined in @*AllowedValue*. See ▸ Section 10.2.7.2.9.1 Value.

10.2.7.2.9.1 Value

Table 10.24: PDFPathState/Value Element

NAME	DATA TYPE	DESCRIPTION
AllowedValue	PDFPath	A legal value for a matrix variable.
ValueUsage ?	enumeration	Defines whether the value defined in @*AllowedValue* means "Present", "Allowed" or both. **Default behavior:** valid for both "Present" and "Allowed". **Allowed values are:** Present – Present configuration is supported. Allowed – Allowed configuration is supported.
Loc *	element	The localization(s) of the string defined in @*AllowedValue*. See ▸ Section 10.2.5.1 Loc.

10.2.7.2.10 RectangleState

New in JDF 1.2

This ***State*** Subelement is used to describe ranges of rectangle values. It inherits from the Abstract ***State*** Element described above.

Table 10.25: RectangleState Element

NAME	DATA TYPE	DESCRIPTION
AllowedHWRelation ?	XYRelation	Allowed relative value of width (X) vs. Height (Y).
AllowedValueList ?	RectangleRangeList	A list of ranges of allowed values that can be chosen.
CurrentValue ?	rectangle	Current value for the current running Job set in the Device.
DefaultValue ?	rectangle	Default value if not specified in a submitted **JDF**. @*DefaultValue* SHALL be specified if @*HasDefault* = "true".
PresentHWRelation ?	XYRelation	Allowed relative value of width (X) vs. Height (Y). If not specified, the value of @*AllowedHWRelation* is applied.
PresentValueList ?	RectangleRangeList	A list of ranges of values that can be chosen without operator intervention. If not specified, the value of @*AllowedValueList* is applied.
ValueLoc *	element	A list of supported values. The ***ValueLoc***/@*Value* Attribute SHALL be a representation of a rectangle. This can also be used to localize (or provide names for) specific rectangles. See ‣ Section 10.2.7.2.1.1 ValueLoc.

10.2.7.2.11 ShapeState

New in JDF 1.1

This ***State*** subelement is used to describe ranges of shape values. It inherits from the abstract ***State*** element described above.

Table 10.26: ShapeState Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AllowedValueList ? Modified in JDF 1.2	ShapeRangeList	A list of values that can be chosen.
AllowedValueMax ? Deprecated in JDF 1.2	shape	Inclusive maximum allowed value. Replaced by @*AllowedValueList* in **JDF** 1.2 and beyond.
AllowedValueMin ? Deprecated in JDF 1.2	shape	Inclusive minimum allowed value. Replaced by @*AllowedValueList* in **JDF** 1.2 and beyond.
AllowedX ? New in JDF 1.2	DoubleRangeList	Allowed X-axis of the @*Shape*.
AllowedY ? New in JDF 1.2	DoubleRangeList	Allowed Y-axis of the @*Shape*.
AllowedZ ? New in JDF 1.2	DoubleRangeList	Allowed Z-axis of the @*Shape*.
CurrentValue ?	shape	Current value for the current running Job set in the Device.
DefaultValue ?	shape	Default value if not specified in a submitted **JDF**. @*DefaultValue* SHALL be specified if @*HasDefault* = "true".
PresentValueList ? Modified in JDF 1.2	ShapeRangeList	A list of values that can be chosen without operator intervention. If not specified, the value of @*AllowedValueList* is applied.
PresentValueMax ? Deprecated in JDF 1.2	shape	Inclusive maximum allowed value that can be chosen without operator intervention. If not specified, the value of @*AllowedValueMax* is applied. Replaced by @*AllowedValueList* in **JDF** 1.2 and beyond.

Table 10.26: ShapeState Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
PresentValueMin ? Deprecated in JDF 1.2	shape	Inclusive minimum allowed value that can be chosen without operator intervention. If not specified, the value of @*AllowedValueMin* is applied. Replaced by @*AllowedValueList* in **JDF** 1.2 and beyond.
PresentX ? New in JDF 1.2	DoubleRangeList	Present X-axis of the @*Shape* that can be chosen without operator intervention. If not specified, the value of @*AllowedX* is applied.
PresentY ? New in JDF 1.2	DoubleRangeList	Present Y-axis of the @*Shape* that can be chosen without operator intervention. If not specified, the value of @*AllowedY* is applied.
PresentZ ? New in JDF 1.2	DoubleRangeList	Present Z-axis of the @*Shape* that can be chosen without operator intervention. If not specified, the value of @*AllowedZ* is applied.
ValueLoc * New in JDF 1.2	element	A list of supported shapes. See ▸ Section 10.2.7.2.1.1 ValueLoc.

10.2.7.2.12 StringState

New in JDF 1.1

This ***State*** Subelement is used to describe ranges of string values. It inherits from the Abstract ***State*** Element described above.

Table 10.27: StringState Element

NAME	DATA TYPE	DESCRIPTION
AllowedLength ? New in JDF 1.2	IntegerRange	Inclusive minimum and maximum length of valid string in multi-byte characters. Note that this is the length in characters, and not in bytes of the internal encoding of an application. For instance, the length of the string "Grün" is 4 and not 6 (UTF-8 with a terminating 0 and a double byte "ü").
AllowedRegExp ? New in JDF 1.2	regExp	Regular expression that limits the allowed values.
CurrentValue ?	string	Current value for the current running Job set in the Device.
DefaultValue ?	string	Default value if not specified in a submitted **JDF**. @*DefaultValue* SHALL be specified if @*HasDefault* = "true".
PresentLength ? New in JDF 1.2	IntegerRange	Inclusive minimum and maximum length of valid string in characters that can be chosen without operator intervention. If not specified, the value of @*AllowedLength* is applied.
PresentRegExp ? New in JDF 1.2	regExp	Regular expression that limits the present values that can be chosen without operator intervention. If not specified, the value of @*AllowedRegExp* is applied.
Value * Modified in JDF 1.2	element	A list legal values. See ▸ Section 10.2.7.2.12.1 Value.

10.2.7.2.12.1 Value

New in JDF 1.1

Table 10.28: StringState/Value Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AllowedValue	string	A legal value for a string variable.
PresentValue ? Deprecated in JDF 1.2	string	A legal value for a string variable that can be chosen without operator intervention. If not specified, the value of @*AllowedValue* is applied. In **JDF** 1.2 and beyond, use @*ValueUsage*.

Table 10.28: StringState/Value Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ValueUsage ? New in JDF 1.2	enumeration	Defines whether the value defined in @*AllowedValue* means "Present", "Allowed" or both. **Default behavior:** valid for both "Present" and "Allowed". **Allowed values are:** Present – Present configuration is supported. Allowed – Allowed configuration is supported.
Loc * New in JDF 1.2	element	The localization(s) of the string defined in @*AllowedValue*. See ▸ Section 10.2.5.1 Loc.

10.2.7.2.13 XYPairState

New in JDF 1.1

This ***State*** Subelement is used to describe ranges of XYPair values. It inherits from the Abstract ***State*** Element described above.

Table 10.29: XYPairState Element

NAME	DATA TYPE	DESCRIPTION
AllowedValueList ? Modified in JDF 1.2	XYPair-RangeList	A list of values that can be chosen.
AllowedValueMax ? Deprecated in JDF 1.2	XYPair	Inclusive maximum allowed value. Replaced with @*AllowedValueList* in **JDF** 1.2 and beyond.
AllowedValueMin ? Deprecated in JDF 1.2	XYPair	Inclusive minimum allowed value. Replaced with @*AllowedValueList* in **JDF** 1.2 and beyond.
AllowedXYRelation ? New in JDF 1.2	XYRelation	Relative value of X vs. Y.
CurrentValue ?	XYPair	Current value for the current running Job set in the Device.
DefaultValue ?	XYPair	Default value if not specified in a submitted **JDF**. @*DefaultValue* SHALL be specified if @*HasDefault* = "true".
PresentValueList ? Modified in JDF 1.2	XYPair-RangeList	A list of values that can be chosen without operator intervention. If not specified, the value of @*AllowedValueList* is applied.
PresentValueMax ? Deprecated in JDF 1.2	XYPair	Inclusive maximum allowed value that can be chosen without operator intervention. If not specified, the value of @*AllowedValueMax* is applied. Replaced with @*PresentValueList* in **JDF** 1.2 and beyond.
PresentValueMin ? Deprecated in JDF 1.2	XYPair	Inclusive minimum allowed value that can be chosen without operator intervention. If not specified, the value of @*AllowedValueMin* is applied. Replaced with @*PresentValueList* in **JDF** 1.2 and beyond.
PresentXYRelation ? New in JDF 1.2	XYRelation	Relative value of X vs. Y that can be chosen without operator intervention. If not specified, the value of @*AllowedXYRelation* is applied.
UnitType ? New in JDF 1.2	NMTOKEN	Specifies the unit type that this ***State*** Element represents. Used to enable an application to localize the representation of the units. @*UnitType* SHALL be specified if the ***IntegerState*** represents a value that has units. **Values include those from:** ***IntegerState***/@*UnitType* (▸ Table 10.18 IntegerState Element).
ValueLoc * New in JDF 1.2	element	A list of supported shapes. See ▸ Section 10.2.7.2.1.1 ValueLoc.

10.2.8 DisplayGroupPool

New in JDF 1.2

The ***DisplayGroupPool*** Element declares set(s) of related features that are intended to be displayed as a group in user interfaces. These declarations are references to individual features declared in ***State*** Elements.

Table 10.30: DisplayGroupPool Element

NAME	DATA TYPE	DESCRIPTION
DisplayGroup *	element	Declares a set of references to ***State*** Elements that are intended to be displayed as a group in user interfaces.

Example 10.1: DisplayGroupPool

In this example, a single ***DisplayGroup*** is specified. This ***DisplayGroup*** declares that the ***State*** Attributes with @*ID*'s "btd", "cmp", "mag", "colorspace" and "outputres" are all to be grouped together in any user interface. The English string "*ScanningParameters*" is associated with this ***DisplayGroup***, though no explicit assumptions are made about how to display this group of Attributes. The ***DisplayGroup*** Element merely states that there is a user-significant relationship between the Attributes.

```
<Device Class="Implementation" ID="Link0003" Status="Available">
  <DeviceCap>
    <DisplayGroupPool>
      <DisplayGroup rRefs="btd cmp mag colorspace outputres">
        <Loc HelpText="Parameters for scanning configuration" Lang="en"
             Value="ScanningParameters"/>
      </DisplayGroup>
    </DisplayGroupPool>
  </DeviceCap>
</Device>
```

10.2.8.1 DisplayGroup

Each ***DisplayGroup*** Element declares a group of features that are intended to be displayed together in user interfaces.

Table 10.31: DisplayGroup Element

NAME	DATA TYPE	DESCRIPTION
rRefs	IDREFS	References to ***State*** Elements. See ▸ Section 10.2.7 State for details of the ***State*** Element.
Loc *	element	Localized strings describing the ***DisplayGroup***. See ▸ Section 10.2.5.1 Loc.

10.2.9 FeaturePool

New in JDF 1.2

Modified in JDF 1.5

The ***FeaturePool*** Element describes Message or Resource Subelements that represent composite features for user manipulation when describing capabilities. These features typically do not directly represent any **JDF** Resources or parameters., but rather trigger macros that manipulate related sets of parameters. For more information on macro definitions, see ▸ Section 10.2.10 MacroPool.

These features can be mapped to ▸ NamedFeatures (see ▸ NamedFeature in ▸ Table 1.4 Glossary). A feature from ▸ NamedFeatures is selected by specifying a ***GeneralID***[@*DataType*="NamedFeature"] where ***GeneralID***/@*IDUsage* maps to @*Name* and ***GeneralID***/@*IDValue* is restricted by the ***State*** elements in the ***FeaturePool***.that matches entries from ***FeaturePool***/***State***/@*Name* and ***FeaturePool***/***State***/@*AllowedValueList*.

Table 10.32: FeaturePool Element

NAME	DATA TYPE	DESCRIPTION
State *	element	***State*** Elements that define the accepted parameter space for the Messages or Resources Subelements. These ***State***Subelements are identical in form to other ***State*** Elements, but typically are only "macro" features that control other features through ***macro*** Elements. For more information on ***macro*** definitions, see ▸ Section 10.2.10 MacroPool. For details of the ***State*** Element, see ▸ Section 10.2.7 State.

Example 10.2: FeaturePool

In this example, *ScanMode* is a feature that doesn't map directly to any **JDF** Resource or Attribute, but provides a "shell" feature that allows users to control a set of **JDF** Resources and/or Attributes to indicate a common or preferred grouping based on the user's desired task. The actual corresponding **JDF** Resource Attribute Values are determined and set by the *ScanModeMacro* ***macro*** that is called when the *ScanMode* feature is manipulated.

```
<Device Class="Implementation" ID="Link0003" Status="Available">
<DeviceCap>
<FeaturePool>
<EnumerationState
AllowedValueList="Mono ColorTransparency Photo" ID="sm"
HasDefault="false" MacroRefs="ScanModeMac" Name="ScanMode"
UserDisplay="Display"/>
</FeaturePool>
</DeviceCap>
</Device>
```

10.2.10 MacroPool

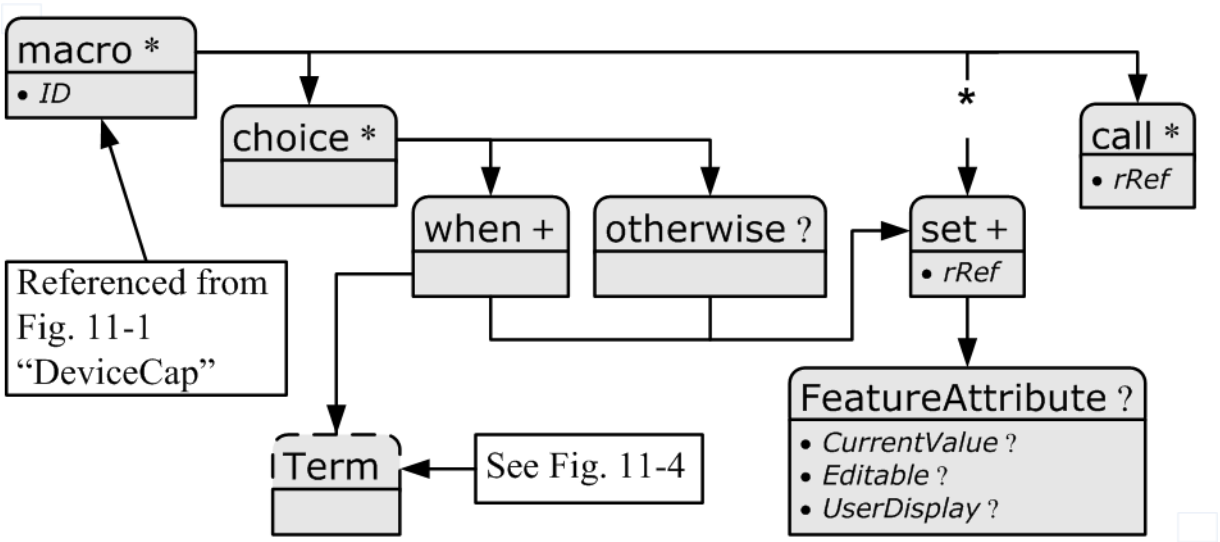
New in JDF 1.2

The ***MacroPool*** Element is used to contain descriptions of macro expressions. Each macro declares a set of conditional operations that are used to change ***State*** Element Attribute Values.

Table 10.33: MacroPool Element

NAME	DATA TYPE	DESCRIPTION
macro *	element	A list of independent macros.

Figure 10-4: macro Element – a diagram of its structure



10.2.10.1 macro

New in JDF 1.2

The ***macro*** Subelement is used to contain a set of conditional operations that are used to change ***State*** Element Attribute Values. Each ***macro*** contains one or more of the following Elements:

- ***choice*** — Declares one or more ***when*** statements, each of which contains a Boolean expression (as defined in ▸ Section 10.2.13 Term) and a ***set*** Element. When the expression evaluates to "true", the action specified in the ***set*** Element SHALL be performed. If no evaluation in any ***when*** Element in a ***choice*** evaluates to "true", the action(s) specified in the ***otherwise*** Element SHALL be performed.
- ***set*** — sets the condition of one or more ***State*** Element Attributes.
- ***call*** — calls another ***macro*** to be executed.

When executing a ***macro***, consumers SHALL execute ***choice***, ***set*** and ***call*** Elements in the order in which they are specified in the actual XML document. Note that the ordering provided in the actual capabilities description SHOULD be honored. The following shows the logical layout of the ***macro*** Subelement:

Table 10.34: macro Element

NAME	DATA TYPE	DESCRIPTION
ID	ID	Unique identifier of a ***macro*** Element. This @*ID* is used to refer to the ***macro*** Element.
choice *	element	A set of conditional operations that set (or not) feature values. At least one of ***choice***, ***set*** or ***call*** SHALL be specified in ***macro***.
set *	element	An Element that sets one or more ***State*** Attribute Values. At least one of ***choice***, ***set*** or ***call*** SHALL be specified in ***macro***.
call *	element	An Element that calls another ***macro***, allowing for ***macro*** reuse and chaining. At least one of ***choice***, ***set*** or ***call*** SHALL be specified in ***macro***.

10.2.10.2 choice

The ***choice*** Subelement is used to contain expressions that declare conditional operations that can cause ***State*** Element Attribute Values to be changed. The ***choice*** includes one or more ***when*** statements that are evaluated in order, each of which contains a Boolean expression (as defined in ▸ Section 10.2.13 Term) and a ***set*** Element. When the expression evaluates to "true", the action specified in the ***set*** Element SHALL be performed and no further ***when*** statements are evaluated. If no evaluation in any ***when*** Element in a ***choice*** evaluates to "true", the action(s) specified in the ***otherwise*** Element SHALL be performed.

Table 10.35: choice Element

NAME	DATA TYPE	DESCRIPTION
when +	element	A set of conditional operations that set (or not) feature values.
otherwise ?	element	An Element that sets one or more ***State*** Element Attribute Values if none of the ***when*** expressions evaluate to "true".

10.2.10.3 otherwise

The ***otherwise*** Subelement sets one or more feature values if none of the ***when*** expressions in a ***choice*** Element evaluate to "true".

Table 10.36: otherwise Element

NAME	DATA TYPE	DESCRIPTION
set +	element	An element that sets one or more feature values.

10.2.10.4 when

The ***when*** Subelement is used to contain expressions that declare conditional operations to enforce sets of feature behaviors. The ***when*** Element includes a Boolean expression (as defined in ▸ Section 10.2.13 Term) and a ***set*** Element. When the ***Term*** evaluates to "true", the action specified in the ***set*** Element SHALL be performed.

Table 10.37: when Element

NAME	DATA TYPE	DESCRIPTION
Term	element	A Boolean expression that evaluates a set of feature values.
set +	element	An Element that sets one or more feature values.

10.2.10.5 set

The ***set*** Subelement sets one or more ***State*** Element Attribute Values.

Table 10.38: set Element

NAME	DATA TYPE	DESCRIPTION
rRef	IDREF	Reference to a ***State*** Element referring to the feature value to set
FeatureAttribute ?	element	Specifies one or more Attributes within the ***State*** Element that are to have their value changed (along with the value they change to).

10.2.10.6 FeatureAttribute

FeatureAttribute specifies one or more Attributes of a ***State*** Element that are to have their value changed. The following Attributes can be changed:

Table 10.39: FeatureAttribute Element

NAME	DATA TYPE	DESCRIPTION
CurrentValue ?	string	The value to change the @*CurrentValue* Attribute of the ***State*** Element to. Note that the mapping of the string to the actual data type of the ***State*** Element SHALL be performed by the application processing the capabilities.
Editable ?	boolean	When "true", the feature and its current value can be edited by the user. If "false", the user interface SHALL NOT allow user modification of the current value of the ***State*** Element.
UserDisplay ?	enumeration	Indicates under which conditions the feature SHALL be displayed in user interfaces. **Allowed values are from:** ***State***/@*UserDisplay*.

10.2.10.7 call

The ***call*** Subelement is used to call other ***macro*** Elements, effectively using them as macro "templates".

Table 10.40: call Element

NAME	DATA TYPE	DESCRIPTION
rRef	IDREF	Reference to a ***macro***.

10.2.11 Performance

New in JDF 1.1

The ***Performance*** Element describes speed as the capability to consume or produce a **JDF** Resource.

Table 10.41: Performance Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AverageAmount ?	double	Average amount produced/consumed per hour assuming an average Job.
AverageCleanup ?	duration	Average time needed to clean the Device after a Job.
AverageSetup ?	duration	Average time needed to setup the Device before a Job.

Table 10.41: Performance Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
DevCapsRef ? New in JDF 1.2	IDREF	Reference to the ***DevCaps*** Element that describes the Resource whose performance is specified by this ***Performance*** Element.
MaxAmount ?	double	Maximum amount produced/consumed per hour, assuming an ideal Job. The default value of "0" translates to the value of @*AverageAmount*.
MaxCleanup ?	duration	Maximum time needed to clean the Device after a Job, assuming a worst case Job. Defaults to @*AverageCleanup*.
MaxSetup ?	duration	Maximum time needed to setup the Device before a Job, assuming a worst case Job. Defaults to @*AverageSetup*.
MinAmount ?	double	Minimum amount produced/consumed per hour, assuming a worst case Job. Defaults to @*AverageAmount*.
MinCleanup ?	duration	Minimum time needed to clean the Device after a Job, assuming an ideal Job. Defaults to @*AverageCleanup*.
MinSetup ?	duration	Minimum time needed to setup the Device before a Job, assuming an ideal Job. Defaults to @*AverageSetup*.
Name ? Deprecated in JDF 1.2	NMTOKEN	Name of the Input Resource type that is processed by the Device (e.g., ***Media***, ***Ink***, ***RunList***). **Deprecation note:** Starting with **JDF** 1.2, use @*DevCapsRef*.
Unit ?	NMTOKEN	Unit of measure of Resource consumption per hour. **Default value is from:** Resource's generic units as defined in ‣ Table 1.7 Units Used in JDF.

10.2.12 TestPool

New in JDF 1.2

The ***TestPool*** Subelement is used to contain Boolean expressions that are used to describe "templates" for use in ***Action*** Elements.

Table 10.42: TestPool Element

NAME	DATA TYPE	DESCRIPTION
Test *	element	A list of independent ***Test*** Elements.

10.2.12.1 Test

The ***Test*** Subelement is used to contain Boolean expressions that are for use only when referenced by another ***Test*** or ***Action*** and are not evaluated independently. Its purpose is to simplify the description of other ***Test*** Elements and ***macro*** Elements by representing a commonly used Boolean expression.

Table 10.43: Test Element

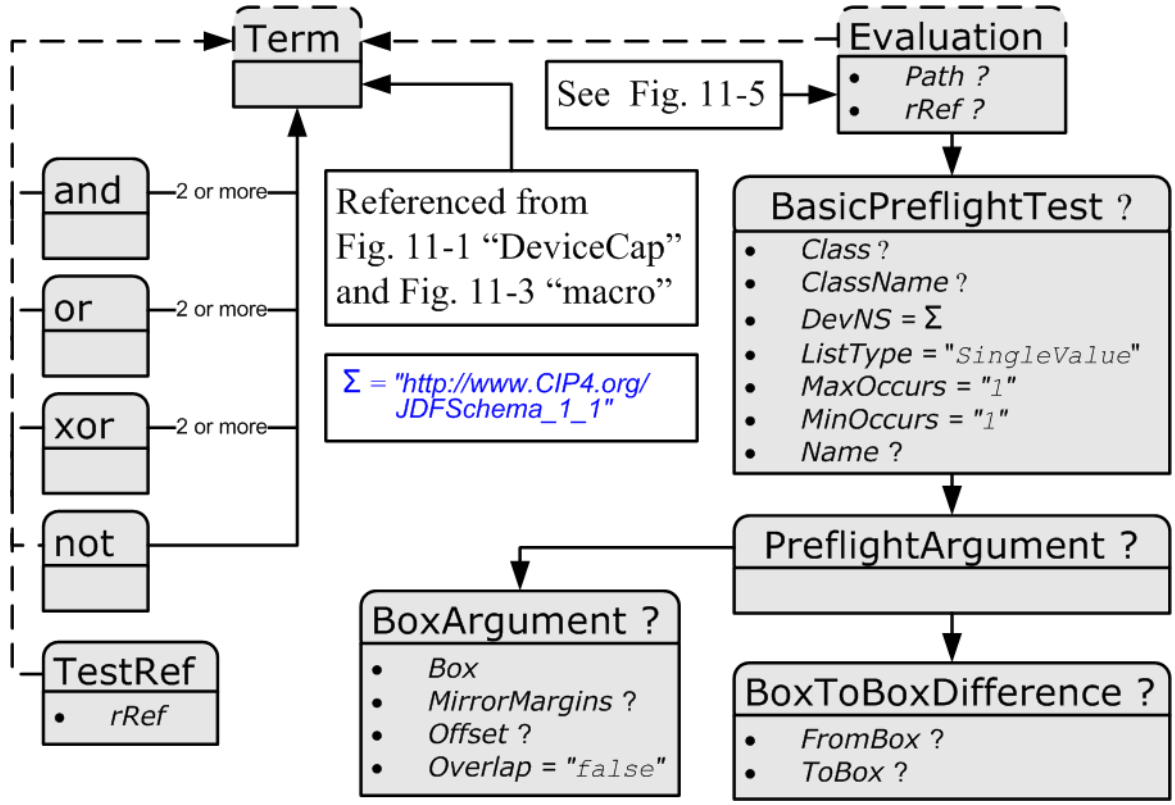
NAME	DATA TYPE	DESCRIPTION
ID	ID	Unique identifier of a ***Test*** Element. This ID is used to refer to the ***Test*** Element.
Term	element	Any Element derived from an Abstract ***Term*** (e.g., "***not***", "***and***" or one of the explicit ***Evaluation*** Elements).

10.2.13 Term

10.2.13.1 Abstract Term

‣ Figure 10-5: Abstract Term Element – a diagram of its structure shows the ***Abstract Term*** and all ***Term*** Elements

Figure 10-5: Abstract Term Element – a diagram of its structure



10.2.13.2 Term Elements

The ***Term*** Element serves as the basis for all constraint expressions and conditional macro expressions. It describes a (potentially) nested Boolean expression that evaluates as a whole to either "true" or "false". This expression is then used inside constraint or ***macro*** Elements to determine proper action given the evaluation of the ***Term***. ***Term*** Elements are composed of Boolean combinations of Elements in ▸ Table 10.44 List of Term Elements. The ***Term*** Elements that are Boolean operators MAY be nested. They are used both in Device capabilities and preflighting context.

Note: In the actual **JDF** schema, several abstract element definitions are used to create an appropriate inheritance structure. Rather than reproduce this here, only the actual non-Abstract Elements that will appear in **JDF** files will be described

Table 10.44: List of Term Elements

NAME	PAGE	DESCRIPTION
and	page 711	Boolean AND operator.
not	page 712	Boolean negation.
or	page 711	Boolean OR operator.
xor	page 711	Boolean exclusive or (XOR) operator.
TestRef	page 712	Reference to a constraint ***Test*** Element to be evaluated as a nested Boolean expression inside a larger expression.
Evaluation	page 712	Elements, which evaluate a **JDF** ***State*** Attribute Value to create a simple Boolean expression (e.g., "Is the value of @*BitDepth* equal to 8?"). Each ***XXXExpression*** Element is derived from the ***Abstract Evaluation*** Element.

Example 10.3: ActionPool and TestPool

Term is an Abstract Element, so it will never appear in a **JDF** document. In this "ctcmp" constraint example, the ***Term*** is represented by the ***and*** Element. Since the ***Term*** Element itself is Abstract, what will actually appear in constraints will be Boolean expressions. In this example, the logic is, "We can not use CCITT compression if the bit depth is not 1 bit." The check for compression type uses an ***EnumerationEvaluation*** Element, which evaluates an ***EnumerationState*** value against "CCITTFaxEncode". If the value of the ***EnumerationState*** Element referred to by "cmp" = "CCITTFaxEncode", the

EnumerationEvaluation evaluates as "true". The check for "btd" is accomplished through a *@TestRef* to the "is1bit" constraint. The *and* and *not* Elements behave according to the standard semantics for Boolean combinatorial logic.

```
<Device Class="Implementation" ID="Link0003" Status="Available">
<DeviceCap>
<ActionPool>
<Action ID="MyAction" TestRef="ctcmp">
<Loc HelpText="Only select CCITTFaxEncoding for 1 bit documents"
Lang="en" ShortValue="Ouch!"
Value="CCITTFaxEncoding not supported on
grayscale images"/>
</Action>
</ActionPool>
<TestPool>
<Test ID="ctcmp">
<!-- Can't CCITT compress anything but 1 bit grayscale -->
<and>
<not>
<TestRef rRef="is1bit"/>
</not>
<EnumerationEvaluation ValueList="CCITTFaxEncode" rRef="cmp"/>
</and>
</Test>
<Test ID="is1bit">
<IntegerEvaluation ValueList="1" rRef="btd"/>
</Test>
</TestPool>
</DeviceCap>
</Device>
```

10.2.13.3 and

The *and* Element evaluates two or more *Term* Elements to determine if, as a set, they evaluate to "true" when combined in a Boolean "and" function.

Table 10.45: and Element

NAME	DATA TYPE	DESCRIPTION
Term	element	Any Element derived from an Abstract *Term*.
Term +	element	Any Element derived from an Abstract *Term*.

10.2.13.4 or

The *or* Element evaluates two or more *Term* Elements to determine if, as a set, they evaluate to "true" when combined in a Boolean "or" function.

Table 10.46: or Element

NAME	DATA TYPE	DESCRIPTION
Term	element	Any Element derived from an Abstract *Term*.
Term +	element	Any Element derived from an Abstract *Term*.

10.2.13.5 xor

The *xor* Element evaluates two or more *Term* Elements to determine if, as a set, they evaluate to "true" when combined in a Boolean "xor" function. For more than two arguments, exactly one *Term* SHALL evaluate to "true" for the *xor* to evaluate to "true". Note that this is different from the mathematical behavior of "xor".

Table 10.47: xor Element

NAME	DATA TYPE	DESCRIPTION
Term	element	Any Element derived from an Abstract *Term*.
Term +	element	Any Element derived from an Abstract *Term*.

10.2.13.6 not

The ***not*** Element inverts the Boolean state of a ***Term***.

Table 10.48: not Element

NAME	DATA TYPE	DESCRIPTION
Term	element	Any Element derived from an Abstract ***Term***.

10.2.13.7 TestRef

The ***TestRef*** Element refers to another constraint that SHALL be evaluated as part of the parent constraint.

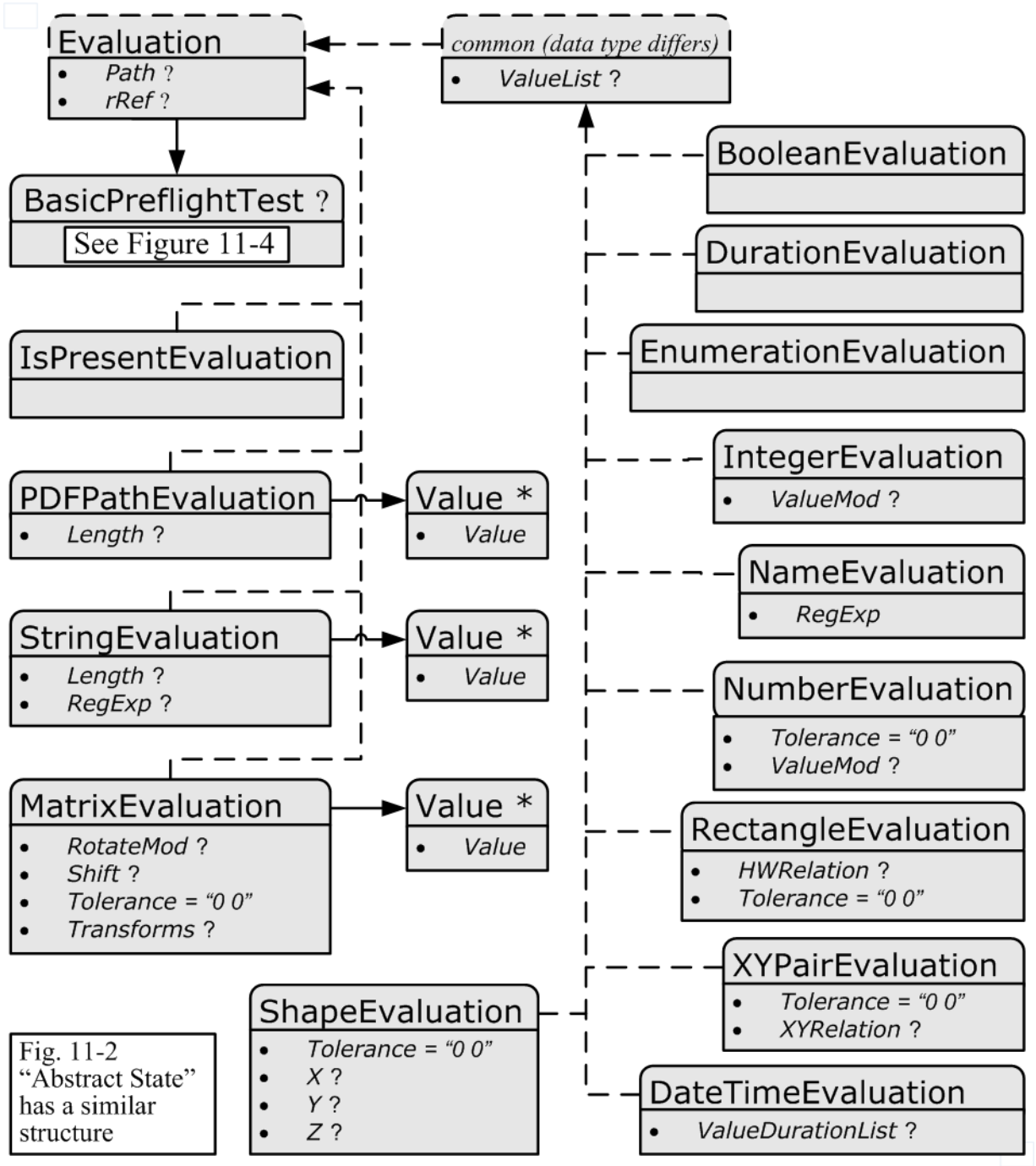
*Table 10.49: **TestRef** Element*

NAME	DATA TYPE	DESCRIPTION
rRef	IDREF	Reference to a ***Test*** to be evaluated as a nested Boolean expression inside a larger expression.

10.2.13.8 Evaluation

Figure 10-6: Abstract Evaluation Element – a diagram of its structure shows all ***Evaluation*** Elements.

Figure 10-6: Abstract Evaluation Element – a diagram of its structure



10.2.13.8.1 Abstract Evaluation

The following table describes the common, data type-independent parameters of all ***Evaluation*** Elements.

Table 10.50: Abstract Evaluation Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Path ? New in JDF 1.4	XPath	When present, describes an XPath within the file where the value to be evaluated may be found. **Constraint:** Exactly one of *@Path*, *@rRef* or ***BasicPreflightTest*** SHALL be specified.

Table 10.50: Abstract Evaluation Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
rRef ? Modified in JDF 1.4	IDREF	A reference to ***State***, ***DevCap***, ***DevCaps*** or ***Module*** Elements when used in the context of Device capability descriptions. **Constraint:** Exactly one of @*Path*, @*rRef* or ***BasicPreflightTest*** SHALL be specified. **Modification note:** Starting with **JDF** 1.4, ***DevCap*** and ***DevCaps*** can also be referenced and @*Path* added to the all constraints in this table.
BasicPreflightTest ?	element	Definition of the preflight basic test to which the ***Evaluation*** refers. ***BasicPreflightTest*** is only valid when ***Evaluation*** Elements are used in the context of preflighting. The ***Evaluation*** Elements in capability descriptions SHALL reference the appropriate ***State*** Element using @*rRef*. For details of the ***BasicPreflightTest***, see ▸ Section 8.112 PreflightParams. **Constraint:** Exactly one of @*Path*, @*rRef* or ***BasicPreflightTest*** SHALL be specified.

10.2.13.8.2 Evaluation Elements

Evaluation Elements map generalized tests against a condition to form a true or false Boolean state that can be evaluated using the Boolean logic defined below.

Table 10.51: List of Evaluation Elements

NAME	PAGE	DESCRIPTION
BooleanEvaluation	page 715	Describes operations on a set of Boolean values.
DateTimeEvaluation	page 715	Describes operations on a set of dateTime values.
DurationEvaluation	page 716	Describes operations on a set of duration values.
EnumerationEvaluation	page 716	Describes operations on a set of enumeration values.
IntegerEvaluation	page 716	Describes operations on a numerical range of integer values.
IsPresentEvaluation	page 716	Checks for the existence of a tag, Element or feature.
MatrixEvaluation	page 717	Describes operations on a range of matrices. Generally used to define valid orientations of ***Component*** Resources.
NameEvaluation	page 717	Describes operations on a set of NMTOKEN values
NumberEvaluation	page 718	Describes operations on a numerical range of values.
PDFPathEvaluation	page 718	Describes operations on PDFPath.
RectangleEvaluation	page 718	Describes operations on a set of four-value rectangle values.
ShapeEvaluation	page 719	Describes operations on a set of three-value shape values.
StringEvaluation	page 719	Describes operations on a set of string values.
XYPairEvaluation	page 719	Describes operations on a set of XYPair values.

Mapping of Evaluation Element to State Element

When used in a Device capabilities context, the ***Evaluation*** Elements map to the ***State*** Elements (i.e., ***BooleanState***, ***IntegerState***, etc.). These Elements each declare individual **JDF** Attributes for a Device capabilities description. The ***Evaluation*** Elements are instances of ***Term*** Elements that compare the value of a given ***State*** Attribute against a condition to form a true or false Boolean statement. The form of the condition depends on the type of the ***Evaluation***–***State*** Element pairing — different types of pairings need different condition declarations, depending on the structure of the logic and the data type of the ***Evaluation*** and ***State*** Elements.

When used in a preflighting context, ***Evaluation*** Elements map named preflight tests against a condition to form a true or false Boolean statement.

Table 10.52: Mapping of Evaluation Element to State Element

NAME	CORRESPONDING STATE ELEMENT	DESCRIPTION
BooleanEvaluation	***BooleanState***	Describes operations on a set of Boolean values.
DateTimeEvaluation	***DateTimeState***	Describes operations on a set of dateTime values.
DurationEvaluation	***DurationState***	Describes operations on a set of duration values.
EnumerationEvaluation	***EnumerationState***	Describes operations on a set of enumeration values.
IntegerEvaluation	***IntegerState***	Describes operations on a numerical range of integer values.
IsPresentEvaluation	***State*** (all, IsPresent-State (ElementState))	Checks for the existence of a tag, Element or feature.
MatrixEvaluation	***MatrixState***	Describes operations on a range of matrices. Generally used to define valid orientations of ***Component*** Resources.
NameEvaluation	***NameState***	Describes operations on a set of NMTOKEN values
NumberEvaluation	***NumberState***	Describes operations on a numerical range of values.
PDFPathEvaluation	***PDFPathState***	Describes operations on PDFPath.
RectangleEvaluation	***RectangleState***	Describes operations on a set of four-value rectangle values.
ShapeEvaluation	***ShapeState***	Describes operations on a set of three-value shape values.
StringEvaluation	***StringState***	Describes operations on a set of string values.
XYPairEvaluation	***XYPairState***	Describes operations on a set of XYPair values.

10.2.13.8.2.1 BooleanEvaluation

The ***BooleanEvaluation*** Element declares a Boolean value for comparison in an expression to a ***BooleanState*** Element in constraints. It inherits from the ***Abstract Evaluation*** Element described above.

Table 10.53: BooleanEvaluation Element

NAME	DATA TYPE	DESCRIPTION
ValueList ?	enumerations	A list of all supported values. **Allowed values are:** true false

10.2.13.8.2.2 DateTimeEvaluation

The ***DateTimeEvaluation*** Element declares a Boolean value for comparison in an expression to a ***DateTimeState*** Element in constraints. It inherits from the ***Abstract Evaluation*** Element described above.

Table 10.54: DateTimeEvaluation Element

NAME	DATA TYPE	DESCRIPTION
ValueDurationList ?	DurationRangeList	List of inclusive minimum and maximum allowed values relative to the current system time.
ValueList ?	DateTimeRangeList	A list of all supported values.

10.2.13.8.2.3 DurationEvaluation

The ***DurationEvaluation*** Element declares a Boolean value for comparison in an expression to a ***DurationState*** Element in constraints. It inherits from the ***Abstract Evaluation*** Element described above.

Table 10.55: DurationEvaluation Element

NAME	DATA TYPE	DESCRIPTION
ValueList ?	DurationRangeList	A list of all supported values.

10.2.13.8.2.4 EnumerationEvaluation

The ***EnumerationEvaluation*** Element declares an enumeration value for comparison in an expression to an ***EnumerationState*** Element in constraints. It inherits from the ***Abstract Evaluation*** Element described above.

Table 10.56: EnumerationEvaluation Element

NAME	DATA TYPE	DESCRIPTION
ValueList ?	enumerations	A list of all potential supported values. If not specified all enumerations defined by the XML schema are valid. In order to enable capabilities to be specified without access to the **JDF** XML schema, it is strongly RECOMMENDED to specify @*ValueList*, even when the entire range of schema-valid values is supported.

10.2.13.8.2.5 IntegerEvaluation

The ***IntegerEvaluation*** Element declares an integer value for comparison in an expression to a ***IntegerState*** Element in constraints.

Table 10.57: IntegerEvaluation Element

NAME	DATA TYPE	DESCRIPTION
ValueList ?	IntegerRangeList	A list of all supported values.
ValueMod ?	XYPair	X defines the Modulo and Y the offset of the allowed value. In other words, if @*AllowedValueMod* = "10 2", only the values ... -8,2,12,22 ... are allowed. If not specified all values in the range are valid. If ((N%X)−Y = 0) then N is a valid value. **Note:** "Modulo" is the remainder of an integer division. For example: 4 mod 3 = 4 -3 = 1; 17 mod 3 = 17-5*3 = 2; and 3 mod 3 = 3-3 = 0.

10.2.13.8.2.6 IsPresentEvaluation

The ***IsPresentEvaluation*** Element checks for the existence of a tag, module or feature. It inherits from the ***Abstract Evaluation*** Element described above and has no additional Attributes. ***IsPresentEvaluation***/@*rRef* MAY reference a ***DevCap*** Element in order to test for the existence of an Element.

IsPresentEvaluation/@*rRef* MAY reference a ***DevCaps*** Element in order to test for the existence of a Resource.

Table 10.58: IsPresentEvaluation Element

NAME	DATA TYPE	DESCRIPTION

10.2.13.8.2.7 MatrixEvaluation

The ***MatrixEvaluation*** Element declares a matrix value for comparison in an expression to a ***MatrixState*** Element in constraints. It inherits from the ***Abstract Evaluation*** Element described above.

Table 10.59: MatrixEvaluation Element

NAME	DATA TYPE	DESCRIPTION
RotateMod ?	double	Allowed Modulo of the allowed rotations and offset in degrees. **Note:** Although this seems counter-intuitive and contrary to the convention set in **JDF** coordinate systems, the application of @*RotateMod* in practice will involve subtracting values by the value of the @*RotateMod*. Hence, any number is reduced by "0" and is unaffected by the subtraction. **Values include:** 360 – No rotation is allowed. 90 – Any orthogonal rotation. 0 – Interpreted to mean that any rotation is allowed.
Shift ?	DoubleList	If @*Transforms* is specified, the implied shift defined in ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component is subtracted from @*Shift*, thus all in-place rotations have an implied Shift value of "0 0 0 0".
Tolerance = "0 0"	XYPair	The tolerance between the real and actual values that are defined as equal. Used to account for rounding errors and such. The first value is a positive value representing the negative tolerance, and the second value represents the positive tolerance. The tolerance applies to all of the matrix values.
Transforms ?	Orientations	Any of the eight predefined transforms for ***PhysicalResource***s as defined in ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component.
Value *	element	A list supported values. The ***Value***/@*Value* Attribute SHALL be a representation of a matrix. See ▸ Section 10.2.13.8.2.7.1 Value.

10.2.13.8.2.7.1 Value

Table 10.60: MatrixEvaluation/Value Element

NAME	DATA TYPE	DESCRIPTION
Value	matrix	A supported value for a matrix variable.

10.2.13.8.2.8 NameEvaluation

The ***NameEvaluation*** Element declares a NMTOKEN value for comparison in an expression to a ***NameState*** Element in constraints. It inherits from the ***Abstract Evaluation*** Element described above.

Table 10.61: NameEvaluation Element

NAME	DATA TYPE	DESCRIPTION
RegExp	regExp	Regular expression that limits the allowed values.
ValueList ?	NMTOKENS	A list of supported values.

10.2.13.8.2.9 NumberEvaluation

The ***NumberEvaluation*** Element declares a number value for comparison in an expression to a ***NumberState*** Element in constraints. It inherits from the ***Abstract Evaluation*** Element described above.

Table 10.62: NumberEvaluation Element

NAME	DATA TYPE	DESCRIPTION
Tolerance = "0 0"	XYPair	The tolerance between the real and actual values that are defined as equal. Used to account for rounding errors and such. The first value is a positive value representing the negative tolerance, and the second represents the positive tolerance.
ValueList ?	DoubleRange List	A list of supported values.
ValueMod ?	XYPair	X defines the Modulo and Y the offset of the allowed value. In other words, if @*AllowedValueMod* = "10 2", only the values ... -8,2,12,22 ... are allowed. If not specified all values in the range are valid. If ((N%X)−Y = 0) then N is a valid value. **Note:** "Modulo" is the remainder of an integer division. For example: 4 mod 3 = 4 -3 = 1; 17 mod 3 = 17-5*3 = 2; and 3 mod 3 = 3-3 = 0.

10.2.13.8.2.10 PDFPathEvaluation

The ***PDFPathEvaluation*** Element declares a PDF path value for comparison in an expression to a ***PDFPathState*** Element in constraints. It inherits from the ***Abstract Evaluation*** Element described above.

Table 10.63: PDFPathEvaluation Element

NAME	DATA TYPE	DESCRIPTION
Length ?	IntegerRange	Inclusive minimum and maximum length of valid PDF path in characters.
Value *	element	PDF path values for comparison in an expression to a ***PDFPathState*** Element. See ▸ Section 10.2.13.8.2.10.1 Value.

10.2.13.8.2.10.1 Value

Table 10.64: PDFPathEvaluation/Value Element

NAME	DATA TYPE	DESCRIPTION
Value	PDFPath	A supported value for a PDF path Attribute.

10.2.13.8.2.11 RectangleEvaluation

The ***RectangleEvaluation*** Element declares a Boolean value for comparison in an expression to a ***RectangleState*** Element in constraints. It inherits from the ***Abstract Evaluation*** Element described above.

Table 10.65: RectangleEvaluation Element

NAME	DATA TYPE	DESCRIPTION
HWRelation ?	XYRelation	Allowed relative value of width (X) versus height (Y).
Tolerance = "0 0"	XYPair	The tolerance between the real and actual values that are defined as equal. Used to account for rounding errors and such. The first value is a positive value representing the negative tolerance, and the second represents the positive tolerance. The tolerance applies to both sides of the rectangle.
ValueList ?	Rectan-gleRangeList	A list of ranges of allowed values that can be chosen.

10.2.13.8.2.12 ShapeEvaluation

The ***ShapeEvaluation*** Element declares a shape value for comparison in an expression to a ***ShapeState*** Element in constraints. It inherits from the ***Abstract Evaluation*** Element described above.

Table 10.66: ShapeEvaluation Element

NAME	DATA TYPE	DESCRIPTION
Tolerance = "0 0"	XYPair	The tolerance between the real and actual values that are defined as equal. Used to account for rounding errors and such. The first value is a positive value representing the negative tolerance, and the second represents the positive tolerance. The tolerance applies to all values tested.
ValueList ?	ShapeRange-List	A list of ranges of values that can be chosen.
X ?	Dou-bleRangeList	Allowed X-axis of the ***Shape***.
Y ?	Dou-bleRangeList	Allowed Y-axis of the ***Shape***.
Z ?	Dou-bleRangeList	Allowed Z-axis of the ***Shape***.

10.2.13.8.2.13 StringEvaluation

The ***StringEvaluation*** Element declares a string value for comparison in an expression to a ***StringState*** Element in constraints. It inherits from the ***Abstract Evaluation*** Element described above.

Table 10.67: StringEvaluation Element

NAME	DATA TYPE	DESCRIPTION
Length ?	IntegerRange	Inclusive minimum and maximum length of valid string in characters. Note that this is the length in characters, and not in bytes of the internal encoding of an application. For instance, the length of the string "Grün" is 4 and not 6 (UTF-8 with a terminating 0 and a double byte "ü").
RegExp ?	regExp	Regular expression that limits the allowed values.
Value *	element	A string value for comparison in an expression to a ***StringEvaluation*** Element. See ▸ Section 10.2.13.8.2.13.1 Value.

10.2.13.8.2.13.1 Value

Table 10.68: StringEvaluation/Value Element

NAME	DATA TYPE	DESCRIPTION
Value	string	A supported value for a string Attribute.

10.2.13.8.2.14 XYPairEvaluation

The ***XYPairEvaluation*** Element declares a XYPair value for comparison in an expression to a ***XYPairState*** Element in constraints. It inherits from the ***Abstract Evaluation*** Element described above.

Table 10.69: XYPairEvaluation Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Tolerance = "0 0"	XYPair	The tolerance between the real and actual values that are defined as equal. Used to account for rounding errors and such. The first value is a positive value representing the negative tolerance, and the second represents the positive tolerance. These tolerance values apply to both the X and Y values of the evaluation being performed.

Table 10.69: XYPairEvaluation Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
ValueList ?	XYPair-RangeList	A list of values that can be chosen.
XYRelation ?	XYRelation	Relative value of X vs. Y.

10.2.14 Examples of Device Capabilities

New in JDF 1.1

All of the examples in this section are based on a simple definition of a scanner. The **JMF** based hand shaking is also illustrated. ***NodeInfo***, ***ExposedMedia*** and ***ScanParams*** are restricted.

10.2.14.1 Device Description of a Scanner

This first example shows the general structure and provides an example of user interface localization (the query requests localization for the French language, and localizations are returned for the ***ScanParams*** Resource).

Example 10.4: KnownDevices Query for a Scanner

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" MaxVersion="1.4" Version="1.4"
TimeStamp="2005-04-05T16:45:43+02:00" SenderID="Controller"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<Query ID="DeviceQuery" Type="KnownDevices" xsi:type="QueryKnownDevices">
<DeviceFilter DeviceDetails="Capability" Localization="fre"/>
</Query>
</JMF>
```

Example 10.5: KnownDevices Response for a Scanner

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Scanner"
TimeStamp="2005-06-05T16:45:43+02:00" MaxVersion="1.4" Version="1.4"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<Response ID="xyz" Type="KnownDevices" refID="DeviceQuery"
xsi:type="ResponseKnownDevices" >
<DeviceList>
<DeviceInfo DeviceStatus="Idle">
<Device Class="Implementation" DeviceID="Joe the Drum"
KnownLocalizations="En Fre" ModelName="Bongo" >
<DeviceCap GenericAttributes="ID Class SettingsPolicy
BestEffortExceptions OperatorInterventionExceptions
MustHonorExceptions PartIDKeys DocIndex"
Lang="Fre" Type="Scanning">
<!--   the scanner takes a minute to set up and scans an average
of 2 sheets a min. -->
<Performance AverageAmount="120" AverageSetup="PT2M"
Name="ExposedMedia"/>
<DevCaps Name="NodeInfo">
<DevCap>
<!--NodeInfo only supports JobPriority and
TargetRoute Attributes -->
<StringState Name="TargetRoute" HasDefault="false"/>
<IntegerState Name="JobPriority" HasDefault="false"/>
</DevCap>
</DevCaps>
<DevCaps Name="ExposedMedia">
<DevCap>
<!-- ExposedMedia restrictions -->
<DevCap Name="Media">
<NameState DefaultValue="Sheet" Name="MediaUnit"/>
<XYPairState AllowedValueMax="600 1200"
AllowedValueMin="0 0"
Name="Dimension" HasDefault="false"/>
</DevCap>
</DevCap>
</DevCaps>
<DevCaps Name="ScanParams">
<Loc HelpText="Les parametres pour commander le
procede de balayage."
```

```
Value="Les parametres de module de balayage"/>
<DevCap>
<!-- Black and white 1 bit mode -->
<IntegerState AllowedValueMax="1" AllowedValueMin="1"
DefaultValue="8" Name="BitDepth"/>
<EnumerationState AllowedValueList="CCITTFaxEncode None"
Name="CompressionFilter" HasDefault="false">
<Loc HelpText="Choisissez la compression pour reduire la
taille de donnees."
Value="La compression de donnees"/>
<ValueLoc Value="CCITTFaxEncode">
<Loc Value="Compression de CCITT Fax"/>
</ValueLoc>
<ValueLoc Value="None">
<Loc Value="Aucun compression"/>
</ValueLoc>
</EnumerationState>
<NumberState AllowedValueMax="10" AllowedValueMin="1.e-002"
Name="Magnification" HasDefault="false">
<Loc ShortValue="Rapport optique"
Value="Rapport de rapport optique d'image"/>
</NumberState>
<EnumerationState AllowedValueList="GrayScale"
Name="OutputColorSpace" HasDefault="false">
<Loc ShortValue="Format de couleur"
Value="Configurez le format de couleur de
module de balayage"/>
<ValueLoc Value="GrayScale">
<Loc Value="echelle de gris"/>
</ValueLoc>
</EnumerationState>
<XYPairState DefaultValue="2400 2400"
Name="OutputResolution">
<Loc ShortValue="resolution"
Value="Resolution de module de balayage"/>
</XYPairState>
</DevCap>
<DevCap>
<!-- Grayscale 12 bit mode -->
<IntegerState AllowedValueMax="12" AllowedValueMin="12"
DefaultValue="8" Name="BitDepth">
<Loc Value="Le profondeur de bit"/>
</IntegerState>
<EnumerationState
AllowedValueList="FlateEncode DCTEncode None"
Name="CompressionFilter" HasDefault="false">
<Loc HelpText="Choisissez la compression pour
reduire la taille de donnees."
Value="La compression de donnees"/>
<ValueLoc Value="FlateEncode">
<Loc Value="Compression de Flate"/>
</ValueLoc>
<ValueLoc Value="DCTEEncode">
<Loc Value="Compression de DCTE"/>
</ValueLoc>
<ValueLoc Value="None">
<Loc Value="Aucun compression"/>
</ValueLoc>
</EnumerationState>
<NumberState AllowedValueMax="10" AllowedValueMin="0.001"
Name="Magnification" DefaultValue="1.0">
<Loc ShortValue="Rapport optique"
Value="Rapport de rapport optique d'image"/>
</NumberState>
<EnumerationState AllowedValueList="GrayScale"
Name="OutputColorSpace" HasDefault="false">
<Loc ShortValue="Format de couleur"
Value="Configurez le format de couleur de
module de balayage"/>
<ValueLoc Value="GrayScale">
```

```
<Loc Value="Echelle de gris"/>
</ValueLoc>
</EnumerationState>
<XYPairState AllowedValueMax="2400 2400"
AllowedValueMin="100 100" DefaultValue="600 600"
Name="OutputResolution">
<Loc ShortValue="resolution"
Value="Resolution de module de balayage"/>
</XYPairState>
</DevCap>
<DevCap>
<!-- Color 10 bit mode -->
<IntegerState AllowedValueMax="10" AllowedValueMin="10"
DefaultValue="8" Name="BitDepth">
<Loc Value="Le profondeur de bit"/>
</IntegerState>
<EnumerationState
AllowedValueList="FlateEncode DCTEncode None"
Name="CompressionFilter">
<Loc HelpText="Choisissez la compression pour reduire
la taille de donnees."
Value="La compression de donnees"/>
<ValueLoc Value="FlateEncode">
<Loc Value="Compression de Flate"/>
</ValueLoc>
<ValueLoc Value="DCTEEncode">
<Loc Value="Compression de DCTE"/>
</ValueLoc>
<ValueLoc Value="None">
<Loc Value="Aucun compression"/>
</ValueLoc>
</EnumerationState>
<NumberState AllowedValueMax="10" AllowedValueMin="1.e-002"
Name="Magnification">
<Loc ShortValue="Rapport optique"
Value="Rapport de rapport optique d'image"/>
</NumberState>
<EnumerationState AllowedValueList="CMYK RGB LAB"
Name="OutputColorSpace">
<Loc ShortValue="Format de couleur"
Value="Configurez le format de couleur de
module de balayage"/>
<ValueLoc Value="CMYK">
<Loc Value="Couleur de CMYK"/>
</ValueLoc>
<ValueLoc Value="RGB">
<Loc Value="Couleur de RGB"/>
</ValueLoc>
<ValueLoc Value="LAB">
<Loc Value="Couleur de LAB"/>
</ValueLoc>
</EnumerationState>
<XYPairState AllowedValueMax="2400 2400"
AllowedValueMin="100 100"
DefaultValue="600 600" Name="OutputResolution">
<Loc ShortValue="resolution"
Value="Resolution de module de balayage"/>
</XYPairState>
</DevCap>
</DevCaps>
</DeviceCap>
</Device>
</DeviceInfo>
</DeviceList>
</Response>
</JMF>
```

10.2.14.2 Device Description of a Scanner #2

This second example illustrates the use of constraints, macros and ***DisplayGroup*** Elements in a capability response. For the sake of simplicity, the only localizations returned are for the constraints.

Example 10.6: KnownDevices Query for a Scanner #2

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Controller"
TimeStamp="2005-04-05T16:45:43+02:00" MaxVersion="1.4" Version="1.4"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<Query ID="DeviceQuery" Type="KnownDevices" xsi:type="QueryKnownDevices">
<DeviceFilter DeviceDetails="Capability" Localization="en"/>
</Query>
</JMF>
```

Example 10.7: KnownDevices Response for a Scanner #2

```
<JMF SenderID="Scanner" TimeStamp="2004-10-17T14:30:47Z"
xmlns="http://www.CIP4.org/JDFSchema_1_1" MaxVersion="1.4" Version="1.4"
DescriptiveName="Example from JDF 1.2 Spec Document"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<Response ID="xyz" Type="KnownDevices" refID="DeviceQuery" ReturnCode="0"
Acknowledged="false" xsi:type="ResponseKnownDevices" >
<DeviceList>
<DeviceInfo DeviceStatus="Idle">
<Device DeviceID="Joe the Drum" ModelName="Bongo">
<DeviceCap GenericAttributes="ID Class SettingsPolicy
BestEffortExceptions OperatorInterventionExceptions
MustHonorExceptions PartIDKeys DocIndex"
Type="Scanning" CombinedMethod="None"
ExecutionPolicy="AllFound">
<Performance AverageAmount="120.0" Name="ExposedMedia" />
<FeaturePool>
<EnumerationState MinOccurs="1"
AllowedValueList="Mono ColorTransparency Photo"
UserDisplay="Display" Editable="true" ID="sm"
ListType="SingleValue" HasDefault="true" Name="ScanMode"
DevNS="http://www.CIP4.org/JDFSchema_1_1" MaxOccurs="1"
MacroRefs="ScanModeMacro" />
</FeaturePool>
<DisplayGroupPool>
<DisplayGroup rRefs="btd cmp mag colorspace outputres">
<Loc HelpText="Parameters for scanning configuration"
Lang="en" ShortValue="ScanningParameters" />
</DisplayGroup>
</DisplayGroupPool>
<ActionPool>
<Action Severity="Error" TestRef="BD-bw" ID="BD-bw-action">
<Loc HelpText="For 1 bit grayscale, please select
CCITTFaxEncoding"
Lang="en" ShortValue="Ouch!"
Value="Flate and DCT Encoding not allowed
on 1 bit images" />
</Action>
<Action Severity="Error" TestRef="ctcmp" ID="ctcmp-action">
<Loc HelpText="Only select CCITTFaxEncoding for
1 bit documents"
Lang="en" ShortValue="Ouch!"
Value="CCITTFaxEncoding not supported on
grayscale images" />
</Action>
<Action Severity="Error" TestRef="cd" ID="cd-action">
<Loc HelpText="Choose a bit depth of 10 or less
for color images"
Lang="en" ShortValue="Ouch!"
Value="Bit depths higher than 10 are not
supported for color" />
</Action>
</ActionPool>
<TestPool>
<Test ID="iscolor">
```

```
<EnumerationEvaluation
ValueList="RGB LAB CMYK" rRef="colorspace" />
</Test>
<Test ID="is1bit">
<IntegerEvaluation ValueList="1" rRef="btd" />
</Test>
<Test ID="BD-bw">
<and>
<TestRef rRef="is1bit" />
<EnumerationEvaluation
ValueList="FlateEncode DCTEncode"
rRef="cmp" />
</and>
</Test>
<Test ID="ctcmp">
<and>
<not>
<TestRef rRef="is1bit" />
</not>
<EnumerationEvaluation ValueList="CCITTFaxEncode"
rRef="cmp" />
</and>
</Test>
<Test ID="cd">
<and>
<TestRef rRef="iscolor" />
<IntegerEvaluation ValueList="1 10" rRef="btd" />
</and>
</Test>
</TestPool>
<MacroPool>
<macro ID="ScanModeMacro">
<choice>
<when>
<EnumerationEvaluation ValueList="Mono" rRef="sm" />
<set rRef="btd">
<FeatureAttribute CurrentValue="1" />
</set>
<set rRef="colorspace">
<FeatureAttribute CurrentValue="GrayScale" />
</set>
<set rRef="outputres">
<FeatureAttribute CurrentValue="1200 1200" />
</set>
</when>
<when>
<EnumerationEvaluation ValueList="ColorTransparency"
rRef="sm" />
<set rRef="btd">
<FeatureAttribute CurrentValue="8" />
</set>
<set rRef="colorspace">
<FeatureAttribute CurrentValue="RGB" />
</set>
<set rRef="outputres">
<FeatureAttribute CurrentValue="600 600" />
</set>
</when>
<when>
<EnumerationEvaluation ValueList="Photo" rRef="sm" />
<set rRef="btd">
<FeatureAttribute CurrentValue="10" />
</set>
<set rRef="colorspace">
<FeatureAttribute CurrentValue="LAB" />
</set>
<set rRef="outputres">
<FeatureAttribute CurrentValue="200 200" />
</set>
</when>
```

```
</choice>
</macro>
</MacroPool>
<DevCaps Required="false" Context="Resource"
DevNS="http://www.CIP4.org/JDFSchema_1_1"
Availability="Installed"
Name="NodeInfo" ResourceUpdate="None">
<DevCap MinOccurs="1" Name="NodeInfo"
DevNS="http://www.CIP4.org/JDFSchema_1_1" MaxOccurs="1">
<StringState UserDisplay="Display"
DevNS="http://www.CIP4.org/JDFSchema_1_1"
Editable="true" MinOccurs="1" MaxOccurs="1"
Name="TargetRoute" HasDefault="true"
ListType="SingleValue" />
<IntegerState Name="JobPriority"
DevNS="http://www.CIP4.org/JDFSchema_1_1"
Editable="true" MinOccurs="1" MaxOccurs="1"
UserDisplay="Display" HasDefault="true"
ListType="SingleValue" />
</DevCap>
</DevCaps>
<DevCaps Required="false" ResourceUpdate="None" Context="Resource"
Availability="Installed" Name="ExposedMedia"
DevNS="http://www.CIP4.org/JDFSchema_1_1">
<DevCap MinOccurs="1" Name="ExposedMedia"
DevNS="http://www.CIP4.org/JDFSchema_1_1" MaxOccurs="1">
<DevCap MinOccurs="1" Name="Media"
DevNS="http://www.CIP4.org/JDFSchema_1_1"
MaxOccurs="1">
<NameState MinOccurs="1" DefaultValue="Sheet"
UserDisplay="Display" Editable="true"
ListType="SingleValue" HasDefault="true"
Name="MediaUnit" MaxOccurs="1"
DevNS="http://www.CIP4.org/JDFSchema_1_1"/>
<XYPairState MinOccurs="1" UserDisplay="Display"
Editable="true" AllowedValueMax="600.0 1200.0"
ListType="SingleValue" HasDefault="true"
Name="Dimension" AllowedValueMin="0.0 0.0"
DevNS="http://www.CIP4.org/JDFSchema_1_1"
MaxOccurs="1" />
</DevCap>
</DevCap>
</DevCaps>
<DevCaps Required="false" Context="Resource"
DevNS="http://www.CIP4.org/JDFSchema_1_1"
Availability="Installed" Name="ScanParams"
ResourceUpdate="None">
<DevCap MinOccurs="1" Name="ScanParams"
DevNS="http://www.CIP4.org/JDFSchema_1_1" MaxOccurs="1">
<IntegerState MinOccurs="1" DefaultValue="1"
AllowedValueList="1 4 8 10 12" UserDisplay="Hide"
ActionRefs="BD-bw ctcmp cd" Editable="true"
ID="btd" ListType="SingleValue" HasDefault="true"
Name="BitDepth" MaxOccurs="1"
DevNS="http://www.CIP4.org/JDFSchema_1_1"/>
<EnumerationState ActionRefs="BD-bw ctcmp" MinOccurs="1"
AllowedValueList=
"CCITTFaxEncode FlateEncode DCTEncode None"
UserDisplay="Hide" Editable="true" ID="cmp"
ListType="SingleValue" HasDefault="true"
Name="CompressionFilter" MaxOccurs="1"
DevNS="http://www.CIP4.org/JDFSchema_1_1" />
<NumberState MinOccurs="1" UserDisplay="Display"
Editable="true" ID="mag" ListType="SingleValue"
HasDefault="true" AllowedValueMax="100.0"
AllowedValueMin="0.01" MaxOccurs="1"
Name="Magnification"
DevNS="http://www.CIP4.org/JDFSchema_1_1"/>
<EnumerationState ActionRefs="cd" MinOccurs="1"
AllowedValueList="GrayScale CMYK RGB LAB"
```

```
UserDisplay="Display" Editable="true"
ID="colorspace" ListType="SingleValue"
HasDefault="true" Name="OutputColorSpace"
MaxOccurs="1"
DevNS="http://www.CIP4.org/JDFSchema_1_1" />
<XYPairState MinOccurs="1" DefaultValue="600.0 600.0"
AllowedValueList="100.0 100.0 300.0 300.0 600.0 600.0
1200.0 1200.0 2400.0 2400.0"
UserDisplay="Display" Editable="true" ID="outputres"
ListType="SingleValue" HasDefault="true"
Name="OutputResolution" MaxOccurs="1"
DevNS="http://www.CIP4.org/JDFSchema_1_1" />
</DevCap>
</DevCaps>
</DeviceCap>
</Device>
</DeviceInfo>
</DeviceList>
</Response>
</JMF>
```

Example 10.8: JDF Accepted by Previous Scanner

Example of **JDF** Node that is accepted by the scanner of the previous example. All parameters of the following Scanning Node are compliant with the capabilities.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="GoodScan"
Status="Waiting" Type="Scanning" JobPartID="ID300" Version="1.4">
<ResourcePool>
<ScanParams BitDepth="8" Class="Parameter" ID="Link0007"
OutputColorSpace="RGB" OutputResolution="600. 600." Status="Available"/>
<ExposedMedia Class="Handling" ID="Link0008" Status="Available">
<Media Dimension="425.196850394 566.929133858"/>
</ExposedMedia>
<RunList Class="Parameter" ID="Link0014" Status="Available"/>
</ResourcePool>
<ResourceLinkPool>
<ScanParamsLink Usage="Input" rRef="Link0007"/>
<ExposedMediaLink Usage="Input" rRef="Link0008"/>
<RunListLink Usage="Output" rRef="Link0014"/>
</ResourceLinkPool>
</JDF>
```

Example 10.9: JDF Rejected by Previous Scanner

Example of **JDF** Node that is rejected by the scanner of the previous example. All parameters of the following Scanning Node except `Magnification` are compliant with the Device capabilities. Therefore, the Device can not execute the Job.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="BadScan" Status="Waiting"
Type="Scanning" JobPartID="ID300" Version="1.4">
<ResourcePool>
<ScanParams BitDepth="8" Class="Parameter" ID="Link0012"
Magnification="1000. 1000."
OutputColorSpace="RGB" OutputResolution="600. 600." Status="Available"/>
<ExposedMedia Class="Handling" ID="Link0013" Status="Available">
<Media Dimension="425.196850394 566.929133858"/>
</ExposedMedia>
<RunList Class="Parameter" ID="Link0014" Status="Available"/>
</ResourcePool>
<ResourceLinkPool>
<ScanParamsLink Usage="Input" rRef="Link0012"/>
<ExposedMediaLink Usage="Input" rRef="Link0013"/>
<RunListLink Usage="Output" rRef="Link0014"/>
</ResourceLinkPool>
</JDF>
```

10.3 Concept of the Preflight Process

New in JDF 1.2

Note: This section establishes Elements, Attributes and Attribute Values that are used by the Resources referenced by the **Preflight** Process, including ***PreflightParams***, ***PreflightReportRulePool*** and ***PreflightReport***, as well as extensions of testing methodology established ***Action*** and ***Test*** functions defined in ▸ Section 10.2.1 DeviceCap.

In order to define one ***Test***, you can combine one or more basic tests using the Boolean logic as defined in ▸ Section 10.2 Device Capability Definitions. Each basic test is applied to one defined property with a given data type. Note that document properties defined in this section include one or more Attributes that are extracted from documents (e.g., a client's PDF file) and used by one or more evaluations as part of a preflight test. Each data type can be tested on an object using its matching ***Evaluation***. A document that is preflighted is made of objects. Some of them, like virtual boxes (**TrimBox** or **MediaBox**) are not visible. In order to combine basic tests together, they have been classified by groups of properties. These groups do not necessarily match a class of an object. However, each class of object will implement one or more groups of properties.

The rules to combine basic tests into a ***Test*** can be built on both object classes and groups of properties. Each basic test takes an object as an input and has four different states in output: "false", "true", "TestWrongPDL" or "TestNotSupported". The two last values occur when a basic test has no meaning for the given object or when the application that is executing the test does not support that test. These four different states lead to a more open way of dealing with Boolean logic

"false"	AND	"TestWrongPDL"	=	"false"
"true"	OR	"TestWrongPDL"	=	"true"
"false"	AND	"TestNotSupported"	=	"false"
"true"	OR	"TestNotSupported"	=	"true"
"true"	AND	"TestWrongPDL"	=	"TestWrongPDL"
"false"	OR	"TestWrongPDL"	=	"TestWrongPDL"
"true"	AND	"TestNotSupported"	=	"TestNotSupported"
"false"	OR	"TestNotSupported"	=	"TestNotSupported"
"TestWrongPDL"	OR	"TestNotSupported"	=	"TestNotSupported"
"TestWrongPDL"	AND	"TestNotSupported"	=	"TestNotSupported"
if ("true")		Report according to action.		
if ("false")		Do not report.		
if ("TestWrongPDL")		Report problem if specified in ***PRRule***.		
if ("TestNotSupported")		Report problem if specified in ***PRRule***.		

For instance, "TestWrongPDL" would occur when a test about font size is made on a page. "TestNotSupported" would happen when a **JDF** preflight agent does not support the concept of font size.

10.3.1 Object Classes

▸ Table 10.70 Object Classes for a Document below has a list of the real objects that can be preflighted in a document. The objects are identified by their class name specified in the "Name" column:

Table 10.70: Object Classes for a Document (Sheet 1 of 2)

NAME	DESCRIPTION
Annotation	An annotation is a complex object that adds information to the page of a document. The characteristic of such object is that it is optional to print it. When an annotation is set to be printed, the graphical objects making the annotation are considered separated objects.
Document	The document, which is preflighted.
Font	A font is a set of characters that can be used to draw text. A font can be in a document without being used by any text of the document.
Image	An image is a graphic object drawn with colored pixels.
MaskUsingImage	This object is an object that masks another object using an image.
MaskUsingVector	This object is an object that masks another object using a vector path.
MaskUsingText	This object is an object that masks another object using text components.
Mask	A mask is an object used to mask or clip a graphic object.
Page	A document can be made of finished pages (but could be empty as well).

Table 10.70: Object Classes for a Document (Sheet 2 of 2)

NAME	DESCRIPTION
PageBox	In each finished page, some virtual boxes can be defined (page size and margins). Some tests can be done with these boxes.
PDL	A PDL object is a generic kind of object that can be specific to some types of documents. It is just a way to detect presence or not of such objects.
Shading	A shading is a graphic object drawn using a smooth color change from one point to another.
Text	A text is a set of characters that have exactly the same style (i.e., same size, same font, same fill and stroke, etc.).
Vector	A vector is a graphic object drawn with vector curves. It is made of a fill and a stroke.

10.3.1.1 Properties Implemented by each Class of Object

Table 10.71 Properties Implemented by each Class of Object below, has columns of object Classes and rows of Properties Categories. An "X" in a cell means that an object of the specified Class implements the specified Properties (see Table 10.73 List of Properties Categories).

Table 10.71: Properties Implemented by each Class of Object (Sheet 1 of 2)

PROPERTIES	CLASSES													
	DOCUMENT	PAGE	IMAGE	VECTOR	TEXT	SHADING	IMAGEMASK	ANNOTATION	PAGEBOX	FONT	MASKUSINGIMAGE	MASKUSINGVECTOR	MASKUSINGTEXT	PDL
Logical	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Class	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Document	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Page		X	X	X	X	X	X	X	X	X	X	X	X	X
Reference			X	X										
Colorant	X		X	X	X	X	X							
Box		X	X	X	X	X	X	X	X		X	X	X	X
Graphic			X	X	X	X	X							
Fill				X	X		X							
Stroke				X	X									
Image			X				X				X			
Vector				X								X		
Text					X								X	
Shading						X								
Font					X					X				
Annotation								X						
Page Box									X					

Table 10.71: Properties Implemented by each Class of Object (Sheet 2 of 2)

PROPERTIES	CLASSES													
	DOCUMENT	PAGE	IMAGE	VECTOR	TEXT	SHADING	IMAGEMASK	ANNOTATION	PAGEBOX	FONT	MASKUSINGIMAGE	MASKUSINGVECTOR	MASKUSINGTEXT	PDL
PDL Object														X

10.3.1.2 Checking for the Presence of a Property

In most of the **Preflight** Process, only the "values" of properties are needed. Please note that a property MAY incorporate one or more Attributes, and it is the values (e.g., string or enumeration) of these Attributes that are collectively referred to here as the "value" of the property. In some cases, it is also useful to be able to check if a property has been defined. This happens in some types of documents where the property definition is optional. Before checking its value, you just want to check that this property was defined.

For all the basic tests described in this document where it makes sense to check if they are defined, they are checked "Yes" in the **Tag** column of properties definition tables below. Use the ***IsPresentEvaluation*** to check for the presence of a property.

Example 10.10: Test for Existence of TrappedKey

This example checks if the @*TrappedKey* is defined in a PDF document.

```
<Device Class="Implementation" ID="Link0003" Status="Available">
<DeviceCap>
<TestPool>
<Test ID="PT01">
<IsPresentEvaluation>
<BasicPreflightTest Name="TrappedKey"/>
</IsPresentEvaluation>
</Test>
</TestPool>
</DeviceCap>
</Device>
```

Example 10.11: Test for TrappedKey Equal to "Unknown"

This example checks if the value of the @*TrappedKey* = "Unknown" in a PDF document.

```
<Device Class="Implementation" ID="Link0003" Status="Available">
<DeviceCap>
<TestPool>
<Test ID="PT02">
<EnumerationEvaluation ValueList="Unknown">
<BasicPreflightTest Name="TrappedKey"/>
</EnumerationEvaluation>
</Test>
</TestPool>
</DeviceCap>
</Device>
```

Table 10.72: Mapping between property types (in the preflight spec) and evaluations (Sheet 1 of 2)

PROPERTY TYPE	EVALUATION	EXPECTED USAGE FOR BASICPREFLIGHTTEST LISTTYPE
presence	***IsPresentEvaluation***	-
boolean	***BooleanEvaluation***	SingleValue.
BooleanList	***BooleanEvaluation***	Any of @*ListType*'s value that refers to a list.

Table 10.72: Mapping between property types (in the preflight spec) and evaluations (Sheet 2 of 2)

PROPERTY TYPE	EVALUATION	EXPECTED USAGE FOR BASICPREFLIGHTTEST LISTTYPE
DateTime	***DateTimeEvaluation***	SingleValue.
DateTimeList	***DateTimeEvaluation***	Any of @*ListType*'s value that refers to a list.
enumeration	***NameEvaluation***	SingleValue.
enumerations	***NameEvaluation***	Any of @*ListType*'s value that refers to a list.
integer	***IntegerEvaluation***	SingleValue.
IntegerList	***IntegerEvaluation***	Any of @*ListType*'s value that refers to a list.
Name	***NameEvaluation***	SingleValue.
NameList	***NameEvaluation***	Any of @*ListType*'s value that refers to a list.
double	***NumberEvaluation***	SingleValue.
DoubleList	***NumberEvaluation***	Any of @*ListType*'s value that refers to a list.
rectangle	***RectangleEvaluation***	SingleValue.
RectangleList	***RectangleEvaluation***	Any of @*ListType*'s value that refers to a list.
string	***StringEvaluation***	SingleValue.
StringList	***StringEvaluation***	Any of @*ListType*'s value that refers to a list.
XYPair	***XYPairEvaluation***	SingleValue.
XYPairList	***XYPairEvaluation***	Any of @*ListType*'s value that refers to a list.

10.3.1.3 Basic tests on set of objects

Some properties can be applied to more than one object and have a value when applied to a list of objects which differs from their value when applied to a single object. For instance, this allows you to make tests on the number of separations of objects included in a given area. These properties have the column "Set" checked with "Yes". In order to define a ***Test*** using such properties, a list of objects is filtered first, before applying the test. This is achieved using the ***PreflightArgument*** Element.

10.3.2 Properties

▸ Table 10.73 List of Properties Categoriesspecifies the Properties Categories. In each of the following subsections, there is a table with a list of Attributes belonging to the specified Properties Category. Each such Attribute can be found, extracted, and evaluated from a document. The Attributes of each Properties Category apply to Objects of certain specified Classes (see ▸ Table 10.72 Mapping between property types (in the preflight spec) and evaluations).

Note: Each table of Properties in the subsections below has a different meaning from a table for an Element or Resource, which describes an XML element along with its member attributes or subelements. A Properties table does not describe an XML element or any other structure. Rather each table row describes an Attribute that is a potential Attribute of some Element derived from ***Abstract PRGroupOccurrenceBase*** Element (see ▸ Table 8.203 List of PRGroupOccurrenceBase Elements).

Note: For each Properties tables, the "Set" column is described in ▸ Section 10.3.1.3 Basic tests on set of objects, and the "Tag" column is described in ▸ Section 10.3.1.2 Checking for the Presence of a Property.

Table 10.73: List of Properties Categories

NAME	PAGE	DESCRIPTION
▸ Annotation Properties	page 731	Describes Annotations.
▸ Box Properties	page 732	Describes a container box
▸ Class Properties	page 732	Describes the Class name and Property name
▸ Colorant Properties	page 733	Describes color and separation information.
▸ Document Properties	page 734	Describes a document.
▸ Fill Properties	page 737	Describes fill for graphic objects
▸ Font Properties	page 738	Describes fonts in a document:
▸ Graphic Properties	page 739	Describes display and graphic information
▸ Image Properties	page 741	Describes images displayed using pixels
▸ Logical Properties	page 742	Mainly used with "**Set**" to count the number of objects
▸ PageBox Properties	page 743	Describes virtual boxes for each page.
▸ Pages Properties	page 743	Describes a page in a document
▸ PDLObject Properties	page 744	Describes particular PDF objects in a document
▸ Reference Properties	page 744	Describes references to external objects.
▸ Shading Properties	page 745	Describes shading that is applied graphic objects.
▸ Stroke Properties	page 745	Describes strokes applied to graphic objects with vector primitives
▸ Text Properties	page 746	Describes text.
▸ Vector Properties	page 747	Describes graphic objects with vector primitives.

10.3.2.1 Annotation Properties

Annotation objects are specific objects that can be displayed or printed according to the user's choice. When they are displayed or printed, they add graphical objects to the document that can be preflighted.

Table 10.74: Annotation Properties (Sheet 1 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
AnnotationPrintFlag	boolean	Is "true" when it will be printed on the final document.	—	—	PDF
AnnotationType	NMTOKEN	The type of annotations. **Values include those from:** ▸ Table 10.75 AnnotationType Attribute Values.	—	—	PDF

Table 10.74: Annotation Properties (Sheet 2 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
TrapnetAnnotationPDFX	NMTOKENS	The PDF/X versions to which the @*TrapNet* annotation complies (e.g., "PDF/X-1a:2003").	—	—	PDF

10.3.2.1.1 AnnotationType Attribute Values

*Table 10.75: **AnnotationType** Attribute Values*

VALUE	DESCRIPTION	VALUE	DESCRIPTION
Circle		Sound	
FileAttachment		Square	
FreeText		Squiggly	
Highlight		Stamp	
Ink		StrikeOut	
Link		Text	
Line		TrapNet	
Movie		Underline	
Popup		Widget	
PrinterMark			

10.3.2.2 Box Properties

All visible objects can be described at least by a box in which they can be contained. In a page, some kind of boxes can define some basic Box Properties that are extracted as Attributes for use in a test.

Table 10.76: Box Properties

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
BoundingBox	rectangle	The bounding box of the object is the smallest rectangle containing the object. When used with group of objects, this is the smallest box containing boxes of all objects.	Yes	—	—
DifferentBoxSize	enumerations	This is the list of boxes, which are different on one page from the same boxes on another page. **Allowed values are from:** ***BoxArgument***/@*Box* (▸ Table 8.195 BoxArgument Element).	Only	—	—
InsideBox	boolean	Is "true" when an object is inside a given box. @*InsideBox* SHALL be qualified by ***BoxArgument*** Subelement.	—	—	—
OutsideBox	boolean	Is "true" when an object is outside a given box. @*OutsideBox* SHALL be qualified by ***BoxArgument*** Subelement.	—	—	—

10.3.2.3 Class Properties

Each object can define the name of the class of objects it belongs to:

Table 10.77: Class Properties

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
ClassName	NMTOKEN	The name of the class to which the object belongs. **Values include those from:** ▸ Table 10.78 ClassName Attribute Values.	—	—	—
PropertyList	enumerations	The list of Properties the object has. **Allowed values are from:** ▸ Table 10.79 PropertyList Attribute Values.	—	—	—

10.3.2.3.1 ClassName Attribute Values

*Table 10.78: **ClassName** Attribute Values*

VALUE	DESCRIPTION	VALUE	DESCRIPTION
Annotation		MaskUsingVector	
Document		Page	
Font		PageBox	
Image		PDL	
ImageMask		Shading	
MaskUsingImage		Text	
MaskUsingText		Vector	

10.3.2.3.2 PropertyList Attribute Values

*Table 10.79: **PropertyList** Attribute Values*

VALUE	DESCRIPTION	VALUE	DESCRIPTION
Annotation		Logical	
Box		Page	
Class		PageBox	
Colorant		PDLObject	
Document		Reference	
Fill		Shading	
Font		Stroke	
Graphic		Text	
Image		Vector	

10.3.2.4 Colorant Properties

Every visible object or group of objects will imply a given number of separations.

Table 10.80: Colorant Properties (Sheet 1 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
AliasSeparations	boolean	Is "true" when some of the separations have different names but the same color values.	Yes	—	—

Table 10.80: Colorant Properties (Sheet 2 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
AmbiguousSeparations	boolean	Is "true" when some of the separations have the same name but different color values.	Yes	—	—
InkCoverage	double	This is the maximum percentage of ink coverage for one object. In case of a group of objects, this is the maximum amount of ink coverage for the list of objects. The method of calculation can be application-dependant and can differ from one application to another. Some applications MAY check the coverage object by object without taking into account overprint or transparencies between objects; some others MAY use a rasterization Process to get the coverage of the combined objects.	Yes	—	—
SeparationList	string	List of all separations necessary to print one object or a group of objects.	Yes	—	—

10.3.2.5 Document Properties

This is the list of Properties (Attributes) that define parts of a document.

Table 10.81: Document Properties (Sheet 1 of 4)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
Author	string	A string describing the author of the document.	—	Yes	—
Binding	enumeration	The binding of the document: **Allowed values are:** Left Right	—	Yes	PDF
CreationDate	dateTime	The date when the document was created according to the file system.	—	—	—
CreationDateInDocument	dateTime	The date when the document was created according to data inside the document.	—	Yes	—
CreationID	NMTOKEN	An NMTOKEN which identifies a document when created. The @*CreationID* SHALL be unique within the workflow. In case of a PDF, it matches exactly the first element of ID array.	—	Yes	—
Creator	string	A string describing the creator of the document. This is usually the name and version of the authoring application used. In case of PS and PDF files, it matches exactly the Creator key.	—	Yes	—
DocumentCompression	enumerations	A list of all compression types used in the document (including image compression referenced by @*CompressionTypes* in ***Image Properties***). **Allowed values are from:** @*CompressionTypes* in ▸ Table 10.87 Image Properties.	—	—	—

Table 10.81: Document Properties (Sheet 2 of 4)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
DocumentCorruption	NMTOKENS	The list of recoverable errors against the document format that were found in this document. An empty list means the document is not corrupted. **Values include:** InvalidOffsets – Some offsets are invalid, but the preflight agent was able to load the document nonetheless. Note that the absence of this value does not mean that all document structures are valid, only that the offsets are correct)	—	—	—
DocumentEncoding	enumeration	The document encoding which can be either: **Allowed values are:** ASCII Binary	—	—	PS, PDF
DocumentIsGoodCompression	boolean	Is "true" when a strong compression algorithm is used (not just an ASCII filter) for all objects in the document where it makes sense to have compression.	—	—	—
EncryptedDocument	boolean	Is "true" if document is encrypted.	—	—	—
EncryptionFilter	NMTOKEN	The Filter name of encryption for a PDF file.	—	Yes	PDF
EncryptionLength	integer	The length of the encryption key of a PDF file in bits.	—	Yes	PDF
EncryptionRestrictions	NMTOKENS	The actions that are forbidden by the encryption. **Values include:** Assembly – Inserting or removing pages. Copying – Extracting part of the content. DisabledAccess – Allowing copying specifically for providing access to the disabled. EditingAnnotations EditingContent FillingIn – Filling in forms. HighResPrinting Printing	—	—	PDF
EncryptionSubFilter	NMTOKEN	The SubFilter name of encryption for a PDF file.	—	Yes	PDF
EncryptionV	integer	The V integer of encryption for a PDF file.	—	Yes	PDF
FileName	string	The file name, including file extension, in the file system. This is not the full path.	—	—	—
FileSize	integer	The file size expressed in bytes.	—	—	—

Table 10.81: Document Properties (Sheet 3 of 4)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
Keywords	string	A string made of keywords describing the document.	—	Yes	—
Linearized	boolean	Is "true" if the document is linearized (i.e., prepared for web download).	—	—	PDF
ModificationDate	dateTime	The date when the document was last modified according to the file system.	—	—	—
ModificationDateInDocument	dateTime	The date when the document was last modified according to data inside the document.	—	Yes	—
ModificationID	NMTOKEN	A name that which can uniquely identify the current document instance. In case of a PDF, it matches exactly the second element of ID array.	—	Yes	—
NumberOfPages	integer	The number of finished pages contained in the document.	—	—	—
OutputIntentColorSpace = "None"	NMTOKEN	The color space belonging to the output intent of the document. **Values include:** None – The default value to be used if this property is not present. CMYK Gray RGB	—	Yes	PDF
OutputIntentStandard	string	The standards the output intent is compliant with (e.g., PDF/X-1a:2001). The version of the standard is assumed to be in the string accordingly to the standard's notation.	—	—	—
PagesHaveSameOrientation	boolean	Is "true" when all pages have the same orientation.	—	—	—
PDFXVersion	NMTOKEN	The PDF/X version key present in the document.	—	Yes	PDF
PDLType	NMTOKEN	The type of document expressed as a MIME-type. **Values include those from:** ▸ Table G.1 MimeType Attribute Values (IANA Registered) and ▸ Table G.2 MimeType and File Type Combinations. **Example:** @*PDLType* value is "application/pdf".	—	—	—
PDLVersion	string	The version of document according to the @*PDLType*. **Values include those from:** ▸ Table G.1 MimeType Attribute Values (IANA Registered) and ▸ Table G.2 MimeType and File Type Combinations.	—	—	—

Table 10.81: Document Properties (Sheet 4 of 4)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
Producer	string	A string describing the producer of the document. This is usually the name of the software used to create file. In case of PDF files, it matches exactly the Producer key.	—	Yes	—
SeparationFlag	boolean	Is "true" if the document is made of separations or is not composite.	—	—	PS, PDF
Subject	string	A string describing the subject of the document.	—	Yes	—
Title	string	A string describing the title of the document.	—	Yes	—
TrappedKey	enumeration	A value explaining the use of trapping on the document. **Allowed values are:** true false Unknown **Note:** The values match exactly the @*TrappedKey* information of PDF.	—	Yes	—

10.3.2.6 Fill Properties

Fill property values are derived from graphic objects with vector primitives. They can have a fill color and a stroke color, with given colors. This is a list of Properties that specifically apply to this kind of object:

Table 10.82: Fill Properties

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
FillColorName	string	The name of the color of the fill of the vector object.	—	—	—
FillColorType	enumeration	This is an enumeration of known colors to draw fill. **Allowed values are from:** ▸ Table 10.83 FillColorType Attribute Values.	—	—	—
HasFillColor	boolean	Is "true" if the vector object is drawn with a fill color.	—	—	—

10.3.2.6.1 FillColorType Attribute Values

Table 10.83: FillColorType Attribute Values

VALUE	DESCRIPTION
CMYBlack	Will print with 100% on Cyan, Magenta and Yellow separations and less than 100% on the Black separation.
CMYGray	Will print with the same percentage 0–100% exclusive on Cyan, Magenta and Yellow separations.
Other	Any other combinations of separations.
PureBlack	Will print as 100% on the black separation with 0% on the other separation(s).
PureGray	Will print as 1–99% on the black separation with 0% on the other separation(s).
RegistrationBlack	Will print as 100% on all the separations.
RegistrationGray	Will print as 0–100% exclusive on all the separations (assuming all the separations use the same value).
RichBlack	Will print as 100% on the black separation with more than 0% on one or more of the other separations.
White	Will print as 0% on all the separations.

10.3.2.7 Font Properties

The following is the list of property Attributes that can be applied to a font contained in, or referenced into, a document:

Table 10.84: Font Properties (Sheet 1 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
EmbeddingRestrictionFlag	boolean	Is "true" if a font cannot be embedded.	—	—	—
FontCorrupted	boolean	Is "true" if a font is corrupted or invalid. The implementation of this check MAY vary from one application to another.	—	—	—
FontCreator	string	The font creator.	—	—	—
FontEmbedded	boolean	Is "true" if a font is embedded into the document.	—	—	—
FontIsStandardLatin	boolean	Is "true" when all characters belong to the standard Latin character set.	—	—	—
FontName	string	The font name.	—	—	—
FontNotUsed	boolean	Is "true" if a font is not used to draw characters from the document.	—	—	—
FontSubset	boolean	Is "true" if a font is only a subset of a main font.	—	—	PS, PDF
FontType = "Other"	enumeration	This is the type of the font. **Allowed values are from:** ▸ Table 10.85 FontType Attribute Values.	—	—	—
FontVendor	string	The font vendor.	—	—	—
IsDoubleByteFont New in JDF 1.4	boolean	Some fonts need double-byte encoding to store characters internally	—	—	—

Table 10.84: Font Properties (Sheet 2 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
IsFontScreenOnly	boolean	Is "true" if a font referenced in the document contains only screen description.	—	—	Authoring
PSFontName	NMTOKEN	The PostScript font name.	—	—	PS, PDF

10.3.2.7.1 FontType Attribute Values

*Table 10.85: **FontType** Attribute Values*

VALUE	DESCRIPTION	VALUE	DESCRIPTION
CIDFontType0		Type1	
CIDFontType1		Type1CMultipleMaster	
CIDFontType2		Type2C	
CIDFontType3		Type3	
CIDFontType4		PDFType3	
OpenType		Type42	Embedded TrueType into a PostScript font.
TrueType		Unknown	Type of font that can not be resolved for any reason (i.e., missing font, etc.).
Type0	PostScript Type0 without the CID	Other	To be used when the property is not any of the values listed above.

10.3.2.8 Graphic Properties

This is a list of property Attributes that specifically apply to objects that can be displayed or printed.

Table 10.86: Graphic Properties (Sheet 1 of 3)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
AlphaIsShape	boolean	The @*AlphaIsShape* of a PS or PDF object.	—	—	PS, PDF
AlternateColorSpace	enumeration	The alternate color space of the object is one of the given. **Allowed values are from:** @*ColorSpace*.	—	Yes	PS, PDF
BelongsToAnnotation	boolean	Is "true" when this object belongs to an annotation.	—	—	—
BlackGeneration	enumeration	The @*BlackGeneration* function of a PS or PDF object. **Allowed values are:** Identity – Defines identity function. Custom – Used when the function is described.	—	Yes	PS, PDF
BlendMode	NMTOKEN	The @*BlendMode* of a PS or PDF object.	—	—	PS, PDF

Table 10.86: Graphic Properties (Sheet 2 of 3)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
ColorSpace	enumeration	The color space of the object. **Allowed values are:** CalGray CalRGB CIEBasedA CIEBasedABC CIEBasedDEFG DeviceCMYK DeviceGray DeviceN DeviceRGB ICCBased Lab Separation	—	—	PS, PDF
EmbeddedPS	boolean	Is "true" if a PDF object uses PostScript to be drawn.	—	—	PDF
Flatness	double	A number giving the value of PS or PDF *Flatness*.	—	Yes	PS, PDF
HalfTone	NMTOKEN	The value of the Halftone used in a document: "Named", "1", "5", "6", "10", "16".	—	Yes	PS, PDF
HalfTonePhase	XYPair	The value of the @*HalftonePhase* associated with the object.	—	Yes	PS, PDF
HasColorLUT	boolean	Is "true" when an object is using indexed colors in a table to describe color.	—	—	—
HasSoftMask	boolean	Is "true" when the object is using a soft-mask using pixel values.	—	—	—
NumberOfColorsInLUT	integer	The number of colors in the color table used to display an indexed image.	—	—	—
OverPrintFlag	boolean	Is "true" when one object has been set to overprint.	—	—	—
OverPrintMode	integer	An integer giving the PostScript or PDF value for overprint mode.	—	—	PS, PDF
RenderingIntent	NMTOKEN	The rendering intent of a PS or PDF object.	—	Yes	PS, PDF
Smoothness	double	A number giving the value of PS or PDF @*Smoothness*.	—	Yes	PS, PDF
TransferFunction	enumeration	The transfer function of a PS or PDF object. **Allowed values are:** Custom – Used when the function is described. Identity – Defines identity function.	—	Yes	PS, PDF
TransparencyFlag	boolean	Is "true" when the object has transparency. A transparency that is null has the "false" value.	—	—	—

Table 10.86: Graphic Properties (Sheet 3 of 3)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
UnderColorRemoval	enumeration	The @*UnderColorRemoval* function of a PS or PDF object. **Allowed values are:** Custom – Used when the function is described. Identity – Defines identity function.	Yes	Yes	PS, PDF

10.3.2.9 Image Properties

This group of property Attributes is very specific to images displayed using pixels:

Table 10.87: Image Properties (Sheet 1 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
AlternateImages	NMTOKENS	When to draw some of the alternate images that correspond with the given image. The PDF specification defines "Print" as a value, but any other application-specific value could be used. **Values include:** Print	—	Yes	PDF
BitsPerSample	integer	The number of bits used to represent color on every separation.	—	—	—
CompressionRatio	double	For all compression types to which it makes sense, the tests apply to the quality expressed as percentage of compression.	—	—	—
CompressionTypes	enumerations	The type of method used to compress or encode the image. **Allowed values are:** ASCII85 ASCIIHex CCITT JBIG2 JPEG JPEG2000 LZW None RunLength ZIP **Note:** Where JPEG, JPEG2000 and/or JBIG2 are specified, they can be concatenated and only JPEG need be used.	—	—	—
EffectiveResolution	XYPair	The horizontal and vertical resolutions of the scaled image, in dots per inch.	—	—	—
EstimatedJPEGQuality	integer	For "JPEG" compression type, use algorithm provided below to obtain the estimated JPEG quality by doing a "reverse statistic" on the IJG library's quality-to-matrix routine. This value will be expressed as an integer, where "0" is the worse quality and "100" is the best quality.	—	—	—

Table 10.87: Image Properties (Sheet 2 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
ImageFlipped	enumeration	The way the image is flipped. **Allowed values are:** None Horizontal Vertical	—	—	—
ImageMaskType	enumeration	The type of masks used by image. **Allowed values are:** NoMask – Used when the image does not use specific mask. BitmapMask – Used when the image is masked using a bitmap image ColorKeyMask – Used when some colors are masked out to display the image (such like video chroma-key).	—	—	—
ImageRotation	integer	The number of degrees an image is rotated. A positive number represents a counterclockwise rotation. A negative number represents a clockwise rotation. **Note:** A 540^0 rotation is valid (e.g., one full rotation + 180^0 rotation).	—	—	—
ImageScalingRatio	double	The ratio between X and Y scaling of an image.	—	—	—
ImageSkew	double	The skew angle of the image ("0" is not skewed). A positive number represents a clockwise skewing. A negative number represents a counterclockwise skewing.	—	—	—
OriginalResolution	XYPair	The horizontal and vertical resolutions of the image before scaling.	—	—	—
PixelHeight	integer	Image height in pixels.	—	—	—
PixelWidth	integer	Image width in pixels.	—	—	—

The JPEG quality algorithm is based on a technique used by the IJG library (http://www.ijg.org/) — which uses a quality value in the range 0–100 and translates image data into a 8x8 matrix. The following algorithm performs a "reverse statistic" on the IJG library's quality-to-matrix routine, which gives a matrix-to-quality routine. The formula's used are as follows:

```
//DCTSIZE2 is the size of the matrix, 64
derived = 0.0;
for (i = 0; i < DCTSIZE2; i++){
derived += (*qtblptr0)->quantval[i];
}
derived = derived / DCTSIZE2;
xq = (100.0 * derived - 50.0) / 57.625;
if (xq < 100.0){
quality = (long) ((200.0 - xq) / 2.0);
} else {
quality = (long) (5000.0 / xq);
}
```

The algorithm calculates the average value in the quantization matrix and then derives a quality value in the range of 0–100 from that average.

10.3.2.10 Logical Properties

The logical Properties are mainly used with "**Set**" to count the number of objects.

Table 10.88: Logical Properties

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
Count	integer	The number of objects contained in the referenced set of objects.	Yes	—	—

10.3.2.11 PageBox Properties

The page box represents virtual boxes for each page. The following is a list of Attributes that specifically apply to this kind of objects.

Table 10.89: PageBox Properties

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
PageBoxType	enumeration	**Note:** When not known, the default SHALL leave @*PageBoxType* empty. **Allowed values are from:** ***BoxArgument***/@*Box* (▸ Table 8.195 BoxArgument Element).	—	—	—

10.3.2.12 Pages Properties

This is the list of Elements and Attributes related to the page object in a document.

Table 10.90: Pages Properties (Sheet 1 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
BlankPage	boolean	Is "true" when the trim box and the bleed box area, when defined, do not output any marks.	—	—	—
BlendColorSpace	enumeration	The page blend color space. **Allowed values are from**: @*ColorSpace* in ▸ Table 10.86 Graphic Properties.	—	Yes	PDF
PageHasOptionalContent New in JDF 1.4	boolean	Detect if a PDF has optional content (commonly called PDF layers).	—	—	—
PageHasUnknownObjects	boolean	Page contains unknown objects but the PDL was set to ignore these errors. Examples are the use of BX/EX in PDF.	—	—	—
PageNumber	integer	The page index in the ***RunList***.	—	—	—
PageScalingFactor New in JDF 1.4	double	In PDF file, one way of scaling a page is to use a page scale factor. This factor can be ambiguous because it is not always used by all applications.	—	—	—
ReversePageNumber	integer	A special page numbering which starts from the last page. The last page is "-1". This has been added to allow filtering of last page or the before last page, which is "-2". It is used to apply specific test on a document cover.	—	—	—

Table 10.90: Pages Properties (Sheet 2 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
BoxToBoxDifference	element	The rectangle from calculating the differences between two rectangles: @*FromBox* and @*ToBox*. The calculation is made using the following formula: @*FromBox* (left)–@*ToBox* (left), @*FromBox* (bottom)–@*ToBox* (bottom), @*ToBox* (right)–@*FromBox* (right), @*ToBox* (top)–@*FromBox* (top). To define the two boxes used, options are given in ***BoxToBoxDifference*** argument. See ▸ Table 8.197 BoxToBoxDifference Element.	—	—	—

Note that ***BoxToBoxDifference*** Element is always a Subelement of a ***PreflightArgument***.

Example 10.12: Test with BoxToBoxDifference Element

```
<Device Class="Implementation" ID="Link0003" Status="Available">
<DeviceCap>
<TestPool>
<Test ID="PT01">
<RectangleEvaluation ValueList="0 0 10 10">
<BasicPreflightTest Name="BoxToBoxDifference">
<PreflightArgument>
<BoxToBoxDifference FromBox="TrimBox"
ToBox="BleedBox"/>
</PreflightArgument>
</BasicPreflightTest>
</RectangleEvaluation>
</Test>
</TestPool>
</DeviceCap>
</Device>
```

10.3.2.13 PDLObject Properties

The PDL object is used to check whether select objects are defined or not defined in the document, but does not check anything else as these objects are specific to one given PDL.

Table 10.91: PDLObject Properties

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
PDLObjectType	NMTOKEN	The type of specific PDL object. **Values include:** AcroForm – The PDF AcroForm. Actions – The PDF Actions. Bookmarks – The PDF Bookmarks. JavaScript – The PDF JavaScript. Thread – The PDF Thread. Thumbnails – The PDF Thumbnails.	—	—	PDF

10.3.2.14 Reference Properties

Reference property Attributes describe objects that have links to external references on other objects. It only deals with OPI links and references in page to other graphical contents. This is not describing the Font Properties (see ▸ Section 10.3.2.7 Font Properties).

Table 10.92: Reference Properties

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
ExternalReferenceMissing	boolean	Is "true" when the target of an external reference is missing.	—	—	—
HasExternalReference	boolean	Is "true" when some of the page graphical contents have a link on files.	—	—	—
HasOPI	boolean	Is "true" if there is OPI information associated with the object.	—	—	PS, PDF
OPIMissing	boolean	Is "true" when the target of OPI comments associated with the object is missing.	—	—	PS, PDF
OPIType	NMTOKEN	The OPI type of OPI comments associated with the object. Sometimes in PS, the comments are not OPI comments. **Values include:** OPIComments OtherComments	—	—	PS, PDF
OPIVersion	NMTOKENS	The OPI versions of OPI comments associated with the object.	—	—	PS, PDF

10.3.2.15 Shading Properties

Shading property Attributes are derived from graphic objects with applied shading, which is usually defined as of either smooth or vector type.

Table 10.93: Shading Properties

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
ShadingType	enumeration	The type of shading. **Allowed values are:** Smooth Vector	—	—	—

10.3.2.16 Stroke Properties

Stroke property Attributes are linked with graphic objects with vector primitives. They can have a fill color and a stroke color with given colors. This is a list of Properties that specifically apply to this kind of object:

Table 10.94: Stroke Properties (Sheet 1 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
HasStrokeColor	boolean	Is "true" if the vector object is drawn with a stroke color.	—	—	—
StrokeAlternateColorSpace	enumeration	The alternate color space of the stroke of one object. **Allowed values are from**: *@ColorSpace* in ▸ Table 10.86 Graphic Properties.	—	Yes	PS, PDF
StrokeColorName	string	The name of the color of the stroke of the vector object.	—	—	—

Table 10.94: Stroke Properties (Sheet 2 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
StrokeColorSpace	enumeration	The color space of the stroke of one object. **Allowed values are from**: *@ColorSpace* in ▸ Table 10.86 Graphic Properties.	—	—	PS, PDF
StrokeColorType	enumeration	This is an enumeration of known colors used to draw stroke. **Allowed values are from**: *@FillColorType* in ▸ Table 10.82 Fill Properties.	—	—	—
StrokeOverprintFlag	boolean	Is "true" when the stroke of one object has been set to overprint.	—	—	—
StrokeShadingType	enumeration	The type of shading used in the stroke. **Allowed values are:** Smooth Vector	—	—	—
StrokeThickness	double	The thickness of the stroke of the vector object.	—	—	—

10.3.2.17 Text Properties

"Text" refers to a consecutive set of one or more characters that share the same style (i.e., font, size, fill, stroke, etc.). The following are the Attributes that can be applied to text:

Table 10.95: Text Properties (Sheet 1 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
CharacterProblem	enumeration	Problem encountered to render character. **Allowed values are:** Corrupted – Used when a character was found but could not be rendered. IncorrectEncoding – Used when encoding information is missing, incomplete or otherwise incorrect. Missing – Use when the character could not be found in font. Others – Used in all other cases.	—	—	—
MissingPrinterFont	boolean	Is "true" if a referenced font has no printer information.	Yes	—	—
MissingScreenFont	boolean	Is "true" if a referenced font has no screen information.	—	—	—
TextSize	double	The size in points of the character.	—	—	—

Table 10.95: Text Properties (Sheet 2 of 2)

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
UseArtificialTextEffect	enumerations	The artificial text effects list used to draw a character. **Allowed values are:** Bold Italic Outline Shadow Underline **Note:** The authoring applications can apply the text effect directly, whereas in PS or PDF, the effect will be calculated.	—	—	—

10.3.2.18 Vector Properties

Vector property Attributes are derived from graphic objects with vector primitives. They can have a fill color and a stroke color, with given colors. This is a list of Attributes that specifically apply to this kind of object:

Table 10.96: Vector Properties

NAME	TYPE	DESCRIPTION	SET	TAG	DOCUMENTS
NumberOfPathPoints	integer	The number of points used to create a vector path.	—	—	—

Building a System

11.1 Implementation Considerations and Guidelines

JDF parsing. JDF Devices SHALL implement **JDF** parsing. At a minimum, a Device SHALL be able to search the **JDF** to find a Node whose Process type it is able to execute. The details of the search algorithm are implementation dependent and can be as simple as searching only in the **JDF** Root Node. In addition, a ▸ Device SHALL be able to consume the inputs and produce the outputs for each process type it is able to execute. See ▸ Section 4.2.1 Determining Executable Nodes.

Test run. To reduce failures during processing, it is RECOMMENDED that either individual Devices or their Controller support the test-run functionality. This prevents the case where a Device begins processing a Node that is incomplete or malformed.

11.2 JDF and JMF Interchange Protocol

A system of vendor-independent elements SHOULD define a protocol that allows them to interchange information based on **JDF** and **JMF**.

Controllers and devices SHOULD provide insecure http without an SSL layer for better interoperability. Controllers and devices.

11.2.1 File-Based Protocol (JDF)

The file-based protocol is a solution for **JDF** Job tickets. A file-based protocol MAY be based on hot folders. A Device that implements hot holders SHALL define an input hot folder and an output folder for **JDF**. In addition, the ***SubmitQueueEntry*** Message contains a URL Attribute that allows specification of arbitrary **JDF** locators. Implementation of **JDF** file-based protocol is simple, but it is important to note that the protocol does not support acknowledgement receipts for protocol error handling. It requires that the receiver polls the output folder of the processor. Finally, granting read/write access to your hot folder negates the security functions.

11.2.2 HTTP-Based Protocol (JDF + JMF)

HTTP ▸ [RFC2616] is a stable, vendor-independent protocol, and it supports a variety of advantageous features. For example, it offers a wide availability of tools. It is already a common technology among vendors who use HTTP, and it has a well defined query-response mechanism (HTTP post message). It also offers widespread firewall support and secure connections via SSL (see ▸ [SSL3]) when using HTTPS.

11.2.2.1 Protocol Implementation Details

JDF Messaging does not specify a standard port.

Implementation of Messages

Only HTTP servers SHALL be targeted by ***Query*** Messages, ***Registration*** Messages or ***Command*** Messages. This is done with a standard HTTP Post request. The **JMF** is the body of the HTTP post message. The ***Response*** Message is the body of the response to the initiated HTTP post. ***Signal*** and ***Acknowledge*** Messages are also implemented as HTTP post messages. The body of the HTTP response to these Messages MAY be empty.

If reliable Signaling (see ▸ Section 5.2.3 Signal) is implemented, the ***Response*** to a ***Signal*** SHALL NOT be empty.

HTTP Push Mechanisms

Since HTTP is a stateless protocol, push mechanisms, such as regular status bar updates, are non-trivial when communicating with a client. Workarounds can, however, be implemented. For example, a client application that polls the server in regular intervals MAY be used.

11.2.3 HTTPS-Based Protocol – SSL with two-way authentication

New in JDF 1.3

11.2.3.1 Purpose

The addition of support for the HTTPS Protocol for use in **JMF** systems from **JDF** Spec version 1.3 onwards is not so much about Encryption as about Authentication. Customers of **JMF** based system have a need to be able to exchange Messages securely between systems in their facility without fear of intervention from outside sources or from malicious acts. The solution needs to be able to sustain authentication without having to exchange username and password on every call, is

platform and implementation language independent and is capable of working across firewalls (though configuration of firewalls might be required in an individual installation).

Support for **JMF** over HTTPS does not require the implementation of any additional **JMF** messages, though the ***RequestForAuthentication*** Message (which is new in 1.4) may be used to exchange certificates and establish a secure connection.

On a web server, the server provides its certificate to you. The client decides whether to accept communication. With two-way authentication client authentication is required.

11.2.3.2 Certificates

JMF over HTTPS requires both parties to provide exchange and validate certificates. The certificates SHALL contain the core four fields of the X.509 format and the UserID. Any additional fields are OPTIONAL. These fields are:

- Common Name (Abbreviation CN) (i.e., hostname which could be an IP address or DNS name by which the receiver knows the sender),
- Organization Unit (Abbreviation OU)
- Organization (Abbreviation O)
- Country (Abbreviation C)
- UserID (Abbreviation UID) - this SHALL be the SenderID that messages from the sender will be identified by. This would be the client's SenderID for commands, queries, and signals, and the server's SenderID for responses and acknowledges
- givenName? - The vendor name, product name, and any other information about the product MAY be optionally included in the certificate using the givenName certificate field.

Example for XYZ Software's XYZImpose product:

```
CN=impose7.printinginc.internal OU=Prepress O=Printing, Inc.
C=US UID=XYZImpose7 givenName=XYZ Software XYZImpose v7.0
```

More information can be found at http://www.rsasecurity.com/rsalabs/node.asp?id=2307

Certificates can be generated by any certificate generation tool such as Sun Keytool. See ▸ Section 11.2.3.5.2 Example of Java Keytool Usage.

The certificates should be self-signed to remove the need to access third-party Certificate Authorities.

11.2.3.2.1 Verification of Certificates

Certificates should be verified against the hostname of the machine. Therefore certificates should reference the machine and may need to be generated on site.

Note: The difference between the hostname and the IP address is that if the IP address changes, this will effectively revoke the certificate. However, if the hostname is used, then the name SHALL be resolvable by the receiver using either DNS or local name resolution.

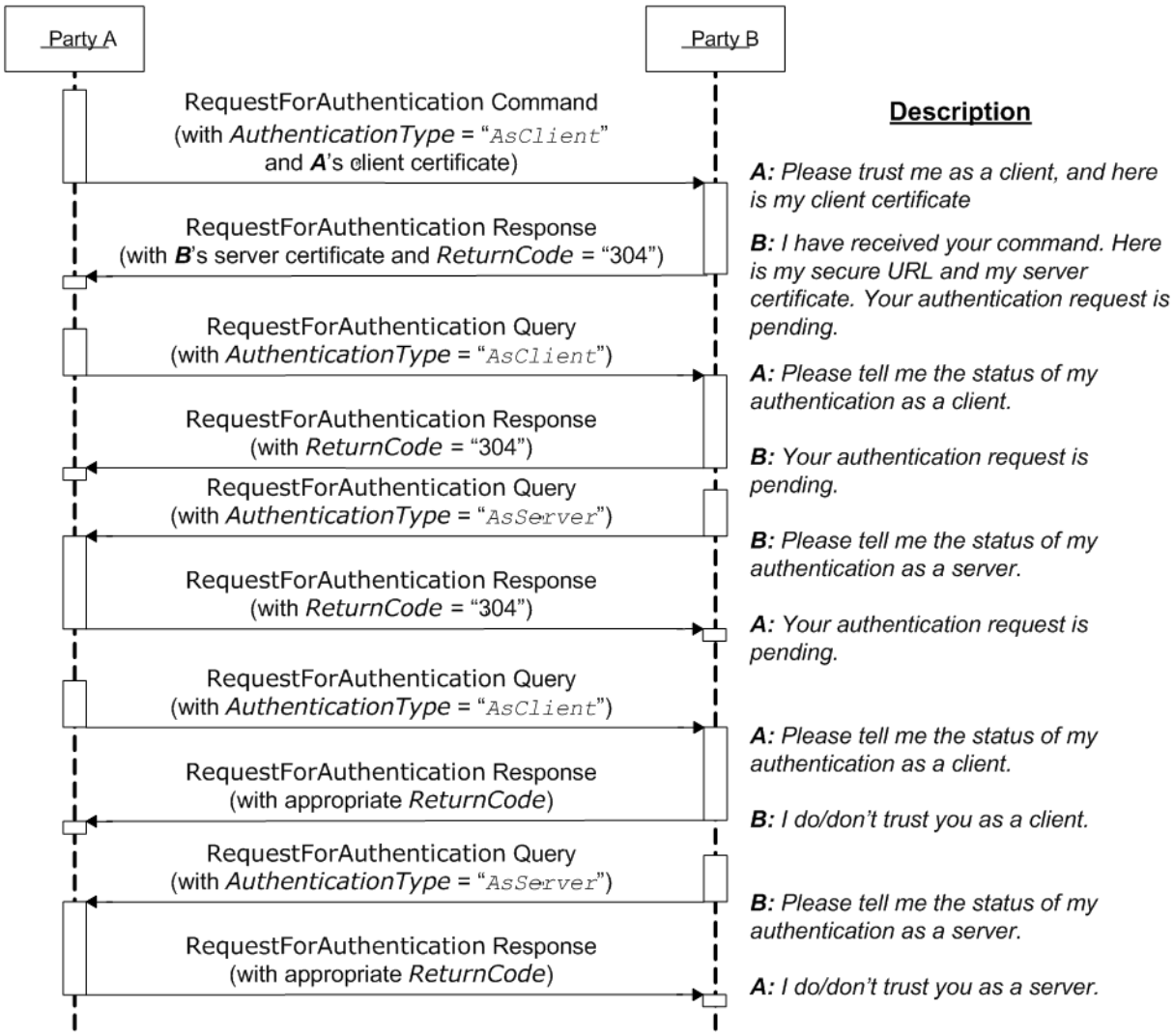
11.2.3.3 Exchange of Certificates

Certificates may be exchanged and authenticated by the following sequence which makes use of the ***RequestForAuthentication*** Message.

The ***RequestForAuthentication*** Message includes a requirement that the recipient return its appropriate certificate on receipt of the sender's certificate, based on the value of the *@AuthenticationType* Attribute.

The likely sequence of events between two parties, A and B, can be summarized as follows:

Figure 11-1: Example of Exchange of Certificates



We now have 2 way authentication in one direction with *A* as the client, and *B* as the server. To complete the other direction, there are two possibilities:

- The Process is repeated with *B* sending a ***RequestForAuthentication*** to *A* using the same steps.
- *A* to initiate the same steps, but sets the @*AuthenticationType* Attribute to "AsServer", and provides its Secure URL in the ***AuthenticationCmdParams*** Element.

If the certificate received by *A* in the response from *B* is bad, then *B*'s trust of *A* SHALL be manually deleted. Then *A* can repeat the above steps.

If the certificate received by *A* at some later time goes bad, then *A* can repeat the above steps over a secure channel, with the @*Reason* Attribute set appropriately to indicate a problem. Effectively it is saying "I'm serving you notice that your certificate is bad; send me a new one". *B*'s response will be to present a certificate that should be different to the one previously sent.

If party *B* realizes that it needs to re-issue it's server certificate, it MAY send a ***RequestForAuthentication*** Command to party *A*'s secure URL, with the @*AuthenticationType* Attribute set to "AsServer". *A* should then respond appropriately.

Reconnection: if certificates have been exchanged, but the secure URL has been lost, reconnection can be facilitated by sending a ***KnownDevices*** Query to the system whose URL has been lost. If the signed certificate has been lost, then the existing trust relationships SHALL be manually deleted, then a repeat of the above steps.

11.2.3.4 Standards

See ▸ [SSL3] and ▸ [X.509].

11.2.3.5 Implementation

If a client communicates with a server over an HTTPS connection and at some point the client receives a "permission denied" HTTP response, this indicated that the secure connection has been revoked and that the client needs to resubmit the ***RequestForAuthentication*** message.

11.2.3.5.1 Discovery Messages

The ***KnownDevices*** Message has been extended so that ***Device*** Resource has a new Attribute @*SecureJMFURL*.

The ***KnownMessages*** Message has been extended to indicate which Messages are supported under which protocols, by adding the @*URLSchemes* Attribute to the ***MessageService*** Element.

11.2.3.5.2 Example of Java Keytool Usage

A command line example of using the Java keytool:

1 Use Java keytool to generate a public/private key pair and wrap the public key into an X.509 v1 self-signed certificate. The private key and certificate are stored in a JKS key store.

```
keytool -genkey -alias impose7 -dname "CN=xyzimpose7.myCompany.internal
OU=Prepress O=Printing, Inc. C=US UID=XYZImpose7
givenName=XYZ Software XYZImpose" -validity 365 -keystore keystore.jks
```

2 Export the self-signed certificate to the base64 encoded PEM format:

```
keytool -export -keystore keystore.jks -rfc -alias impose7 -file impose7.cer
```

For full documentation, see http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/keytool.html

11.2.4 Managing Persistent Channels.

A Controller MAY request information about currently active subscriptions by sending a ***KnownSubscriptions*** Query to a Device. A Controller SHOULD NOT send a new ***Subscription*** if a matching ***Subscription*** is already in place in the Device. If the device does not support ***KnownSubscriptions*** Query, the Controller MAY create a new ***Subscription***. A Device that receives a ***Subscription*** of the same type to the same URL SHOULD replace the existing ***Subscription*** with the new ***Subscription***

A Controller SHOULD remove persistent channels that are no longer evaluated by sending a ***StopPersistentChannel*** Command to a device.

Persistent channels SHOULD be maintained even when a device is powered off and powered on again.

11.2.5 Deleting Persistent Channels

A persistent channel SHALL be deleted by sending a ***StopPersistentChannel*** Command Message, as described in ▸ Section 5.56 StopPersistentChannel.

11.3 JDF Packaging

New in JDF 1.2

JDF messaging supports combining into a single package:

- the **JMF** message,
- the **JDF** job ticket(s) to which it refers, and
- the digital assets to which the **JDF** job tickets refer.

The following external data file types are identified, although any valid MIME file type MAY be referenced:

- Preview images (They SHOULD be encoded using the PNG format.)
- ICC Profiles
- Preflight Profiles
- PDL (Page Description Language)

Currently MIME Multipart/Related packaging is supported. ▸ [RFC2387]

All packaging methods use a consistent design pattern. The package contains one or more parts and there SHALL be at least one **JDF** or **JMF** part. If a **JMF** part is included there SHALL be only one. If the packaging has ordered parts (Multipart/Related) the **JMF** part SHALL be first. The **JDF** parts SHALL follow the **JMF** part (if present) and any other parts follow the **JDF** parts.

When the content parts of a **JDF** Package are extracted, the ***QueueSubmissionParams*** (at a provided URL) or ***ResubmissionParams*** (at a provided URL) within the **JMF** Message and ***FileSpec*** (at a provided URL) within the **JDF** ticket(s) SHALL be updated with the URL at which the referenced items are stored.

11.3.1 MIME Basics

MIME (Multipurpose Internet Mail Extensions) ▸ [RFC2045] is an Internet standard that defines mechanisms for specifying and describing the format of Internet message bodies. MIME is comprised of headers and content. In case of Multipart messages, the content consists of multiple body parts, each with its own MIME headers and content. A unique boundary string precedes each body part and follows the last one.

11.3.2 MIME Types and File Extensions

The following MIME types and extensions SHOULD be used when storing **JDF** or **JMF** as files or when a MIME type is required, e.g.when setting the http Content-Type header.:

Table 11.1: MIME Types and File Extensions

MIME TYPE	EXTENSION	USAGE
application/ vnd.cip4-jdf+xml	jdf	unpackaged **JDF**.
application/ vnd.cip4-jmf+xml	jmf	unpackaged **JMF**.

It is RECOMMENDED that the Controller use a file extension of "jdf" when using file-based **JDF** in an environment that supports file name extensions. Agents that serialize **JMF** to a file SHOULD use a file extension of "jmf".

When a MIME package containing **JDF** or **JMF** is serialized to a file, it is RECOMMENDED to use the "mjd" file extension for packages where a **JDF** is the first entity. It is RECOMMENDED to use the "mjm" file extension when a **JMF** Message is the first package. CIP4 will also register a mime type for CIP3 ppf: application/vnd.cip3-ppf. It is RECOMMENDED that the Controller use a file extension of "ppf" when writing CIP3 ppf files.

11.3.2.1 MIME Headers

New in JDF 1.2

This section defines the normative extensions when using MIME to package **JMF** or **JDF**.

11.3.2.1.1 Content-Type Header

This MIME header is REQUIRED for an individual **JDF** or **JMF**, the root, and the individual bodyparts of a MIME Multipart/Related package. "Content-Type" identifies the MIME type of the message or body part). The "Content-Type" header can identify a message as a MIME Multipart message and each body part also has a "Content-Type" header to identify its content. The following "Content-Type" are used with **JDF**:

Table 11.2: MIME Content-Types

MIME TYPE	DESCRIPTION
`application/vnd.cip4-jdf+xml`	A **JDF** File. The root XML element SHALL be **JDF** .
`application/vnd.cip4-jmf+xml`	A **JMF** File. The root XML element SHALL be **JMF** .
Multipart/Related	A package of a **JDF** or **JMF** file + optional additional referenced data ▸ [RFC2387]. The root XML element of the first bodypart SHALL be **JDF** or **JMF** .

11.3.2.1.2 Content-ID Header

This field is REQUIRED for every body part that is referenced from another body part in a Multipart/Related message. "Content-ID" identifies each different body part within a MIME Multipart message. Its value SHALL be an Email address as long as it is defined using US-ASCII. Each value of "Content-ID" SHALL be unique within the message, but it need not be a working Email address. Thus "Content-ID" can be a somewhat random sequence and need not be related to the original filename. It is good practice to limit yourself to using only alphanumeric characters or only the first 127 characters of the US-ASCII character set in order to avoid confusing less intelligent MIME Agents.

11.3.2.1.3 Content-Transfer-Encoding

This field is OPTIONAL. ▸ [RFC2045]. It defines the following different encodings:

- "7bit"
- "quoted-printable"
- "base64"
- "8bit": This specifies that no additional encoding is applied to the data. Use "8bit" if the **JDF** stream contains CR or LF separators (e.g., for body parts containing **JDF** or **JMF**).
- "binary": This specifies that no additional encoding is applied to the data. Use "binary" if there is no CR or LF separators in the stream (e.g., for body parts containing JPEG).

Private encodings MAY be defined and begin with the prefix "X-". When no encoding is used, the data are only encapsulated by MIME headers. "base64" and "quoted-printable" encodings are commonly used algorithms for converting eight-

bit and binary data into seven-bit data and vice versa. Consumers that support MIME SHOULD support "8bit" and "binary" and SHALL support "base64". The other encodings are OPTIONAL.

It is RECOMMENDED to also specify the encoding for the **JDF/JMF** parts of a Multipart/Related package.

11.3.2.1.4 Content-Disposition Header

This field is OPTIONAL. See ▸[RFC2231] It allows a filename to be specified for a body part. The "Disposition-Type" SHALL be set to "attachment".

The Disposition filename parameter contains a suggested file name for storing the attachment. This file name MAY be the original file name when creating the MIME file and can be visible to the operator. Note that the filename is a value that needs special MIME encoding rules, these are ▸[RFC2822]and ▸[RFC2231].

It is RECOMMENDED to use quoted-strings for file names with only US-ASCII characters see ▸[RFC2822] and ▸[RFC2231]for file names with non-USASCII characters.

Example for ▸[RFC2822]:

A name = "Cover page.pdf" becomes:

```
Content-Disposition: attachment; filename="Cover page.pdf";
```

Example for ▸[RFC2231]:

A name = "Dollar€_1.pdf" becomes:

```
Content-Disposition: Attachment; filename*=UTF-8''Dollar%E2%82%AC_1.pdf;
```

Example 11.1: Packaging of Individual JDF/JMF files in MIME

New in JDF 1.2

The following example displays MIME packaging of a **JDF** file as an individual MIME object:

```
MIME-Version: 1.0
Content-Type: multipart/related; boundary=abcdefg0123456789
Content-Transfer-Encoding: 8bit
--abcdefg0123456789
Content-Type: application/vnd.cip4-jdf+xml
<JDF ... >
   <PreviewImage Separation = "PANTONE 128" URL="cid:123456.png" />
</JDF>
--abcdefg0123456789--
```

11.3.2.2 CID URL Scheme

New in JDF 1.2

One of the benefits of the MIME Multipart/Related *@MediaType* is the ability of a URL in one body part to refer to the content of another body part. This is done by using a "cid" scheme in a URL, specified in ▸[RFC2392]. Please look at the example to see how it is used.

Example 11.2: CID URL Scheme

```
MIME-Version: 1.0
Content-Type: multipart/related; boundary=abcdefg0123456789
Content-Transfer-Encoding: 8bit
--abcdefg0123456789
Content-Type: application/vnd.cip4-jdf+xml

<JDF ... >
   <PreviewImage Separation="PANTONE 128" URL="cid:123456.png@cip4.org" />
</JDF>

--abcdefg0123456789
Content-Type: image/png
Content-Transfer-Encoding: base64
Content-ID: <123456.png@cip4.org>

BASE64DATA
BASE64DATA

--abcdefg0123456789--
```

Note: ▸[RFC2392] *requires* that the value of the Content-ID be enclosed in angle brackets (<>). Also the characters that ▸[RFC2392] allows in Content-ID include characters that ▸[RFC3986] does not permit in URLs; any such character (such as "+" or "&") SHALL be hex-encoded using the %hh escape mechanism in the URL (see ▸[RFC3986]). Therefore,

matching the cid URL with the Content-ID SHALL take account of the escaped equivalencies. Case-insensitive matching SHALL be used.

11.3.2.3 Ordering of Body Parts in MIME Multipart/Related

New in JDF 1.2

The first body part of the MIME Multipart message SHALL be the **JMF** Message. Internal links are defined using the cid URL and a corresponding Content-ID MIME header. Subsequent sections are the **JDF** Jobs followed by the linked entities, such as the preview images shown in the following example:

Example 11.3: MIME Multipart/Related

A Multipart/Related message is received that contains:

- Message.jmf
- Ticket01.jdf
- Pages.pdf

```
MIME-Version: 1.0
Content-Type: multipart/related; boundary=unique-boundary

--unique-boundary
Content-Type: application/vnd.cip4-jmf+xml
Content-Transfer-Encoding: 8bit
...
<?xml version="1.0" encoding="UTF-8"?>
<JMF SenderID="JMFClient" TimeStamp="2016-07-07T13:15:56+01:00"
  Version="1.4" xmlns="http://www.CIP4.org/JDFSchema_1_1" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
  <Command ID="C0001" Type="SubmitQueueEntry" xsi:type="CommandSubmitQueueEntry">
    <QueueSubmissionParams Hold="true" URL="cid:JDF1@hostname.com"/>
  </Command>
</JMF>


--unique-boundary
Content-Type: application/vnd.cip4-jdf+xml
Content-Transfer-Encoding: 8bit
Content-ID: <JDF1@hostname.com>
Content-Disposition: attachment; filename="Ticket01.jdf";
```

```
<?xml version="1.0" encoding="UTF-8"?>
<JDF Activation="Active" ID="JDF_c" JobID="Job1" JobPartID="345"
  Status="Waiting" Type="Product" Version="1.4" xmlns="http://www.CIP4.org/JDFSchema_1_1">
  <JDF ID="JDF-3" JobPartID="400" Status="Waiting" Type="DigitalPrinting">
    <ResourceLinkPool>
      <DigitalPrintingParamsLink Usage="Input" rRef="ID123"/>
      <RunListLink Usage="Input" rRef="RunList4"/>
      <ComponentLink Amount="3" Usage="Output" rRef="ID125"/>
    </ResourceLinkPool>
    <ResourcePool>
      <DigitalPrintingParams Class="Parameter" ID="ID123" Status="Available"/>
    </ResourcePool>
  </JDF>
  <ResourceLinkPool>
    <ComponentLink Usage="Output" rRef="ID125"/>
  </ResourceLinkPool>
  <ResourcePool>
    <Component Class="Quantity" ComponentType="Sheet" ID="ID125" Status="Unavailable"/>
    <RunList Class="Parameter" DocCopies="1" FirstPage="0" ID="RunList4"
      IsPage="true" PageCopies="1" Status="Available">
      <LayoutElement ElementType="Document" HasBleeds="false"
        IgnorePDLCopies="true" IgnorePDLImposition="true" IsPrintable="true">
        <FileSpec AppOS="Windows" Compression="None"
          URL="cid:Asset01@hostname.com" UserFileName="Christmas Cards"/>
      </LayoutElement>
    </RunList>
  </ResourcePool>
</JDF>


--unique-boundary
Content-type: application/pdf
Content-ID: <Asset01@hostname.com>
Content-Transfer-Encoding: binary
Content-Disposition: attachment; filename="Pages 1.pdf";

The pdf goes in here.
--unique-boundary--
```

When such a stream arrives at the server, it is decoded and the parts stored locally either in memory or persistent storage. The contents of the stream are extracted. The designer of the Controller chose to save package contents into a uniquely named directory.

- Assets are saved first — Pages.pdf is placed in /root/temp/a39e9503-a96b-4e86-9c1d-f4188d19810e/Assets/
- The Controller then internally maps cid:Asset01@hostname.com in the ticket into file:///root/temp/a39e9503-a96b-4e86-9c1d-f4188d19810e/Assets/Pages.pdf.
- Then Ticket01.jdf is placed in a directory /root/temp/a39e9503-a96b-4e86-9c1d-f4188d19810e/
- The Controller then internally maps cid:JDF1@hostname.com in the message into file:///root/temp/a39e9503-a96b-4e86-9c1d-f4188d19810e/Ticket01.jdf and either executes or stores the Message.

11.4 MIS Requirements

MIS systems MAY:

- Ignore ***Audit*** Elements if they receive complete information about a Process execution via **JMF**.
- Decompose **JDF** into an internal format such as database tables.

.

11.5 Interoperability Conformance Specifications

Interoperability Conformance Specifications (i.e., ICS documents) are developed by CIP4 working committees. They establish the minimum **JDF** support requirements for Devices of a common class, including expected behavior. An ICS document can subset **JDF** but can not expand upon **JDF**. For instance, an ICS that covers desktop printers can either omit or prohibit all of the postpress Processes related to case binding. ICS documents can also establish minimum **JDF** support requirements for a class of Devices.

Once published, ICS documents will form the basis for testing and certification by CIP4-sanctioned facilities. **JDF**-enabled products that pass these tests will be deemed "JDF Certified" to conform to an identified level of one or more ICS

documents and will be permitted upon certification to use a "JDF Certified" logo in connection with certified **JDF**-enabled products.

The development of ICS documents are done in parallel, but not in synchronization, with the development of editions of the **JDF** specification (e.g., an ICS is related to a specific edition of the **JDF** Specification, but might be released at a later date). Once approved, all published ICS documents will be available at http://www.cip4.org/document_archive/ics.php.

Appendix A

Data Types and Values

This appendix lists a number of commonly used **JDF** data types and structures and their XML encoding. Data types are simple data entities such as strings, numbers (as doubles) and dates. They have a very straightforward string representation and are used as XML Attribute Values. Data structures, on the other hand, describe more complex structures that are built from the defined data types, such as colors.

A.1 Notes About Encoding

All of the **JDF** types are derived from XML Schema types either by extension, use of lists or by restriction. Each type will refer back, either directly or indirectly, to such a type and reference ought to be made to "XML Schema Part 2 – Datatypes" ▸ [XMLSchema].

A.1.1 List, Range and Range List Data Types

Some data types are derived from a base type that represents a single value. Such data types include a list, a range and a range list. For a data type *X*, the name of such data types are *X*List, *X*Range and *X*RangeList, respectively. Each data type represents a set of values of the base data type. A list is an enumerated set of values, which is expressed as a list of space separated values. A range is a continuous inclusive range of values, which is expressed as a pair of values separated by a '~' character. A range list is a set of values that includes range values and may also include individual values. A range list is expressed as a list of space separated ranges and individual values. Some data types with a range and range list data types do not have a list data type. In this case, the range list may allow only range values.

A.1.2 Whitespace

The addition of whitespace characters for single types is NOT RECOMMENDED. Items in a list of values are separated by whitespace. A range consists of two items separated by a '~'; although not mandatory (to maintain compatibility with **JDF** 1.1), it is strongly RECOMMENDED that whitespace is used between the items and the '~'.

Note: The **JDF** 1.2 schema will only correctly validate ranges if whitespace is used around the '~'.

A.1.3 Infinity Limits

Several types require the ability to set an unbounded range, or to select a single terminating value (e.g., Integer or date ranges). These types have been extended with the tokens "-INF" or "INF" to indicate the maximum negative and positive limits of the values in question, details are shown where appropriate for each value.

A.2 Simple Types — Attribute ValuesXJDF Data Types

A.2.1 boolean

Has the value space for supporting the mathematical concept of binary-valued logic:

Encoding

Values of type boolean are encoded as either of the string values "true" or "false". The XML Schema data type boolean values of "1" or "0" are not permitted.

Example A.1: boolean

```
<Example Enable="true"/>
```

A.2.2 CMYKColor

XML attributes of type CMYKColor are used to specify CMYK colors.

Encoding

Values of type CMYKColor attributes are primitive data types and are encoded as a list of four numbers (as doubles) in the range of [0...1.0] separated by whitespace. A value of 0.0 specifies no ink and a value of 1.0 specifies full ink. The sequence of colors is "C M Y K".

Example A.2: CMYKColor

```
<Color cmyk = "0.3 0.6 0.8 0.1"/><!--brick red-->
```

A.2.3 date

A calendar date, it represents a time period that starts at midnight on a specified day and lasts for 24 hours. Based on ▸[ISO8601:2004].

Encoding

It is represented identically to the XML Schema type: *date*.

Example A.3: date

```
<Example StartDate="1999-05-31"/>
```

A.2.4 dateTime

Represents a specific instant of time. It SHALL be a Coordinated Universal Time (UTC) or the time zone SHALL be indicated by the offset to UTC. In other words, the time SHALL be unique in all time zones around the world. It also allows infinity limits to allow for explicit 'don't care' values (i.e., it SHALL be finished before 'anytime').

Encoding

Values of type dateTime are represented as a union of the XML Schema type: *dateTime* and the infinity value tokens *INF* and *-INF*.

Note: That ▸[ISO8601:2004] allows a wider range of time zone specifications than XML. *dateTime* SHALL adhere to the stricter limitations defined in ▸[XMLSchema]. For instance the colon ':' in the time zone field SHALL be present when writing time zones in the format "hh:mm".

Example A.4: dateTime

```
<Example Start="1999-05-31T18:20:00Z"/>

<Example Start="1999-05-31T13:20:00-05:00"/>
```

A.2.5 DateTimeRange

New in JDF 1.2

XML Attributes of type DateTimeRange are used to describe a range of points in time. More specifically, it describes a time span that has an absolute start and end. Unbounded ranges can use the infinity value tokens *INF* and *-INF*

Encoding

A DateTimeRange is represented by two dateTime or infinity tokens separated by the whitespace "~" whitespace sequence.

Example A.5: DateTimeRange

```
<XXX range="1999-05-31T18:20:00Z ~ 1999-05-31T18:20:00Z"/>

<XXX range="1999-05-31T18:20:00Z ~ INF"/>

<XXX range="-INF ~ 1999-05-31T18:20:00Z"/>
```

A.2.6 DateTimeRangeList

New in JDF 1.2

XML Attributes of type DateTimeRangeList are used to describe a list of ranges of points in time. More specifically, it describes a list of time spans, which each have a relative start and end.

Encoding

A DateTimeRangeList is represented by sequence of either DateTimeRange values (See 1.5), separated by whitespace or dateTime values.

Example A.6: DateTimeRangeList

```
<XXX RangeList=
   "1999-05-31T18:20:00Z ~ 1999-05-31T18:20:00Z 1999-05-31T13:20:00-05:00 ~ INF"/>
```

A.2.7 double

Values of type double correspond to IEEE double-precision 64-bit floating point type. It includes the infinity limit tokens *INF* and *-INF*, but does not allow the not a number token *NaN*.

Encoding

It is represented similarly to the XML Schema type: *double*. However string value *NaN*, is not permitted.

Example A.7: double

```
<Example NegativePi="-3.14"/>
```

A.2.8 DoubleList

New in JDF 1.2

Values of type DoubleList are used to describe a variable length list of numbers (as doubles). This type is used as the base for other **JDF** types that use a fixed length list of number (e.g., CMYKColor which is restricted to four number in the list).

Encoding

A DoubleList is encoded as a string of whitespace-separated double values as defined in ▸ Section A.2.7 double.

Example A.8: DoubleList

```
<XXX list="3.14 1 .6"/>
```

A.2.9 DoubleRange

New in JDF 1.2

XML Attributes of type DoubleRange are used to describe a range of numbers (as doubles). Mathematically spoken, the two numbers define a closed interval.

Encoding

A DoubleRange is represented by two double values separated by a "~" (tilde) character and OPTIONAL additional whitespace.
Note: It is now RECOMMENDED that the '~' is surrounded by whitespace to aid validation and parsing.

Example A.9: DoubleRange

```
<XXX range="-3.14 ~ 5.13"/>

<XXX range="0 ~ INF"/>
```

A.2.10 DoubleRangeList

New in JDF 1.2

XML Attributes of type DoubleRangeList are used to describe a list of DoubleRange values and/or enumerated numbers as doubles).

Encoding

A DoubleRangeList is a sequence of DoubleRange values and single double values separated by whitespace.

Example A.10: DoubleRangeList

```
<XXX list="-1 ~ -6 3.14 ~ 5.13 7 9 ~ 128 131 255 ~ INF"/>
```

A.2.11 duration

Values of type duration represent a period of time. Based on ▸ [ISO8601:2004]. The single infinity limit token *INF* is permitted.

Encoding

It is represented as a union of the XML Schema type: "duration" and the string value "INF"
Note: That ▸[XMLSchema] explicitly allows negative durations. Thus a value of -PT15M is valid and describes a negative duration of 15 minutes in the past.

Example A.11: duration

```
<Example Duration= "P1Y2M3DT10H30M"/>
```

A.2.12 DurationRange

XML Attributes of type DurationRange are used to describe a range of time durations. More specifically, it describes a time span that has a relative start and end.

Encoding

A DurationRange is represented by two duration values, separated by the "~" (tilde) space character and optional additional whitespace.
Note: It is now RECOMMENDED that the '~' is surrounded by whitespace to aid validation and parsing.

Example A.12: DurationRange

```
<XXX range="P1Y2M3DT10H30M ~ P1Y2M3DT10H35M"/>

<XXX range="P1Y2M3DT10H30M ~ INF"/>
```

A.2.13 DurationRangeList

New in JDF 1.2

XML Attributes of type DurationRangeList are used to describe a list of ranges of time durations. More specifically, it describes a list of time spans that have a relative start and end.

Encoding:

A DurationRangeList is represented by sequence of DurationRange values and durations, separated by whitespace.

Example A.13: DurationRangeList

```
<XXX RangeList="P1Y2M3DT10H30M ~ P1Y2M3DT10H35M P1Y3M2DT10H30M"/>
```

A.2.14 gYearMonth

Represents a specific Gregorian month in a specific Gregorian year. Based on ▸[ISO8601:2004].

Encoding

It is represented identically to the XML Schema type: *gYearMonth*

Example A.14: gYearMonth

```
<Example Month="2002-11"/>
```

A.2.15 hexBinary

Values of type hexBinary represents arbitrary hex encoded binary data.

Encoding

It is represented identically to the XML Schema type: *hexBinary*

Example A.15: hexBinary

```
<Example Hex="0A1C"/>
```

A.2.16 ID

Modified in JDF 1.3

Represents the @*ID* Attribute from ▸[XMLSchema]. It represents a name or string that contains no space characters and starts with a letter, or '_'. Each ID value SHALL be unique within a **JDF** document and thus uniquely identify the elements that bear them.
Note: that the @*ID* Attribute definition in ▸[XMLSchema] is more restrictive than the @*ID* Attribute definition in ▸[XML]. ▸[XMLSchema] explicitly forbids the use of ':' in ID.

Encoding

It is represented identically to the XML Schema type: *ID*

Example A.16: ID

```
<Example ID="R-16"/>
```

A.2.17 IDREF

IDREF Represents the IDREF Attribute from ▸[XMLSchema]. For a valid XML-document, an element with the ID value specified in IDREF SHALL be present in the scope of the document.

Encoding

It is represented identically to the XML Schema type: *IDREF*

Example A.17: IDREF

```
<Example IDREF="R-16"/>
```

A.2.18 IDREFS

IDREFS Represents the IDREFS Attribute from ▸[XMLSchema]. More specifically, this is a whitespace-separated list of IDREF values.

Encoding

It is represented identically to the XML Schema type: *IDREFS*

Example A.18: IDREFS

```
<Example IDREFS="R-12 R-16"/>
```

A.2.19 integer

Represents numerical integer values with tokens for representing infinity limits.
Implementation note: Except where explicitly noted otherwise, integers are not expected to exceed a value that can be represented as signed 32 bits.

Encoding

It is represented as a union of the XML Schema type: *integer* and the infinity value tokens *INF* and *-INF*

Example A.19: integer

```
<Example Copies="36"/>
```

A.2.20 IntegerList

XML Attributes of type IntegerList are used to describe a variable length list of integer values.

Encoding

An IntegerList is encoded as a string of integers separated by whitespace.

Example A.20: IntegerList

```
<XXX list="-INF 0 1 2 3 4 INF 1 3 0"/>
```

A.2.21 IntegerRange

XML Attributes of type IntegerRange are used to describe a range of integers. In some cases, ranges are defined for an unknown number of objects. In these cases, a negative value denotes a number counted from the end. For example, -1 is the last object, -2 the second to last and so on. IntegerRanges that follow this convention are marked in the respective Attribute descriptions.

If the first element of an IntegerRange specifies an element that is behind the second element, the Range specifies a list of integers in reverse order, counting backwards. For example "6 ~ 4" = "6 5 4" and "-1 ~ 0" = "last... 2 1 0".

Encoding

An IntegerRange is represented by two integers, separated by a "~" (tilde) character and optional additional whitespace.
Note: It is now RECOMMENDED that the '~' is surrounded by whitespace to aid validation and parsing.

Example A.21: IntegerRange

```
<RunList ID="RL1" Class="Parameter" Status="Available" Pages="-3 -5"/>
<RunList ID="RL2" Class="Parameter" Status="Available" Pages="INF -5"/>
```

A.2.22 IntegerRangeList

XML Attributes of type IntegerRangeList are used to describe a list of IntegerRanges and/or enumerated integers.

Encoding

An IntegerRangeList is represented by a sequence of IntegerRanges and integers, separated by whitespace.

Example A.22: IntegerRangeList

```
<XXX list="-1 ~ -6 3 ~ 5 7 9 ~ 128 131"/>
```

A.2.23 LabColor

Values of type LabColor are used to specify absolute Lab colors. The Lab values are normalized to a Light of D50 and an angle of 2 degrees as specified in ▸ [CIE 15:2004] and ▸ [ISO13655:1996].

This corresponds to a white point of X = 0.9642, Y = 1.0000 and Z = 0.8249 in CIEXYZ color space. The value of L is restricted to a range of [0..100]; a and b are unbounded.

Encoding

LabColors are primitive data types and are encoded as a list of three numbers (as doubles) separated by whitespace in the sequence: "L a b"

Example A.23: LabColor

```
<Color Lab="51.9 12.6 -18.9"/>
```

A.2.24 language

Values of type language represent a natural language defined in ▸ [RFC1766].

Encoding

It is represented identically to the XML Schema type: *language*

Example A.24: language

```
<Example Language="de"/> <!-- German -->
<Example Language="de-CH"/> <!-- Swiss German -->
<Example Language="en"/> <!-- English -->
<Example Language="en-GB"/> <!-- British English -->
```

A.2.25 languages

New in JDF 1.4

Values of type languages represent a list of natural languages, each defined in ▸ [RFC1766].

Encoding

A languages value is encoded as a string of languages, each language separated by whitespace.

Example A.25: languages

```
<Example Languages ="de-CH de en-GB en"/>
```

A.2.26 matrix

Coordinate transformation matrices are widely used throughout the whole printing Process, especially in ***Layout*** Resources. They represent two dimensional transformations as defined by [PS] and [PDF1.6]. For more information, refer to the respective reference manuals, and look for "Coordinate Systems and Transformations." The "identity matrix", which is "1 0 0 1 0 0", is often used as a default throughout this specification. When another matrix is factored against a matrix with the identity matrix value, the result is that the original matrix remains unchanged.

Encoding

Coordinate transformation matrices are primitive data types and are encoded as a list of six numbers (as doubles), separated by whitespace: "a b c d Tx Ty". The variables *Tx* and *Ty* describe distances and are defined in points.

Example A.26: matrix

```
<ContentObject CTM="1 0 0 1 3.14 21631.3" />
```

A.2.27 NameRange

XML Attributes of type NameRange are used to describe a range of NMTOKEN data that are acquired from a list of named elements, such as named pages in a PDL file. It depends on the ordering of the targeted list, which names are assumed to be included in the NameRange. The following two possibilities exist:

1. There is no explicit ordering. In this case, case sensitive alphabetical ordering [Unicode5.0] is implied. This behavior is the default unless called out explicitly in the specification.
2. There is explicit ordering, such as in a list of named pages in a ***RunList***. In this case, the ordering of the ***RunList*** defines the order and all pages between the end pages are included in the NameRange.

Modification note: Starting with **JDF** 1.4, the first item is specified as the default behavior.

Encoding

A NameRange typed Attribute is represented by two NMTOKEN values separated by a "~" (tilde) character and optional additional whitespace.
Note: It is now RECOMMENDED that the '~' is surrounded by whitespace to aid validation and parsing.

Example A.27: NameRange

```
<XXX NameRange="Jack ~ Jill"/>
```

A.2.28 NameRangeList

XML Attributes of type NameRangeList are used to describe a list of NameRanges.

Encoding

A NameRangeList is represented by a sequence of NameRanges and NMTOKEN, separated by whitespace.

Example A.28: NameRangeList

```
<XXX list="A brian ~ fred x z"/>
```

A.2.29 NMTOKEN

Values of type NMTOKEN represent a name or string that contains no space characters.
Note: NMTOKEN values MAY begin with any non whitespace character, including numerical characters.

Encoding

It is represented identically to the XML Schema type: *NMTOKEN*.

Example A.29: NMTOKEN

```
<Example Alias="ABC_6"/>
```

A.2.30 NMTOKENS

Represents the NMTOKENS attribute type from ▸ [XML]. More specifically, this is a whitespace-separated list of NMTOKEN values.

Encoding

It is represented identically to the XML Schema type: *NMTOKENS*

Example A.30: NMTOKENS

```
<Example AliasList="ABC_6 ABCD_3 DEGF"/>
```

A.2.31 PDFPath

Modified in JDF 1.3

Values of type PDFPath are used in **JDF** for describing parameters such as trap zones and clip paths. In PJTF, PDFPaths are encoded as a series of **moveto-lineto** operations. **JDF** has a different encoding, which is able to describe more complex paths, such as Bezier curves. The non-zero winding rule is used to fill closed paths.

Encoding

PDFPaths are encoded by restricting an XML *string* attribute formatted with PDF path operators. This allows for easy adoption in PS and PDF workflows. PDF operators are limited to those described in "Path Construction Operators" in ▸ [PDF1.6].

Example A.31: PDFPath

```
<ElementWithPath path="0 0 m 10 10 l 20 20 l"/>
```

A.2.32 rectangle

Values of type rectangle are used to describe rectangular locations on the page, sheet or other printable surface. A rectangle is represented as a list of four numbers — llx lly urx ury — specifying the lower-left x, lower-left y, upper-right x and upper-right y coordinates of the rectangle, in that order. This is equivalent to the ordering: Left Bottom Right Top. All numbers are defined in points.

Encoding

To maintain compatibility with PJTF, rectangles are primitive data types and are encoded as a string of four *numbers*, separated by whitespace: "llx lly urx ury" or "l b r t".

Example A.32: rectangle

```
<ContentObject ClipBox="0 0 3.14 21631.3" />
```

Implementation Remark

Since all numbers are real numbers, any comparison of boxes SHOULD take into account certain rounding errors. For example, different XYPair values MAY be considered equal when all numbers are the same within a range of 1 point.

A.2.33 RectangleRange

New in JDF 1.2

XML Attributes of type RectangleRange are used to describe a range of rectangles.

Encoding

A RectangleRange is represented by one or two Rectangles, separated by a "~" (tilde) character and optional additional whitespace.
Note: It is now RECOMMENDED that the '~' is surrounded by whitespace to aid validation and parsing.

Example A.33: RectangleRange

```
<XXX range="1 2 3 4 ~ 5 6 7 8"/>
<XXX range="-INF -INF 3 4 ~ 0 1 INF INF"/>
```

A.2.34 RectangleRangeList

New in JDF 1.2

XML Attributes of type RectangleRangeList are used to describe a list of rectangle ranges.

Encoding

A RectangleRangeList is represented by sequence of RectangleRange values and Rectangle values, separated by whitespace.

Example A.34: RectangleRangeList

```
<XXX RectangleRangeList="1 2 3 4 ~ 5 6 7 8 9 10 11 12 13 14 15 16"/>
```

A.2.35 regExp

Values of type regExp represent a regular expression as defined in ▸ [XMLSchema].

Encoding

It is represented identically to the XML Schema type: *normalizedString*

Example A.35: regExp

```
<Example expression="Foo({1|2}*)"/>
```

A.2.36 shape

Values of type shape are used to describe a three dimensional box.

Encoding

A shape is represented as an array of three (positive or zero) *numbers* — x y z — specifying the Width x, height y and depth z coordinates of the shape, in that order.

Example A.36: shape

```
<XXX Dimensions="10 20 40"/>
```

A.2.37 ShapeRange

XML Attributes of type ShapeRange are used to describe a range of shapes (three dimensional boxes). The range "x1 y1 z1 ~ x2 y2 z2" describes the area x1 <= x <= x2 and y1 <= y <= y2 and z1 <= z <= z2. Thus the shape "2 3 4" is within "1 2 1 ~ 3 4 4".

Note that this implies that all three values of the second entry SHALL be >= the corresponding values of the first entry. The following example is therefore invalid: "1 2 1 ~ 0 4 4".

Encoding

A ShapeRange is represented by two shapes, separated by a "~" (tilde) character and optional additional whitespace. Note: It is now RECOMMENDED that the '~' is surrounded by whitespace to aid validation and parsing.

Example A.37: ShapeRange

```
<XXX Shaperange="1 2 3 ~ 4 5 6"/>
<XXX Shaperange="1 2 3 ~ 4 INF 6"/>
```

A.2.38 ShapeRangeList

XML Attributes of type ShapeRangeList are used to describe a list of ShapeRange and/or shapes.

Encoding

A ShapeRangeList is a sequence of ShapeRange and shapes separated by whitespace.

Example A.38: ShapeRangeList

The brackets below the example illustrate the grouping of shapes and ShapeRange values.

```
<XXX Shapelist="100 200 300 ~ 110 220 330  150 300 150  2 3 0 ~ 3 4 5"/>
               [                          ][           ][             ]
```

A.2.39 sRGBColor

XML Attributes of type sRGBColors are used to specify sRGB colors.

Encoding

sRGBColors are primitive data types and are encoded as a string of three numbers in the range of [0...1.0] separated by whitespace. A value of 0 specifies no intensity (black) and a value of 1 specifies full intensity. The sequence is defined as: "r g b"

Example A.39: sRGBColor

```
<Color sRGB="0.3 0.6 0.8" />
```

A.2.40 string

Values of type string represents sequences of characters.

Encoding

It is represented identically to the XML Schema type: *normalisedString*.

Example A.40: string

```
<Example Name="Test With Space"/>
```

A.2.41 TimeRange

Deprecated in JDF 1.2

A.2.42 TransferFunction

Values of type TransferFunction are functions that have a one-dimensional input and output. In **JDF** , they are encoded as a simple kind of sampled functions and used to describe transfer curves of image transfer processes from one medium to the next (e.g., film to plate, or plate to press).

A transfer curve consists of a series of XY pairs where each pair consist of the stimuli (X) and the resulting value (Y). To calculate the result of a certain stimuli, the following algorithms SHALL be applied:

1. If x < = first stimuli, then the result is the y value of the first xy pair.
2. If x > = the last stimuli, then the result is the y value of the last xy pair.
3. Search the interval in which x is located.
4. Return the linear interpolated value of x within that interval.

Encoding

A TransferCurve is encoded as a string of space-separated *numbers* (as doubles). The numbers are the XY pairs that build up the transfer curve. Note that the end points of a TransferFunction SHALL be explicitly specified and are NOT defaulted to "0 0" or "1 1".

Example A.41: TransferFunction

```
<someElementWithTransferCurve someCurve="0 0 .1 .2 .5 .6 .8 .9 1 1"/>
```

A.2.43 URI

Modified in JDF 1.3

Values of type URI represent a Uniform Resource Identifier (URI) Reference as defined in ▸ [RFC3986]. In **JDF** 1.3 and above, the URI data typed is represented as an Internationalized Resource Identifier (IRI) as defined in ▸ [RFC3987].

Encoding

A URI is represented identically to the XML Schema type: *anyURI*.

Example A.42: URI

```
<Example URI="http://www.w3.org/1999/XMLSchema"/>
```

A.2.44 URL

Short for URL-reference. Represents a Uniform Resource Locator (URL) Reference as defined in ▸ [RFC3986]. In **JDF** 1.3 and above, the URL data typed is represented as an Internationalized Resource Identifier (IRI) as defined in ▸ [RFC3987].

Encoding

A URL is represented identically to the XML Schema type: *anyURI*.
Note: Some characters in a URL SHALL be escaped and all characters MAY be escaped by encoding their UTF-8 representation into a '%' followed by the double digit hex representation of the character. The list of characters that SHALL be encoded is dependent on the URL scheme. Non-escaped characters SHALL be encoded in the encoding of the containing **JDF** document.

Example A.43: URL

New in JDF 1.4

A UNC path to be displayed as a URL:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Example URL="\\\\myHost\\a\\c äöü%.pdf"/>
```

Example A.44: URL: UTF-8

New in JDF 1.4

The UNC path encoded as an IRL with internationalized characters in UTF-8:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Example URL="file://myHost/a/c%20äöü%25.pdf"/>
```

Example A.45: URL: Windows Locale 1252

New in JDF 1.4

The same UNC path encoded as an IRL with internationalized characters in UTF-8 viewed in a Windows locale 1252:

```
<Example URL="file://myHost/a/c%20Ã¤Ã¶Ã¼%25.pdf"/>
```

Example A.46: URL: Escaped Characters

New in JDF 1.4

The same UNC path encoded as an IRL with internationalized characters escaped:

```
<Example URL="file://myHost/a/c%20%c3%a4%c3%b6%c3%bc%25.pdf"/>
```

A.2.45 XPath

New in JDF 1.2

Values of type XPath represent an XPath expression as described in ▸ [XPath]

Encoding

It is represented identically to the XML Schema type: *token*

Example A.47: XPath

```
<Example xpath= "JDF/AuditPool/Created/@TimeStamp" />
```

A.2.46 XYPair

Values of type XYPair are used to describe sizes, e.g. @*Dimensions* and @*StartPosition*. They can also be used to describe positions on a page. All numbers that describe lengths are defined in points.

Encoding

XYPair attributes are primitive data types and are encoded as a string of two *numbers*, separated by whitespace: "x y"

Example A.48: XYPair

```
<CutBlock BlockSize="612 792"/>
```

Implementation Remark

Since all numbers are real numbers, comparison of XYPair values SHOULD take into account certain rounding errors. For example, different XYPair values MAY be considered equal when all numbers are the same within a range of 1 point.

A.2.47 XYPairRange

XML Attributes of type XYPairRange are used to describe a range of XYPair values. The range "x1 y1 ~ x2 y2" describes the area x1 <= x <= x2 and y1 <= y <= y2. Thus the XYPair "2 3" is within "1 2 ~ 3 4". Note that this implies that both values of the second entry SHALL be >= the corresponding values of the first entry. The following example is therefore invalid: "1 2 ~ 0 4".

Encoding

An XYPairRange is represented by two XYPair values, separated by a "~" (tilde) character and optional additional whitespace.
Note: It is now RECOMMENDED that the '~' is surrounded by whitespace to aid validation and parsing.

Example A.49: XYPairRange

```
<XXX XYrange="1 2 ~ 3 4"/>
<XXX XYrange="-INF 2 ~ 3 INF"/>
```

A.2.48 XYPairRangeList

XML Attributes of type XYPairRangeList are used to describe a list of XYPairRange and/or XYPair values.

Encoding

A XYPairRangeList is a sequence of XYPairRange and XYPair values separated by whitespace.

Example A.50: XYPairRangeList

The brackets below the example illustrate the grouping of XYPair values and XYPairRange values.

```
<XXX XYlist="100 200 ~ 110 220  150 300  150 350  200 300 ~ INF INF"/>
           [                 ][       ][       ][                ]
```

A.3 Enumerations

This section contains tables each with a closed set of values for an enumeration/enumerations type. If there are any implications to the order of the values this will be detailed in the description, otherwise no order is implied.

The values in the closed sets are encoded as a restriction of xsd:NMTOKEN.

A.3.1 Action

Attributes with a data type Action, determine what action if any to take as a result of a particular event.

Table A.1: Action Enumeration Values

ENUMERATION VALUE	COMMENT
Abort	Abort the ongoing activity and do not proceed with any other further avtivity.
Continue	Continue with the present activity. Details SHOULD be logged.
Repair	Repair the condition before proceeding with the activity. Details SHOULD be logged. **Note:** The actions required to perform the repair are system specified.

A.3.2 Anchor

New in JDF 1.4

Attributes with a data type of Anchor describe the 9 anchor points of a rectangle.

Table A.2: Anchor Enumeration Values

ENUMERATION VALUE	COMMENT
TopLeft	
TopCenter	
TopRight	
CenterLeft	
Center	
CenterRight	
BottomLeft	
BottomCenter	
BottomRight	

A.3.3 Automation

Automation determines how complete an item is.

Table A.3: Automation Enumeration Values

ENUMERATION VALUE	COMMENT
Dynamic	The item is incomplete and should be completed automatically.
Static	The item is complete.

A.3.4 Axis

Axis determines the notional line around which an operation, such as mirroring, will be performed.

Table A.4: Axis Enumeration Values

ENUMERATION VALUE	COMMENT
Both	The operation is performed around both axes.
FeedDirection	The operation is performed around the feed direction axis.
MediaWidth	The operation is performed around the medis width axis.
None	No operation is to be performed.

A.3.5 BinderMaterial

Attributes of this type define the material to be used for loose binding.

Table A.5: BinderMaterial Enumeration Values

ENUMERATION VALUES	COMMENT
ColorCoatedSteel	Coated steel.
Plastic	Any kind of plastic.
Steel	Plain steel

A.3.6 ChannelMode

Attributes of this type define the reliability mode of a message channel.

Table A.6: ChannelMode Enumeration Values

ENUMERATION VALUES	COMMENT
FireAndForget	The receiver of the signal MAY respond using a **JMF** response message.
Reliable	Indicates that the signal is the result of a subscription where reliable signaling was specified. The receiver of the signal SHALL respond using a **JMF** response message.

A.3.7 Coating

Coating specifies the coating of a substrate.

Table A.7: Coating Enumeration Values

ENUMERATION VALUE	COMMENT
Coated	A coating of a system specified type.
HighGloss	A high gloss coating.
InkJet New in JDF 1.2 Deprecated in JDF 1.4	A coating intended for use with inkjet technology.
Glossy	A glossy coating.
Matte	A matte coating.
None	No coating.
Satin	A coating between gloss and matte.
Semigloss	A semi gloss coating.

A.3.8 Compensation

Attributes of type Compensation define how a process is to apply transfer curve compensation..

Table A.8: Compensation Enumeration Values

ENUMERATION VALUES	COMMENT
Film	Compensated until film exposure.
None	No Compensation.
Plate	Compensated until plate exposure.
Press	Compensated until press.
Unknown Deprecated in JDF 1.2	

A.3.9 Drying

Attributes with a datatype of Drying specify the method employed to dry an item.

Table A.9: Drying Enumeration Values

ENUMERATION VALUE	COMMENT
Heatset	Heatset dryer.
IR	Infrared dryer.
Off	No dryer is used.
On	The device's default drying unit is used.

Table A.9: Drying Enumeration Values

ENUMERATION VALUE	COMMENT
UV	Ultraviolet dryer.

A.3.10 Edge

Attributes of this type select the edge of an object.

Table A.10: Edge Enumeration Values

ENUMERATION VALUES	COMMENT
Bottom	Bottom edge of a sheet or product.
Left	Left edge of a sheet or product.
Right	Right edge of a sheet or product.
Top	Top edge of a sheet or product.

A.3.11 EmbossDirection

Attributes of this type determine the type and direction of embossing.

Table A.11: EmbossDirection Enumeration Values

ENUMERATION VALUES	COMMENT
Both	Both debossing and embossing using one stamp.
Depressed	Debossing only.
Flat New in JDF 1.3	The embossing foil is applied flat. Used for foil stamping.
Raised	Embossing only.

A.3.12 EmbossLevel

Attributes of this type determine the profile of the embossing.

Table A.12: EmbossDirection Enumeration Values

ENUMERATION VALUES	COMMENT
MultiLevel	
Sculpted	
SingleLevel	

A.3.13 EmbossType

Attributes of this type determine the type of embossing requored.

Table A.13: EmbossType Enumeration Values

ENUMERATION VALUES	COMMENT
BlindEmbossing	Embossed forms are not inked or foiled. The color of the image is the same as the substrate.
Braille New in JDF 1.3 (Errata)	Six dot braille embossing.
FoilEmbossing	Combines embossing and foil stamping in a single operation.
FoilStamping	Uses a heated die to place a metallic or pigmented image from coated foil on to the substrate.

Table A.13: EmbossType Enumeration Values

ENUMERATION VALUES	COMMENT
RegisteredEmbossing	Creates an embossed image that is exactly registered to a printed image.

A.3.14 FeedQuality

Attributes of this type define how the quality of a supply of object are to be evaluated.

Table A.14: FeedQuality Enumeration Values

ENUMERATION VALUES	COMMENT
Check	Check the quality and register.
NotActive	Quality control is not active.
StopNoWaste	Check the quality and register. The consuming device will stop after the predefined number of consecutive errors. The error MUST be corrected e.g. manually.
StopWaste	Check the quality and register. The object failing the test is wasteThe consuming device will stop after the predefined number of consecutive errors. The error MUST be corrected e.g. manually.
Waste	

A.3.15 FitPolicy

Attributes of this type define how an object should be manipulated to enable it to fit into a given area.
Note: The 'given direction' in the following text is derived from the attributes context, i.e. for *@HorizontalFitPolicy* this would be horizontal.

Table A.15: FitPolicy Enumeration Values

ENUMERATION VALUES	COMMENT
NoRepeat	The object is niether resized nor repeated. If it is bigger that the given area then it is clipped.
RepeatToFill	The object is placed in the requested position. It is then repeated in the given direction, allowing clipping to occur, until all the allocated space is filled.
RepeatUnclipped	The object is placed in the requested position. It is then repeated in the given direction, without clipping, to fill as much of the allocated space as possible.
StretchToFit	The object is stretched along the given direction to entirely fill the allocated space. **Note:** If used in isolation this can result in distortion of the object's aspect ratio.
UndistortedScaleToFit	The object is resized to fit in the given direction. **Note:** For the orthogonal direction this may result in either the object being clipped or the object not filling the allocated space.

A.3.16 GangPolicy

Attributes of this type define how multiple jobs are to be ganged.

Table A.16: GangPolicy Enumeration Values

ENUMERATION VALUES	COMMENT
Gang	The job SHALL be ganged and MAY be submitted to the device.
GangAndForce	The job SHALL be ganged and SHALL be submitted to the device.
NoGang	The job SHALL NOT be ganged.

A.3.17 Glue

Attributes of this type define how multiple jobs are to be ganged .

Table A.17: Glue Enumeration Values

ENUMERATION VALUES	COMMENT
ColdGlue	
Hotmelt	
PUR	Polyurethane rubber.

A.3.18 IncludeResources

Attributes of this type define how fonts are to be embedded.

Table A.18: IncludeResources Enumeration Values

ENUMERATION VALUES	COMMENT
IncludeNever	Never embed fonts.
IncludeOncePerDoc	Embed once per document.
IncludeOncePerPage	Embed one per page.

A.3.19 ISOPaperSubstrate

Attributes with a data type of ISOPaperSubstrate describe a print substrate according to [ISO12647-2:2013].
Note: See Section E.3 Paper Grade for a mapping to the paper grade values defined in [ISO12647-2:2004].

Table A.19: ISOPaperSubstrate Enumeration Values

ENUMERATION VALUE	COMMENT
PS1	Premium Coated
PS2	Improved Coated
PS3	Standard Coated Glossy
PS4	Standard Coated Matte
PS5	Wood-free Uncoated
PS6	Super Calendered
PS7	Improved Uncoated
PS8	Standard Uncoated

A.3.20 JDFJMFVersion

Describes the schema version of a **JDF** or **JMF** instance.

Table A.20: JDFJMFVersion Enumeration Values

ENUMERATION VALUE	COMMENT
1.1	JDF 1.1
1.2	JDF 1.2
1.3	JDF 1.3
1.4	JDF 1.4
1.5	JDF 1.5

A.3.21 MappingSelection

Attributes with a data type of MappingSelection determine how a device should construct a color.

Table A.21: MappingSelection Enumeration Values

ENUMERATION VALUE	COMMENT
UsePDLValues	Use color values specified in the PDL. See ▸ [ColorPS]
UseLocalPrinterValues	Use the device's best local mapping.
UseProcessColorValues	Use the values define in the associated process.

A.3.22 NamedColor

Colors of preprocessed products such as Wire-O binders and cover leaflets. The entries in the following table MAY be prefixed by either "Dark" or "Light". The result MAY additionally be prefixed by "Clear" to indicate translucent material. For example, "ClearDarkBlue" indicates a translucent dark blue, "ClearBlue" a translucent blue and "Blue" indicates an opaque blue.

Table A.22: NamedColor Enumeration Values

COLOR NAME/ ENUMERATION VALUE	COMMENT	COLOR NAME/ ENUMERATION VALUE	COMMENT
Black	—	MultiColor New in JDF 1.1	
Blue	—	Mustard New in JDF 1.1	
Brown	—	NoColor	—
Buff	—	Orange	—
Cyan New in JDF 1.2		Pink	—
Gold	—	Red	—
Goldenrod	—	Silver	—
Gray	—	Turquoise	—
Green	—	Violet	—
Ivory	—	White	—
Magenta New in JDF 1.2		Yellow	—

A.3.23 Opacity

Attribute of this type describe the opacity of a resource..

Table A.23: Opacity Enumeration Values

ENUMERATION VALUE	COMMENT
Opaque	The media or resource is opaque and does not transmit light under normal incident lighting conditions.
Translucent New in JDF 1.2	The media or resource is translucent to a system specified degree. For example, translucent material can be used for back lit viewing.
Transparent	The media is transparent to a system specified degree.

A.3.24 Orientation

Orientation of a ***PhysicalResource***. For details see ▸ Table 2.4 Matrices and Orientation values for describing the orientation of a Component.

Table A.24: Orientation Enumeration Values

ENUMERATION VALUE	COMMENT
Rotate0	
Rotate90	
Rotate180	
Rotate270	
Flip0	
Flip90	
Flip180	
Flip270	

A.3.25 Polarity

Attributes with a datatype of Polarity describe whether a given image is color inverted.

Table A.25: Polarity Enumeration Values

ENUMERATION VALUE	COMMENT
Negative	The image is color-inverted.
Positive	The image is not color-inverted

A.3.26 PositionPolicy

Attributes with a datatype of PositionPolicy specify the level of freedom when applying placement or positioning values.

Table A.26: Policy Enumeration Values

ENUMERATION VALUE	COMMENT
Exact	The values SHALL be followed precisely.
Free	The values are used as guidance and MAY be modified by the designer.

A.3.27 RenderingIntent

Identifies the rendering intent that SHALL be applied when rendering the selected object. Values are defined in ▸ [ICC.1], File Format for Color Profiles.

Table A.27: RenderingIntent Enumeration Values

ENUMERATION VALUE	COMMENT
Saturation	
Perceptual	
RelativeColorimetric	
AbsoluteColorimetric	
ColorSpaceDependent Modified in JDF 1.3	The rendering intent is dependent on the color space. The dependencies are implementation specific.

A.3.28 Scope

Attributes of type define the availability of features or resources in a device.

Table A.28: Scope Enumeration Values

ENUMERATION VALUES	COMMENT
Allowed	The feature is potentially available but currently not available without operator intervention.
Job	The feature is currently available within the scope of a job.
Present	The feature is currently available without operator intervention.

A.3.29 Severity

Attributes of this type define the severity of an error.

Note: This table is not ordered alphabetically - it is ordered by increasing level of severity.

Table A.29: Severity Enumeration Values

ENUMERATION VALUES	COMMENT
Event	Normal operating event.
Information	Informational event worthy of being logged.
Warning	A minor error. The executing device is able to repair the condition and continue.
Error	A significant error. Operator intervention is required to allow the device to continue.
Fatal	A fatal error. The device has aborted the operation and cannot continue.

A.3.30 SheetLay

SheetLay specifies the reference edge where media or components are placed in a device.

Table A.30: SheeLay Enumeration Values

ENUMERATION VALUE	COMMENT
Center	The media is placed in the center. This is most commonly used in web devices.
Left	The media is placed so that it is guided on the left.
Right	The media is placed so that it is guided on the right.

A.3.31 Side

Attributes of type Side define which side is to be used for an action.

Table A.31: Side Enumeration Values

ENUMERATION VALUES	COMMENT
Back	The back surface.
Front	The front surface.

A.3.32 Sides

Sides defines the sides of the media or product that SHALL be imaged.

Table A.32: Sides Enumeration Values

ENUMERATION VALUE	COMMENT
OneSided	Page contents SHALL be imposed on the front side.
OneSidedBack	Page contents SHALL be imposed on the back side.

Table A.32: Sides Enumeration Values

ENUMERATION VALUE	COMMENT
TwoSidedHeadToFoot	Page contents SHALL be imposed on the front and back sides of media sheets so that the head (top) of the front backs up to the foot (bottom) of the back.
TwoSidedHeadToHead	Page contents SHALL be imposed on the front and back sides so that the head (top) of page contents backs up to each other.

A.3.33 SourceObjects

Attributes of this type identify the class of a graphical object. Multiple tokens specify that the action that is filtered by ▸ SourceObjects applies to all of the listed classes.

Table A.33: SourceObjects Enumeration Values

ENUMERATION VALUES	COMMENT
All Deprecated in JDF1.6	All types are allowed.
ImagePhotographic	Contone images.
ImageScreenShot	Images largely comprised of rasterized vector art.
LineArt	Vector objects other than text.
SmoothShades	Gradients and blends.
Text	Text objects.

A.3.34 StripMaterial

Attributes of type StripMaterial define the material to be used for a resource or process.

Table A.34: StripMaterial Enumeration Values

ENUMERATION VALUES	COMMENT
Calico	
Cardboard	
CrepePaper	
Gauze	
Paper	
PaperlinedMules	
Tape	

A.3.35 TightBacking

Attributes of this type define the required geometry for the back of a book block.
Note: This table is not ordered alphabetically – it is ordered by the pressure required; lowest first.

Table A.35: TightBacking Enumeration Values

ENUMERATION VALUES	COMMENT
Round	Rounding way.
RoundBacked	Rounding way, backing way.
Flat	A flat backing – no tight backing is applied.
FlatBacked	Backing way.

A.3.36 Usage

Attributes of this type define how a resource is used.

Table A.36: Usage Enumeration Values

ENUMERATION VALUES	COMMENT
Input	The resource is used as an input.
Output	The resource is used as an output.

A.3.37 WorkingDirection

Attributes of this type define the direction of an action or of the application of a resource.

Table A.37: WorkingDirection Enumeration Values

ENUMERATION VALUES	COMMENT
Bottom	From below.
Top	From above.

A.3.38 WorkStyle

Workstyle is defined in the press coordinate system, where the sheet moves parallel to the Y axis. In the simple case of a single unrotated page per surface this implies that a flip around the Y-axis (WorkAndTurn, WorkAndBack) will result in head to head images for the the back side, whereas a flip around the X-axis (WorkAndTumble) will result in a head to foot images.

Table A.38: WorkStyle Enumeration Values

ENUMERATION VALUE	COMMENT
Simplex	No turning.
Perfecting	Perfecting specifies the use of a device dependent perfecting module. Many offset sheet fed printing presses have perfecting cylinder(s) built in. The leading edge of the print sheet changes as the sheet is turned by the perfecting cylinder, but the side lays remain unaltered. In this regard, this @*WorkStyle* is similar to "WorkAndTumble", but "Perfecting" is an in-line operation during the press run. Therefore, an additional plate (set) is needed during this press run. The perfecting modules of digital presses are not standardized and therefore perfecting can be implemented either as "WorkAndTurn" or "WorkAndBack".
WorkAndBack	This @*WorkStyle* describes the printing on both sides of the substrate with a different plate (set) in the second run. After the first run the side lays are altered but the front lays stay as they were. Lays can be turned by hand or using a pile reverser. Two plate sets are necessary for "WorkAndBack".
WorkAndTurn	"WorkAndTurn" refers to the turning of the first-run sheet for subsequent perfecting. The front lays remain unchanged but the side lays SHALL be altered. The alteration can be made by hand or using a pile turner. Turning happens after the first press run and the plate (set) is used again in the second press run, imaging the other sheet surface.
WorkAndTumble	The "WorkAndTumble" method is also used for perfecting. The leading edge of the print sheet changes as the sheet is turned, but the side lays remain unaltered. Tumbling happens after the first press run and the plate (set) may be used again in the second press run, imaging the other sheet surface. **Note:** "WorkAndTumble" is the most common @*WorkStyle* for perfecting, in which case two plate sets are required.
WorkAndTwist	Done between two press runs. The sheets are twisted 180 degrees before the second run is performed so that the front lay and the side lay both change. The surface to be imaged is the same at both runs. Each run prints only part of the surface. The plate (set) stays in the machine. This @*WorkStyle* is used for saving plate or film material. It is no longer a common @*WorkStyle*.

A.3.39 XYRelation

New in JDF 1.2

XML Attributes of type XYRelation define the relationship between two ordered numbers.

Table A.39: XYRelation Enumeration Values

ENUMERATION VALUE	COMMENT
gt	X > Y
ge	X >= Y
eq	X = Y
le	X <= Y
lt	X < Y
ne	X != Y

A.4 Preferred String and NMTOKEN Values

This section contains the preferred values for items of type string or NMTOKEN. Although these types are open lists the values in these tables SHOULD be used where possible.

A.4.1 Comb and Coil Shape Names

When specifying the shape of a comb for CombBiding, values from the following table are recommended.

Table A.40: CombShape Names

NAME	DESCRIPTION
Single	Each "tooth" is made with one wire
SingleCalendar	Each "tooth" is made with one wire and an extension for hanging the bound product is provided in the center.
Twin	The shape of each "tooth" is made with a double wire (e.g., *Wire-O*®).
TwinCalendar	The shape of each "tooth" is made with a double wire and an extension for hanging the bound product is provided in the center.

A.4.2 DeviceClass and ModuleType Supported Strings

CIP4 supports many device classes. The following values SHOULD be used when filling ***Device***/@*DeviceClass*.

The ***ModuleStatus*** element (see ▸ Table 5.106 ModuleStatus Element), the ***ModulePhase*** element (see ▸ Table 3.39 ModulePhase Element) and ***VarnishingParams*** (see ▸ Section 8.167 VarnishingParams) contain a @*ModuleType* attribute that defines individual modules within a machine. The following tables provide lists of individual ***Module***/@*ModuleType* values.

Table A.41: Device Class Values

DEVICECLASS NAME	DESCRIPTION
CaseMaker	
Cutter	
DieCutter	
EndsheetFeeder	
FilmSetter	
Folder	
Gatherer	
GathererBinder	
Hardcover	
HardcoverBookLine	

Table A.41: Device Class Values

DEVICECLASS NAME	DESCRIPTION
HolePuncher	
Inserter	
IntegratedDigitalPrinter	
Jacketer	
MultipleWebConventional-Press	
PerfectBinder	
PerfectBinderLine	
PlateSetter	
PrintDelivery	
PrintingPress	
Scanner	
SheetFedConventionalPress	
SheetFedDigitalPrinter	
SingleWebConventional-Press	
Stacker	
Stitcher	
ThreadSewer	
Trimmer	
WebDigitalprinter	
WideFormatPrinter	

Table A.42: ModuleType Attribute Values for Conventional Printing Devices

MODULETYPE	DESCRIPTION
CoatingModule	Unit for coatings, for example, full coating of varnish.
Delivery	Delivery module, unit for gathering the printed sheets.
Drier	Module for drying the previously printed color or varnish.
ExtensionModule	Unit for extending the distance between modules, for example to increase the distance between the last printing module and the delivery module.
Feeder	Feeder module, feeds the device with paper.
Imaging	Imaging module in a direct to plate machine.
Numbering	Numbering unit.
PerfectingModule	Unit for perfecting, reversing device.
PrintModule	Unit for printing a color. Describes one cylinder and one side.

Table A.43: ModuleType Attribute Values for Postpress

MODULETYPE	DESCRIPTION
BlockPreparer New in JDF 1.4	The block preparer prepares the book block for a hardcover book. See ▸ Section 6.5.2 BlockPreparation.
BoxFolder New in JDF 1.4	The box folder folds and glues blanks into folded boxes for packaging. See ▸ Section 6.5.3 BoxFolding.
CaseMaker New in JDF 1.4	The case maker produces the hard case for books. See ▸ Section 6.5.6 CaseMaking.
Caser New in JDF 1.4	The caser joins the hard cover book case and the book block. (**CasingIn**). See ▸ Section 6.5.7 CasingIn.
Chain New in JDF 1.2	Transport chain or conveyer to transport gathered / collected product.
EndSheetGluer New in JDF 1.4	The end sheet gluer merges the front-end sheet, the book block and the back-end sheet together. See ▸ Section 6.5.17 EndSheetGluing.
Feeder New in JDF 1.2	Feeder module, feeds the device with paper. See ▸ Section 6.5.18 Feeding.
Gluer New in JDF 1.4	The gluer applies glue to a component. See ▸ Section 6.5.21 Gluing.
HeadBandApplicator New in JDF 1.4	The head band applicator applies a head band to the book block. See ▸ Section 6.5.22 HeadBandApplication.
InkjetPrinter New in JDF 1.4	Prints images or texts on a component. (**Numbering**, **DigitalPrinting**)
Inserter New in JDF 1.4	The inserter inserts one or more “child” components to one “mother” component. See ▸ Section 6.5.24 Inserting.
Jacketer New in JDF 1.4	The jacketer wraps a jacket around a book. See ▸ Section 6.5.25 Jacketing.
PaperPath New in JDF 1.2	Paper path module, path that paper follows through the machine.
PressingStation New in JDF 1.4	The pressing station presses the cover to the book block.
ShapeCutter New in JDF 1.4	The shape cutter produces special shapes like an envelope window or a heart-shaped beer mat. Note that the shape cutter module may contain tools that correspond to the actual dies etc. See ▸ Section 6.5.36 ShapeCutting.
SpinePreparer New in JDF 1.4	The spine preparer prepares the spine of a book for hard and soft cover production. See ▸ Section 6.5.39 SpinePreparation.
SpineTaper New in JDF 1.4	The spine taper applies a tape strip to the spine of a book block. See ▸ Section 6.5.40 SpineTaping.
Strapper New in JDF 1.4	The strapper straps a bundle of products. See ▸ Section 6.5.44 Strapping.
ThreadSealer New in JDF 1.4	The thread sealer sews and seals a signature at the spine. See ▸ Section 6.5.46 ThreadSealing.

Table A.43: ModuleType Attribute Values for Postpress

MODULETYPE	DESCRIPTION
ThreadSewer New in JDF 1.4	The thread sewer sews all signatures of a book block together. See ▸ Section 6.5.47 ThreadSewing.

Table A.44: ModuleType Attribute Values for DigitalPrinting

MODULETYPE	DESCRIPTION
FarmPrinter New in JDF 1.3	Individual printer in a printer farm of printers.
Fuser New in JDF 1.2	Fuser module — fuses the toner onto the media.
Marker New in JDF 1.4	Marker module, excluding in-line finishing.
MimeUnpacker New in JDF 1.4	Module that receives and unpacks the MIME package and fetches the **JDF** if it is referenced from the **JMF** .
ReferencedDataCollector New in JDF 1.4	Module that fetches data referenced from the **JDF** and MAY include data referenced from the PDL. Does not include accepting MIME, unpacking MIME, or fetching the **JDF** itself.
RIP New in JDF 1.4	RIP module. See ▸ Section 6.3.33 RIPing.

Table A.45: ModuleType Attribute Values for PrintingUnitWebPath Modules of Web Printing Devices

MODULETYPE	DESCRIPTION
ChillUnit New in JDF 1.3	Chill unit that chills down the heated printed paper.
ImprintUnit New in JDF 1.3	Printing unit that allows changing plates during production run, doing imprints.
PrintUnit New in JDF 1.3	A print unit consists of multiple print module units.
Rollstand New in JDF 1.3	The roll stand feeds the web into the process-unit chain.
RemoisteningModule New in JDF 1.3	Module that can be used for high gloss varnish, remoistened glue, rub-off ink or encapsulated fragrances. The remoistening module is located between last printing unit and dryer.
UVCoater New in JDF 1.3	The UV-Coater module applies UV-varnish with subsequent drying in a UV-dryer.

Table A.46: ModuleType Attribute Values for FolderSuperstructureWebPath Modules of Web Printing Devices (Sheet 1 of 2)

MODULETYPE	DESCRIPTION
CrossCutter New in JDF 1.3	Cuts the web / ribbon n-times into sheets and transports the sheets to inline postpress-equipment
Delivery New in JDF 1.3	Delivers the printed and/or folded sheets out of the folder

Table A.46: ModuleType Attribute Values for FolderSuperstructureWebPath Modules of Web Printing Devices (Sheet 2 of 2)

MODULETYPE	DESCRIPTION
Folder New in JDF 1.3	Module for cutting the collected ribbons into sheets, in some cases collecting these sheets, and folding the sheets (quarter and cross folds)
Former New in JDF 1.3	Module for gathering ribbons and in most instances doing the first fold of the ribbons (quarter fold).
GluingAndSofteningModule New in JDF 1.3	Consists multiple heads, spread out in the press for gluing or/and softening of ribbons or folded sheets
MoebiusDeinfinitizer New in JDF 1.3	Used to resolve the infinite loops caused by printing on interleaving surfaces of Möbius banded webs.
PerforatingModule New in JDF 1.3	Module for doing cross, longitudinal or diagonal perforations and die cuts on a web. Module is placed between chill unit and folder.
PlanoModule New in JDF 1.3	The plano module cuts the web / ribbon into sheets and stacks the sheets to a pile
PloughFoldModule New in JDF 1.3	The plough fold module does a quarter fold to ribbons or webs, mostly found in front of a folder module
Rewinder New in JDF 1.3	Rewinds the printed web to a roll.
RibbonCompensator New in JDF 1.3	Controls the web / ribbons in running direction regarding the cross cut
Slitter New in JDF 1.3	Module for cutting in machine direction
Stitcher New in JDF 1.3	Stitches folded sheets together
Superstructure New in JDF 1.3	Module in which a web will be cut into ribbons and these will be moved to the correct position for folding.
TurnerBar New in JDF 1.3	Turns the front side of a web to the back side and vice versa.
TurnerBarUnit New in JDF 1.3	Turns the front side of a web to the back side and vice versa in a separate unit.

Table A.47: ModuleType Attribute Values for PostPressComponentPath Modules of Web Printing Devices (Sheet 1 of 2)

MODULETYPE	DESCRIPTION
BundlingModule New in JDF 1.3	The bundling module is used for bundling components
LabelingModule New in JDF 1.3	The labeling module is used for labelling a bundle.
PalletizingModule New in JDF 1.3	The palletizing module collects the bundles on a pallet. See ▸ Section 6.5.30 Palletizing.
PrintRoll New in JDF 1.3	The print roll is used for rolling components. See ▸ Section 6.5.33 PrintRolling.
Stacker New in JDF 1.3	Stacks the component to a pile. See ▸ Section 6.5.41 Stacking.

Table A.47: ModuleType Attribute Values for PostPressComponentPath Modules of Web Printing Devices (Sheet 2 of 2)

MODULETYPE	DESCRIPTION
Trimmer New in JDF 1.3	Trims the component to its final size. See ▸ Section 6.5.48 Trimming.

A.4.3 Input Tray and Output Bin Names

New in JDF 1.2

Location/@*LocationName* MAY be used to specify a location within a device (e.g., a paper tray). When specifying paper trays, the following locations are predefined. When specifying input paper trays (indicated with "I") and/or output bins (indicated with "O"), the following values for ***Location***/@*LocationName* locations are predefined . When specifying input tray names, the following values for ***Location***/@*LocationName* are suggested. The input tray names that specify a position (e.g., Top) are identified by an asterisk (*). These positional input tray names SHOULD NOT be used if devices are clustered because the position of the input tray might not be the same for all of the devices in the cluster. (See ▸ Section 3.10.6.4 Locations of PhysicalResources for more details on the use of ***Location***.)

Table A.48: Input Tray and Output Bin Names (Sheet 1 of 2)

NAME	I/O	DESCRIPTION
AnyLargeFormat	IO	The location that holds larger format media with one dimension larger than 11 inches. The media dimensions SHALL be specified. "AnyLargeFormat" is defined for a PPD.
AnySmallFormat	IO	The location that holds smaller format media. The media dimensions SHALL be specified. "AnySmallFormat" is defined for a PPD.
AutoSelect	IO	The location that the device selects based on the ***Media*** specification.
Back	IO*	The value "Rear" is analogous; "Rear" SHOULD be used instead when possible.
Booklet	O	The bin where the device places booklets.
Bottom	IO*	The bin that, when facing the device, can best be identified as 'bottom'.
BypassTray	I	The input tray used to handle odd or special papers. MAY be used to specify the input tray that is used for inserts sheets that are not to be imaged.
BypassTray-N	I	The input tray used to handle odd or special papers. MAY be used to specify the input tray that is used for inserts sheets that are not to be imaged. N = '1', '2', ...
Cassette	IO	The value "Tray-N" is analogous; "Tray-N" SHOULD be used instead when possible.
Center	—	The bin that, when facing the device, can best be identified as 'center.' Deprecated in JDF 1.2 — use Middle instead.
Continuous	IO	The location to handle continuous media (i.e., continuously connected sheets).
Disc	IO	The location to handle CD or DVD discs to be printed on.
Disc-N	IO	The location to handle CD or DVD discs to be printed on. N = '1', '2', ...
Envelope	IO	The location that is to contain envelopes.
Envelope-N	IO	The location that is to contain envelopes. N = '1', '2', ...
FaceDown	O	The bin that can best be identified as 'face down' with respect to the device.
FaceUp	O	The bin that can best be identified as 'face up' with respect to the device.
FitMedia	O	Requests the device to select a bin based on the size of the media.
Front	IO*	The location that, when facing the device, can best be identified as 'front.'
InsertTray	I	The input tray that can best be identified as 'insert tray.' Used to specify the input tray that is used for inserts sheets (insert sheets are never imaged).

Table A.48: Input Tray and Output Bin Names (Sheet 2 of 2)

NAME	I/O	DESCRIPTION
InsertTray-N	I	The input tray that can best be identified as 'insert tray-1', 'insert tray-2', ... etc. Used to specify the input tray that is used for inserts sheets (insert sheets are never imaged).
LargeCapacity	IO	The bin that can best be identified as the 'large capacity' bin (in terms of the number of sheets) with respect to the device.
LargeCapacity-N	IO	The location that can best be identified as the 'large capacity-1', 'large-capacity-2', ... etc., input tray (in terms of the number of sheets) with respect to the device.
Left	IO*	The bin that, when facing the device, can best be identified as 'left.'
Lower	IO*	The value "Bottom" is analogous; "Bottom" SHOULD be used instead when possible.
Mailbox-N	O	The job will be output to the bin that is best identified as "Mailbox #1", "Mailbox #2", etc.
Main	IO	The value "LargeCapacity" is analogous; "LargeCapacity" SHOULD be used instead when possible.
Middle	IO*	The bin that, when facing the device, can best be identified as "middle".
MyMailbox	O	The job will be output to the bin that is best identified as "my mailbox"
PostMarkerInserter New in JDF 1.4	I	The input tray that is downstream of the marking engine and allows the user to pass media through a non-marking paper path for covers and/or inserts.
Rear	IO*	The bin that, when facing the device, can best be identified as "rear".
Right	IO*	The bin that, when facing the device, can best be identified as "right".
Roll	IO	The location to handle web-fed media.
Roll-N	IO	The Nth location to handle the Nth web-fed media.
Side	IO*	The bin that, when facing the device, can best be identified as "side".
Stacker-N	O	The job will be output to the bin that is best identified as "Stacker #1", "Stacker #2", etc.
Top	IO*	The bin that, when facing the device, can best be identified as "top".
Tray	IO	The location for a single tray device.
Tray-N	IO	The job will be output to the tray that is best identified as "Tray #1", "Tray #2", etc.
Upper	IO*	The value "Top" is analogous; "Top" SHOULD be used instead when possible.

A.4.4 Media Coating Names

When specifying media coating types and variations, values from the following table are recommended.

Table A.49: Media Coating Names

NAME	DESCRIPTION
Aqueous	Water based coating
Coated New in JDF 1.6	A coating of a system specified type
DullUV Deprecated in JDF 1.6	

Table A.49: Media Coating Names

NAME	DESCRIPTION
DullVarnish Deprecated in JDF 1.6	
Gloss New in JDF 1.6	A glossy coating
GlossUV Deprecated in JDF 1.6	
GlossVarnish Deprecated in JDF 1.6	
Matte New in JDF 1.6	A matte coating
None	No coating is requested
Primer New in JDF 1.6	A coating that is applied beneath the image
Protective Deprecated in JDF 1.6	
RubResistant New in JDF 1.6	Attribute of the ink
Satin New in JDF 1.6	A coating between Gloss and Matte
SatinUV Deprecated in JDF 1.6	
SatinVarnish Deprecated in JDF 1.6	
Silicone	Liquid that is similar to ink
UV Modified in JDF 1.5	Ultra Violet cured polymers
Varnish	Unpigmented ink
WaterResistant New in JDF 1.6	Attribute of the ink

A.4.5 Milestone Values

The following table defines a list of values that are valid for ***PageList***/***PageData***/@*PageStatus* and ***Milestone***/@*MilestoneType*. The column "JDF Process" specifies the @*Category* or @*Type* of the node that the ***Milestone*** applies to. "PageStatus" specifies whether the value MAY be used as ***PageList***/***PageData***/@*PageStatus* ***ContentList***/@*ContentStatus*. "Milestone" specifies whether the value MAY be used as ***Milestone***/@*MilestoneType*. Note that milestones usually refer to events involving multiple objects, although the ***Milestone***/@*MilestoneType* is specified as a singular. The scope of the ***Milestone*** is defined by the parent ***Notification***element.

Note: The following symbols are used in the table below:

- **DigDel** means **DigitalDelivery** in the **JDF** Process column.
- **Delivery** means **Delivery** in the **JDF** Process column.
- **ArtDelI** means ***ArtDeliveryIntent*** in the **JDF** Intent Resource column.
- **DeliveryI** means ***DeliveryIntent*** in the **JDF** Intent Resource column.
- **ProofingI** means ***ProofingIntent*** in the **JDF** Intent Resource column

Table A.50: MessageEvents and MilestoneType Values (Sheet 1 of 2)

MESSAGEEVENTS AND MILESTONETYPE VALUES	JDF PROCESS	MILE STONE	PAGE STATUS	DESCRIPTION
Accepted	**DigDel**	—	Yes	The receiver acknowledged that the files are accessible for their destination.
BindingCompleted	—	Yes	Yes	All binding worksteps including packing of the job have been completed. Postpress worksteps are defined according to ▸ Section 6.5 Postpress Processes.
BindingInProgress	—	Yes	Yes	At least one of the binding worksteps of the job is in progress status.
Delivered	**DigDel**	Yes	Yes	The files were delivered to the destination.
DeviceStopped	All	—	—	The device that executes the workstep has been stopped.
DigitalArtArrived	—	Yes	Yes	Digital content has been received.
JobCompletedSuccessfully	All	Yes	Yes	Job completed successfully.
JobCompletedWithErrors	All	Yes	Yes	Job completed with errors.
JobCompletedWithWarnings	All	Yes	Yes	Job completed with warnings.
JobInProgress	All	Yes	Yes	Job is in progress.
PageApproved	—	Yes	Yes	Planned page proofs have been approved.
PageCompleted	—	Yes	Yes	Pages are ready (no further page processing or page proofing required).
PageDeleted	—	—	Yes	Specifies that this, originally planned, page was deleted. For instance, in the past, the page status was "PagePreliminary". Due to a reduction of the total number of pages, this specific page may have been deleted.
PagePlanned	—	Yes	Yes	Specifies that this page is ready for further processing. Its planning process is finished.
PagePreliminary		Yes	Yes	It is planned to produce this page, but its planning process is not finished yet.
PageProofed	—	Yes	Yes	Planned page proofs have been made
PDLProduced	All	Yes	Yes	Indicates that content data has been produced and is ready for production.
PostPressCompleted	—	Yes	Yes	All Postpress worksteps including packing of the job have been completed. Postpress worksteps are defined according to ▸ Section 6.5 Postpress Processes.
PostPressInProgress	—	Yes	Yes	At least one of the Postpress worksteps of the job is in progress status.
PrePressCompleted	—	Yes	Yes	All Prepress worksteps of the job have been completed. Prepress worksteps are defined according to ▸ Section 6.3 Prepress Processes. In conventional prepress, this is the case when all plates have been made.
PrePressInProgress	—	Yes	Yes	At least one of the Prepress worksteps of the job is in progress status.

Table A.50: MessageEvents and MilestoneType Values (Sheet 2 of 2)

MESSAGEEVENTS AND MILESTONETYPE VALUES	JDF PROCESS	MILE STONE	PAGE STATUS	DESCRIPTION
PressCompleted	—	Yes	Yes	All Press worksteps of the job have been completed. Press worksteps are defined according to ▸ Section 6.4 Press Processes.
PressInProgress	—	Yes	Yes	At least one of the Press worksteps of the job is in progress status.
ProofSent	—	Yes	Yes	Planned proofs sent to customer.
ShippingCompleted	**Delivery**	Yes	Yes	Final product was delivered to the customer or distributors.
ShippingInProgress	**Delivery**	Yes	Yes	Final product is being shipped.
SurfaceApproved	—	Yes	Yes	Planned imposition proofs have been approved.
SurfaceAssigned	—	Yes	Yes	Surfaces have their corresponding pages assigned (e.g., could be proofed).
SurfaceCompleted	—	Yes	Yes	Planned surfaces are ready (i.e., plates could be made).
SurfaceProofed	—	Yes	Yes	Planned imposition proofs have been made.

A.4.6 NotificationDetails

The ***Notification*** Element is used for messaging and logging of events. It is defined in ▸ Section 3.11.4.5 Notification. Notifications are grouped into five Classes: "Event", "Information", "Warning", "Error" and "Fatal". For more about ***Notification*** Classes, see ***Notification***/@*Class* in ▸ Table 3.11.4.5 Notification. In addition to the classes, the @*Type* attribute and ***Abstract NotificationDetails*** element provide a container for detailed information about the notification.

Elements derived from the ***Abstract NotificationDetails*** element represent a structured and extensible data type. The structure of various predefined ***NotificationDetails*** types and their descriptions are listed in the following sections.

A.4.6.1 Abstract NotificationDetails

The ***Abstract NotificationDetails*** element is empty.

Table A.51: Abstract NotificationDetails

NAME	PAGE	DESCRIPTION

A.4.6.2 NotificationDetails

▸ Table A.52 List of NotificationDetails Elementsdefines the elements that are derived from the ***Abstract NotificationDetails*** element. The value of ***Notification***/@*Type* is the same as the element name for the corresponding ***Notification***/***NotificationDetails***.

Table A.52: List of NotificationDetails Elements (Sheet 1 of 2)

NAME	PAGE	DESCRIPTION
Barcode	page 791	A bar code has been scanned
FCNKey	page 791	A function key has been activated at a console.
SystemTimeSet	page 791	The system time of a device/Controller/Agent has been set
CounterReset	page 791	The production counter of a device has been reset.
Error	page 791	This element provides additional information for common errors
Event	page 792	This element provides additional information for common events.

Table A.52: List of NotificationDetails Elements (Sheet 2 of 2)

NAME	PAGE	DESCRIPTION
Milestone	page 792	Tracks certain overall milestones concerning the entire job across all resources and processes.

A.4.6.2.1 Barcode

A bar code has been scanned.

Table A.53: Barcode Element

NAME	DATA TYPE	DESCRIPTION
Code	string	Contains the scanned bar code.

A.4.6.2.2 FCNKey

A function key has been activated at a console.

Table A.54: FCNKey Element

NAME	DATA TYPE	DESCRIPTION
Key	integer	Contains the number of that function key.

A.4.6.2.3 SystemTimeSet

The system time of a device/Controller/Agent has been set (e.g., readjusted, changed to daylight saving time, etc.).

Table A.55: SystemTimeSet Element

NAME	DATA TYPE	DESCRIPTION
NewTime	dateTime	Contains the new time.
OldTime ?	dateTime	Contains the old time.

A.4.6.2.4 CounterReset

The production counter of a device has been reset.

Table A.56: CounterReset Element

NAME	DATA TYPE	DESCRIPTION
CounterID ?	string	Identification of the counter that has been set.
LastCount ?	integer	Last counter value before reset.

A.4.6.2.5 Error

This element provides additional information for common errors.

Table A.57: Error Element

NAME	DATA TYPE	DESCRIPTION
ErrorID ? Modified in JDF 1.3	string	Internal Error ID of the application that declares the error.
Resend ? New in JDF 1.3	enumeration	Expected re-sending policy to fix the error. **Allowed values are:** Required – A corrected version of the offending **JMF** SHALL be resent. Prohibited – A corrected version of the offending **JMF** SHALL NOT be resent.
ReturnCode ? New in JDF 1.2	integer	**JDF** defined return code for an error. See ▸ Section Return Values and Error Codes.
ErrorData * New in JDF 1.3	element	Additional details of the error.

A.4.6.2.6 ErrorData

This element provides additional information for locating errors.

Table A.58: ErrorData Element

NAME	DATA TYPE	DESCRIPTION
ErrorType	enumeration	Details of the error of the attribute or element specified in @*Path*. **Allowed values are:** Invalid – the attribute or element has an invalid value Missing – the attribute or element is missing. Unsupported – the attribute or element is not known by the receiver.
ErrorURL ?	URL	URL of the referenced entity (e.g., **JDF** or PDL) where the error occurred. If not specified, the error occurred in the received **JMF** .
FixExpression ?	regExp	Expression that defines the acceptable valid values for the attribute defined by @*Path*. @*FixExpression* SHALL NOT be specified if @*Path* specifies an element.
Path ?	XPath	XPath location of the erroneous attribute or element in the offending **JMF** or referenced file. If @*ErrorURL* is specified, @*Path* refers to the XML that is referenced by @*ErrorURL*, otherwise it refers to the **JMF** that caused the error. @*Path* SHALL NOT be specified if @*ErrorURL* references a format other than XML.

A.4.6.2.7 Event

New in JDF 1.2

This element provides additional information for common events.

Table A.59: Event Element

NAME	DATA TYPE	DESCRIPTION
EventID	string	Internal Event ID of the application that emits the event.
EventValue ?	string	Additional user defined value related to this event.

A.4.6.2.8 Milestone

New in JDF 1.3

In addition to the concrete **JMF** feedback both from production to MIS and MIS to production with respect to finished processes (see ▸ Section 5.55 Status) and available/consumed resources (see ▸ Section 5.46 Resource), many actors in the workflow want to track certain overall milestones concerning the entire job across all resources and processes in order to display this to the operator. Sometimes the **JMF** recipients cannot determine these milestones from the detailed **JDF**/**JMF**. Therefore a more abstract representation of job status is described by ***Milestone*** events. Note that ***Milestone*** elements usually refer to events involving multiple objects, although the ***Milestone***/@*MilestoneType* is specified as a singular. The scope of the ***Milestone*** is defined by the parent ***Notification*** element.

Table A.60: Milestone Element

NAME	DATA TYPE	DESCRIPTION
MilestoneType	NMTOKEN	Type of ***Milestone***. **Values include those from:** ▸ Table A.50 MessageEvents and MilestoneType Values.
TypeAmount ?	integer	Indication of how many elements have been processed (if the milestone refers to certain resources) (e.g., number of pages proofed, number of different printed sheets (not the cumulative amount)).

Example A.51: Milestone in JMF

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="WorkflowController"
       TimeStamp="2005-07-25T12:32:48+02:00" Version="1.4"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
    <Signal ID="S1" Type="Notification" xsi:type="SignalNotification">
       <Notification Class="Event" JobID="myJobID"
              TimeStamp="2005-05-25T12:32:48+02:00"
              Type="Milestone">
          <Comment>All Proofs sent to customer</Comment>
          <Milestone MilestoneType="ProofSent" TypeAmount="24"/>
       </Notification>
    </Signal>
</JMF>
```

A.4.7 StatusDetails Supported Strings

The @*StatusDetails* attribute refines the concept of a job status to be job specific or a device status to be device specific. The following tables define individual @*StatusDetails* values and map them to the appropriate job specific state ***JDF***/@*Status* or device specific state ***DeviceInfo***/@*DeviceStatus*.

A.4.7.1 **Note:** *JDF*/@*Status* = "Setup", "Cleanup" and "Stopped" can include the description of a device with no job assigned to it. StatusDetails for Generic Devices

Table A.61: StatusDetails Mapping for Generic Devices (Sheet 1 of 3)

STATUSDETAILS	JDF/ @STATUS	DEVICESTATUS	DESCRIPTION
AbortedBySystem New in JDF 1.3	Aborted	Stopped	The job is being or has been aborted by the device.
BreakDown	Stopped	Offline	Breakdown of the device, repair needed.
Calibrating	Setup	Production	The device is calibrating, either manually or automatically.
ControlDeferred Modified in JDF 1.4	–	Offline	The machine is not accessible by the device. **Note: JDF**/@*Status* is unknown if the device is not accessible. **Modification note:** Starting with **JDF** 1.4, the @*DeviceStatus* value changed from "Stopped" to "Unknown".
CoverOpen New in JDF 1.3	Stopped	Stopped	One or more covers on the device are open.
DocumentAccessError New in JDF 1.3	Aborted	Stopped	The device could not access one or more documents passed by reference.
DoorOpen New in JDF 1.3	Stopped	Stopped	One or more doors on the device are open.
Failure	Stopped	Stopped	Failure of the device. Requires some maintenance in order to restart the device. "Failure" has specialized subcategories: "PaperJam", "DoubleFeed", "BadFeed", "BadTrim", "ObliqueSheet", "IncorrectComponent", "IncorrectThickness".
Good	InProgress	Production	Production of products in progress, good copy counter is on, waste copy counter is off
Idling	Stoppe	Production	Device is running, but no products are produced or consumed. Good and waste copy counter are off.
InputTrayMissing New in JDF 1.3	Stopped	Stopped	One or more input trays are not in the device
InterlockOpen New in JDF 1.3	Stopped	Stopped	One or more interlock devices on the printer are unlocked.
IterationPaused New in JDF 1.4	Suspended	Production	"Suspended" specifies that at least one iteration cycle has completed but additional iteration cycles MAY still occur.
JobCanceledByOperator New in JDF 1.3	Aborted	Production	The job was canceled by the device operator using ***AbortQueueEntry*** or means local to the device.
JobCanceledByUser New in JDF 1.3	Aborted	Production	The job was canceled by the owner of the job using ***AbortQueueEntry***.
JobCompletedSuccessfully New in JDF 1.3	Completed	Production	The job completed successfully.
JobCompletedWithErrors New in JDF 1.3	Completed	Production	The job completed with errors (and possibly warnings too)

Table A.61: StatusDetails Mapping for Generic Devices (Sheet 2 of 3)

STATUSDETAILS	JDF/ @STATUS	DEVICESTATUS	DESCRIPTION
JobCompletedWithWarnings New in JDF 1.3	Completed	Production	The job completed with warnings.
JobHeld New in JDF 1.3	Waiting	Production	The device held the job that had been waiting (by performing a ***HoldQueueEntry*** request on a waiting ***QueueEntry***).
JobHeldOnCreate New in JDF 1.3	Waiting	Production	The job was submitted to the queue with the ***Queue***/@*Status* = "Held", the job's ***QueueSubmissionParams***/@*Held* = "true", or **JDF**/@*Activation* = "Held".
JobIncoming New in JDF 1.3	Waiting	Production	The device is retrieving/accepting document data.
JobResuming New in JDF 1.3	Waiting	Production	The device is in the process of moving the job from a suspended condition to a candidate for processing (***ResumeQueueEntry***).
JobScheduling New in JDF 1.3	Waiting	Production	The device is scheduling the job for processing.
JobStreaming New in JDF 1.3	InProgress	Production	Same as "JobIncoming" with the specialization that the device is processing the document data as it is being received (that is, the job data is not being spooled, but rather is being processed in chunks by the output device and is being imaged during reception).
JobSuspended New in JDF 1.3	Suspended	Production	The device suspended the job that had been processing (e.g., by performing a ***SuspendQueueEntry*** request on a running ***QueueEntry***) and other jobs can be processed by the device.
JobSuspending New in JDF 1.3	InProgress	Production	The device is in the process of moving the job from a processing condition to a suspended condition where other jobs can be processed.
Maintenance	Stopped	Stopped	Maintenance of the device. "Maintenance" has specialized subcategories: "BlanketChange" and "SleeveChange".
MissResources	Stopped	Stopped	Production has been stopped because resources are missing or unavailable. Waits for new resources; subcategory of "Pause".
MovingToPaused New in JDF 1.3	InProgress	Production	The device has been paused, but the machine(s) are taking an appreciable time to stop.
OutputAreaFull New in JDF 1.3	Stopped	Stopped	One or more output areas are full (e.g., tray, stacker, collator).
OutputTrayMissing New in JDF 1.3	Stopped	Stopped	One or more output trays are not in the device.
PaperJam	Stopped	Stopped	Media jam in the device; subcategory of "Failure".
Pause	Stopped	Stopped	Machine paused; restart is possible. "Pause" has specialized subcategories: "MissResources" and "WaitForApproval".
ProcessingToStopPoint New in JDF 1.3	InProgress	Production	The requester has issued an ***AbortQueueEntry*** request or the device has aborted the job, but is still performing some actions on the job until a specified stop point occurs or job termination/cleanup is completed.

Table A.61: StatusDetails Mapping for Generic Devices (Sheet 3 of 3)

STATUSDETAILS	JDF/ @STATUS	DEVICESTATUS	DESCRIPTION
Repair	Stopped	Offline	The device is being repaired after a break down.
ShutDown	Stopped	Offline	Machine stopped (can be switched off), restart requires a run up.
SizeChange	Setup	Production	Changing setup for media size.
StandyBy		Offline	The device hase been switched into power saving mode and cannot process jobs without prior intervention such as ***Command***[@*Type*="WakeUp"].
StandBy		Idle	The device has been switched into power save mode and is still accepting new jobs.
WaitForApproval	Stopped	Stopped	Production has been stopped because a necessary approval is still missing, subcategory of "Pause".
WarmingUp	Setup	Production	Device is warming up after power up or power saver mode wake-up.
Waste	InProgress	Production	Production of products in progress, good copy counter is off, waste copy counter is on.
WasteFull	Stopped	Stopped	The device waste receptacle is full.

A.4.7.2 StatusDetails for Printing Devices

Table A.62: StatusDetails Mapping for Printing Devices

STATUSDETAILS	JDF/ @STATUS	DEVICESTATUS	DESCRIPTION
BlanketChange	Stopped	Stopped	Changing of blankets; subcategory of "Maintenance" (e.g., a 'specialization').
BlanketWash	Cleanup	Production	Washing of the blanket; subcategory of "WashUp".
CleaningInkFountain	Cleanup	Production	Cleaning of the ink fountain; subcategory of "WashUp".
CylinderWash	Cleanup	Production	Washing of impression cylinders; subcategory of "WashUp".
DampeningRollerWash	Cleanup	Production	Washing of the dampening roller; subcategory of "WashUp".
FormChange	Setup	Production	In conventional printing, changing of plates; in direct imaging printing, imaging or re-imaging of plates.
InkRollerWash	Cleanup	Production	Washing of the inking roller; subcategory of "WashUp".
PlateWash	Cleanup	Production	Washing of the plate; subcategory of "WashUp".
Processing New in JDF 1.4	InProgress	Production	Other productive processing (RIP, etc.) is taking place but no final output is being produced. All input data has arrived (not "InProgress"/"JobStreaming" nor "Waiting"/"JobIncoming").
SleeveChange	Stopped	Stopped	Changing of sleeves; subcategory for "Maintenance".
WashUp	Cleanup	Production	Machine is washed before, during or after production. "WashUp" has specialized subcategories: "BlanketWash", "CleaningInkFountain", "CylinderWash", "DampeningRollerWash", "InkRollerWash", or "PlateWash". "WashUp" is the default which is assumed if @*StatusDetails* is not specified.
WaitingForMarker New in JDF 1.4	Suspended or Ready	Production	The @*Status* is "Suspended" if the printing device models any module prior to the marker module, otherwise, the @*Status* is "Ready". Processing is automatically suspended by the worker because it is waiting behind other jobs for the marker module and the worker will resume processing when a marker module becomes available.
WaitingForReferencedDataCollector New in JDF 1.4	Suspended or Ready	Production	The @*Status* is "Suspended" if the printing device models any module prior to the referenced data collector module, otherwise, the @*Status* is "Ready". Processing is automatically suspended by the worker because it is waiting behind other jobs for theeferenced data collector module and the worker will resume processing when a referenced data collector module becomes available.
WaitingForRIP New in JDF 1.4	Suspended or Ready	Production	The @*Status* is "Suspended" if the printing device models any module prior to the RIP module, otherwise, the @*Status* is "Ready". Processing is automatically suspended by the worker because it is waiting behind other jobs for a RIP module (process slot) and the worker will resume processing when a RIP module becomes available.

A.4.7.3 StatusDetails for Postpress Devices

Table A.63: StatusDetails Mapping for Postpress Devices

STATUSDETAILS	JDF/ @STATUS	DEVICESTATUS	DESCRIPTION
BadFeed New in JDF 1.2	Stopped	Stopped	Bad feed on a feeder; subcategory of "Failure".
BadTrim New in JDF 1.2	Stopped	Stopped	Bad trimmed components; subcategory of "Failure".
DoubleFeed New in JDF 1.2	Stopped	Stopped	Double feeds on a feeder; subcategory of "Failure".
IncorrectComponent New in JDF 1.2	Stopped	Stopped	Incorrect components on a feeder; subcategory of "Failure".
IncorrectThickness New in JDF 1.2	Stopped	Stopped	Incorrect thickness of components; subcategory of "Failure".
ObliqueSheet New in JDF 1.2	Stopped	Stopped	Oblique sheets on components; subcategory of "Failure". Oblique sheets are sheets or signatures which are not properly aligned within a pile (e.g., on a gathering or collecting chain).

A.5 JDF File Formats

This section describes the specific file formats used by **JDF**. **JDF** uses TIFF and JPEG file formats, as well as the PNG image file format. The following sections explain in what ways PNG is used in **JDF**.

A.5.1 PNG Image Format

JDF uses the PNG images for representing preview images. CIP3 defined two formats: composite CMYK and separated. With PNG, only the separated format is supported for color spaces other than RGB. The composite CMYK or spot color representations SHALL be represented as separated CMYK or spot colors. Thus, preview images are stored as separate PNG images and **JDF** links them together. Viewable images and thumbnails can be represented as composite RGB PNG images.

References: http://www.w3.org/Graphics/png.

Appendix B

Schema

XML Schema for **JDF** (and **JMF**) will be published on: http://www.CIP4.org.

The XML Schema is not sufficient to completely validate a **JDF** Job. For example, Partitioned Resources or Process Node types as defined in **JDF** cannot be validated by XML Schema processors. In other words, the structure of some elements depends on the context of usage which cannot be completely described by XML Schema. Thus, the XML Schema for **JDF** will be structured in a way that it enables a pre-validation of valid **JDF** -candidates but does not preclude all syntactically invalid files to be validated.

Using JDF Schema

Any JDF processor SHOULD be capable of validating whether or not a JDF Job meets JDF requirements. This can be accomplished by using a schema when parsing or by using an application derived from a schema. The schema itself MAY be subsetted into multiple schemas that are used for validation purposes at different points in the workflow. For instance, a JMF schema subset MAY be used to test JDF-compliant Devices on your shop floor. A Product Intent subset MAY be used to check customer submitted Job specifications.

B.1 Using xsi:type

New in JDF 1.2

XML Schema permits that multiple type definitions be derived from a base type. Wherever the schema has define an element of that base type, it is possible for the document to indicate to a validator the particular derived type that it has used. This it does by using the *@xsi:type* Attribute with a value of the name of the type, where the "xsi" tag is associated with the Schema Instance namespace that has to be declared in the document.

Note: Use of "xsi" as the tag is normal practice.

Note: The selected type is namespace qualified (which permits extensions)

B.1.1 Using xsi:type with JDF Nodes

New in JDF 1.2

When used with **JDF** Nodes then all Processes defined in Section 6 are supported. Furthermore the value to be used is identical to the Process type, thus a **JDF** Node that has a *@Type* of "DigitalPrinting" can inform validators to use the schema definition for **DigitalPrinting** Nodes by also setting *@xsi:type* to "DigitalPrinting".

Some **JDF** Nodes are general in their nature and do not have a restricted definition (i.e., Product Intent Nodes, Combined Process Nodes and so on). General definitions with the appropriate name are provided to enable consistent use of *@xsi:type*.

The **JDF** Schema defines types for **JDF** Process Nodes and **JMF** Messages. It is RECOMMENDED that these types are used with *@xsi:type*.

Example B.1: JDF Nodes: xsi:type

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="BackCover"
Status="InProgress"
Type="DigitalPrinting" Version="1.4" JobPartID="345"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.CIP4.org/Schema/JDFSchema_1_1"
xsi:type="DigitalPrinting">
<ResourceLinkPool>
<DigitalPrintingParamsLink Usage="Input" rRef="ID123"/>
<RunListLink Usage="Input" rRef="ID124"/>
<ComponentLink Usage="Output" rRef="ID125"/>
</ResourceLinkPool>
<ResourcePool>
<DigitalPrintingParams ID="ID123" Class="Parameter" Status="Available" />
<RunList ID="ID124" Class="Parameter" Status="Available" />
<Component ID="ID125" Class="Quantity" Status="Unavailable"
ComponentType="Sheet" />
</ResourcePool>
</JDF>
```

Example B.2: JDF Nodes: xsi:type (not in Default Namespace)

If the **JDF** is not in the default namespace then the type name needs to be altered accordingly:

```
<jdf:JDF xmlns:jdf="http://www.CIP4.org/JDFSchema_1_1" ID="BackCover"
Status="InProgress"
Type="DigitalPrinting" Version="1.4" JobPartID="345"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="jdf:DigitalPrinting">
<jdf:ResourceLinkPool>
<jdf:DigitalPrintingParamsLink Usage="Input" rRef="ID123"/>
<jdf:RunListLink Usage="Input" rRef="ID124"/>
<jdf:ComponentLink Usage="Output" rRef="ID125"/>
</jdf:ResourceLinkPool>
<jdf:ResourcePool>
<jdf:DigitalPrintingParams ID="ID123" Class="Parameter"
Status="Available" />
<jdf:RunList ID="ID124" Class="Parameter" Status="Available" />
<jdf:Component ID="ID125" Class="Quantity" Status="Unavailable"
ComponentType="Sheet" />
</jdf:ResourcePool>
</jdf:JDF>
```

B.1.2 Using xsi:type with JMF Messages

New in JDF 1.2

JMF Messages are organized into families — ***Command***, ***Query***, etc. (See ▸ Section 5.2 JMF Message Families) — and each of these families has Messages for each Message @*Type* — ***Events***, etc. Because it is the convolution of these two that are the unique derived types, the name used in @*xsi:type* has to be the convolution of the Message Family and Type.

To query an event a Query Message with an ***Events***/***QueryTypeObj*** would be used. The type definition name employed by the **JDF** Schema would therefore be "QueryEvents".

Note **JMF** Messages also do not have to be in the default namespace as in the **JDF** Node example below.

Example B.3: JMF : xsi:type

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="TestSender"
TimeStamp="2003-11-07T12:15:56Z" MaxVersion="1.4" Version="1.4"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<Query ID="Message_001Q" Type="Events" xsi:type="QueryEvents">
<NotificationFilter/>
</Query>
<Response ID="Message_001R" Type="Events" refID="Q001"
xsi:type="ResponseEvents">
<NotificationDef Classes="Error" Type="Barcode"/>
</Response>
</JMF>
```

Appendix C

Return Values and Error Codes

The following list defines the standard *ReturnCode* for messaging. Return code values are integers. Error Messages below 100 are reserved for protocol errors. Error messages above 100 are used for Device and Controller errors and error messages above 200 for Job and pipe specific errors. Error Codes above 300 are used for errors related to authentication and certificate exchange.

Table C.1: Return codes for JMF (Sheet 1 of 3)

RETURNCODE	DESCRIPTION
0	Success
1 – 99	Protocol errors
1	General error
2	Internal error
3	XML parser error (e.g., if a MIME file is sent to an XML Controller).
4	XML validation error
5	Query Message/Command Message not implemented
6	Invalid parameters
7	Insufficient parameters
8	Device not available (Controller exists but not the Device or queue)
9	Message incomplete.
10 New in JDF 1.3	Message Service is busy.
11 New in JDF 1.4	Synchronous mode not supported for message. No @*AcknowledgeURL* is specified and the Message can only be processed asynchronously and was not processed. (Error)
12 New in JDF 1.4	Asynchronous acknowledge not supported for message. No @*AcknowledgeURL* is specified and the Message was processed. The resulting ***Acknowledge*** can only be emitted asynchronously. (Warning)
13 New in JDF 1.4	Reliable Signals not supported. Subscription denied.
100 – 199	Device and Controller errors
100	Device not running
101	Device incapable of fulfilling request (e.g., a RIP that has been asked to cut a Sheet).
102	No executable node exists in the **JDF**.
103	@*JobID* not known by Controller
104	@*JobPartID* not known by Controller
105	Queue entry not in queue
106	Queue request failed because the queue entry is already executing
107	The queue entry is already executing. Late change is not accepted

Table C.1: Return codes for JMF (Sheet 2 of 3)

RETURNCODE	DESCRIPTION
108	Selection or applied filter results in an empty list
109	Selection or applied filter results in an incomplete list. A buffer cannot provide the complete list queried for.
110	Queue request of a Job submission failed because the requested completion time of the Job cannot be fulfilled.
111	Subscription request denied.
112 New in JDF 1.1	Queue request failed because the Queue is "Closed" or "Blocked" and does not accept new entries.
113 New in JDF 1.2	Queue entry is already in the resulting status.
114 Modified in JDF 1.4	***QueueEntry***/@*Status* is already "PendingReturn", "Completed" or "Aborted" and therefore does not accept changes. **Modification note:** Starting with **JDF** 1.4, "PendingReturn" added.
115 New in JDF 1.2	Queue entry is not running.
116 New in JDF 1.3	Queue entry already exists. Used when a ***QueueEntry*** with identical @*JobID*, @*JobPartID* and ***Part*** already exists.
120 New in JDF 1.3	Cannot access referenced URL. URI Reference cannot be resolved. Used when a referenced entity (e.g., a **JDF** in a ***SubmitQueueEntry*** cannot be found).
121 New in JDF 1.3	Unknown *DeviceID*. No Device is known with the *DeviceID* specified.
130 New in JDF 1.3	Ganging is not supported. A gang Job has been submitted to a queue that does not support ganging.
131 New in JDF 1.3	*GangName* not known. A Job has been submitted with an unknown *GangName*.
200 – ...	Job and pipe specific errors
200	Invalid Resources
201	Insufficient Resource parameters
202	*PipeID* unknown
203	Unlinked ***ResourceLink***.
204 New in JDF 1.3	Could not create new **JDF** node.
205 New in JDF 1.6	Pipe request failed because the Pipe is closed and does not accept new requests
300 New in JDF 1.3	Authentication denied.
301 New in JDF 1.3	Secure channel not supported - I don't support secure channel for this Message.
302 New in JDF 1.3	Secure channel required - I require secure channel for this Message.
303 New in JDF 1.3	Certificate expired (Some implementations might not be able to send this response because the SSL layer will reject the Message before passing it to the **JMF** implementation for parsing)

Table C.1: Return codes for JMF (Sheet 3 of 3)

RETURNCODE	DESCRIPTION
304 New in JDF 1.4	Authentication pending.
305 New in JDF 1.4	Authentication already established.
306 New in JDF 1.4	No authentication request in process.
307 New in JDF 1.4	Certificate Invalid

Appendix D

Color Adjustment

New in JDF 1.2

This appendix describes several alternative usages of some attributes in the ***ColorCorrectionOp*** element (see ***ColorCorrectionParams***/***ColorCorrectionOp*** in ▸ Section 8.22 ColorCorrectionParams). that are intended to allow simple, late-in-the-workflow, minor adjustments to the overall color appearance of a job or portions of a job.

Note: These color adjustments are not available in any intent resource, such as ***ColorIntent***. In order to request such adjustment in a product intent job ticket supplied to a print provider, attach to a product intent node an incomplete **ColorCorrection** process with a ***ColorCorrectionParams*** resource specifying the requested ***ColorCorrectionOp*** element attributes.

D.1 Adjustment Using Direct Attributes

This section describes the following attributes that provide direct adjustments to various aspect of the color space:

Table D.1: Attributes for Color Space Adjustment

ATTRIBUTE NAME	ALLOWED VALUE RANGE
AdjustCyanRed	-100 to +100
AdjustMagentaGreen	-100 to +100
AdjustYellowBlue	-100 to +100
AdjustContrast	-100 to +100
AdjustHue	-180 to +180
AdjustLightness	-100 to +100
AdjustSaturation	-100 to +100

These attributes can be applied at a point where an abstract profile would be applied (following any abstract profiles used) in the order: @*AdjustLightness*, @*AdjustContrast*, @*AdjustSaturation*, @*AdjustHue*, {@*AdjustCyanRed*/@*AdjustMagentaGreen*/@*AdjustYellowBlue*}. The operation of each adjustment attribute is described in relation to colors expressed in the L*a*b* connection color space (with L* expressed on a scale of 0 to 100).

Note: In the C-language-like assignment statements below, the variables L, a and b are used to represent values of the L*, a* and b* channels to avoid ambiguity with the "*" used to denote multiplication in these statements.

- @*AdjustLightness* offsets the L* channel. [L += @*AdjustLightness*]
- @*AdjustContrast* scales the L* channel about mid-scale (where L = 50).
  [L = 50 + (L - 50) * (@*AdjustContrast* / 100 + 1)]
- @*AdjustSaturation* scales the a* and b* channels about zero. [a *= (@*AdjustSaturation* / 100 + 1)] and
  [b *= (@*AdjustSaturation* / 100 + 1)]

@*AdjustCyanRed*, @*AdjustMagentaGreen* and @*AdjustYellowBlue* offset the colors in the a*b* plane along the respective color vector. Lightness (L*) is not changed. Positive values offset towards red, green or blue, and negative values offset towards cyan, magenta or yellow. The adjustment vectors are aligned with the standard SWOP inks. When adjusting device colors, these adjustments can be approximated by offsets along the vectors of the actual ink colors being used. The angles and unit vectors for SWOP inks (from the CGATS TR001 print characterization) are:

	Red-cyan	**Green-Magenta**	**Blue-yellow**
Angle	-129.9	-5.3	94.5
a*	0.641	-0.996	0.078
b*	0.767	0.092	-0.997

So

a* += 0.641 * @*AdjustCyanRed*
- 0.996 * @*AdjustMagentaGreen*

+ 0.078 * @*AdjustYellowBlue*

b* += 0.767 * @*AdjustCyanRed*
+ 0.092 * @*AdjustMagentaGreen*
- 0.997 * @*AdjustYellowBlue*

@AdjustHue offsets the hue angle value when the colors have been transformed to the CIE- L* C* H* (luminance, chroma and hue) color space from the L*a*b* connection color space. The *@AdjustHue* angle is expressed in degrees.
Note: In the C-language-like assignment statements below, the variables L, a and b are used to represent values of the L*, a* and b* channels to avoid ambiguity with the "*" used to denote multiplication in these statements.

- a = a * cos(@*AdjustHue*) - b * sin(@*AdjustHue*)
- b = a * sin(@*AdjustHue*) + b * cos(@*AdjustHue*)

D.2 Adjustment using ICC Profile Attributes

This section describes two alternatives to the direct color adjustment attributes providing adjustments of the same nature using ICC profiles. The ICC profile approach provides a standard mechanism for applying a set of multi-dimensional adjustments with a single operation. The ICC profile approach also has an advantage in that it minimizes algorithm and interpretation dependency on the receiving end.

D.3 Adjustment using an ICC Abstract Profile Attribute

A color adjust can be encapsulated in an ICC abstract profile that is applied in ICC Profile Connection Space (PCS). The ***FileSpec*** element of the ***ColorCorrectionOpColorCorrectionOp*** element with the *@ResourceUsage* attribute set to "AbstractProfile" references an ICC profile to be used in this manner.

D.4 Adjustment using an ICC DeviceLink Profile Attribute

A color adjust can be encapsulated in an ICC DeviceLink profile that is applied in device space. The ***FileSpec*** element of the ***ColorCorrectionOp*** element with the *@ResourceUsage* attribute set to "DeviceLinkProfile" references an ICC profile to be used in this manner.

Paper Weight and Grade

In North America and Japan, each grade of paper has one basic size used to compute its basis weight per ream. For example, Bond basic size is 17" x 22" and Shiroku-ban basic size is 788 mm x 1091 mm.

E.1 North American Media Weight

New in JDF 1.2

In North America, a paper's basis weight is the weight of five hundred Sheets of its basic size. For example, if five hundred 25" x 38" Sheets of offset paper weigh 60 pounds, it is called 60# offset. Paper mills outside of North America use the metric system to designate paper weight. The basis weight of foreign papers is grams per square meter (g/m2) known as the Sheet's grammage. Papers made to metric standards don't convert to basis weights familiar to North Americans. For example, 100 g/m2 equals a basis weight of 67.5. Following is the English/grammage conversion formula:

Basis Weight (lb.) x (1406.5 / Square inches in basic size) = grams per square meter

For example, the grammage of 65 lb. cover stock when the cover is 20 x 26 can be calculated as follows:

65 x (1406.5 / (20 x 26)) = 65 x 2.70 = 176 g/m2

The following table defines the basic sizes and the factor that @*USWeight* is multiplied by to calculate @*Weight* for various stock types. Stock type is specified in ***Media***/@*StockType* or ***MediaIntent*** /@*StockType*.

Table E.1: Conversion Factor from Stock Weight (lbs) to Weight (g/m²)

STOCK TYPE	BASIS SIZE IN INCHES	STOCK WEIGHT	EQUIVALENT
"Bond"	17" x 22"	3.76	"Ledger", "Manifold"
"Book"	25" x 38"	1.48	"Bible", "Coated", "Offset", "Text"
"Bristol"	22½" x 28½"	2.19	
"Cover"	20" x 26"	2.70	
"Index"	25½" x 30½"	1.81	
"Newsprint"	24" x 36"	1.63	"Tag"

In the following table, the right columns of each column pair list common basis weights for North American papers while the left columns list their corresponding grammage. The rows are ordered by grammage. Basis weights for bond, book, cover and other grades of papers are computed using different basic sizes, so the progression of weights down the right columns is untidy.

Table E.2: Grammage Equivalents for Common (US) Stock Weights (Sheet 1 of 2)

GRAMMAGE (G/M²)	STOCK WEIGHT	GRAMMAGE (G/M²)	BASIS WEIGHT
30	20# Book	150	40# Ledger
34	9# Manifold	152	60# Cover
36	24# Book	163	90 # Index
44	30# Book	163	100 # Tag
45	12# Manifold	175	80# Bristol
49	13# Bond	176	65# Cover
49	33# Book	178	120# Book
52	35# Book	197	90# Bristol
59	40# Book	199	110# Index
60	16# Bond	204	125# Tag
67	45# Bond	216	80# Cover

Table E.2: Grammage Equivalents for Common (US) Stock Weights (Sheet 2 of 2)

GRAMMAGE (G/M²)	STOCK WEIGHT	GRAMMAGE (G/M²)	BASIS WEIGHT
74	50# Book	219	100# Bristol
75	20# Bond	244	150# Tag
81	55# Book	253	140# Index
89	60# Book	263	120# Bristol
90	24# Bond	270	100# Cover
104	70# Book	285	175# Tag
105	28# Ledger	307	140# Bristol
108	40# Cover	307	170# Index
118	80# Book	325	200# Tag
120	32# Ledger	350	160# Bristol
133	90# Book	352	130# Cover
135	36# Ledger	394	180# Bristol
135	50# Cover	398	220# Index
147	67# Bristol	407	250# Tag
148	100# Book	438	200# Bristol
		488	300# Tag

E.2 Japanese Media Weight

New in JDF 1.3

In Japan, a paper's basis weight is the weight of 1000 Sheets of its basic size and ream weights are given in kg.

The following table is originally published by EDS Inc., Editorial & Design Services see ▸ [Japanese Paper Sizes]. For more help with grammage and basis weight conversion, see also ▸ [Grammage Conversion].

Following is the Japanese/grammage conversion formula:

Basis Weight (kg) / Basic Size (m^2) = grams per square meter

For example, the grammage of 70 kg Shiroku-ban stock when the size is 0.788 x 1.091 can be calculated as follows:

70 / (0.788 x 1.091) = 81.4 g/m^2

In the table below, trade-sheet size is given in mm.

Table E.3: Japanese Media Weight (Sheet 1 of 2)

STOCK TYPE	SHIROKU-BAN 788 X 1091	JIS B-BAN 765 X 1085	KIKU-BAN 636 X 939	JIS A-BAN 625 X 880	GRAMMAGE (G/M²)
上質紙 Joushitsushi	40	--	--	--	46.5
	45	--	31	20.5	52.3
	55	53	38	35	64.0
	70	67.5	48.5	44.5	81.4
	90	--	62.5	47.5	104.7
	110	--	71.5	70.5	127.9
	135	--	93.5	80.5	157.0
	180	--	--	--	209.3

Table E.3: Japanese Media Weight (Sheet 2 of 2)

STOCK TYPE	SHIROKU-BAN 788 X 1091	JIS B-BAN 765 X 1085	KIKU-BAN 636 X 939	JIS A-BAN 625 X 880	GRAMMAGE (G/M^2)
中質紙 Chuushitsushi	--	45	--	30	54.2
	--	55	--	36.5	66.3
アート紙 Aatoshi	73	70.5	50.5	46.5	84.9
	90	87	62.5	57.5	104.7
	110	106	76.5	70.5	127.9
	135	130.5	93.5	86.5	157.0
マシンコート紙 Mashinkootoshi	63	61	--	--	73.3
	68	65.6	47	43.5	79.1
	73	70.5	50.5	46.5	84.9
	90	87	62.5	57.5	104.7
	110	106	76.5	70.5	127.9
	135	130.5	93.5	86.5	157.0
アートポスト紙 Aatoposutoshi	180	--	125	--	209.3
	200	--	139	--	232.6
	220	--	153	--	255.0

* The following describes the five stock types in the above table:

- **上質紙** Joushitsushi ("top-quality paper") contains 100% chemical pulp;
- **中質紙** Chuushitsushi ("medium-quality paper") contains a minimum of 70% chemical pulp;
- **アート紙**Aatoshi ("art paper") is machine coated paper, available in top quality and medium quality (Joushitsu and Chuushitsu);
- **マシンコート紙** Mashinkootoshi ("machine coated paper"), also called Kootoshi (**コート紙**), is machine coated paper given only a thin coat of clay;
- **アートポスト紙** Aatoposutoshi ("art-post paper") is cover stock coated on one side.

E.3 Paper Grade

[ISO12647-2:2004] provides a rough classification of paper with 5 classes, which is generally referred to as paper grade. [ISO12647-2:2013]was updated in 2013 and a new set of 8 standard papers was defined that are more appropriate for paper types that are used today. The following table provides a rough and non-normative translation between the two standard sets:

Table E.4: Translation of Paper grades between [ISO12647-2:2004] and [ISO12647-2:2013]

ISO12647-2:2013		PRESS	ISO12647-2:2004	
ID	TYPE		GRADE	TYPE
PS1	Premium Coated	Sheet	1	Gloss coated
PS2	Improved Coated	Web	3	Gloss coated, web
PS3	Standard Coated Glossy	Web	3	Gloss coated, web
PS4	Standard Coated Matte	Web	2	Matte coated
PS5	Wood free Uncoated	Sheet	4	Uncoated white
PS6	Super Calendered	Web	4	Uncoated white

Table E.4: Translation of Paper grades between [ISO12647-2:2004] and [ISO12647-2:2013]

ISO12647-2:2013		PRESS	ISO12647-2:2004	
ID	TYPE		GRADE	TYPE
PS7	Improved Uncoated	Web	4	Uncoated white
PS8	Standard Uncoated	Web	4/5	Uncoated white/yellowish

Appendix F

Media Sizes

The following table defines a set of named media sizes as defined by ▸ [PPD].

Implementation Remark

Modified in JDF 1.5

Since Media sizes may be real numbers, comparison of Media sizes SHOULD take into account certain rounding errors. For example, different Media sizes SHOULD be considered equal when all numbers are the same within a range of 5 point.
Modification note: Starting with **JDF** 1.5, the recommended range has been changed from 1 point to 5 points.

Key for Notes

- I — Size is defined by ISO standards, including ▸ [ISO216:1975].
- J — Size is defined by JIS standards. ▸ [JIS P0138:1998]
- E — This is an envelope size ▸ [ISO269:1985].

Table F.1: Media Sizes (Sheet 1 of 4)

MEDIA SIZE	SIZE IN POINTS	SIZE IN MILLIMETERS	SIZE IN INCHES	NOTES
A0	2384 x 3370	841 x 1189	33.11 x 46.81	I, J
A1	1684 x 2384	594 x 841	23.39 x 33.11	I, J
A2	1191 x 1684	420 x 594	16.54 x 23.39	I, J
A3	842 x 1191	297 x 420	11.69 x 16.54	I, J
A3Extra	913 x 1262	322 x 445	12.67 x 17.52	
A4	595 x 842	210 x 297	8.27 x 11.69	I, J
A4Extra	667 x 914	235.5 x 322.3	9.27 x 12.69	
A4Plus	595 x 936	210 x 330	8.27 x 13	
A4Tab New in JDF 1.4	638 x 842	225 x 297	8.86 x 11.69	
A5	420 x 595	148 x 210	5.83 x 8.27	I, J
A5Extra	492 x 668	174 x 235	6.85 x 9.25	
A6	297 x 420	105 x 148	4.13 x 5.83	I, J
A7	210 x 297	74 x 105	2.91 x 4.13	I, J
A8	148 x 210	52 x 74	2.05 x 2.91	I, J
A9	105 x 148	37 x 52	1.46 x 2.05	I, J
A10	73 x 105	26 x 37	1.02 x 1.46	I, J
AnsiC	1224 x 1584	431.8 x 558.8	17 x 22	
AnsiD	1584 x 2448	558.8 x 863.6	22 x 34	
AnsiE	2448 x 3168	863.6 x 1118	34 x 44	
ARCHA	648 x 864	228.6 x 304.8	9 x 12	

Table F.1: Media Sizes (Sheet 2 of 4)

MEDIA SIZE	SIZE IN POINTS	SIZE IN MILLIMETERS	SIZE IN INCHES	NOTES
ARCHB	864 x 1296	304.8 x 457.2	12 x 18	
ARCHC	1296 x 1728	457.2 x 609.6	18 x 24	
ARCHD	1728 x 2592	609.6 x 914.4	24 x 36	
ARCHE	2592 x 3456	914.4 x 1219	36 x 48	
B0	2920 x 4127	1030 x 1456	40.55 x 57.32	J
B1	2064 x 2920	728 x 1030	28.66 x 40.55	J
B2	1460 x 2064	515 x 728	20.28 x 28.66	J
B3	1032 x 1460	364 x 515	14.33 x 20.28	J
B4	729 x 1032	257 x 364	10.12 x 14.33	J
B5	516 x 729	182 x 257	7.17 x 10.12	J
B6	363 x 516	128 x 182	5.04 x 7.17	J
B7	258 x 363	91 x 128	3.58 x 5.04	J
B8	181 x 258	64 x 91	2.52 x 3.58	J
B9	127 x 181	45 x 64	1.77 x 2.52	J
B10	91 x 127	32 x 45	1.26 x 1.77	J
C4	649 x 918	229 x 324	9.02 x 12.75	I, E
C5	459 x 649	162 x 229	6.38 x 9.02	I, E
C6	323 x 459	114 x 162	4.51 x 6.38	I, E
Comm10	297 x 684	104.8 x 241.3	4.125 x 9.5	E
DL	312 x 624	110 x 220	4.33 x 8.66	I, E
DoublePostcard	567 x 419	200 x 148	7.87 x 5.83	
Env9	279 x 639	98.4 x 225.4	3.875 x 8.875	E
Env10	297 x 684	104.8 x 241.3	4.125 x 9.5	E
Env11	324 x 747	113.3 x 263.5	4.5 x 10.375	E
Env12	342 x 792	120.7 x 279.4	4.75 x 11	E
Env14	360 x 828	127 x 292.1	5 x 11.5	E
EnvC0	2599 x 3676	917 x 1297	36.10 x 51.06	I, E
EnvC1	1837 x 2599	648 x 917	25.51 x 36.10	I, E
EnvC2	1298 x 1837	458 x 648	18.03 x 25.51	I, E
EnvC3	918 x 1296	324 x 458	12.75 x 18.03	I, E
EnvC4	649 x 918	229 x 324	9.02 x 12.75	I, E
EnvC5	459 x 649	162 x 229	6.38 x 9.02	I, E

Table F.1: Media Sizes (Sheet 3 of 4)

MEDIA SIZE	SIZE IN POINTS	SIZE IN MILLIMETERS	SIZE IN INCHES	NOTES
EnvC6	323 x 459	114 x 162	4.51 x 6.38	I, E
EnvC65	324 x 648	114 x 229	4.51 x 9	E
EnvC7	230 x 323	81 x 113	3.19 x 4.49	I, E
EnvChou3	340 x 666	120 x 235	4.72 x 9.25	E
EnvChou4	255 x 581	90 x 205	3.54 x 8	E
EnvDL	312 x 624	110 x 220	4.33 x 8.66	I, E
EnvInvite	624 x 624	220 x 220	8.66 x 8.66	E
EnvISOB4	708 x 1001	250 x 353	9.84 x 13.9	E
EnvISOB5	499 x 709	176 x 250	6.9 x 9.8	E
EnvISOB6	499 x 354	176 x 125	6.9 x 4.9	E
EnvItalian	312 x 652	110 x 230	4.33 x 9	E
EnvKaku2	680 x 941	240 x 332	9.45 x 13	E
EnvKaku3	612 x 785	216 x 277	8.5 x 10.9	E
EnvMonarch	279 x 540	98.43 x 190.5	3.875 x 7.5	E
EnvPersonal	261 x 468	92.08 x 165.1	3.625 x 6.5	E
EnvPRC1	289 x 468	102 x 165	4 x 6.5	E
EnvPRC2	289 x 499	102 x 176	4 x 6.9	E
EnvPRC3	354 x 499	125 x 176	4.9 x 6.9	E
EnvPRC4	312 x 590	110 x 208	4.33 x 8.2	E
EnvPRC5	312 x 624	110 x 220	4.33 x 8.66	E
EnvPRC6	340 x 652	120 x 230	4.7 x 9	E
EnvPRC7	454 x 652	160 x 230	6.3 x 9	E
EnvPRC8	340 x 876	120 x 309	4.7 x 12.2	E
EnvPRC9	649 x 918	229 x 324	9 x 12.75	E
EnvPRC10	918 x 1298	324 x 458	12.75 x 18	E
EnvYou4	298 x 666	105 x 235	4.13 x 9.25	E
Executive	522 x 756	184.2 x 266.7	7.25 x 10.5	
FanFoldGerman	612 x 864	215.9 x 304.8	8.5 x 12	
FanFoldGermanLegal	612 x 936	215.9 x 330	8.5 x 13	
FanFoldUS	1071 x 792	377.8 x 279.4	14.875 x 11	
Folio	595 x 935	210 x 330	8.27 x 13	
ISOB0	2835 x 4008	1000 x 1414	39.37 x 55.67	I

Table F.1: Media Sizes (Sheet 4 of 4)

MEDIA SIZE	SIZE IN POINTS	SIZE IN MILLIMETERS	SIZE IN INCHES	NOTES
ISOB01	2004 x 2835	707 x 1000	27.83 x 39.37	I
ISOB2	1417 x 2004	500 x 707	19.68 x 27.83	I
ISOB3	1001 x 1417	353 x 500	13.90 x 19.68	I
ISOB4	709 x 1001	250 x 353	9.84 x 13.90	I
ISOB5	499 x 709	176 x 250	6.9 x 9.8	I
ISOB5Extra	569 x 782	201 x 276	7.9 x 10.8	
ISOB6	354 x 499	125 x 176	4.92 x 6.93	I
ISOB7	249 x 354	88 x 125	3.46 x 4.92	I
ISOB8	176 x 249	62 x 88	2.44 x 3.46	I
ISOB9	125 x 176	44 x 62	1.73 x 2.44	I
ISOB10	88 x 125	31 x 44	1.22 x 1.73	I
Ledger	1224 x 792	431.8 x 279.4	17 x 11	
Legal	612 x 1008	215.9 x 355.6	8.5 x 14	
LegalExtra	684 x 1080	241.3 x 381	9.5 x 15	
Letter	612 x 792	215.9 x 279.4	8.5 x 11	
LetterExtra	684 x 864	241.3 x 304.8	9.5 x 12	
LetterPlus	612 x 913	215.9 x 322.3	8.5 x 12.69	
LetterTabThreeEighthsInch New in JDF 1.4	639 x 792	225.4 x 279.4	8.875 x 11	
LetterTabHalfInch New in JDF 1.4	648 x 792	228.6 x 279.4	9 x 11	
LetterTabFiveEighthsInch New in JDF 1.4	657 x 792	231.8 x 279.4	9.125 x 11	
Monarch	279 x 540	98.43 x 190.5	3.875 x 7.5	E
Postcard	284 x 419	100 x 148	3.94 x 5.83	
PRC16K	414 x 610	146 x 215	5.75 x 8.5	
SRA3 New in JDF 1.4	907 x 1276	320 x 450	12.60 x 17.72	

Appendix G

MimeTypes

New in JDF 1.2

This appendix lists examples values for the following Attributes of the ***FileSpec*** Resource: @*MimeType* and @*MimeTypeVersion*. The preferred file name extension is also indicated for use in the ***FileSpec***/@*URL* Attribute. The tables below apply to the values of @*PDLType* and @*PDLVersion* defined in ▸ Section 10.3.2.5 Document Properties, respectively.

The listing is intended to be exhaustive for the most likely document formats that are routinely used in **JDF** applications. However, other document formats and other combinations of the listed document formats can be used as well. When these format standards are revised with new version numbers, they MAY be used and SHOULD follow the patterns established in the following tables.

Many @*MimeTypeVersion* values are taken from the *Printer MIB* ▸ [RFC1759] by using the a language (e.g., PS, PCL, etc.) as a prefix followed by the level or version defined for **prtInterpreterLangLevel** separated by a "/" character (e.g.,"PS/3" for PostScript Level 3). For file formats not in the *Printer MIB*, the prefix is the common acronym for the format with "/" changed to "-" so that the prefix always ends with the first "/" (e.g. "DCS/2.0" for DCS version 2.0 and "TIFF-IT/BL/P1:1998" for TIFF/IT — Binary Line art image data — profile 1).

▸ Table G.1 MimeType Attribute Values (IANA Registered) lists the @*MimeType* values that are MIME Media Types registered with IANA (as opposed to file types which are not registered with IANA) in alphabetical order, as well as possible @*MimeTypeVersion* values. A blank @*MimeTypeVersion* table entry indicates that there is no recognized version number for the @*MimeType*. ▸ Table G.1 MimeType Attribute Values (IANA Registered) also lists the associated RECOMMENDED file name extensions commonly used by **JDF** applications.

Note: According to ▸ [RFC2046] the initial set of MIME media types start with the substrings: "application/", "audio/", "image/", "message/", "model/", "multipart/", "text/" or "video/". File Types will not start with these strings. The @*Compression* values that do have a corresponding IANA MIME type are also listed, so that a file that is so compressed or encoded has an appropriate @*MimeType* value for the file, as shown below.

Modification note: Starting with **JDF** 1.4, the second column "Sample MimeType Version" replaces "MimeType Version" and rows with same value of MimeType, but with different values of MimeType Version are reduced to a single row with just a sample MimeType Version

Table G.1: MimeType Attribute Values (IANA Registered) (Sheet 1 of 3)

MIMETYPE	SAMPLE MIMETYPE-VERSION	FILE EXTENSION	DESCRIPTION [IANA-MT] INDICATES IANA REGISTRATION
application/mac-binhex40	HQX/4.0	.hqx	Macintosh BinHex 4.0 7-bit encoding ▸ [RFC1741] **Note:** BinHex encoding converts an 8-bit file into a 7-bit format ▸ [RFC1741], similar to Uuencoding. BinHex format preserves file Attributes, as well as Macintosh resource forks, and includes CRC (Cyclic Redundancy Check) error-checking. This encoding method works on any type of file, including formatted word processing and spreadsheet files, graphics files and even executable files (i.e., programs or applications). **Note:** BinHex is not to be confused with MacBinary encoding, which is an 8-bit format.
application/msword	MSWORD/XP	.doc	Microsoft Word
application/pdf	PDF/1.6, PDF/X-3:2003	.pdf	Adobe Portable Document Format ▸ [PDF1.6] and Portable Document Format (PDF) PDF/X-3 ▸ [ISO15930-6:2003]
application/postscript	PS/3	.ps	Adobe PostScript™ See ▸ [RFC2045] and ▸ [RFC2046]

Table G.1: MimeType Attribute Values (IANA Registered) (Sheet 2 of 3)

MIMETYPE	SAMPLE MIMETYPE-VERSION	FILE EXTENSION	DESCRIPTION [IANA-MT] INDICATES IANA REGISTRATION
application/vnd.cip4-jdf+xml Modified in JDF 1.5	JDF 1.5	.jdf	CIP4 Job Definition Format (**JDF**) version 1.5.
application/vnd.cip4-jmf+xml Modified in JDF 1.5	JMF 1.5	.jmf	CIP4 Job Definition Format (**JDF**) version 1.5 (See Job Messaging Format).
application/vnd.cip4-ptk+xml New in JDF 1.5	PrintTalk 1.5	.ptk	CIP4 PrintTalk version 1.5
application/vnd.cip3-ppf	PPF/3.0	.ppf	CIP3 Print Production Format (PPF) version 3.0, 1998 ▸ [CIP3 - PPF]
application/vnd.hp-PCL	PCL/X	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.iccprofile New in JDF 1.4		.icc .icm	International Color Consortium (ICC) File Format for Color Profiles taken from the binary coded decimal Profile Header Profile Version Number field (bytes 8 through 11) ▸ [ICC.1] **Creation note:** Starting with **JDF** 1.4 this MimeType replaces "ICC Profile". See ▸ Table G.2 MimeType and File Type Combinations.
application/vnd.podi-ppml+xml	PPML/2.1	.ppml	Personalized Print Markup Language ▸ [PPML]
application/vnd.Quark.QuarkXPress	XPress/6.0	.qxd .qxt .qwd .qwt .qxl .qxb	QuarkXPress ▸ [Quark]
application/zip		.zip	ZIP packaging — The actual compression used for each file in a ZIP package is stored in the ZIP package as metadata for each file. Therefore, the ***FileSpec***/@*Compression* Attribute for the contained file MAY use any @*Compression* value, including "None", "Compress", "Gzip" and "ZLIB".
image/jpeg		.jpeg .jpg	JPEG See ▸ [RFC2045] and ▸ [RFC2046]. Note: The mime type of image/jpeg is really an image format, not a file format. JFIF and EXIF are file formats that contain image/jpeg image format data, and some applications have their own formats that are similar to JFIF and EXIF but which are proprietary. None the less, the "image/jpeg" @*MimeType* value is used to identify these file types.
image/tiff	tiff/6.0	.tiff .tif	Tag Image File Format ▸ [RFC3302] **Note:** The image/tiff MIME @*MediaType* is assumed to be TIFF Revision 6.0 as defined in detail by Adobe in ▸ [TIFF6]. TIFF/IT is a different MIME type.
multipart/related		.mjd .mjm	Multipart/Related with **JDF** as the first part ▸ [RFC2387]

Table G.1: MimeType Attribute Values (IANA Registered) (Sheet 3 of 3)

MIMETYPE	SAMPLE MIMETYPE-VERSION	FILE EXTENSION	DESCRIPTION [IANA-MT] INDICATES IANA REGISTRATION
x-world/x-vrml New in JDF 1.4			

▸ Table G.2 MimeType and File Type Combinations lists the @*MimeType* values that are file types assigned by CIP4 (as opposed to MIME Media Types which are registered with IANA) and possible @*MimeTypeVersion* values commonly used in **JDF** applications. A blank @*MimeTypeVersion* table entry indicates that there is no recognized version number for the @*MimeType*. ▸ Table G.2 MimeType and File Type Combinations also lists associated RECOMMENDED file name extensions values. A blank file extension column entry indicates that there is no recognized file name extension for the @*MimeType*. The @*Compression* values that do not have a corresponding IANA MIME type are also assigned a file type value, so that a file that is so compressed or encoded has an appropriate @*MimeType* value for the file, as shown in the table below.

Table G.2: MimeType and File Type Combinations (Sheet 1 of 3)

MIMETYPE	FILE EXTENSION	DESCRIPTION [IANA-MT] INDICATES IANA REGISTRATION
Base64	.mme	Base64 — A format for encoding arbitrary binary information for transmission by electronic mail. ▸ [RFC3548]
Compress		Compress — UNIX compression ▸ [RFC1977].
DCS	.eps	Document Color Separation (DCS), version 2.0. ▸ [DCS2.0]
Deflate		Deflate — The file is compressed using ZIP public domain compression format ▸ [RFC1951].
GZip	.gz	Gzip — GNU zip compression technology ▸ [RFC1952].
ICC Profile Deprecated in JDF 1.4	.icc .icm	International Color Consortium (ICC) File Format for Color Profiles taken from the binary coded decimal Profile Header Profile Version Number field (bytes 8 through 11) ▸ [ICC.1] **Deprecation note:** Starting with **JDF** 1.4 this MimeType becomes "application/vnd.iccprofile". See ▸ Table G.1 MimeType Attribute Values (IANA Registered).
MacBinary	.bin	MacBinary — An encoding format that combines the two forks of a Mac file, together with the file information (Name, Creator Application, File Type, etc.) into a single binary data stream that is suitable for storage or transferring through non-Mac systems. ▸ [macbinary]
Tar	.tar	UNIX packaging format.
TIFF/IT	.fp	TIFF/IT ▸ [ISO12639:2004] — Full Page — baseline **Note:** The file format TIFF/IT SHALL NOT use the "application/tiff" @*MimeType*. The "image/tiff" @*MimeType* conforms to baseline TIFF 6.0 ▸ [RFC3302], whereas TIFF/IT does not conform to TIFF 6.0. Consequently, the widely-deployed TIFF 6.0 readers are not able to read TIFF/IT. The ▸ [RFC3302] requires that an RFC be published in order to extend image/tiff with a parameter that would be needed in order to distinguish TIFF/IT from TIFF. There is no plan by the ISO committee that oversees TIFF/IT to register TIFF/IT with either a parameter to image/tiff or as new separate MIME type. Therefore, TIFF/IT will use the @*FileType* Attribute instead of the @*MimeType* Attribute.
TIFF/IT	.ct	TIFF/IT ▸ [ISO12639:2004] — Continuous Tone picture data — baseline
TIFF/IT	.lw	TIFF/IT ▸ [ISO12639:2004] — Continuous Line art — baseline
TIFF/IT	.hc	TIFF/IT ▸ [ISO12639:2004] — High-resolution Continuous tone image data — baseline
TIFF/IT	.mp	TIFF/IT ▸ [ISO12639:2004] — monochrome picture image data — baseline
TIFF/IT	.bp	TIFF/IT ▸ [ISO12639:2004] — Binary Picture image data — baseline

Table G.2: MimeType and File Type Combinations (Sheet 2 of 3)

MIMETYPE	FILE EXTENSION	DESCRIPTION [IANA-MT] INDICATES IANA REGISTRATION
TIFF/IT	.bl	TIFF/IT ▸[ISO12639:2004] — Binary Line art image data — baseline
TIFF/IT	.fp	TIFF/IT ▸[ISO12639:2004] — Full Page — profile 1
TIFF/IT	.ct	TIFF/IT ▸[ISO12639:2004] — Continuous Tone picture data — profile 1
TIFF/IT	.lw	TIFF/IT ▸[ISO12639:2004] — Color Line art data — profile 1
TIFF/IT	.hc	TIFF/IT ▸[ISO12639:2004] — High-resolution Continuous tone image data — profile 1
TIFF/IT	.mp	TIFF/IT ▸[ISO12639:2004] — monochrome picture image data — profile 1
TIFF/IT	.bp	TIFF/IT ▸[ISO12639:2004] — Binary Picture image data — profile 1
TIFF/IT	.bl	TIFF/IT ▸[ISO12639:2004] — Binary Line art image data — profile 1
TIFF/IT	.fp	TIFF/IT ▸[ISO12639:2004] — Full Page — baseline **Note:** This entry and following ones were created in the context of ▸[ISO12639:2004], whereas preceding entries were created in the context of the 1998 version of ▸[ISO12639:2004]
TIFF/IT	.ct	TIFF/IT ▸[ISO12639:2004] — Continuous Tone picture data — baseline
TIFF/IT	.lw	TIFF/IT ▸[ISO12639:2004] — Color Line art data — baseline
TIFF/IT	.hc	TIFF/IT ▸[ISO12639:2004] — High-resolution Continuous tone image data — baseline
TIFF/IT	.mp	TIFF/IT ▸[ISO12639:2004] — monochrome picture image data — baseline
TIFF/IT	.bp	TIFF/IT ▸[ISO12639:2004] — Binary Picture image data — baseline
TIFF/IT	.bl	TIFF/IT ▸[ISO12639:2004] — Binary Line art image data — baseline
TIFF/IT	.sd	TIFF/IT ▸[ISO12639:2004]
TIFF/IT	.fp	TIFF/IT ▸[ISO12639:2004] — Full Page — profile 1
TIFF/IT	.ct	TIFF/IT ▸[ISO12639:2004] — Continuous Tone picture data — profile 1
TIFF/IT	.lw	TIFF/IT ▸[ISO12639:2004] — Color Line art data — profile 1
TIFF/IT	.hc	TIFF/IT ▸[ISO12639:2004] — High-resolution Continuous tone image data — profile 1
TIFF/IT	.mp	TIFF/IT ▸[ISO12639:2004] — monochrome picture image data — profile 1
TIFF/IT	.bp	TIFF/IT ▸[ISO12639:2004] — Binary Picture image data — profile 1
TIFF/IT	.bl	TIFF/IT ▸[ISO12639:2004] — Binary Line art image data — profile 1. **Note:** There is no TIFF/IT P1 conformance level of SD in ▸[ISO12639:2004]
TIFF/IT	.fp	TIFF/IT ▸[ISO12639:2004] — Full Page — profile 2
TIFF/IT	.ct	TIFF/IT ▸[ISO12639:2004] — Continuous Tone picture data — profile 2
TIFF/IT	.lw	TIFF/IT ▸[ISO12639:2004] — Color Line art data — profile 2
TIFF/IT	.hc	TIFF/IT ▸[ISO12639:2004] — High-resolution Continuous tone image data — profile 2
TIFF/IT	.mp	TIFF/IT ▸[ISO12639:2004] — monochrome picture image data — profile 2
TIFF/IT	.bp	TIFF/IT ▸[ISO12639:2004] — Binary Picture image data — profile 2
TIFF/IT	.bl	TIFF/IT ▸[ISO12639:2004] — Binary Line art image data — profile 2

Table G.2: MimeType and File Type Combinations (Sheet 3 of 3)

MIMETYPE	FILE EXTENSION	DESCRIPTION [IANA-MT] INDICATES IANA REGISTRATION
TIFF/IT	.sd	TIFF/IT ▸ [ISO12639:2004]
Type 1 Font	.pfa .pfb	Type 1 Font ▸ [type1font]
True Type Font	.ttf	True Type Font ▸ [truetypefont]
Open Type Font	.otf	Open Type Font ▸ [opentypefont]
UUEncoded	.uue	Uuencode — A set of encoding algorithms for converting files into a series of 7-bit ASCII characters that can be transmitted over the Internet. Originally, uuencode stood for Unix-to-Unix encode, but it has since become a universal protocol used to transfer files between different platforms such as Unix, Windows and Macintosh. Uuencoding is especially popular for sending Email attachments. ▸ [uuencode]
ZLIB		ZLIB — ZLIB compression ▸ [RFC1950]

Appendix H

Dynamic String Generation

New in JDF 1.3

JDF specifies a set of @*XXXFormat* @*XXXTemplate* pairs that allow dynamic generation of strings.

The function defined when using the attributes @*XXXFormat* and @*XXXTemplate* is based on the standard C printf() function. (See ▸ [K&R].) @*XXXFormat* is the first argument and @*XXXTemplate* is a comma-separated list of the additional arguments. @*XXXTemplate* MAY contain unary operators: "+" and "-", binary operators: "+", "-", "*", "/" and "%", as well as parentheses: "(" and ")", which are evaluated using standard C-operator precedence and the variables defined in the following table which include any valid partition key of a partitioned resource.

When evaluating a mathematical expression involving variables, the format evaluation will convert variable values as necessary into the long float values needed to result in a numeric result, and then convert that result if necessary based on what the format specifier is. For example, if:

1 A template contains "Metadata0 * Metadata1", and @*Metadata0* = "5", and @*Metadata1* = "1.5".

2 Both partition key values will be converted into long float values before the multiplication is performed

3 The result will be 7.5

Then, if the format uses "%d", the value will be truncated to 7. If the format uses "%s", the string "7.5" will be used. Finally, if the format uses "%f", the long float value 7.5 will be used.

If a mathematical operation is attempted on a non numeric value, the results are undefined.

Modification note: Starting with **JDF** 1.4, values from @*ShowList* are added to ▸ Table H.1 Predefined variables used in @XXXTemplate below and when two values differ only in case, the one starting with an uppercase letter is deprecated.

Table H.1: Predefined variables used in @XXXTemplate (Sheet 1 of 3)

NAME	DESCRIPTION
<a Partition Key>	Any partition key that is a value of @*PartIDKeys* in ▸ Table 3.24 Partitionable Resource Element.
<any Imposition Variable> New in JDF 1.4	The value of any variable defined by the **Imposition** process (see ▸ Section 6.3.18.2 Variables for Automated Imposition).
AcknowledgeType	Corresponds to the **JMF** @*AcknowledgeType* in the ***Acknowledge*** message. See ▸ Section 5.2.5 Acknowledge.
all	Selects all matching elements. Valid only when ***FileSpec*** is used as an input resource.
Amount New in JDF 1.4	Amount of the product that was produced.
CustomerID	@*CustomerID*.
Date	Current @*Date* in ▸ [ISO8601:2004] format.
DeviceID New in JDF 1.4	ID of the device. This is a unique name within the workflow.
DeviceName	@DescriptiveName of the device.
element	Integer iterator over all elements in a given page. Restarts at 0 for each page.
EndTime New in JDF 1.4	Actual end time of the job.
Error New in JDF 1.4	Errors that happened during the job.
ErrorStats New in JDF 1.4	Statistics on errors that happened during execution.

Table H.1: Predefined variables used in @XXXTemplate (Sheet 2 of 3)

NAME	DESCRIPTION
ExposedMediaName New in JDF 1.4	@*DescriptiveName* of the exposed media (e.g., plate or proof that is being imaged).
FriendlyName New in JDF 1.4	@*FriendlyName* of the device.
GeneralID:XXX New in JDF 1.4	***GeneralID***/@*IDValue* of a ***GeneralID***[@*IDUsage* = "XXX"] For example if @*Format* = "%i" and @*Template*= "GeneralID:foo" then for: <GeneralID IDUsage="foo" IDValue="1"/> the extracted value is 1.
Generated	System generated file name.
i	Integer iterator over all files produced by this process. 0-based numbering.
input	Local file name of the input file. Valid only when ***FileSpec*** is used as an output resource.
jobID	Job ID string.
JobID Deprecated in JDF 1.4	@*JobID* of the node that is executing. **Deprecation note:** Starting with **JDF** 1.4, use "jobID"**.** Before **JDF** 1.4, "JobID" was only for ***JobField***/@*ShowList*
jobName	***JDF***/@*DescriptiveName* of the node that is being processed.
JobName Deprecated in JDF 1.4	"DescriptiveName" of the node that is executing. **Deprecation note:** Starting with **JDF** 1.4, use "jobName"**.** Before **JDF** 1.4, "JobName" was only for ***JobField***/@*ShowList*.
jobPartID	@*JobPartID* string.
JobRecipientName New in JDF 1.4	Name of the recipient of the job.
JobSubmitterName New in JDF 1.4	Name of the submitter of the job.
LayPartCount2in1 New in JDF 1.3	Number of partitions at level 2 within partition level 1 (the base) within the ***Layout*** Resource. When using the RECOMMENDED @*SignatureName* / @*SheetName* partition keys, this equates to the number of signatures within the ***Layout***. All other variants of @*LayPartCount*<n>*in*<m> MAY be used, where <n> is an integer greater than <m>, and where <n> does not exceed the depth of the partition tree within the ***Layout***.
LayPartIndex2in1 New in JDF 1.3	Index (1-based) of partition level 2 within partition level 1 (the base) within the ***Layout*** Resource. When using the RECOMMENDED @*SignatureName*/@*SheetName* partition keys, this equates to the signature number within the ***Layout***. All other variants of @*LayPartIndex*<n>*in*<m> MAY be used, where <n> is an integer greater than <m>, and where <n> does not exceed the depth of the partition tree within the ***Layout***.
MediaBrand New in JDF 1.4	Brand of the media that is being printed.
MediaType New in JDF 1.4	"DescriptiveName" of the media that is being printed.
MoonPhase New in JDF 1.4	Phase of the moon at the @*StartTime* of the job.
Operator New in JDF 1.4	Name of the operator.
OperatorText New in JDF 1.4	Text from the operator as defined in @*OperatorText*.

Table H.1: Predefined variables used in @XXXTemplate (Sheet 3 of 3)

NAME	DESCRIPTION
page	Integer iterator over the page number of a document. This value is equivalent to value *r* (below) for the case that each run contains exactly one page.
PrintQuality New in JDF 1.4	The quality of the printout. (High, Normal, Draft or Device specific name)
ProoferProfileName New in JDF 1.4	Name of the ICC profile for the proofing device.
PressProfileName New in JDF 1.4	Name of the ICC profile for the final printing (used as intermediate space during proofing).
r	Integer iterator over all ***RunList*** Partitions with a partition key of "Run" in an input ***RunList***.
ri	Integer iterator over all indices in an input "Run" of a ***RunList***. This index is equivalent to looping over a @*RunIndex*.
Resolution New in JDF 1.4	Output resolution.
ResolutionX New in JDF 1.4	Output resolution in X direction.
ResolutionY New in JDF 1.4	Output resolution in Y direction.
ScreeningFamily New in JDF 1.4	Name of the screening family of the output.
sep Deprecated in JDF 1.4	Separation as defined in the separation partition keys of a partitioned resource. **Deprecation note:** Starting with **JDF** 1.4, use the partition key "Separation".
SheetNum New in JDF 1.3	Integer iterator over the sheet number of a document.
StartTime New in JDF 1.4	Actual start time of the job
surf Deprecated in JDF 1.3	Surface string, "Front" or "Back". **Deprecation note:** Starting with **JDF** 1.3, use the partition key "Side" instead.
SystemRoot	Root of system directory file structure. This token provides an operating system, independent way to refer to the root.
TileX	X coordinate of a Tile.
TileY	Y coordinate of a Tile.
Time	Current @*Time* in ▸[ISO8601:2004] format.
TotalPagesInDoc New in JDF 1.3	Total # of pages in a document.
UserText New in JDF 1.4	For ***JobField***, "UserText" references user-defined text in ***JobField***/@*UserText*.
Warning New in JDF 1.4	Warnings that happened during the job. Warnings don't lose information in the resulting job, while errors do.

Example H.1: @FileTemplate and @FileFormat

With @*JobID*="j001" and a ***RunList*** defining 2024 created files, this example will iterate over all created files and place them into:

```
"file://myserver.mydomain.com/next/j001/0000/m0000.pdf"
…
"file://myserver.mydomain.com/next/j001/0020/m0023.pdf"
```

```
<RunList Class="Parameter" ID="R1" Status="Available">
    <LayoutElement>
        <FileSpec FileFormat="file://myserver.mydomain.com/next/%s/%4.i/m%4.i.pdf"
                  FileTemplate="JobID,i/100,i%100"/>
    </LayoutElement>
</RunList>
```

Pagination Catalog

This appendix provides a set of diagrams that explain how pages are arranged in groups when preparing to print on the surfaces of large sheets. The diagrams show a wide range of folding patterns to be used before binding. The folding patterns are specified in the **JDF** Fold Catalog (see ▸ Section Figure 8-33: Fold catalog part 1 and ▸ Section Figure 8-34: Fold catalog part 2) which describes how to paginate single-sheet bindery signatures

The purpose of this appendix is to provide a reference for all agents involved in the use of imposition techniques in the printing industry.

I.1 How to interpret the diagrams

I.1.1 Legend

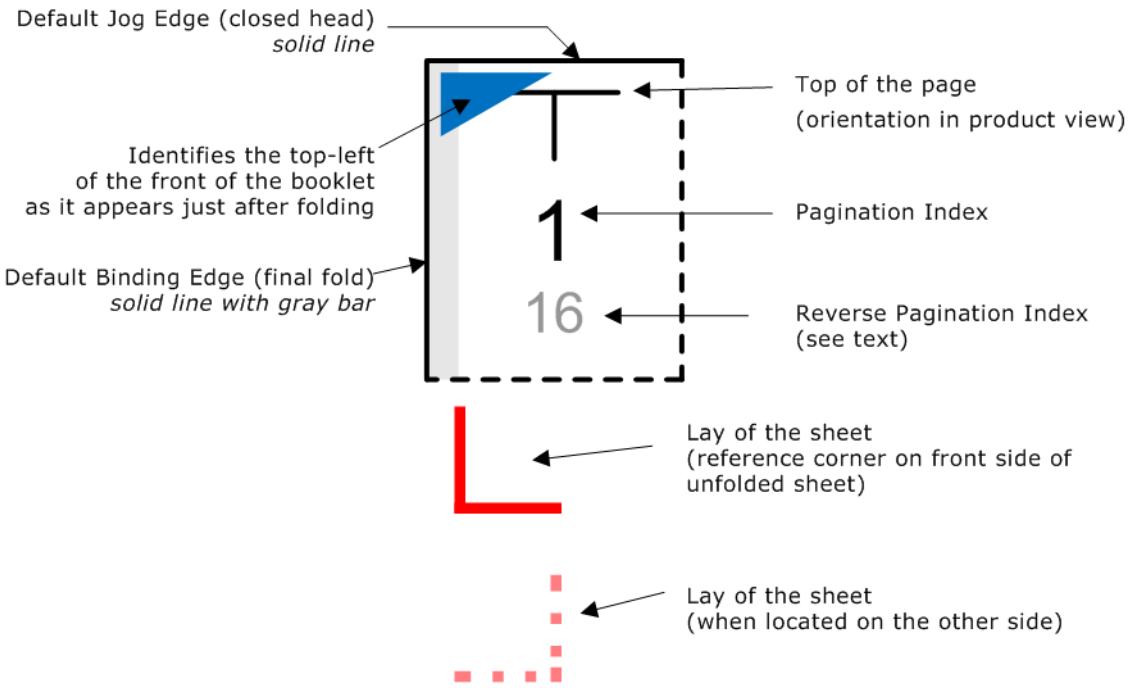
This appendix describes the structure and arrangement of bindery signatures into pagination schemes, which divide sheet surfaces into grids of rectangular areas to be filled by appendix pages during imposition process. These arrangements are the consequence of manipulations made on the sheets by folding, trimming and binding them in order to make booklets ready for assembly.

This appendix uses diagrams to describe the pagination schemes. Each diagram shows a side of an unfolded sheet, illustrating how it is divided into "signature cells". All cells are usually of the same size, allowing the entire sheet to be divided into equal portions, with each portion covering the whole area between surrounding folds. A signature cell is the space that "receives" a single document page and surrounding margins that are part of the gutters.

Each cell shown in the diagram displays how to orient the document page that is to be imposed there, and specifies the index of the page to be imposed. This means that the resulting booklet will have pages that are properly ordered and properly oriented in the product reader view, according to the default values defined in the this [**JDF**] specification.
Note: Pagination indexes start at number 1.

The diagrams also show the pagination to be used when pages are flowed in reverse order because of different binding options, see ▸ Section I.1.3 Settings that Modify the Pagination Schemes and ▸ Section I.1.4 Getting a Specific Pagination Scheme.

Figure I-1: Legend for Interpreting Diagrams



Folding sequences are described using the same notation found in this specification:

- ↑ means "left-hand part goes over right-hand part" or "bottom part goes over top part";
- ↓ means "left-hand part goes under right-hand part" or "bottom part goes under top part";
- The size of the part being folded is expressed as a fraction of unfolded sheet's size;
- First fold is always left-to-right, a "+" sign is used to toggle between left-to-right and bottom-to-top folding directions.

I.1.2 Meaning of a Pagination Scheme

The diagrams in ▸ Section I.2 Pagination Diagrams show the configuration of the page cells that occurs when the bindery signature is specified in the **JDF** file using ***BinderySignature***/@*FoldCatalog*. Each arrangement corresponds to the "fold catalog identifier" that is shown in the left column of the diagram. This identifier is used as a value of the ***BinderySignature***/@*FoldCatalog* attribute, and refers to the recipe used to fold the sheet.

The pagination indexes shown in the diagram correspond to the imposition order, starting with 1, up to the number of pages in a booklet. This index does not correspond to the actual page numbers that will be imposed on the sheets, unless a finished product is made of a single booklet and the first page is numbered "1". These numbers specify the order that pages are imposed into signature cells, from an array of pages associated to a booklet.

These numbers have meaning only if the folded sheet are used as a booklet intended for binding or assembling (i.e., trimmed after folding). In many cases (i.e., accordion folds) the result is mostly theoretical, as those folds are not intended for such use in real life.

When multiple booklets are assembled together, the imposition indexes have to be translated into numbers referring to the list of source document pages. This is calculated using the parameters found in the ***Assembly*** resource and the ***StrippingParams***/@*SectionList* parameter.

The numbers and page orientation shown in the diagram correspond to the finished product view in reader's perspective. The "top of the page", which is a product attribute, does not always correspond to the "head" of the booklet, which is a production attribute. Note, that the finished view is NOT a reference for locating the production measurements (head/foot/face trim and bleed sizes, spine size, overfolds, etc.) as their position is set by the @*BindingOrientation* attribute, independently of the final page flow.

I.1.3 Settings that Modify the Pagination Schemes

I.1.3.1 BindingOrientation

When a sheet has been folded, the last fold is recognized as being the "binding edge", and a perpendicular edge is known as the "jog edge". Both edges join together around a corner known as the "reference corner", which appears at the bottom left of the folded sheet (the last fold always appear either at the bottom or at the left when using the fold catalog).

The attribute ***BinderySignature***/@*BindingOrientation* may be set to indicate that the reference corner is displaced for production purposes. This manipulation is not made on the folded sheet. Only the "virtual" corner is changed, after edges had been identified. This means that the edges that are recognized as "binding" and "jog" are found at new places on the folded sheet, changing the location of the spine, head, face and foot on the booklet before pagination can be applied.

This @*BindingOrientation* attribute is very special because it has two default values, depending on the type of signature being defined: "Flip0" for single-row grids (no closed head), "Rotate0" for all other grids. This particularity reflects common practice of recognizing the jog edge to be at the top of signatures without closed heads.

The diagrams in ▸ Section I.2 Pagination Diagrams are based on these default values. If that parameter is set to another value, use the tables in ▸ Section I.1.4 Getting a Specific Pagination Scheme to convert the pagination scheme to reflect this change.

I.1.3.2 Binding and Jog Sides

To make the bindery signatures be assembled together into a finished product, the pages must have been imposed in the order and orientation needed to get the right reader's perspective after assembling. Before setting the page numbers and orientation in cells to obtain the expected result, the "assembly" is virtually rotated and flipped to make the binding and jog edges be placed as requested, when looking at the very first page in the reader view.

The diagrams in this appendix show the pagination scheme where the front page of the booklet is oriented so that the binding side appears at the left, and the jog side appears at the top, according to the default parameters defined by this [**JDF**] specification. For other values, transformations must be applied on the diagrams to get the right scheme.

These settings are found in the Assembly resource that is used to describe how the booklet is assembled:

- ***Assembly***/@*BindingSide*
- ***Assembly***/@*JogSide*

If one or both of these attributes is set, use the tables in ▸ Section I.1.4 Getting a Specific Pagination Scheme to convert the pagination scheme to reflect this change.

The settings ***BinderySignature***/@*BindingEdge* and ***BinderySignature*** /@*JogEdge* are ignored because they affect production view only. However, if ***Assembly***/@*Order*="None", then ***BinderySignature***/@*BindingEdge* and ***BinderySignature***/@*JogEdge* must be used as replacement settings, because assembly parameters must be ignored in that case, and production view becomes the product view.

I.1.4 Getting a Specific Pagination Scheme

I.1.4.1 Using the Settings to Find the Needed Scheme Transformation

Use the table below to locate the name of the scheme transformation to be applied on the diagram, according to the @*BindingSide*, @*JogSide* and @*BindingOrientation* settings. Default values for these settings are underlined in the table.

The obtained transformation is identified by a "scheme name", which refers to the table in Section I.1.4.2 Scheme Transformations where all pagination schemes are explained, based on the diagrams of the chapter 2.

Table I.1: Schemes Names for Binding Orientations

@BINDING SIDE	@JOGSIDE	SCHEME NAME (FOR @BINDINGORIENTATION SETTING SHOWN IN HEADER) (FOR SINGLE-ROW SIGNATURES: USE COLUMN FOOTERS			
		ROTATE0 *FLIP0*	ROTATE90 *FLIP90*	ROTATE180 *FLIP180*	ROTATE270 *FLIP270*
left	top	Rotate0 *Flip0*	Rotate270/90* *Flip90/270**	Rotate180 *Flip180*	Rotate90/270* *Flip270/90**
	bottom	Flip0 *Rotate0*	Flip270/90* *Rotate90/270**	Flip180 *Rotate180*	Flip90/270* *Rotate270/90**
right	top	Flip180 *Rotate180*	Flip90/270* *Rotate270/90**	Flip0 *Rotate0*	Flip270/90* *Rotate90/270**
	bottom	Rotate180 *Flip180*	Rotate90/270* *Flip270/90**	Rotate0 *Flip0*	Rotate270/90* *Flip90/270**
top	left	Flip90 *Rotate90*	Flip180/0 *Rotate180/0*	Flip270 *Rotate270*	Flip0/180 *Rotate0/180*
	right	Rotate90 *Flip90*	Rotate0/180 *Flip0/180*	Rotate270 *Flip270*	Rotate180/0 *Flip180/0*
bottom	left	Rotate270 *Flip270*	Rotate180/0 *Flip180/0*	Rotate90 *Flip90*	Rotate0/180 *Flip0/180*
	right	Flip270 *Rotate270*	Flip0/180 *Rotate0/180*	Flip90 *Rotate90*	Flip180/0 *Rotate180/0*
		FLIP0 *ROTATE0*	FLIP270 *ROTATE270*	FLIP180 *ROTATE180*	FLIP90 *ROTATE90*

*** Important note**: If binding edges appear horizontally on the diagram, the numbers must be swapped in the scheme names indicated by an asterisk ("Rotate90/270" would become "Rotate270/90"). This happens because the direction of rotation is reversed in those cases (e.g., F8-7, F12-7, F16-10, etc.).

I.1.4.2 Scheme Transformations

Table I.2: Transformations for each Scheme (Sheet 1 of 2)

SCHEME NAME	GETTING THE PAGINATION SCHEME (USING THE DIAGRAM)		
	PAGE NUMBERS	LEFT-BOUND PAGE	RIGHT-BOUND PAGE
Rotate0	Normal	as shown	as shown
Rotate0/180	Normal	as shown	Rotate 180°
Rotate90	Normal	Rotate 90° counterclockwise	Rotate 90° counterclockwise
Rotate90/270	Normal	Rotate 90° counterclockwise	Rotate 90° clockwise
Rotate180	Normal	Rotate 180°	Rotate 180°
Rotate180/0	Normal	Rotate 180°	as shown

Table I.2: Transformations for each Scheme (Sheet 2 of 2)

SCHEME NAME	GETTING THE PAGINATION SCHEME (USING THE DIAGRAM)		
	PAGE NUMBERS	LEFT-BOUND PAGE	RIGHT-BOUND PAGE
Rotate270	Normal	Rotate 90° clockwise	Rotate 90° clockwise
Rotate270/90	Normal	Rotate 90° clockwise	Rotate 90° counterclockwise
Flip0	Reverse	Rotate 180°	Rotate 180°
Flip0/180	Reverse	Rotate 180°	as shown
Flip90	Reverse	Rotate 90° clockwise	Rotate 90° clockwise
Flip90/270	Reverse	Rotate 90° clockwise	Rotate 90° counterclockwise
Flip180	Reverse	as shown	as shown
Flip180/0	Reverse	as shown	Rotate 180°
Flip270	Reverse	Rotate 90° counterclockwise	Rotate 90° counterclockwise
Flip270/90	Reverse	Rotate 90° counterclockwise	Rotate 90° clockwise

In the "Page Numbers" column in the above table, the value "Normal" refers to the main numbers in ▸ Table I.3 Original Diagram, ▸ Table I.5 Original Diagram and ▸ Table I.7 Pagination Diagrams, while "Reverse" refers to the smaller numbers in gray. Note that the small gray numbers have been omitted from ▸ Table I.4 Horizontal Binding Edges and ▸ Table I.6 Vertical Binding Edges. If they were shown, they would be the same as for ▸ Table I.3 Original Diagram and ▸ Table I.5 Original Diagram, respectively.

The "Left-Bound Pages" column in the above table refers to odd pages in the tables below, when looking at the main numbers. The "Right-Bound Pages" column in the above table refers to even pages.

Important note: When a page is rotated 90° (clockwise or counterclockwise), this rotation is made **inside** the signature cell. The cell itself is not rotated because the folding operation remains the same. This means that the aspect ratio of the page must have been designed accordingly. Observe this situation in the examples in the next section.

I.1.5 Examples

The following examples describe the default orientation and pagination of ***BinderySignature*** depending on @*BindingOrientation*.

Note: The orientation of the final fold is defined in the production coordinate system. The binding side of the final product always defaults to left and is modified by @*BindingOrientation* regardless of whether the final fold in production is horizontal or vertical.

I.1.5.1 Signature with Horizontal Binding Edges

The examples below show how to read the diagrams after applying the transformations explained previously. Each diagram is an interpretation of the lay-side diagram defined for fold catalog F8-7, indicating the scheme name above it.

Table I.3: Original Diagram

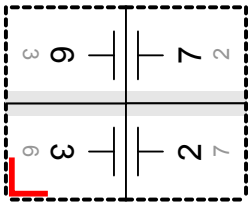
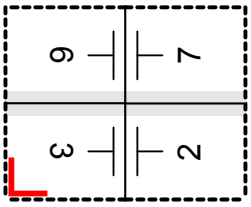
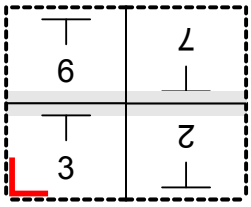
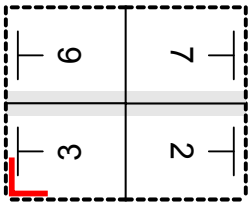
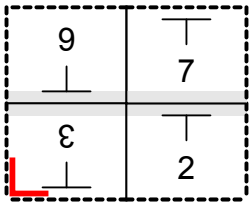
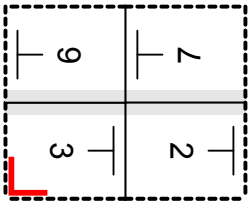
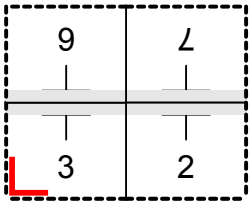
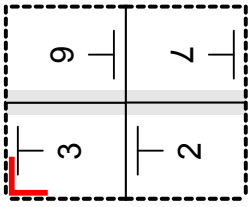
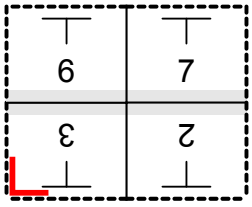
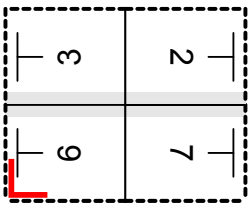
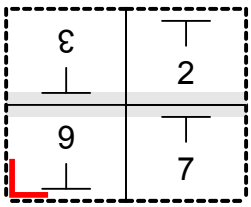
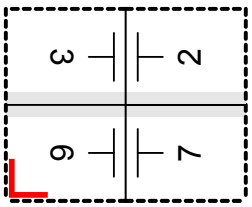
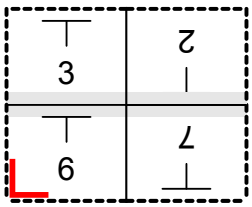
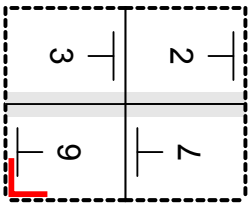
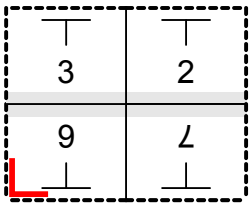
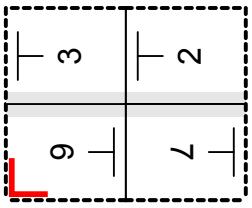
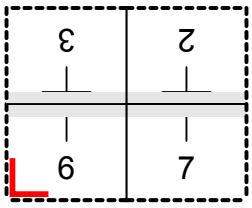
Rotate0


Table I.4: Horizontal Binding Edges

Rotate0	Rotate90	Rotate180	Rotate270
			
Rotate0/180	**Rotate90/270**	**Rotate180/0**	**Rotate270/90**
			
Flip0	**Flip90**	**Flip180**	**Flip270**
			
Flip0/180	**Flip90/270**	**Flip180/0**	**Flip270/90**
			

I.1.5.2 Signature with Vertical Binding Edges

The examples below show how to read the diagrams after applying the transformations explained previously. Each diagram is an interpretation of the lay-side diagram defined for fold catalog F12-11, indicating the scheme name above it.

.

Table I.5: Original Diagram

Rotate0

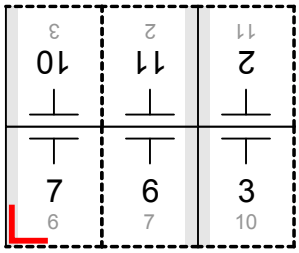
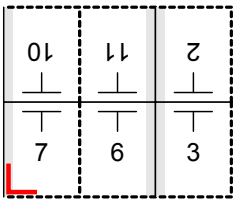
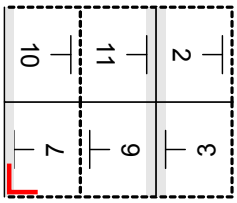
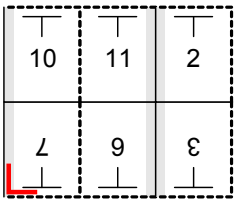
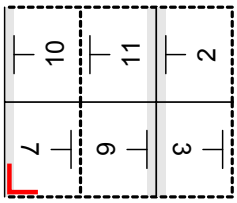
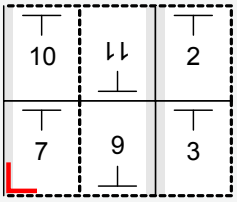
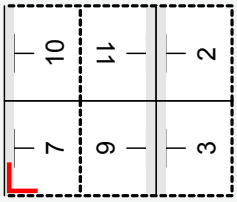
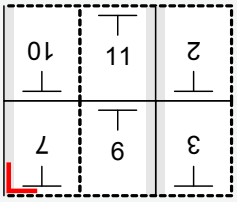
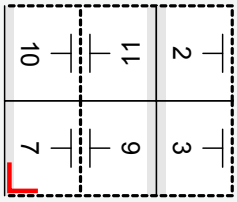
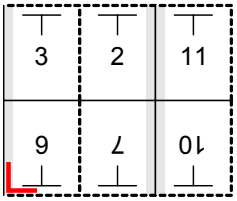
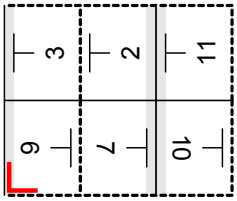
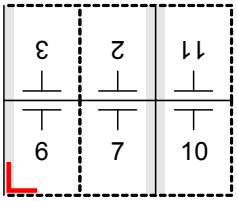
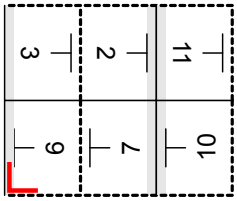
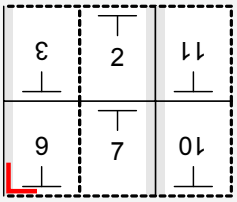
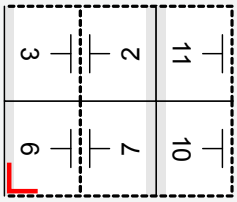
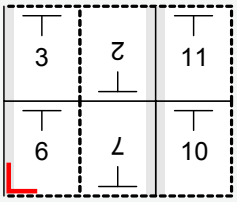
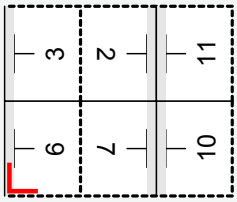


Table I.6: Vertical Binding Edges

Rotate0	Rotate90	Rotate180	Rotate270
			
Rotate0/180	Rotate90/270	Rotate180/0	Rotate270/90
			
Flip0	Flip90	Flip180	Flip270
			
Flip0/180	Flip90/270	Flip180/0	Flip270/90
			

I.2 Pagination Diagrams

Table I.7: Pagination Diagrams (Sheet 1 of 32)

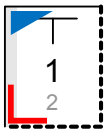
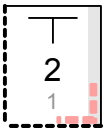
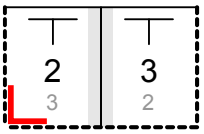
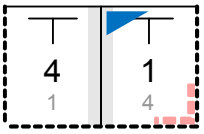
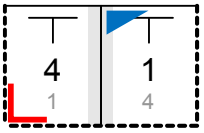
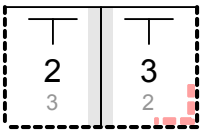
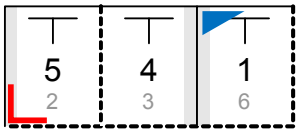
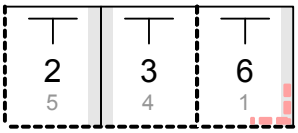
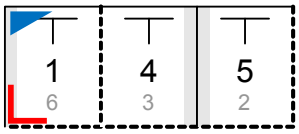
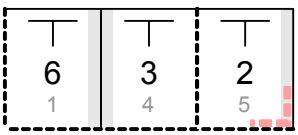
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F2-1	**1x1**	
(no fold)		
F4-1	**2x1**	
$\nearrow^{1}/_{2}$		
F4-2	**2x1**	
$\searrow^{1}/_{2}$		
F6-1	**3x1**	
$\nearrow^{1}/_{3}\searrow^{1}/_{3}$		
F6-2	**3x1**	
$\searrow^{1}/_{3}\nearrow^{1}/_{3}$		

Table I.7: Pagination Diagrams (Sheet 2 of 32)

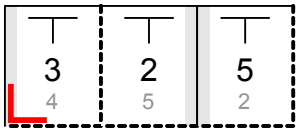
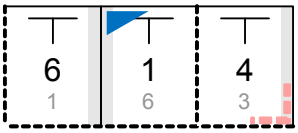
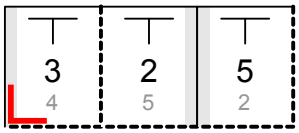
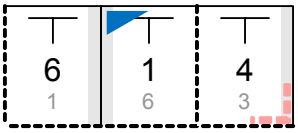
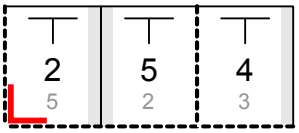
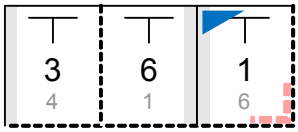
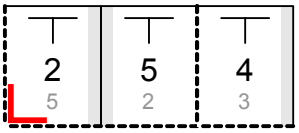
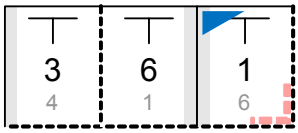
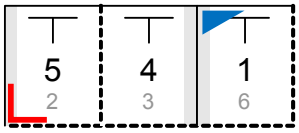
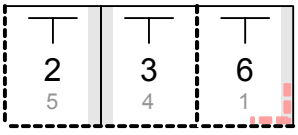
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F6-3	**3x1**	
$\uparrow^1/_4 \uparrow^1/_2$		
F6-4	**3x1**	
$\uparrow^1/_3 \uparrow^1/_3$		
F6-5	**3x1**	
$\uparrow^2/_3 \downarrow^1/_3$		
F6-6	**3x1**	
$\uparrow^3/_4 \downarrow^1/_4$		
F6-7	**3x1**	
$\uparrow^1/_4 \downarrow^1/_4$		

Table I.7: Pagination Diagrams (Sheet 3 of 32)

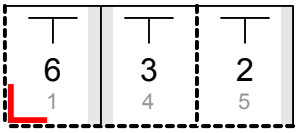
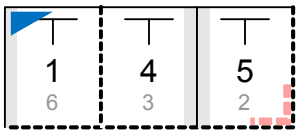
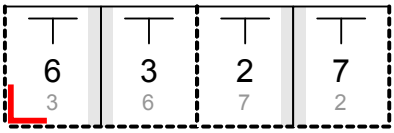
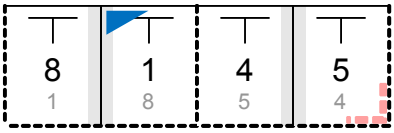
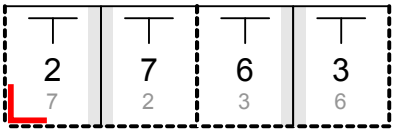
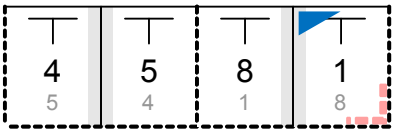
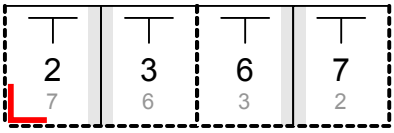
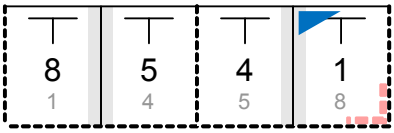
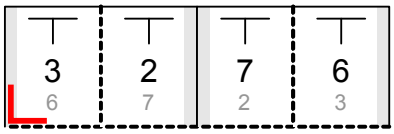
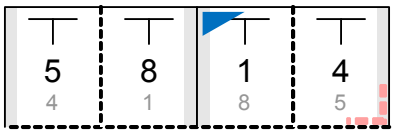
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F6-8	**3x1**	
$\uparrow^{2}/_{3}\ \uparrow^{1}/_{3}$		
F8-1	**4x1**	
$\uparrow^{1}/_{2}\ \uparrow^{1}/_{4}$		
F8-2	**4x1**	
$\uparrow^{1}/_{2}\ \downarrow^{1}/_{4}$		
F8-3	**4x1**	
$\uparrow^{1}/_{4}\ \downarrow^{1}/_{4}\ \uparrow^{1}/_{4}$		
F8-4	**4x1**	
$\uparrow^{1}/_{4}\ \uparrow^{1}/_{2}\ \downarrow^{1}/_{4}$		

Table I.7: Pagination Diagrams (Sheet 4 of 32)

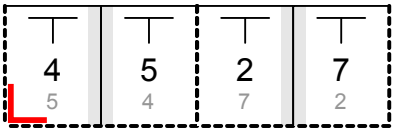
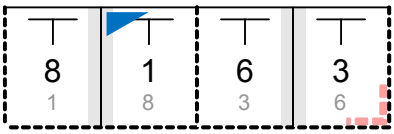
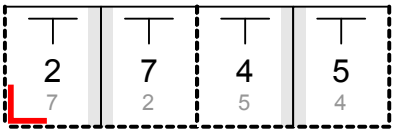
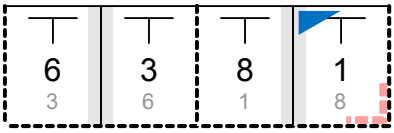
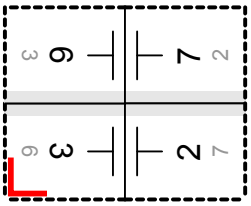
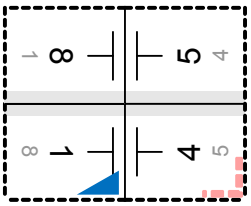
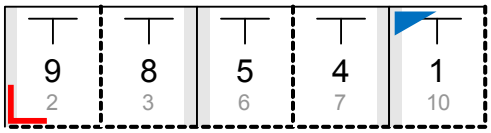
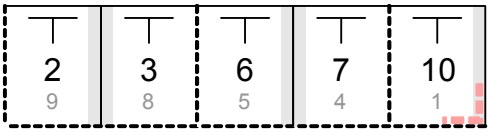
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F8-5	**4x1**	
$\uparrow^1/_4 \uparrow^1/_4 \uparrow^1/_4$		
F8-6	**4x1**	
$\uparrow^3/_4 \downarrow^1/_4 \downarrow^1/_4$		
F8-7	**2x2**	
$\uparrow^1/_2 + \uparrow^1/_2$		
F10-1	**5x1**	
$\uparrow^1/_5 \downarrow^1/_5 \uparrow^1/_5 \downarrow^1/_5$		

Table I.7: Pagination Diagrams (Sheet 5 of 32)

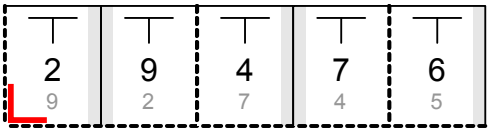
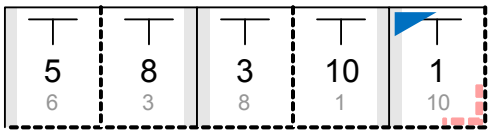
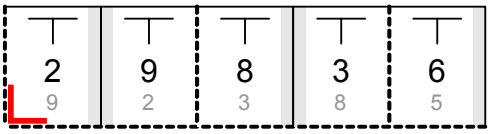
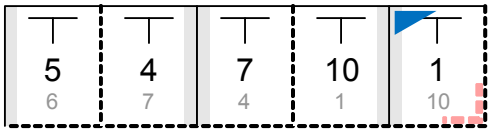
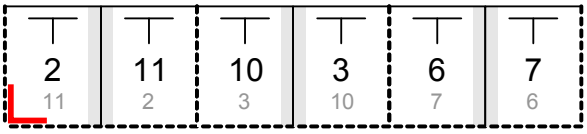
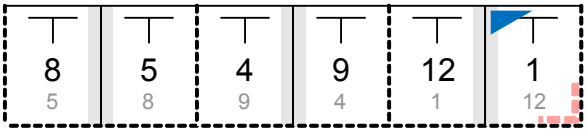
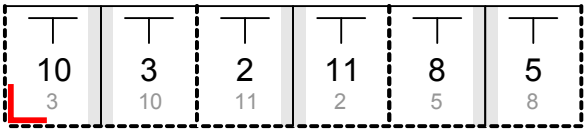
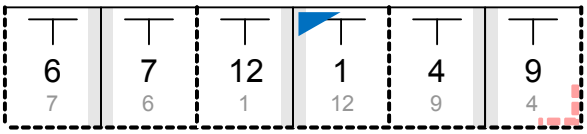
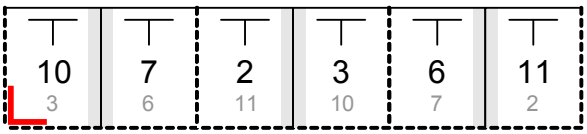
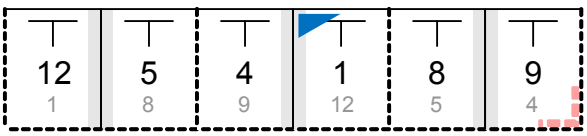
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F10-2	**5x1**	
$\uparrow^{4}/_{5}\downarrow^{1}/_{5}\downarrow^{1}/_{5}\downarrow^{1}/_{5}$		
F10-3	**5x1**	
$\uparrow^{2}/_{5}\downarrow^{2}/_{5}\uparrow^{1}/_{5}$		
F12-1	**6x1**	
$\uparrow^{1}/_{3}\downarrow^{1}/_{3}\uparrow^{1}/_{6}$		
F12-2	**6x1**	
$\uparrow^{1}/_{3}\uparrow^{1}/_{3}\downarrow^{1}/_{6}$		
F12-3	**6x1**	
$\uparrow^{1}/_{2}\downarrow^{1}/_{6}\uparrow^{1}/_{6}$		

Table I.7: Pagination Diagrams (Sheet 6 of 32)

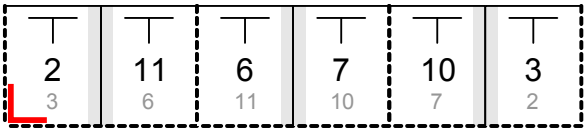
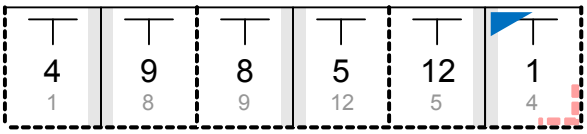
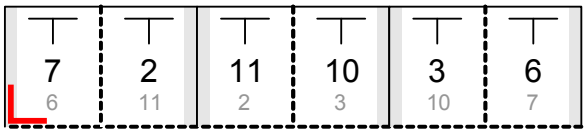
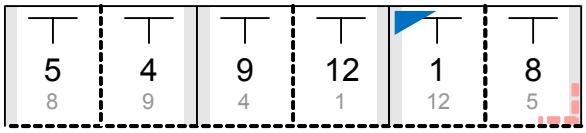
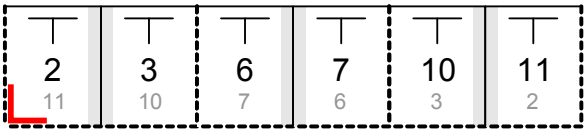
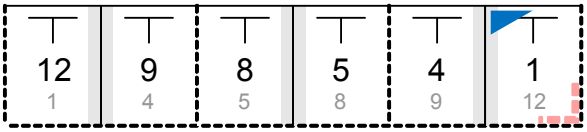
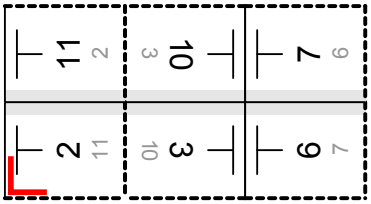
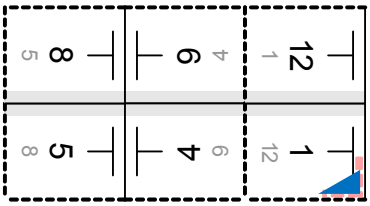
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F12-4	**6x1**	
$\uparrow^1/_2 \downarrow^1/_6 \downarrow^1/_6$		
F12-5	**6x1**	
$\uparrow^1/_2 \downarrow^1/_3 \uparrow^1/_6$		
F12-6	**6x1**	
$\uparrow^1/_6 \downarrow^1/_6 \uparrow^1/_6$ $\downarrow^1/_6 \uparrow^1/_6$		
F12-7	**3x2**	
$\uparrow^1/_3 \downarrow^1/_3 + \uparrow^1/_2$		

Table I.7: Pagination Diagrams (Sheet 7 of 32)

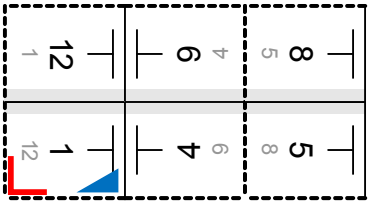
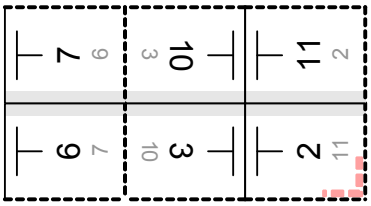
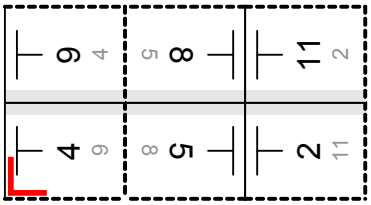
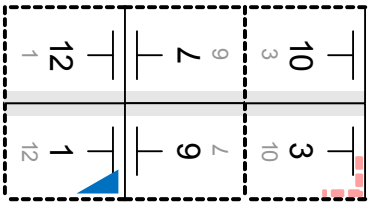
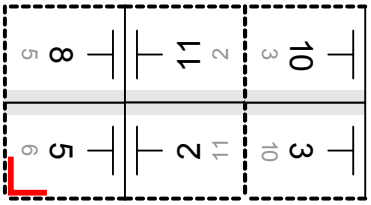
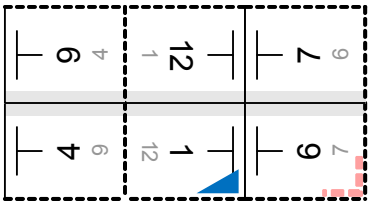
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F12-8	**3x2**	
$\nearrow^{2}/_{3}\ \nearrow^{1}/_{3} + \nearrow^{1}/_{2}$		
F12-9	**3x2**	
$\nearrow^{1}/_{3}\ \nearrow^{1}/_{3} + \nearrow^{1}/_{2}$		
F12-10	**3x2**	
$\nearrow^{2}/_{3}\ \searrow^{1}/_{3} + \nearrow^{1}/_{2}$		

Table I.7: Pagination Diagrams (Sheet 8 of 32)

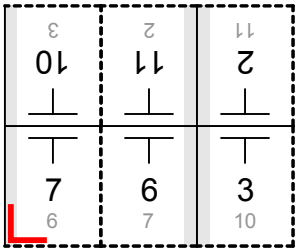
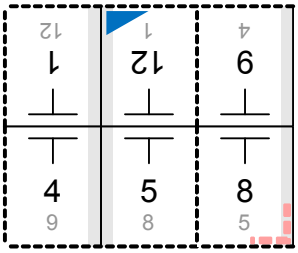
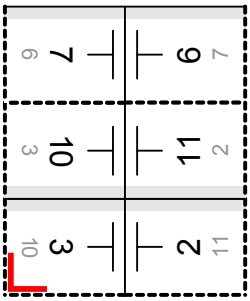
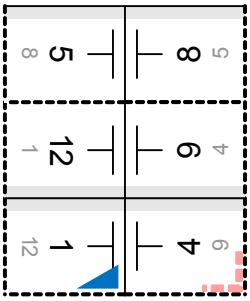
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F12-11	**3x2**	
$\uparrow^1/_3 + \uparrow^1/_2 + \uparrow^1/_3$		
F12-12	**2x3**	
$\uparrow^1/_2 + \uparrow^2/_3 \downarrow^1/_3$		

Table I.7: Pagination Diagrams (Sheet 9 of 32)

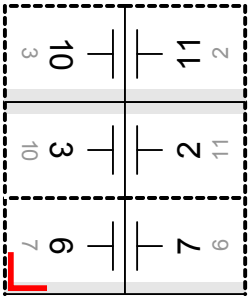
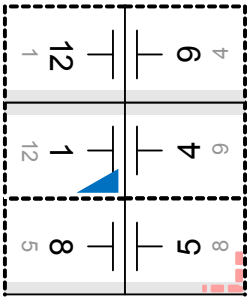
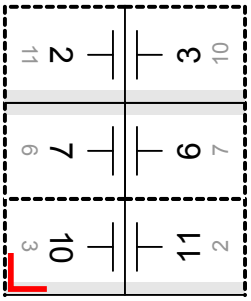
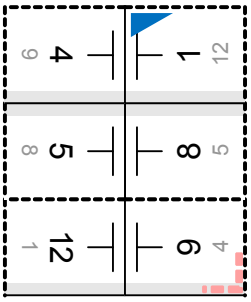
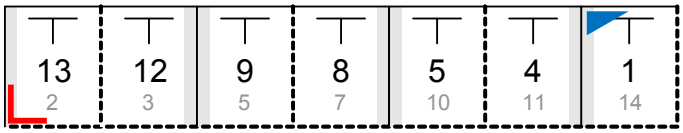
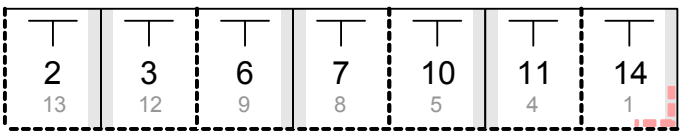
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F12-13	**2x3**	
$\uparrow^1/_2 + \uparrow^1/_3 \uparrow^1/_3$		
F12-14	**2x3**	
$\uparrow^1/_2 + \uparrow^1/_3 \downarrow^1/_3$		
F14-1	**7x1**	
$\uparrow^1/_7 \downarrow^1/_7 \uparrow^1/_7$ $\downarrow^1/_7 \uparrow^1/_7 \downarrow^1/_7$		

Table I.7: Pagination Diagrams (Sheet 10 of 32)

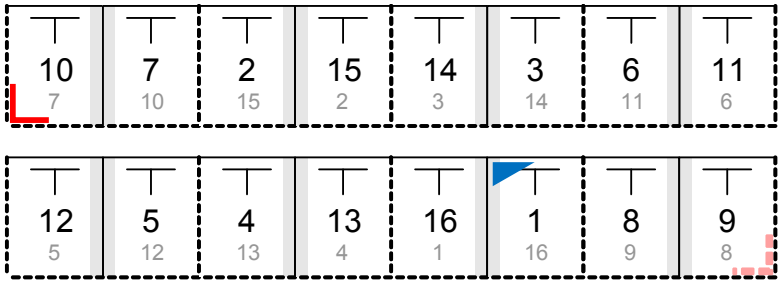
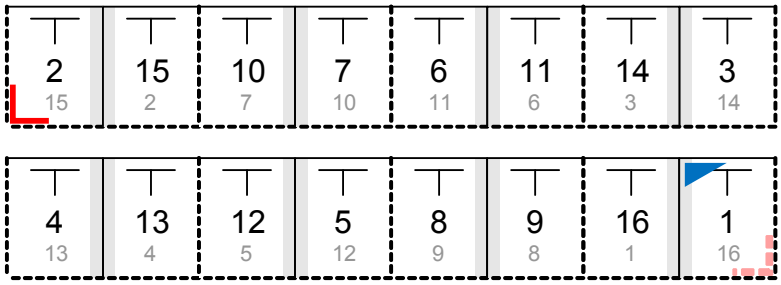
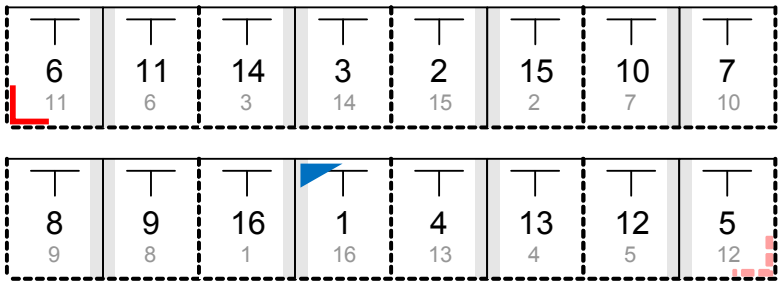
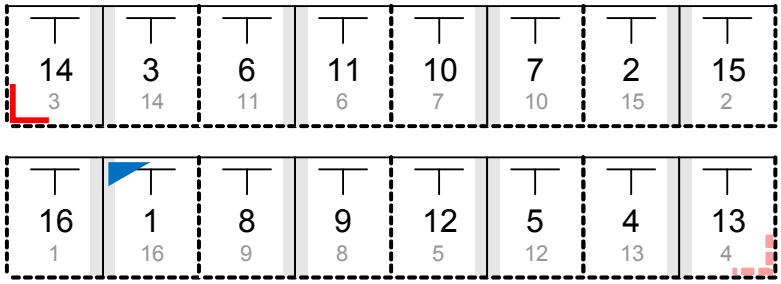
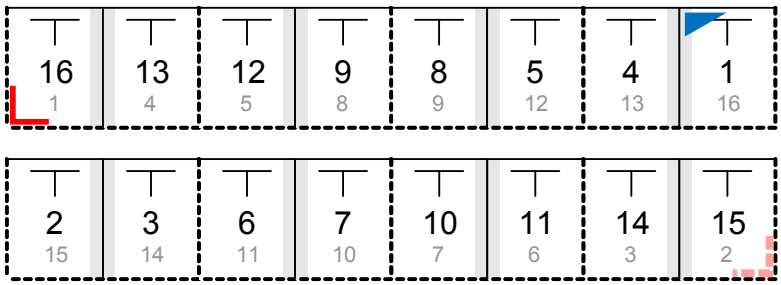
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F16-1	**8x1**	
$\uparrow^1/_2 \downarrow^1/_4 \uparrow^1/_8$		
F16-2	**8x1**	
$\uparrow^1/_2 \downarrow^1/_4 \downarrow^1/_8$		
F16-3	**8x1**	
$\uparrow^1/_2 \uparrow^1/_4 \downarrow^1/_8$		
F16-4	**8x1**	
$\uparrow^1/_2 \uparrow^1/_4 \uparrow^1/_8$		
F16-5	**8x1**	
$\downarrow^1/_8 \uparrow^1/_8 \downarrow^1/_8 \uparrow^1/_8$ $\downarrow^1/_8 \uparrow^1/_8 \downarrow^1/_8$		

Table I.7: Pagination Diagrams (Sheet 11 of 32)

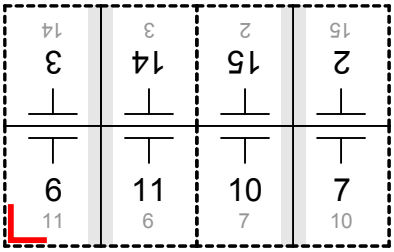
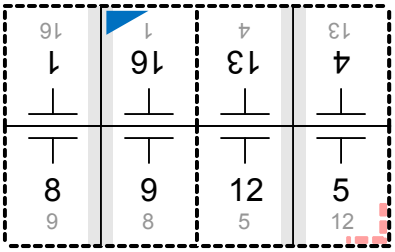
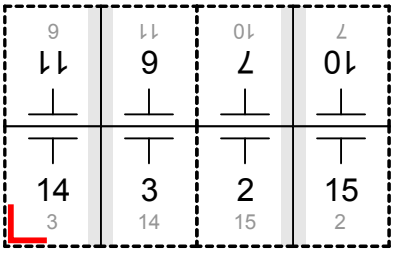
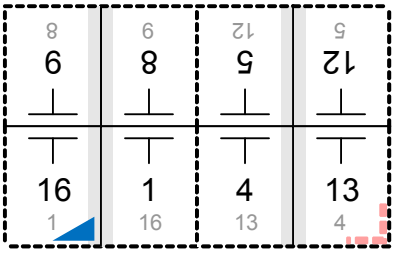
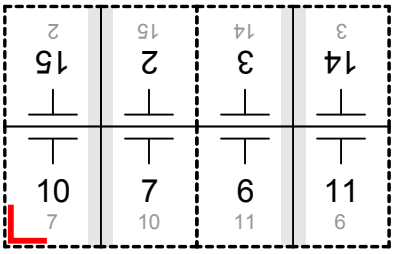
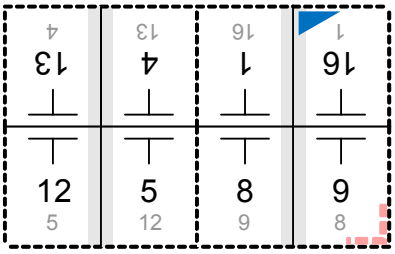
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F16-6	**4x2**	
$\uparrow 1/2 + \uparrow 1/2 + \uparrow 1/4$		
F16-7	**4x2**	
$\uparrow 1/2 + \uparrow 1/2 + \downarrow 1/4$		
F16-8	**4x2**	
$\uparrow 1/2 + \downarrow 1/2 + \downarrow 1/4$		

Table I.7: Pagination Diagrams (Sheet 12 of 32)

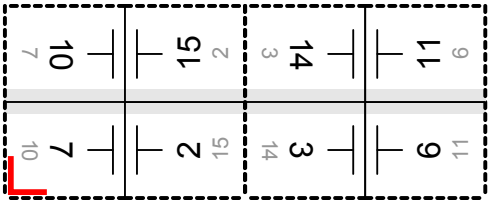
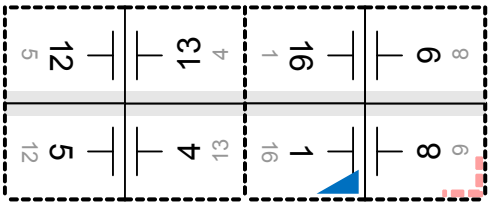
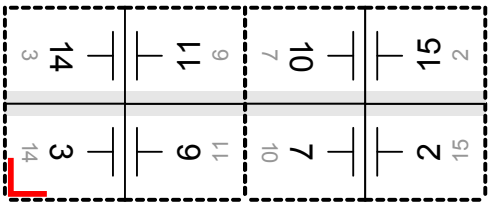
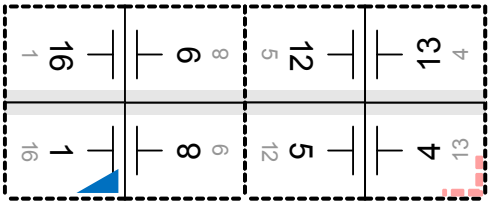
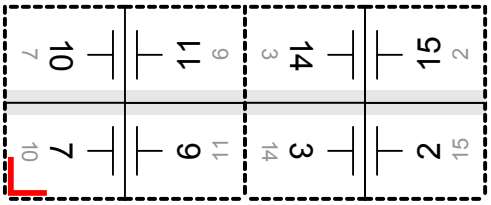
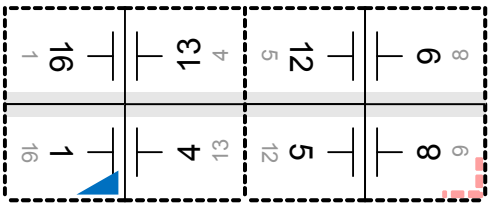
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F16-9	**4x2**	
$\uparrow^1/_2 \downarrow^1/_4 + \uparrow^1/_2$		
F16-10	**4x2**	
$\uparrow^1/_2 \uparrow^1/_4 + \uparrow^1/_2$		
F16-11	**4x2**	
$\uparrow^1/_4 \downarrow^1/_4 \uparrow^1/_4 + \uparrow^1/_2$		

Table I.7: Pagination Diagrams (Sheet 13 of 32)

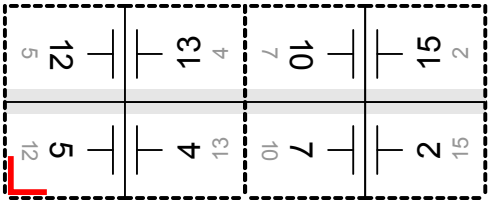
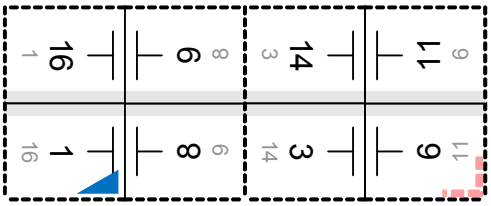
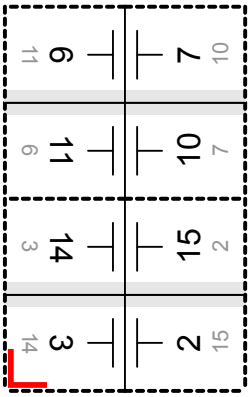
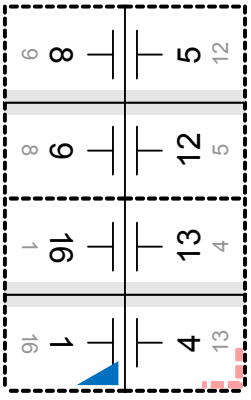
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F16-12	**4x2**	
$\uparrow 1/4\ \uparrow 1/4\ \uparrow 1/4 + \uparrow 1/2$		
F16-13	**2x4**	
$\uparrow 1/2 + \uparrow 1/2\ \downarrow 1/4$		

Table I.7: Pagination Diagrams (Sheet 14 of 32)

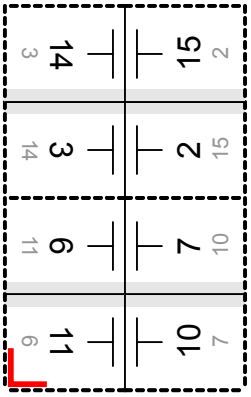
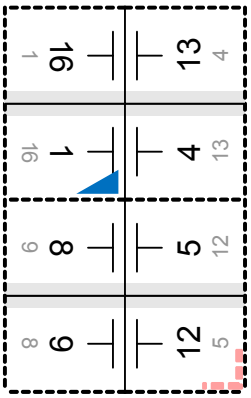
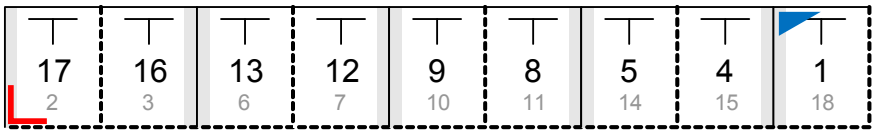
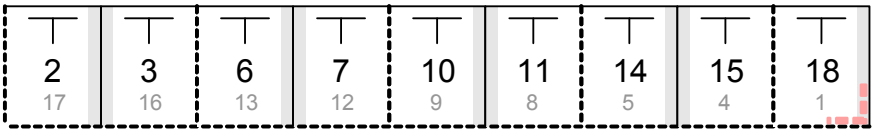
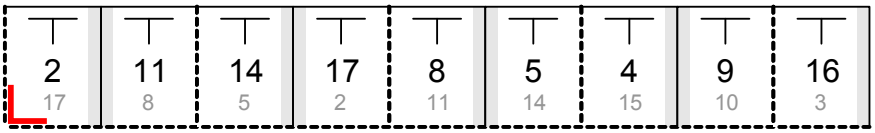
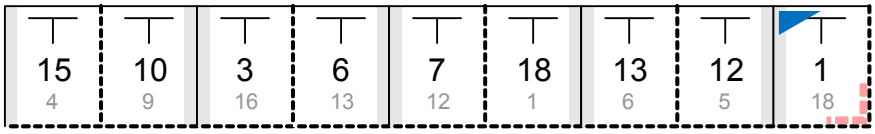
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F16-14	**2x4**	
$\uparrow^1/_2 + \uparrow^1/_2 \uparrow^1/_4$		
F18-1	**9x1**	
$\uparrow^1/_9 \downarrow^1/_9 \uparrow^1/_9 \downarrow^1/_9$ $\uparrow^1/_9 \downarrow^1/_9 \uparrow^1/_9 \downarrow^1/_9$		
F18-2	**9x1**	
$\uparrow^2/_3 \downarrow^1/_3 \uparrow^1/_9 \downarrow^1/_9$		

Table I.7: Pagination Diagrams (Sheet 15 of 32)

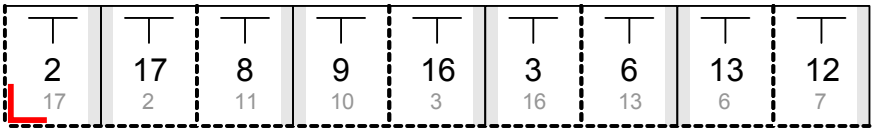
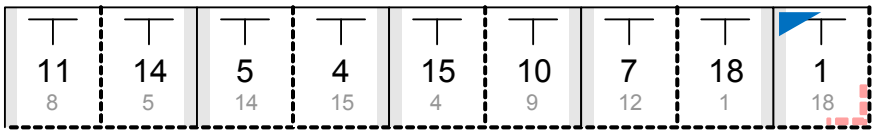
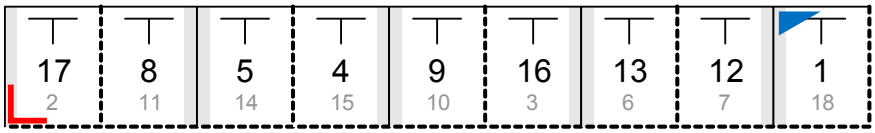
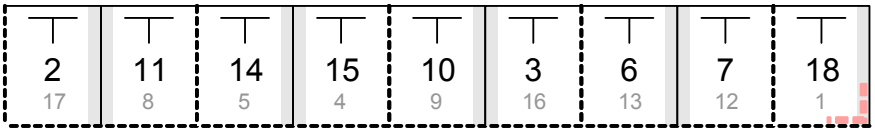
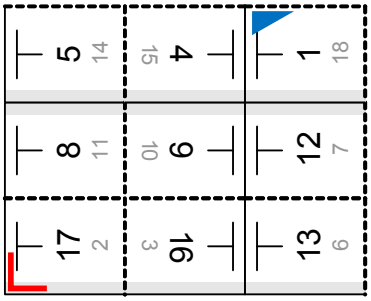
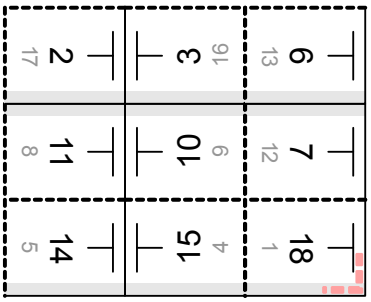
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F18-3	**9x1**	
$\uparrow^1/_3 \downarrow^1/_3 \uparrow^2/_9 \downarrow^1/_9$		
F18-4	**9x1**	
$\uparrow^1/_3 \downarrow^1/_3 \uparrow^1/_9 \downarrow^1/_9$		
F18-5	**3x3**	
↑1/3 ↓1/3 + ↑1/3 ↓1/3		

Table I.7: Pagination Diagrams (Sheet 16 of 32)

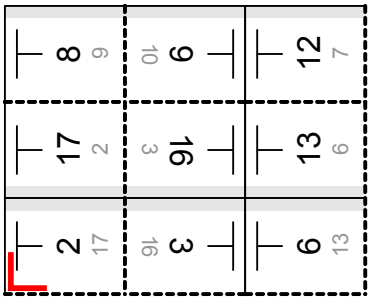
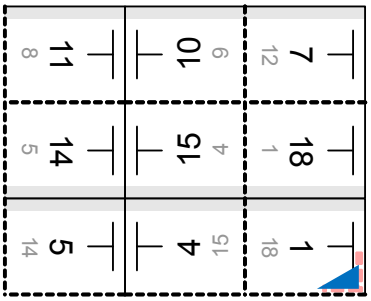
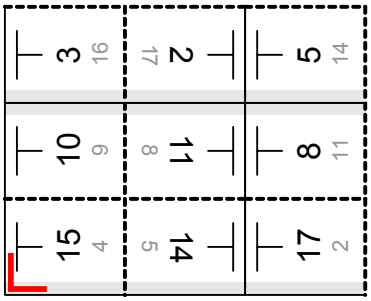
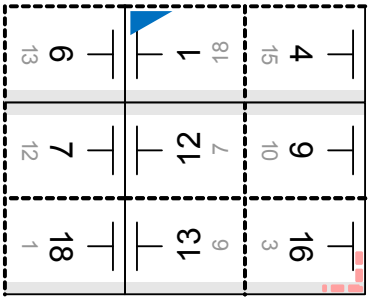
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F18-6	**3x3**	
$\uparrow^{1}/_{3}\downarrow^{1}/_{3} + \uparrow^{2}/_{3}\downarrow^{1}/_{3}$		
F18-7	**3x3**	
$\uparrow^{1}/_{3}\uparrow^{1}/_{3} + \uparrow^{1}/_{3}\downarrow^{1}/_{3}$		

Table I.7: Pagination Diagrams (Sheet 17 of 32)

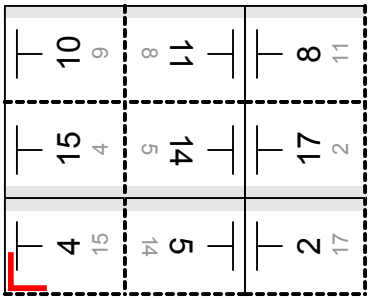
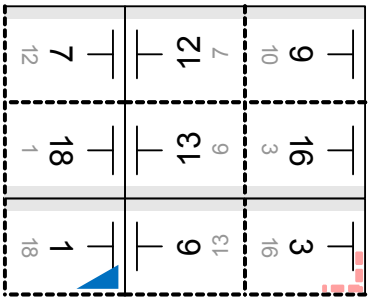
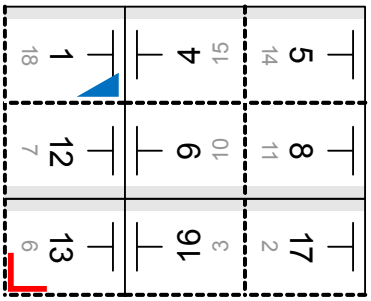
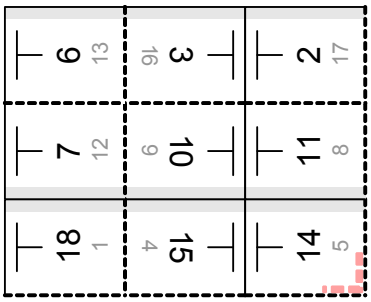
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F18-8	**3x3**	
↑1/3 ↑1/3 + ↑2/3 ↓1/3		
F18-9	**3x3**	
↑2/3 ↑1/3 + ↑2/3 ↑1/3		

Table I.7: Pagination Diagrams (Sheet 18 of 32)

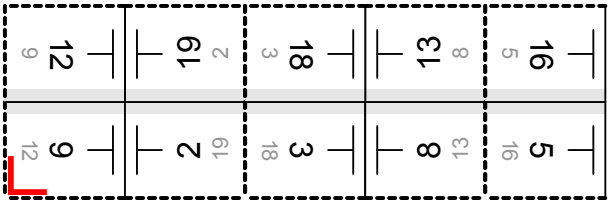
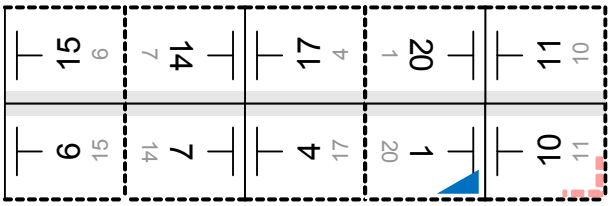
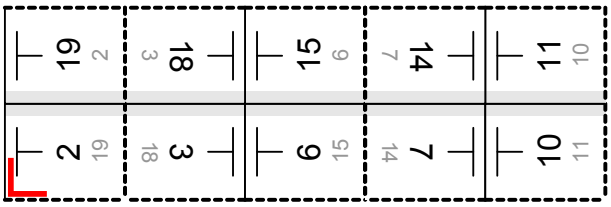
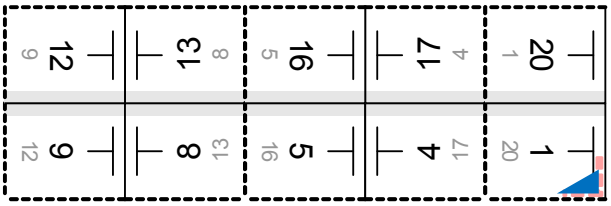
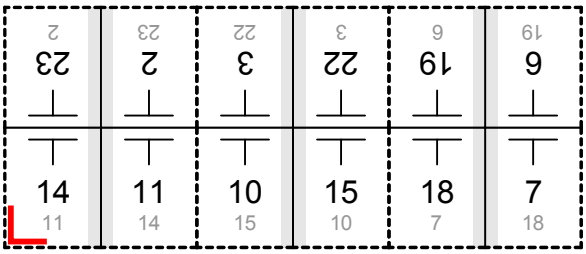
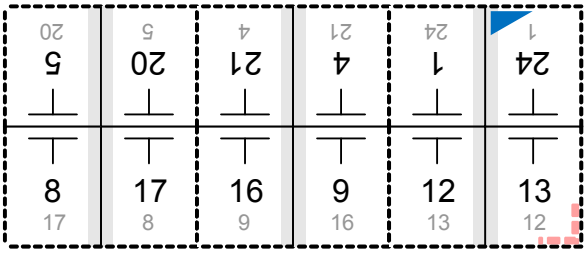
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F20-1	**5x2**	
$\uparrow^{2}/_{5}\downarrow^{2}/_{5}\uparrow^{1}/_{5}+\uparrow^{1}/_{2}$		
F20-2	**5x2**	
$\uparrow^{1}/_{5}\downarrow^{1}/_{5}\uparrow^{1}/_{5}\downarrow^{1}/_{5}+\uparrow^{1}/_{2}$		
F24-1	**6x2**	
$\uparrow^{1}/_{3}\downarrow^{1}/_{3}+$ $\uparrow^{1}/_{2}+\uparrow^{1}/_{6}$		

Table I.7: Pagination Diagrams (Sheet 19 of 32)

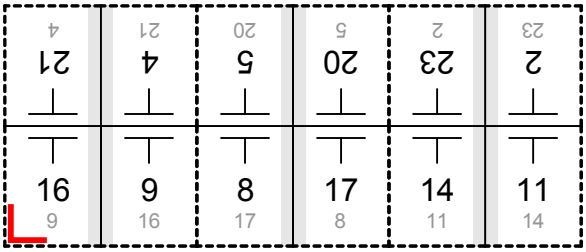
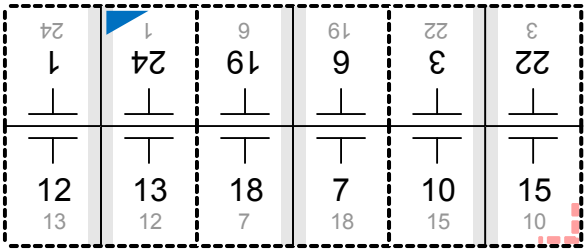
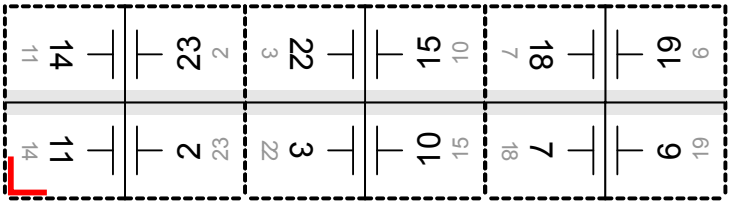
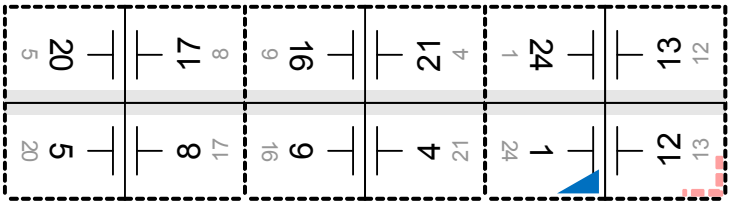
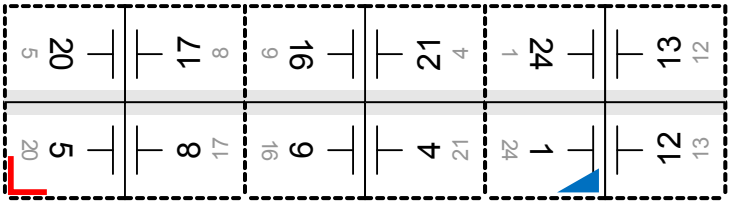
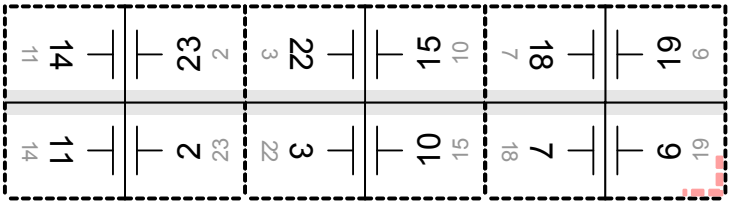
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F24-2	**6x2**	
$\uparrow 1/3\ \uparrow 1/3 + \uparrow 1/2 + \uparrow 1/6$		
F24-3	**6x2**	
$\uparrow 1/3\ \downarrow 1/3\ \uparrow 1/6 + \uparrow 1/2$		
F24-4	**6x2**	
$\uparrow 1/3\ \downarrow 1/3\ \downarrow 1/6 + \uparrow 1/2$		

Table I.7: Pagination Diagrams (Sheet 20 of 32)

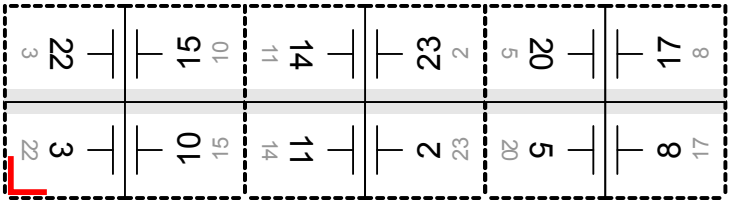
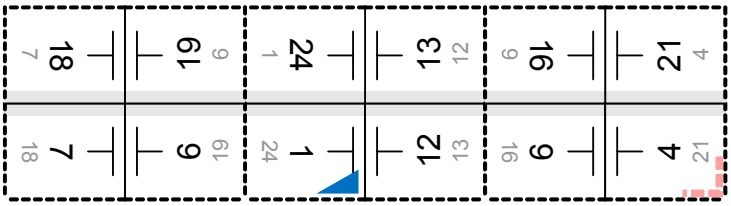
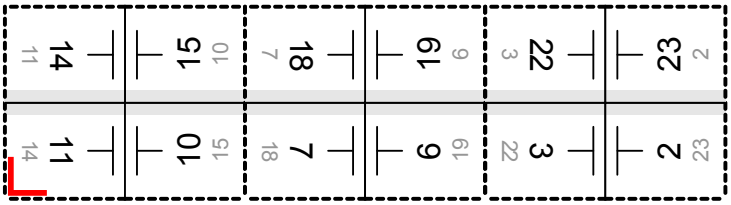
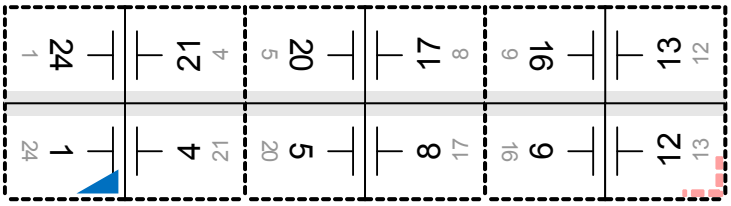
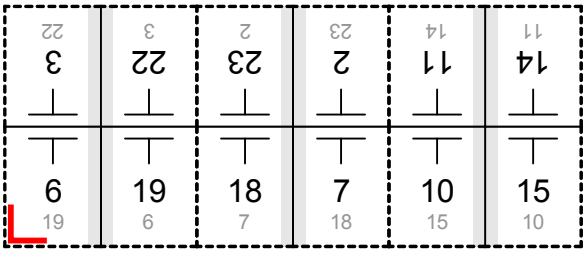
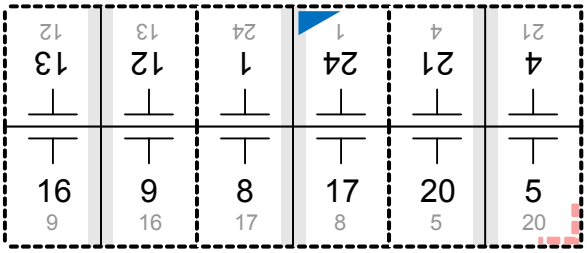
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F24-5	**6x2**	
$\uparrow^1/_3 \uparrow^1/_3 \downarrow^1/_6 + \uparrow^1/_2$		
F24-6	**6x2**	
$\uparrow^1/_6 \downarrow^1/_6 \uparrow^1/_6$ $\downarrow^1/_6 \uparrow^1/_6 + \uparrow^1/_2$		
F24-7	**6x2**	
$\uparrow^1/_3 + \uparrow^1/_2 + \uparrow^1/_3 \downarrow^1/_6$		

Table I.7: Pagination Diagrams (Sheet 21 of 32)

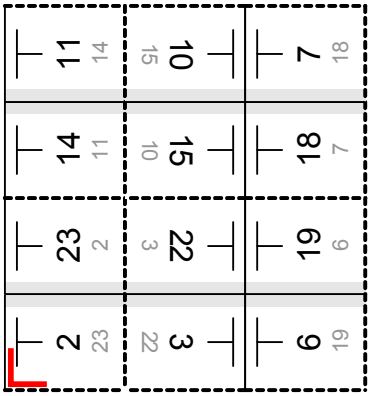
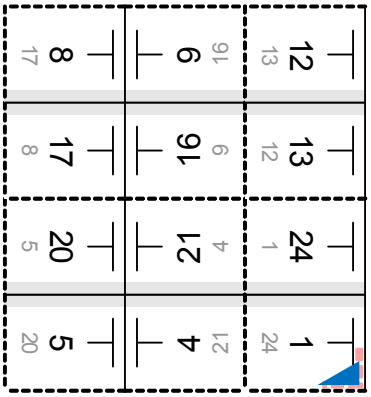
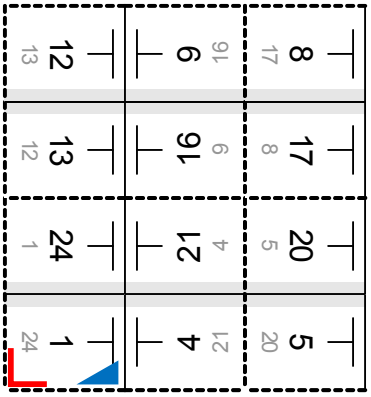
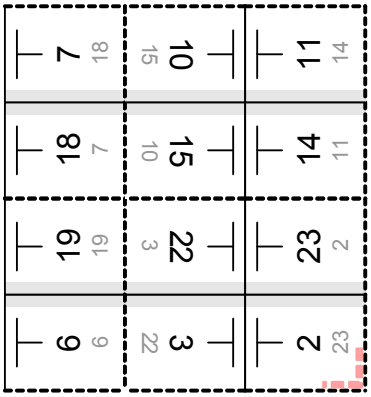
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F24-8	**3x4**	
$\uparrow^{1}/_{3}\downarrow^{1}/_{3}+\uparrow^{1}/_{2}\downarrow^{1}/_{4}$		
F24-9	**3x4**	
$\uparrow^{2}/_{3}\uparrow^{1}/_{3}+\uparrow^{1}/_{2}\downarrow^{1}/_{4}$		

Table I.7: Pagination Diagrams (Sheet 22 of 32)

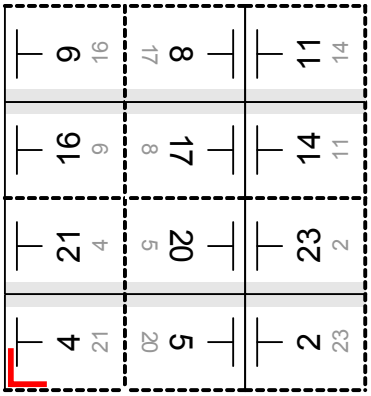
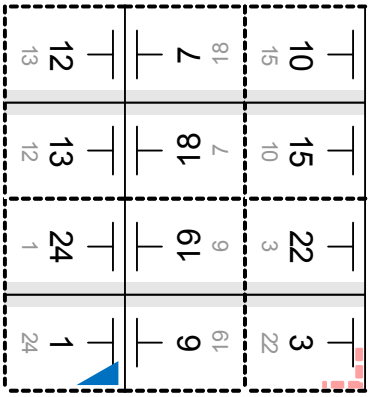
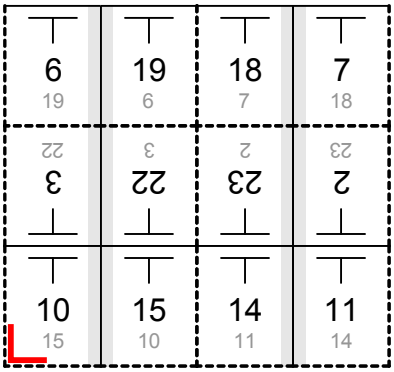
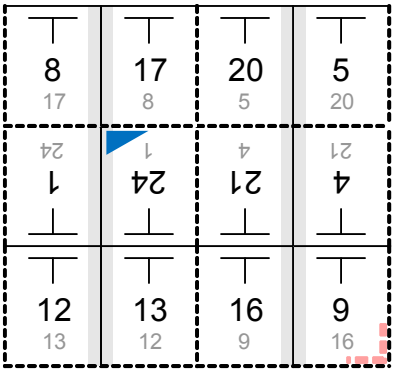
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F24-10	**3x4**	
$\nearrow^{1}/_{3}\ \nearrow^{1}/_{3} + \nearrow^{1}/_{2}\ \searrow^{1}/_{4}$		
F24-11	**4x3**	
$\nearrow^{1}/_{2} + \nearrow^{2}/_{3}\ \searrow^{1}/_{3} + \nearrow^{1}/_{4}$		

Table I.7: Pagination Diagrams (Sheet 23 of 32)

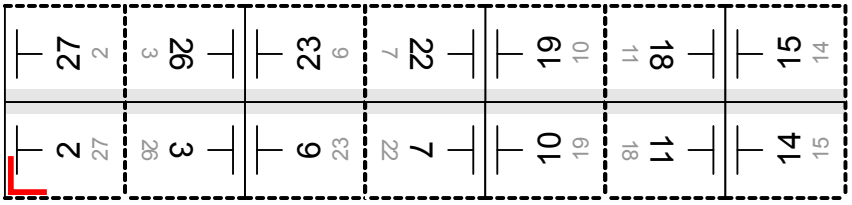
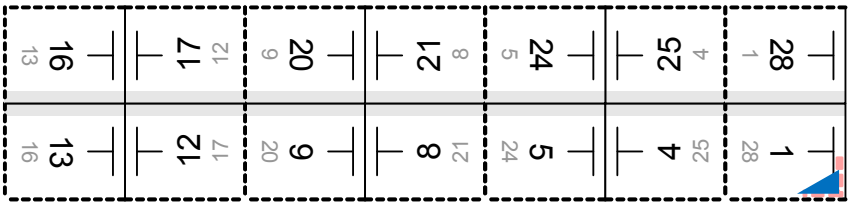
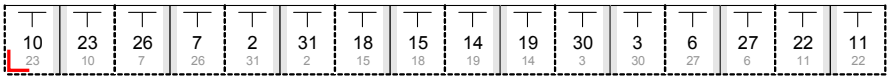
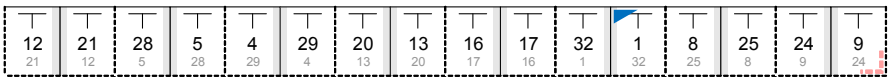
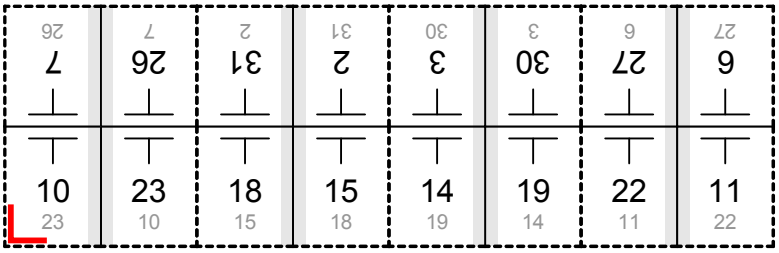
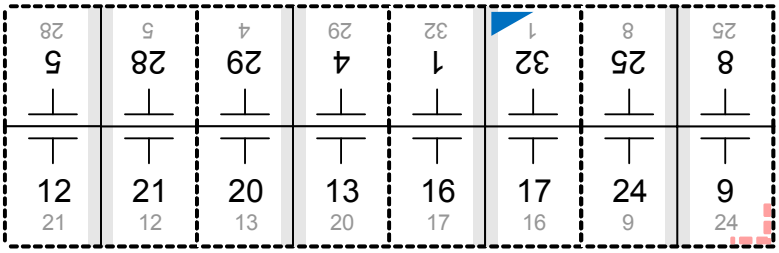
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F28-1	**7x2**	
$\uparrow^1/_7 \downarrow^1/_7 \uparrow^1/_7 \downarrow^1/_7$ $\uparrow^1/_7 \downarrow^1/_7 + \uparrow^1/_2$		
F32-1	**16x1**	
$\uparrow^1/_2 \downarrow^1/_4 \uparrow^1/_8 \downarrow^1/_{16}$		
F32-2	**8x2**	
$\uparrow^1/_2 \downarrow^1/_4 + \uparrow^1/_2 + \uparrow^1/_8$		

Table I.7: Pagination Diagrams (Sheet 24 of 32)

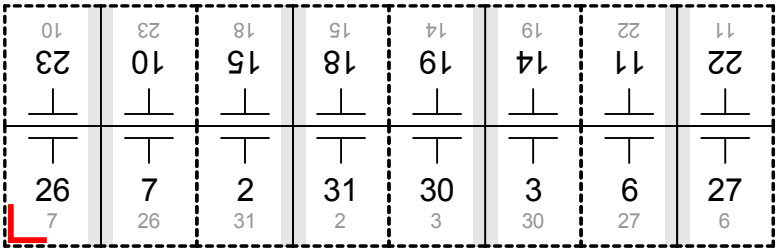
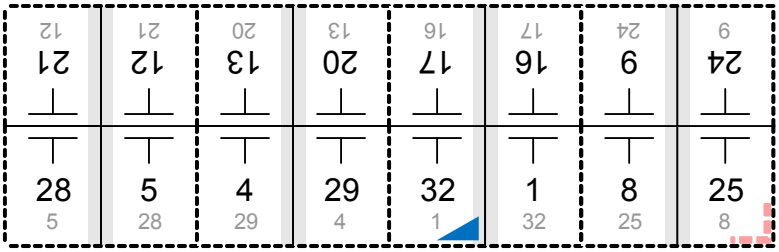
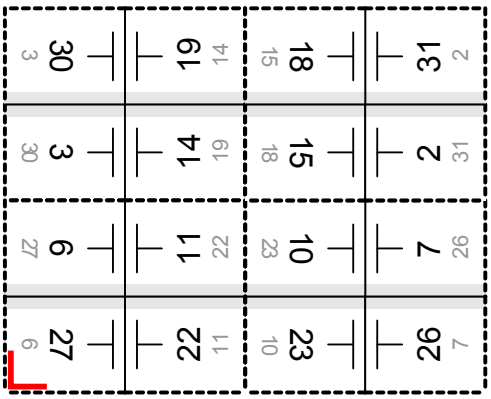
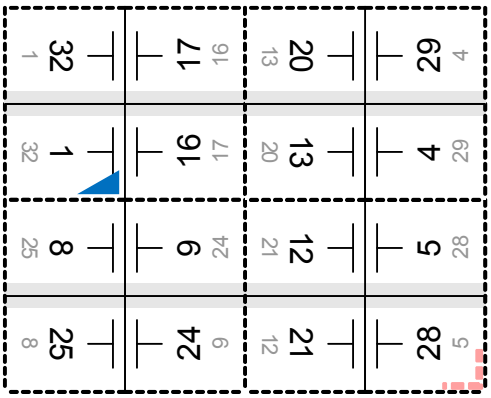
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F32-3	**8x2**	
$\uparrow^1/_2 \downarrow^1/_4 + \uparrow^1/_2 + \downarrow^1/_8$		
F32-4	**4x4**	
$\uparrow^1/_2 + \uparrow^1/_2 + \uparrow^1/_4 + \uparrow^1/_4$		

Table I.7: Pagination Diagrams (Sheet 25 of 32)

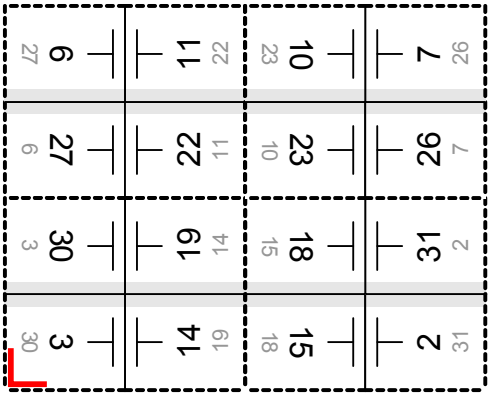
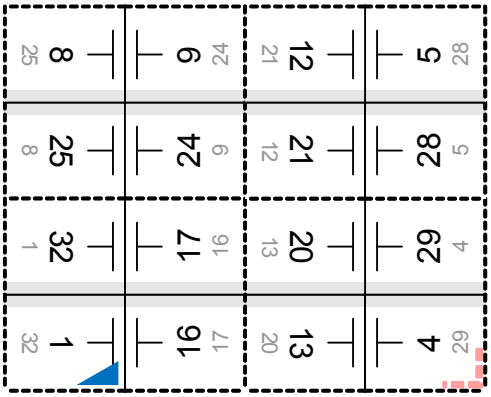
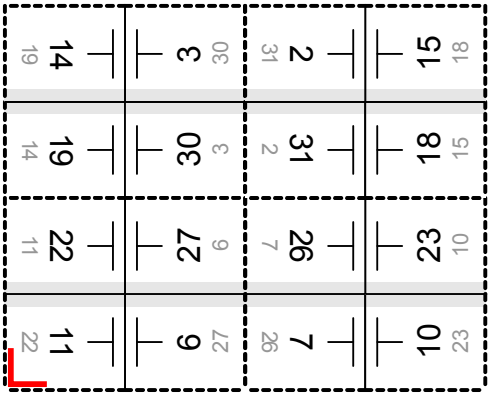
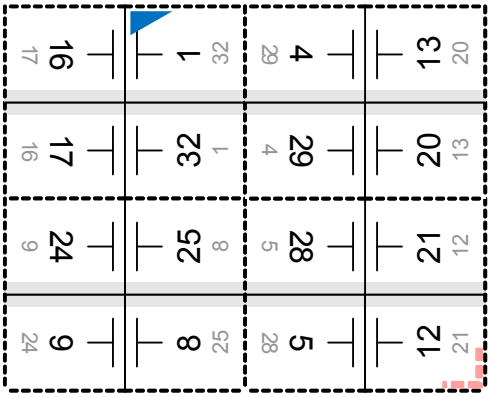
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F32-5	**4x4**	
$\uparrow 1/2 + \uparrow 1/2 + \downarrow 1/4 + \downarrow 1/4$		
F32-6	**4x4**	
$\uparrow 1/2 + \uparrow 1/2 + \uparrow 1/4 + \downarrow 1/4$		

Table I.7: Pagination Diagrams (Sheet 26 of 32)

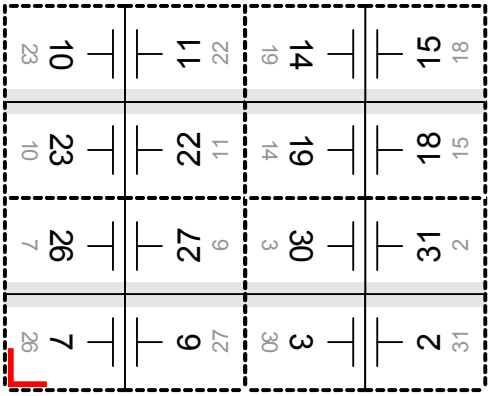
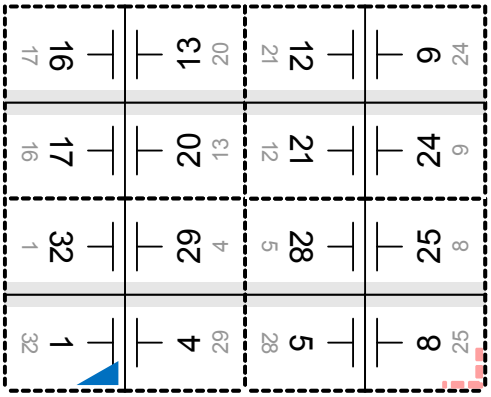
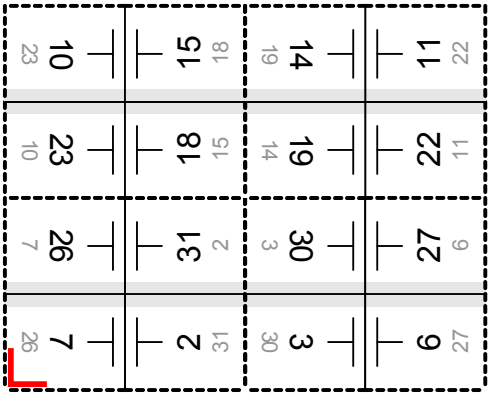
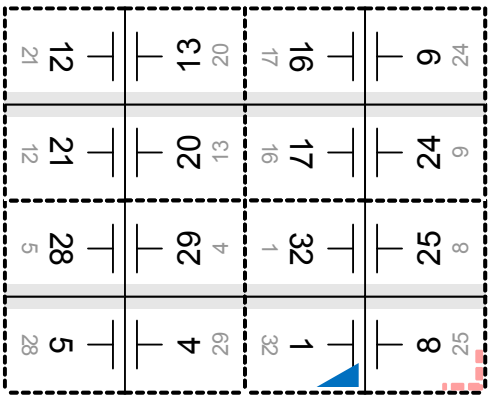
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F32-7	**4x4**	
↑1/4 ↓1/4 ↑1/4 + ↑1/2 ↓1/4		
F32-8	**4x4**	
↑1/2 ↓1/4 + ↑1/2 ↓1/4		

Table I.7: Pagination Diagrams (Sheet 27 of 32)

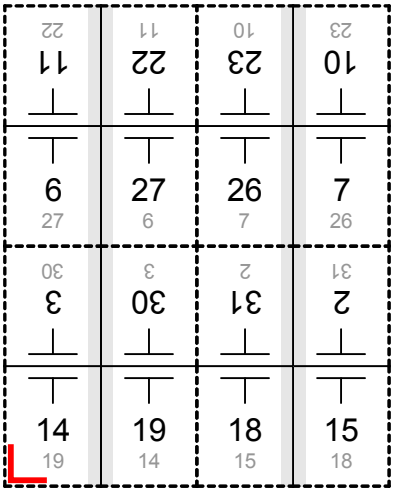
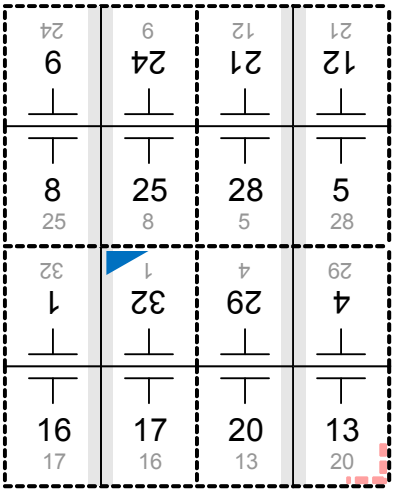
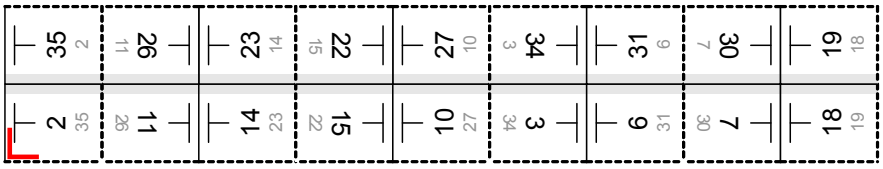
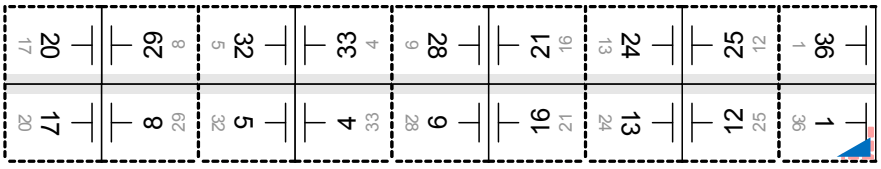
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F32-9	**4x4**	
$\uparrow^1/_2 + \uparrow^1/_2 \downarrow^1/_4 + \uparrow^1/_4$		
F36-1	**9x2**	
$\uparrow^1/_3 \downarrow^1/_3 \uparrow^1/_9$ $\downarrow^1/_9 + \uparrow^1/_2$		

Table I.7: Pagination Diagrams (Sheet 28 of 32)

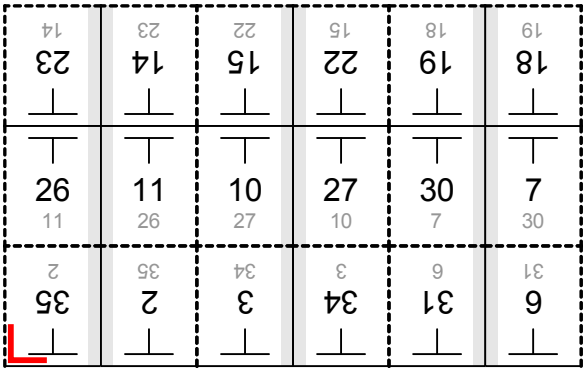
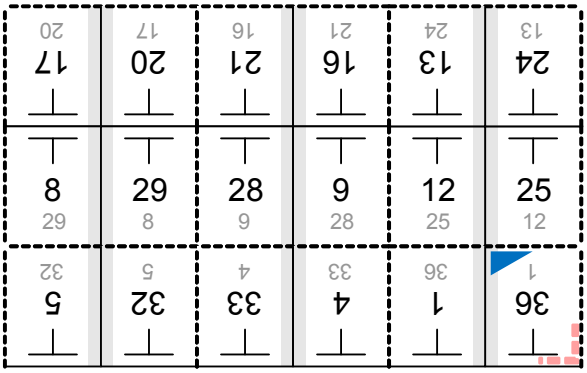
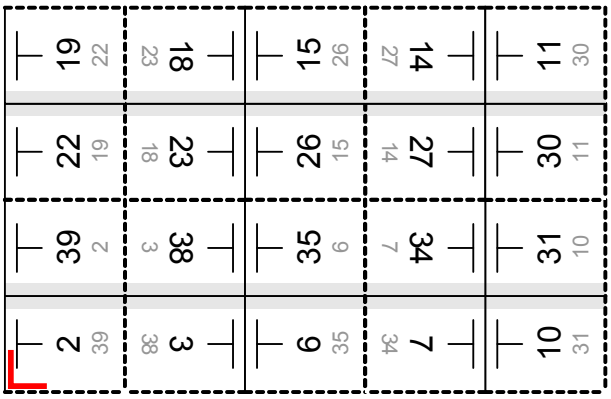
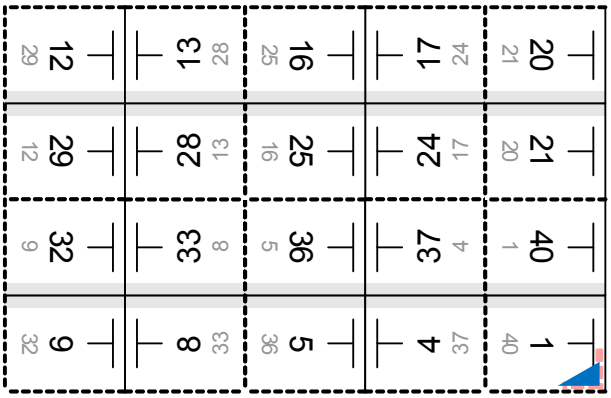
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F36-2	**6x3**	
$\uparrow^1/_3 \downarrow^1/_3 + \uparrow^1/_3$ $\downarrow^1/_3 + \uparrow^1/_6$		
F40-1	**5x4**	
$\uparrow^1/_5 \downarrow^1/_5 \uparrow^1/_5$ $\downarrow^1/_5 + \uparrow^1/_2 \downarrow^1/_4$		

Table I.7: Pagination Diagrams (Sheet 29 of 32)

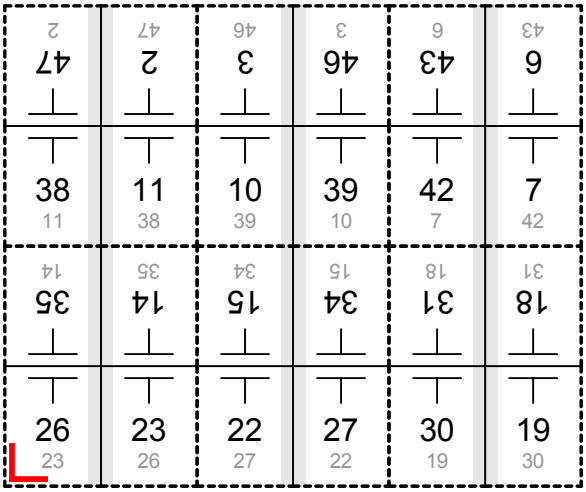
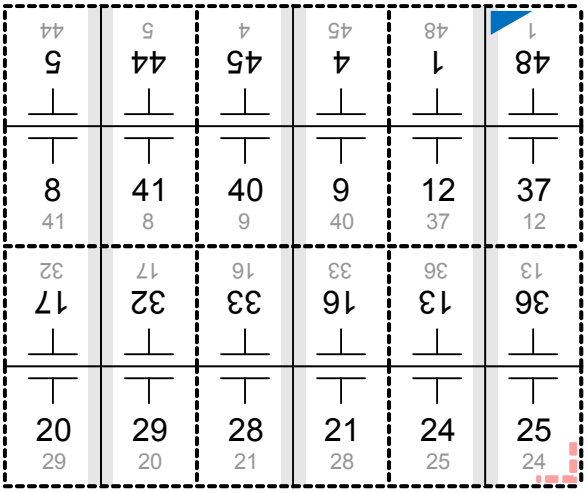
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F48-1	**6x4**	
$\uparrow^1/_3 \downarrow^1/_3 + \uparrow^1/_4$ $\downarrow^1/_4 \uparrow^1/_4 + \uparrow^1/_6$		

Table I.7: Pagination Diagrams (Sheet 30 of 32)

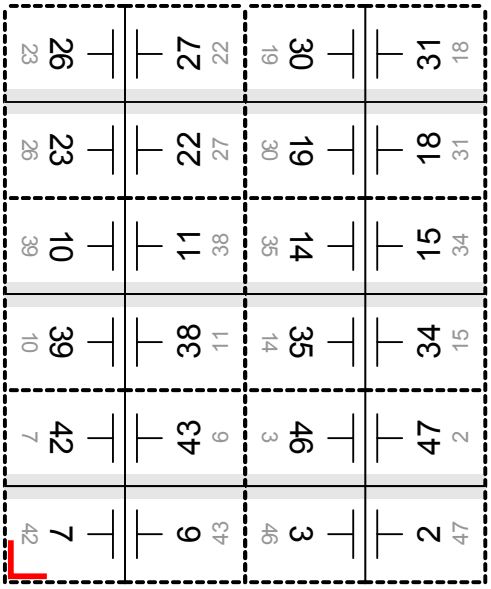
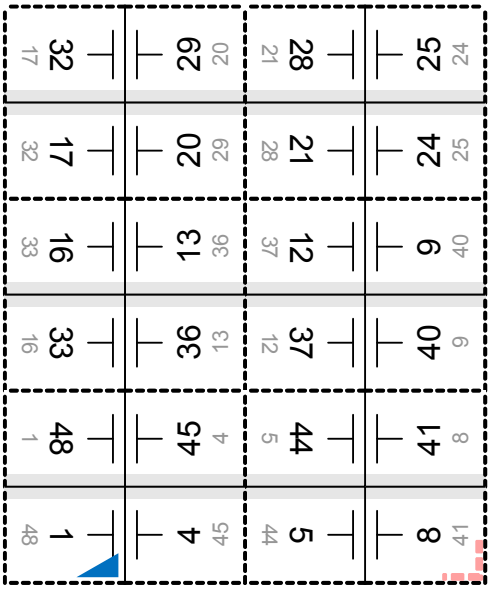
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F48-2	**4x6**	
$\uparrow 1/4 \downarrow 1/4 \uparrow 1/4$ $+ \uparrow 1/3 \downarrow 1/3 \uparrow 1/6$		

Table I.7: Pagination Diagrams (Sheet 31 of 32)

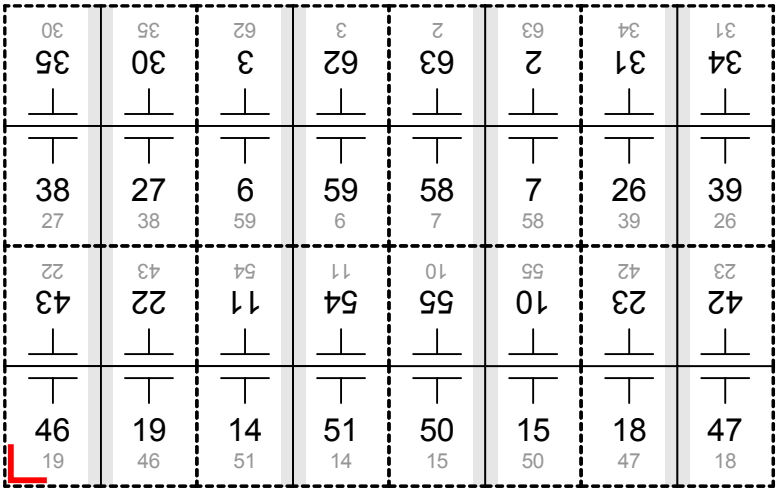
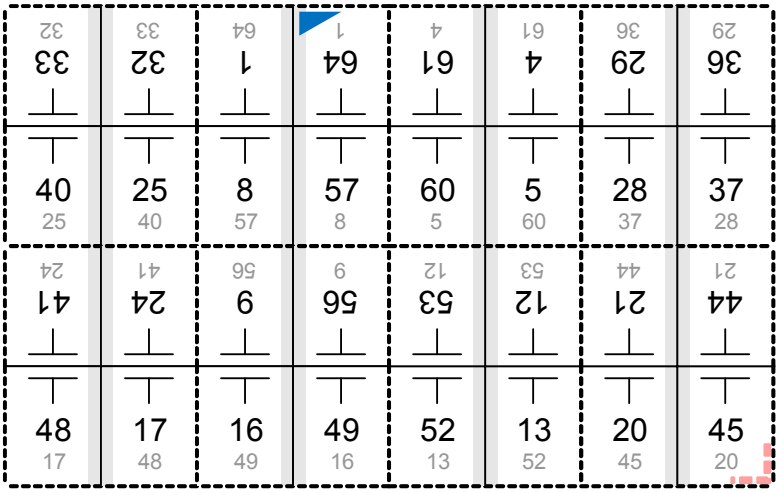
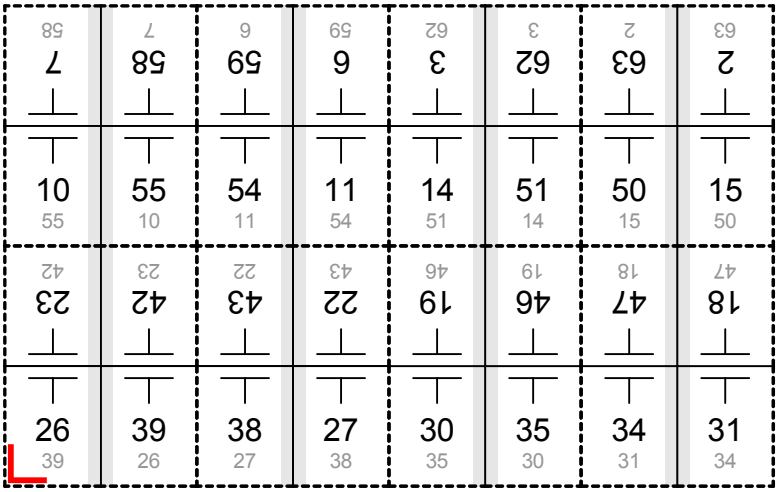
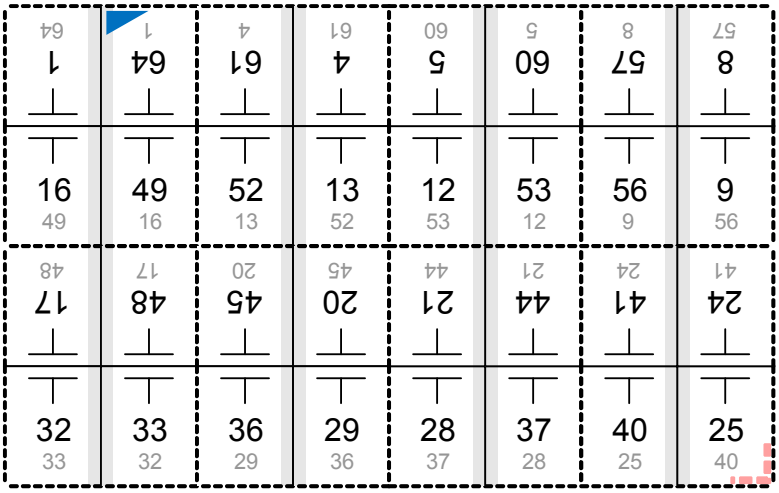
JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F64-1	**8x4**	
$\uparrow^1/_2 + \uparrow^1/_4 \downarrow^1/_4$ $\uparrow^1/_4 + \uparrow^1/_4 \downarrow^1/_8$		

Table I.7: Pagination Diagrams (Sheet 32 of 32)

JDF FOLD CATALOG	GRID SIZE	FRONT SIDE (LAY SIDE)
FOLDING SEQUENCE		BACK SIDE (NON-LAY SIDE)
F64-2	**8x4**	
$\uparrow^1/_4\downarrow^1/_4\uparrow^1/_4$ $+\uparrow^1/_4\downarrow^1/_4\uparrow^1/_4+\uparrow^1/_8$		

Appendix J

Resolving Directory URL References

New in JDF 1.2

This appendix describes the detailed semantics of resolving ***RunList***/@*Directory* and any associated ***FileSpec***/@*URL* URI references in any of the ***RunList*** refelements.

J.1 Semantics of the RunList/@Directory Attribute

The @*Directory* Attribute defines a directory where the files that are associated with this ***RunList*** SHOULD be copied to or from. If @*Directory* is specified, it SHALL be an Absolute URI ▸ [RFC3986] that implicitly also specifies a Base URI that is used to resolve any relative URI Attribute in the ***RunList*** structure. As such, @*Directory* SHALL start with a URL scheme, such as "file" or "ftp", MAY contain an authority, such as "//any.com" and SHOULD contain an absolute path that ends with a "/" to indicate a directory.[1] For example: "file://any.com/pub/doc-archives/" or "file:///pub/doc-archives/".

If @*Directory* is not specified, the Absolute URI that specifies the directory in which the **JDF** file resides is used as the Base URI to resolve each relative @*URI* Attribute in the ***RunList***.

If the ***FileSpec***/***Container***/***FileSpec*** Resource is supplied indicating that the ***FileSpec*** is contained in another file, the Base URI is the Absolute URI of where the **JDF** Consumer extracted the container file (whether or not @*Directory* is specified). See ▸ Section 8.58 FileSpec.

After determining the Base URI depending on the presence or absence of ***RunList***/@*Directory* and ***FileSpec***/***Container***/***FileSpec*** Resource as described above, each @*URL* Attribute in a ***RunList*** refelement (e.g., ***LayoutElement***/***FileSpec***/@*URL* or ***InsertSheet***/***Layout***/***Media***/***QualityControlResult***/***FileSpec***/@*URL*) is used in combination with the Base URI to form the Resolved URI as follows according to one of the following mutually exclusive patterns.[2]

1. ***RunList*** @*URL* starts with a scheme (token ending with ":" (e.g., "file:" or "cid:")):[3]
   - Resolved URI = the entire ***RunList*** URL (and the Base URI is ignored).
2. ***RunList*** @*URL* starts with an authority/host (starts with "//" (e.g., "//www.cip4.org")):
   - Resolved URI = the Base URI scheme, followed by the ***RunList*** URL authority/host followed by its absolute path (which MAY be empty).
3. ***RunList*** @*URL* starts with an absolute path (starts with "/" (e.g., "/pub/document-archives")):
   - Resolved URI = Base URI scheme and its authority (if any) followed by the ***RunList*** URL absolute path.
4. ***RunList*** @*URL* starts with a relative path (starts with something other than "/" (e.g., "foo.pdf", "./folder/foo.pdf", "../foo.pdf", etc.):
   - Resolved URI = Base URI scheme, its authority (if any), and its absolute path (if any) up to and including the right-most "/", followed by the ***RunList*** @*URL* relative path with ".", ".." and "./" segments removed.

The above algorithm is only a summary. See ▸ [RFC3986] for the detailed algorithm. See ▸ [FileURL] for examples.

---

1. According to ▸ [RFC3986] section 5.2 "Resolve Relative References to Absolute Form", the characters following the right-most "/" if any, are removed from the Base URI, in order to resolve a Relative URI with the Base URI. So be sure to end the @*Directory* value with a "/" to make it cleat that @*Directory* is a reference to a directory and not a file, and to ensure that the last path segment won't get removed in resolving the URI reference.
2. The Resolved URI is formed assuming that URI query and fragments are *not* used in JDF.
3. In order to improve interoperability and to simplify implementation, JDF follows the strict-parsing rules of ▸ [RFC3986] so that even if the ***FileSpec***/@*URL* Attribute starts with the same scheme as the Base URI, the entire URL values is always interpreted as an Absolute URI and always replaces the Base URI to form the Resolved URI. This strict rule is especially important for interoperability. Consider the case where the JDF Producer drops the JDF into a hot folder but does NOT specify ***RunList***/@*Directory* so that the JDF Consumer has to generate the Base URI for the hot folder in order to resolve the ***FileSpec***/@*RunList*, but the Producer is supplying a ***FileSpec***/@*URL* that is relative to the hot folder. If the JDF Producer supplies the scheme in the ***FileSpec***/@*URL*, then the JDF Producer would have to supply the same scheme as the JDF Consumer generates for the Base URI for hot folder, in order for the Relative URI semantics to apply. However, under non-strict parsing, if the JDF Producer guesses wrong (say one is "file:" and the other is "ftp:"), the JDF Consumer would interpret ***FileSpec***/@*URI* as an Absolute URI.

Appendix K

References

Throughout this specification references to other documents are indicated by short symbolic names inside square brackets, (e.g., ▸ [ICC.1]). Implementers need to read and conform to such referenced documents when implementing a part of this specification with such a reference. The reader is directed to this Document References section to find the full title, date, source and availability of all such references.

Table K.1: References (Sheet 1 of 9)

TERM	DEFINITION
[Adb-TN5044]	*Adobe Technical note 5044* Date: 24 May 1996 Produced by: Adobe Systems Inc. Available at: http://partners.adobe.com/public/developer/en/ps/sdk/5044.ColorSep_Conv.pdf
[AdsML]	*AdsML 1.0 Specification & Schema* Date: 17 May 2004 Produced by: AdsML Technical Working Group Available at: http://www.adsml.org
[Braille]	*Six dot Braille representation of the ASCII character set* Date: - Available at: https://en.wikipedia.org/wiki/Braille_ASCII
[Unicode Braille Patterns]	*Unicode Braille patterns* Date: - Produced by: The Unicode Consortium Available at: http://www.unicode.org/charts/PDF/U2800.pdf#search=%22braille%20unicode%22
[CCIR601-2]	*CCIR Recommendation 601-2* *Encoding Parameters of Digital Television for Studios, 1990, Volume XI — Part 1, Broadcasting Service (Television), pp. 95-104.* Date: 1990 Produced by: International Telecommunication Union Available at: International Telecommunication Union, General Secretariat — Sales Section, Place des Nations, CH-1211 Geneva 20 (Switzerland)
[CIE 15:2004]	*CIE 15:2004* *Colorimetry, 3rd Edition.* Date: 2004 Produced by: Commission Internationale de l'Eclairage International (CIE) Available at: http://www.cie.co.at
[CIP3 - PPF]	*CIP3 Print Production Format (PPF)* Date: 1 June 1998 Version: CIP3 PPF Specification 3.0 Produced by: CIP4 Organization Available at: https://confluence.cip4.org/display/PUB/PPF
[Characterization Data]	*CMYK Characterization Data* Date: - Version: - Produced by: International Color Consortium Available at: http://www.color.org/chardata/drsection1.xalter
[Color Names]	*Recognized color keyword names* Date: 16 August 2011 Version: SVG (1.1) Second Editon Produced by: World Wide Web Consortium (W3C) Available at: https://www.w3.org/TR/SVG/types.html#ColorKeywords
[ColorPS]	*Color Separation Conventions for PostScript Language Programs* *Technical Note #5044* Date: 24 May 1996 Produced by: Adobe Systems Inc. Available at: http://partners.adobe.com/asn/developer/pdfs/tn/5044.ColorSep_Conv.pdf

Table K.1: References (Sheet 2 of 9)

TERM	DEFINITION
[Corrugated Packaging]	*Corrugatged packaging* Date: - Produced by: Corrugated Packaging Alliance Available at: http://cpc.corrugated.org/
[DCS2.0]	*Document Color Separation (DCS), version 2.0* Date: Revised May 1995 Produced by: Adobe Software Inc. Available at: http://www.npes.org/standards/Tools/DCS20Spec.pdf
[ECMA]	*ECMA Code of Folding Carton Styles* Date: - Produced by: European Carton Makers Association Available at: http://www.ecma.org/download/orderformpublications.pdf
[FEFCO]	*FEFCO European Federation of Corrugated Board Manufacturers* Date: - Produced by: European Federation of Corrugated Board Manufacturers Available at: http://www.fefco.org
[FileURL]	*CIP4 Application Note - Use of FileURL in JDFManufacturers* Date: August 2003 Produced by: CIP4 Organization Available at: http://www.cip4.org
[FINAT]	*Trade Association for the Self-Adhesive and Labeling Industry* Web site: http://www.finat.com
[FIRST]	*Flexographic Image Reproduction Specifications & Tolerances* (FIRST) *Second Edition* Date: November 1999 Produced by: Flexography Technical Association Available at: http://www.fta-ffta.org
[Grammage Conversion]	*Basis Weight and Grammage Conversion Tables.* Date: - Produced by: EDS Inc. Editorial & Design Services. Available at: http://www.edsebooks.com/paper/grammage.html
[iana-character sets]	*IANA Registry of Character Set names* Available at: https://www.iana.org/assignments/character-sets/character-sets.xhtml
[iana-mt]	*IANA Registry of MIME Media Types* Available at: http://www.iana.org/assignments/media-types
[iana-os]	*IANA Registry of Operating System Names* Available at: http://www.iana.org/assignments/operating-system-names
[ICC.1]	*Specification ICC.1:2004-10* *File Format for Color Profiles, Version 4.2.0.0* Date: 2004 Produced by: International Color Consortium (ICC) Available at: http://www.color.org/icc_specs2.xalter
[IEEE754]	*IEEE 754-1985* *Standard for Binary Floating-Point Arithmetic* Date: 1985 Produced by: IEEE Available at: http://grouper.ieee.org/groups/754/
[IEEE1284]	*IEEE 1284-2000* *IEEE Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers* Date: 2000 Produced by: IEEE Available at: http://standards.ieee.org/catalog/olis/busarch.html
[ifra]	*IfraTrack Specification* *Ifra Special Report 6.21.2, Version 2.0* Date: June 1998 Produced by: Ifra Available at: http://www.ifra.com/

Table K.1: References (Sheet 3 of 9)

TERM	DEFINITION
[ISO5-3:1995]	*ISO 5-3:1995* *Photography -- Density measurements -- Part 3: Spectral conditions.* Date: 1995 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO5-4:1995]	*ISO 5-4:1995* *Photography -- Density measurements -- Part 4: Geometric conditions for reflection density.* Date: 1995 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO216:1975]	*ISO 216:1975* *Writing paper and certain classes of printed matter -- Trimmed sizes -- A and B series.* Date: 1975 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO269:1985]	*ISO 269:1985* *Correspondence envelopes -- Designation and sizes* Date: 1985 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO2108:2005]	*ISO 2108:2005* *Information and documentation -- International standard book number (ISBN).* Date: 2005 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO2470:1999]	*ISO 2470:1999* *Paper, board and pulps -- Measurement of diffuse blue reflectance factor (ISO brightness.* Date: 1999 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO2471:1998]	*ISO 2471:1998* *Paper and board—Determination of opacity (paper backing)—Diffuse reflectance method.* Date: 1998 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO3166-1:1997]	*ISO 3166-1:1997* *Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes.* Date: 1997 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO4217:2001]	*ISO 4217:2001* *Codes for the representation of currencies and funds* Date: 2001 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO8254-1:1999]	*ISO 8254-1:1999* *Paper and board -- Measurement of specular gloss -- Part 1: 75 degree gloss with a converging beam, TAPPI method* Date: 1999 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO8601:2004]	*ISO 8601:2004* *Data elements and interchange formats - Information interchange - Representation of dates and times.* Date: 2004 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html

Table K.1: References (Sheet 4 of 9)

TERM	DEFINITION
[ISO12639:2004]	*ISO 12639:2004* *Graphic technology - Prepress digital data exchange — Tag image file format for image technology (TIFF/IT)* Date: 2004 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO12647-2:2004]	*ISO 12647-2:2004* *Graphic technology - Process control for the production of half-tone colour separations, proof and production prints - Part 2: Offset lithographic processes.* Date: 2004 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO12647-2:2013]	*ISO 12647-2:2013* *Graphic technology - Process control for the production of half-tone colour separations, proof and production prints - Part 2: Offset lithographic processes.* Date: 2013 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO13655:1996]	*ISO 13655:1996* *Graphic technology -- Spectral measurement and colorimetric computation for graphic arts images.* Date: 1996 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO14977:1996]	*ISO 14977:1996(E)* *Information technology -- Syntactic metalanguage -- Extended BNF* Date: 1996 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO15930-1:2001]	*ISO 15930-1:2001* *Graphic technology — Prepress digital data exchange — Use of PDF — Part 1: Complete exchange using CMYK data (PDF/X-1 and PDF/X-1a).* Date: 2001 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO15930-3:2002]	*ISO 15930-3:2002* *Graphic technology — Prepress digital data exchange — Use of PDF — Part 3: Complete exchange suitable for colour-managed workflows (PDF/X-3).* Date: 2002 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO15930-6:2003]	*ISO 15930-6:2003* *Graphic technology — Prepress digital data exchange — Use of PDF — Part 3: Complete exchange suitable for colour-managed workflows (PDF/X-3).* Date: 2003 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO15930-7:2010]	*ISO 15930-7:2010* *Graphic technology — Prepress digital data exchange — Use of PDF — Complete exchange suitable for colour-managed workflows (PDF/X-4).* Date: 2010 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO15930-8:2010]	*ISO 15930-8:2010* *Graphic technology — Prepress digital data exchange — Use of PDF — Complete exchange suitable for colour-managed workflows (PDF/X-5).* Date: 2010 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html

Table K.1: References (Sheet 5 of 9)

TERM	DEFINITION
[ISO17972-1:2015]	*ISO 17972-1:2015* *Graphic technology -- Colour data exchange format -- Part 1: Relationship to CxF3 (CxF/X)* Date: 2015 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[ISO19593-1:2016]	*ISO 19593-1:2016* *Graphic technology -- Use of PDF to associate processing steps and content data -- Part 1: Processing steps 2016* Date: 2016 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
[japancolor]	*Japan Color 2001* Date: 2001 Produced by: Japan Printing Machinery Manufacturers Association, Office of JNC for TC130 Available at: Call (81) 03-3434-4661
[Japanese Paper Sizes]	*Japanese Papers for Printing. Metric measurements and inch equivalents.* Date: - Produced by: EDS Inc. Editorial & Design Services. Available at: http://www.edsebooks.com/paper/jpaper.html
[JIS P0138:1998]	*JIS P 0138:1998* *Writing paper and certain classes of printed matter -- Trimmed sizes -- A and B series.* Date: 1998 Produced by: JIS Available at: http://www.webstore.jsa.or.jp/webstore **Note:** If not available try via web seach of the title
[K&R]	*C Programming Language* , by Brian W. Kernighan and Dennis M. Ritchie *Second Edition* Date: March 22, 1988 Produced by: Prentice Hall Available at: Book only - ISBN 0131103628
[macbinary]	*Macintosh Binary Transfer Format ("MacBinary III") Standard Proposal.* Date: December 1996 Produced by: Macintosh Internet Developer Association Available at: http://www.lazerware.com/formats/
[opentypefont]	*OpenType specification* *v.1.4* Date: 11 October 2002 Produced by: Microsoft Corporation Available at: http://www.microsoft.com/typography/specs/
[PDF1.6]	*PDF reference : Adobe portable document format version 1.6 / Adobe Systems Incorporated.* *Version 1.6* Date: November 2004 Produced by: Addison-Wesley Available at: http://partners.adobe.com/public/developer/pdf/index_reference.html
[PJTF]	*The Portable Job Ticket Format* *Version 1.1* Date: 2 April 1999 Produced by: Adobe Systems Inc. Available at: http://partners.adobe.com/asn/developer/pdfs/tn/5620.pjtf.pdf
[PPD]	*Adobe PostScript Printer Description File Format Specification* *Version 4.3* Date: 9 February 1996 Produced by: Adobe Systems Inc. Available at: http://www.cip4.org/documents/technical_info/cip3v3_0.pdf
[PPML]	PPML] Personal Print Markup Language (PPML) Version 2.1 Date: - Produced by: Print On Demand Initiative (PODi) Available at: http://www.podi.org

Table K.1: References (Sheet 6 of 9)

TERM	DEFINITION
[PrintTalk]	*PrintTalk Implementation* *Version 1.1* Date: - Produced by: PrintTalk Consortium Available at: http://www.printtalk.org/
[Pro Carton]	*European Association of Carton and Cartonboard manufacturers.* Date: - Produced by: Pro Carton Available at: http://www.procarton.com/publications-news/publications/
[PS]	*PostScript Language Reference (Redbook)* *Third Edition* Date: - Produced by: Adobe Systems, Inc. Available at: http://partners.adobe.com/asn/developer/pdfs/tn/PLRM.pdf
[PWGFINMIB]	*Printer Finishing MIB* (draft-ietf-printmib-finishing-16.txt — work in progress.) Date: February 2003 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: https://datatracker.ietf.org/public/pidtracker.cgi **Note:** IETF Internet-Drafts have a six month life-time. See above link for latest version.
[Quark]	See http://www.quark.com.
[RFC1738]	*RFC 1738* *Uniform Resource Locators (URL)* Date: 1994 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC1741]	*RFC 1741* *MIME Content Type for BinHex Encoded Files, by Faltstrom, P., Crocker, D. and Fair, E.* Date: December 1994 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC1759]	*RFC 1759* *Printer MIB, Version 2.0 by Smith, R., Wright, F., Hastings, T., Zilles, S. and Gyllenskog, J.* Date: June 2003 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC1766]	*RFC 1766* *Tags for the Identification of Languages, by H. Alvestrand.* Date: March 1995 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC1950]	*RFC 1950* *ZLIB Compressed Data Format Specification version 3.3, by P. Deutsch.* Date: May 1996 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC1951]	*RFC 1951* *DEFLATE Compressed Data Format Specification version 1.3, by Deutsch, P.* Date: May 1996 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC1952]	*RFC 1952* *GZIP file format specification version 4.3, by Deutsch, P.* Date: May 1996 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html

Table K.1: References (Sheet 7 of 9)

TERM	DEFINITION
[RFC1977]	*RFC 1977* *PPP BSD Compression Protocol, by Schryver, V.* Date: May 1996 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC2045]	*RFC 2045* *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, by Freed, N. and Borenstein, N.* (Updated by RFC2184, RFC2231) Date: November 1996 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC2046]	*RFC 2046* *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, by Freed, N. and Borenstein, N.* (Updated by RFC2646) Date: November 1996 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC2231]	*RFC 2231* *MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations* Date: November 1997 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC2368]	*RFC 2368* *The mailto URL scheme by* P. Hoffman, L. Masinter and J. Zawinski. Date: July 1998 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC2387]	*RFC 2387* *The MIME Multipart/Related Content-type, by Levinson, E.* Date: August 1998 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC2392]	*RFC 2392* *Content-ID and Message-ID Uniform Resource Locators, by Levinson, E.* Date: August 1998 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC2616]	*RFC 2616* *Hypertext Transfer Protocol — HTTP/1.1* Date: June 1999 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC2822]	*RFC 2822* *Internet Message Format* Date: April 2001 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC3302]	*RFC 3302* *Tag Image File Format (TIFF) — image/tiff MIME Sub-type Registration, by Parsons, G., Rafferty, J.* Date: September 2002 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC3548]	*RFC 3548* *The Base16, Base32, and Base64 Data Encodings*, by S. Josefsson Date: July 2003 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html

Table K.1: References (Sheet 8 of 9)

TERM	DEFINITION
[RFC3966]	*RFC 3966* *The tel URI for Telephone Numbers by H. Schulzrinne* Date: December 2004 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC3986]	*FC 3986* *Uniform Resource Identifier (URI): Generic Syntax* by T. Berners-Lee, R. Fielding and L. Masinter Date: January 2005 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC3987]	*RFC 3987* *Internationalized Resource Identifiers (IRIs)*, by M. Duerst and M. Suignard Date: January 2005 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[RFC6101]	*RFC 6101* (Category: Historic) *The Secure Sockets Layer (SSL) Protocol Version 3.0*, by A. Freier, P. Karlton and P. Kocher Date: August 2011 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: http://www.rfc-editor.org/rfcsearch.html
[SSL3]	*SSL Specification* Freesoft, SSL Specification 3.0 Date: - Produced by: - Available at: http://www.freesoft.org/CIE/Topics/ssl-draft/3-SPEC.HTM **Note:** Refer to ▸ [RFC6101]
[TAPPI T480]	TAPPI T480 *Specular Gloss of Paper and Paperboard at 75 Degrees, Test Method T 519 om-99* Date: - Produced by: TAPPI Available at: http://www.tappi.org
[TAPPI T519]	TAPPI T519 *Diffuse Opacity of Paper (d/o paper backing), Test Method T 519 om-02* Date: - Produced by: TAPPI Available at: http://www.tappi.org
[TAPPI T527]	TAPPI T527 *Color of Paper and Paperboard (d/o, C/2), Test Method T 527 om-02* Date: - Produced by: TAPPI Available at: http://www.tappi.org
[TAPPI T560]	TAPPI T560 *CIE Whiteness and Tint of Paper and Paperboard (Using d/0°, Diffuse Illumination and Normal Viewing), Test Method T 560 wd-03* Date: - Produced by: TAPPI Available at: http://www.tappi.org
[TIFF6]	*TIFF Revision 6.0* Date: June 1992 Produced by: Adobe Systems, Inc. Available at: http://partners.adobe.com/asn/tech/tiff/specification.jsp
[TIFFPS]	*Adobe Photoshop TIFF Technical Notes* Date: March 2002 Produced by: Adobe Systems, Inc. Available at: http://partners.adobe.com/asn/tech/tiff/specification.jsp
[truetypefont]	*TrueType font file and TrueType Open specification* Date: August 1995 Produced by: Microsoft Corporation Available at: http://www.microsoft.com/typography/specs/

Table K.1: References (Sheet 9 of 9)

TERM	DEFINITION
[type1font]	*Adobe Type 1 Font Format* *Adobe Systems, Inc.* Date: 1990 Produced by: Addison-Wesley Publishing Company, Inc. Available at: http://partners.adobe.com/asn/developer/pdfs/tn/T1_SPEC.PDF
[Unicode5.0]	*The Unicode Standard, Version 5.0,* Date: November 3, 2006 Produced by: The Unicode Consortium Available at: http://www.unicode.org/book/aboutbook.html
[uuencode]	*Unix Uuencode, The Single UNIX ® Specification, Version 2* (Converts binary into the local character set that is suitable to pass through email systems.) Date: 1997 Produced by: The Open Group Available at: http://www.opengroup.org/onlinepubs/007908799/xcu/uuencode.html
[UPNP]	*Printer Device and Printer Basic Service* *Version 1.0* Date: 2002 Produced by: Universal Plug N Play Forum Available at: http://www.upnp.org/standardizeddcps/printer.asp
[URI]	*URIs, URLs, and URNs: Clarifications and Recommendations 1.0)* *Version (W3C Recommendation of 21 September 2001)* Date: 21 September 2001 Produced by: World Wide Web Consortium (W3C) Available at: http://www.w3.org/TR/2001/NOTE-uri-clarification-20010921/
[vCard]	*Standard file format for electronic business cards.* Date: N/A Produced by: Versit Consortium Refer to : https://en.wikipedia.org/wiki/VCard
[XML]	*Extensible Markup Language (XML) 1.0 (Fifth Edition)* *Version (W3C Recommendation of 26 November 2008)* Date: 26 November 2008 Produced by: World Wide Web Consortium (W3C) Available at: http://www.w3.org/TR/2008/REC-xml-20081126/
[XMLNS]	*Namespaces in XML 1.0 (Third Edition)* *Version (W3C Recommendation of 8 December 2009)* Date: *8 December 2009* Produced by: World Wide Web Consortium (W3C) Available at: http://www.w3.org/TR/2009/REC-xml-names-20091208/
[XPath]	*XML Path Language (XPath) 2.0 (Second Edition)* *Version W3C Recommendation 14 December 2010* Date: 14 December 2010 Produced by: World Wide Web Consortium (W3C) Available at: http://www.w3.org/TR/2010/REC-xpath20-20101214/
[XMLSchema]	*XML Schema Part 0+1+2: Primer, Structures and Datatypes* *Version (W3C Recommendation of 28 Oct 2004)* Date: *28 October 2004* Produced by: World Wide Web Consortium (W3C) XML Schema working group Available at: http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/ http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/ http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/
[X.509]	*Public-Key Infrastructure (X.509) (pkix)* *Version 1.1* Date: 14 February 2008 Produced by: IETF. Available at: http://www.ietf.org/html.charters/pkix-charter.html
[ZIP]	*File compression and archiving* *Version W3C Recommendation 14 December 2010* Date: 1 October 2014 Produced by: PKWARE Inc. Available at: https://support.pkware.com/display/PKZIP/APPNOTE

Appendix L

Hole Pattern Catalog

The following table defines the specifics of the predefined holes in ***HoleMakingParams*** and ***HoleMakingIntent***.

Notes:

1 All patterns are centered on the sheet along the reference edge.

2 Reference edge is always defined relative to a portrait orientation of the medium, regardless of the orientation of the printed image or processing path.

3 The pattern axis offset is always specified relative to the reference edge.

4 The default pattern axis offset is always specified in points.

5 Thumbcuts are available in various standard shapes (labeled "No. *N*" where *N* is minimally ranging from 2..7). "No. 3" seems to be the most widely used.

6 Single thumbcuts appear always in the center of the reference edge.

7 Oval shape holes actually look sometimes more like rectangular holes with rounded corners.

8 Generic hole types are dependent on the geographical area where the device is used.

Sources:

1 [PWGFINMIB]

L.1 Naming Scheme:

Table L.1: Naming Scheme for Hole Patterns

NAME	DESCRIPTION
General	<m|i>: m = metric (millimeter is used), i = imperial (inch, where 1 inch = 25.4 mm)
Ring Binding	R<#holes><m|i>-<variant> Example: R2m-DIN = RingBind, 2 hole, metric, DIN
Plastic Comb	P<pitch><m|i>-<shape>-<#thumbcuts>t Example: P16:9m-round-0t = Plastic Comb, 9/16" pitch (16:9), round, no thumbcut
Wire Comb	W<pitch><m|i>-<shape>-<#thumbcuts>t Example: W2:1i-square-1t = Wire Comb, 1/2" pitch (2:1), square, one thumbcut
Coil/Spiral	C<pitch><m|i>-<shape>-<#thumbcuts>t Example: C9.5m-round-0t = Coil, 9.5 mm, round, no thumbcut
Special	S<#holes> Example: S1-generic

L.2 Ring Binding - Two Hole

Table L.2: Hole Details for R2 Series (Sheet 1 of 2)

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
R2-generic	Generic request of a 2-hole pattern	●	5 - 13 mm 0.2-0.51"	N/A	4.5 – 13 mm 0.18 - 0.51"	34.02 (≅ 12 mm)	Left	See note (8)	N/A
R2m-DIN	DIN 2-hole MIB: 6 = twoHoleDIN and 10 = twoHoleMetric	●	5.5 ± 0.1 mm	80 ± 0.1 mm	7 or 11 ± 0.3 mm 7 mm for blocks of <= 15 mm thick	31.18 (≅ 11 mm)	Left	A4 and A5	DIN 5005:1991 DIN 821:1973
R2m-ISO	ISO 2-hole MIB: 6 = twoHoleDIN and 10 = twoHoleMetric	●	6 ± 0.5 mm	80 ± 0.5 mm	12 ± 1 mm Australian Standard AS P5-1969: 10 ± 1 mm	34.02 (≅ 12 mm)	Left	Also used in Japan	ISO 838:1974 (E)
R2m-MIB	Printer Finishing MIB twoHoleDIN and twoHoleMetric	●	5-8 mm	80 ± 0.5 mm	4.5 – 13 mm	31.18 (≅ 11 mm)	Left		Printer Finishing MIB
R2i-US-a	US 2-hole, Variant A MIB: 4 = twoHoldUSTop and 12 = twoHoleUSSide	●	0.2 - 0.32"	2.75"	0.18 - 0.51"	29.25 (≅ 13/32")	Left for letter Top for ledger		Printer Finishing MIB

Table L.2: Hole Details for R2 Series (Sheet 2 of 2)

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
R2i-US-b	US 2-hole, Variant B	●	0.2-0.5" Default: 5/16" Typical: 1/4", 9/32", 11/32", 3/8", 13/32", 1/2"	6"	0.25" + ½ diameter Range: 6/16" - 1/2"	29.25 (≅ 13/32")	Left		

L.3 Ring Binding - Three Hole

Table L.3: Hole Details for R3 Series

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
R3-generic	Generic request of a 3-hole pattern.	●	5 - 13 mm 0.2-0.51”	N/A	4.5 – 13 mm 0.18 - 0.51"	29.25 (≅ 13/32")	Left	See note (8).	N/A
R3i-US	US 3-hole MIB: 5 = threeHoleUS	●	standard: 5/16" Range: 0.2-0.5" Typical: 1/4", 9/32", 11/32", 3/8", 13/32", 1/2"	4.25"	0.25" + ½ diameter range: 6/16" - 1/2"	29.25 (≅ 13/32")	Left		Printer Finishing MIB

L.4 Ring Binding - Four Hole

Table L.4: Hole Details for R4 Series (Sheet 1 of 2)

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
R4-generic	Generic request of a 4-hole pattern.	●	5 - 13 mm 0.2-0.51”	N/A	4.5 – 13 mm 0.18 - 0.51"	31.18 (≅ 11 mm)	Left	See note (8).	N/A
R4m-DIN-A4	DIN 4-hole for A4	●	5.5 ± 0.1 mm	80 ± 0.1 mm	7 or 11 ± 0.3 mm 7 mm for blocks of 15 mm or less	31.18 (≅ 11 mm)	Left	A4	DIN 5005:1991 DIN 821:1973
R4m-DIN-A5	DIN 4-hole for A5	●	5.5 ± 0.1 mm	45-65-45 mm	7 or 11 ± 0.3 mm 7 mm for blocks of 15 mm or less	31.18 (≅ 11 mm)	Left	A5	DIN 5005:1991
R4m-swedish	Swedish 4-hole MIB: 11 = swedish4Hole	●	5 - 8 mm	21-70-21 mm	4.5 - 13 mm	31.18 (≅ 11 mm)	Left for A4 Top for A3	A4, A3	Printer Finishing MIB

Table L.4: Hole Details for R4 Series (Sheet 2 of 2)

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
R4i-US	US 4-hole	●	0.2 - 0.5" Standard: 5/16" Typical: 1/4", 9/32", 11/32", 3/8", 13/32", 1/2"	1.375-4.25-1.375"	0.25" + ½ diameter Range: 6/16" - 1/2"	29.25 (≅ 0.25" + ½ x 5/16" = 13/32")	Left		

L.5 RingBinding - Five Hole

Table L.5: Hole Details for R5 Series

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
R5-generic	Generic request of a 5-hole pattern.	●	5 - 13 mm 0.2-0.51"	N/A	4.5 – 13 mm 0.18 - 0.51"	29.25 (≅ 13/32")	Left	See note (8).	N/A
R5i-US-a	US 5-hole, Variant A MIB: 13 = five-HoleUS	●	0.2 - 0.32"	2-2.25-2.25-2"	0.18 - 0.51"	29.25 (≅ 13/32")	Left for letter Top for ledger		Printer Finishing MIB
R5i-US-b	US 5-hole, Variant B	●	0.2 - 0.5" std: 5/16" typ: 1/4", 9/32", 11/32", 3/8", 13/32", 1/2"	0.75-3.5-3.5-0.75"	0.25" + ½ diameter 0.375 - 0.5"	29.25 (≅ 0.25" + ½ x 5/16" = 13/32")	Left		
R5i-US-c	Combination of R2i-US-a and R3i-US	●	0.2 - 0.5" std: 5/16" typ: 1/4", 9/32", 11/32", 3/8", 13/32", 1/2"	1.25-3-3-1.25"	0.25" + ½ diameter 0.375 - 0.5"	29.25 (≅ 0.25" + ½ x 5/16" = 13/32")	Left		

L.6 Ring Binding - Six Hole

Table L.6: Hole Details for R6 Series (Sheet 1 of 2)

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
R6-generic	Generic request of a 6-hole pattern.	●	5 - 13 mm 0.2-0.51"	N/A	4.5 – 13 mm 0.18 - 0.51"	31.18 (≅ 11 mm)	Left for A4/A5 Top for A3	See note (8).	N/A
R6m-4h2s	Norwegian 4-hole (round) mixed with 2 slots (rectangular) MIB: 16 = norweg6Hole	H: ● S: ■ ■	Holes: 5 - 8 mm Slots: 10 x 5.5 mm	4 holes/2 slots Pattern: H-H-S-S-H-H 64-18.5-75-18.5-64 mm	4.5 - 13 mm	31.18 (≅ 11 mm)	Left for A4 Top for A3		Printer Finishing MIB

Table L.6: Hole Details for R6 Series (Sheet 2 of 2)

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
R6m-DIN-A5	DIN 6-hole for A5	●	5.5 ± 0.1 mm	37.5-7.5-65-7.5-37.5 mm	7 or 11 ± 0.3 mm 7 mm for blocks of <= 15 mm thick	31.18 (≅ 11 mm)	Left	Only used with A5	DIN 5005:1991

L.7 Ring Binding - Seven Hole

Table L.7: Hole Details for R7 Series

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
R7-generic	Generic request of a 7-hole pattern.	●	5 - 13 mm 0.2-0.51"	N/A	4.5 – 13 mm 0.18 - 0.51"	29.25 (≅ 13/32")	Left for letter Top for ledger	See note (8).	N/A
R7i-US-a	US 7-hole, Variant A MIB: 14 = sevenHoleUS	●	0.2 - 0.32"	1-1-2.25-2.25-1-1"	0.18 - 0.51"	29.25 (≅ 13/32")	Left for letter Top for ledger		Printer Finishing MIB
R7i-US-b	US 7-hole, Bell/AT&T Systems. Combination of R3i-US, R4i-US, R5i-US-b	●	0.2 - 0.5" std: 5/16" typ: 1/4", 9/32", 11/32", 3/8", 13/32", 1/2"	0.75-1.375-2.125-2.125-1.375-0.75"	0.25" + ½ diameter 0.375 - 0.5"	29.25 (≅ 0.25" + ½ x 5/16" = 13/32")	Left for letter Top for ledger		
R7i-US-c	US 7-hole, Variant C	●	0.2 - 0.5" std: 5/16" typ: 1/4", 9/32", 11/32", 3/8", 13/32", 1/2"	1.25-0.875-2.125-2.125-0.875-1.25"	0.25" + ½ diameter 0.375 - 0.5"	29.25 (≅ 13/32")	Left for letter Top for ledger		

L.8 Ring Binding - Eleven Hole

Table L.8: Hole Details for R11 Series

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
R11m-7h4s	7-hole (round) mixed with 4 slots (rectangular) MIB: 15 = mixed7H4S	H: ● S: ■ ■	Holes: 5 - 8 mm Slots: 12 x 6 mm	7 holes/2slots Pattern: H-S-H-H-S-H-S-H-H-S-H 15-25-23-20-37-37-20-23-25-15 mm	4.5 - 13 mm	31.18 (≅ 11 mm)	Left for A4 Top for A3		Printer Finishing MIB

L.9 Plastic Comb Binding

Table L.9: Hole Details for P Series

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
P16_9i-rect-0t	US spacing, no thumbcut MIB: 9 = nineteenHo-leUS A4: 21 Hole Letter: 19 Hols	■ ■	5/16" x 1/8" (8 x 3.2 mm)	9/16"	3/16"	13.54 (≅ 0.188")	Left		Printer Finishing MIB
P12m-rect-0t	European spacing, no thumbcut	■ ■	7 x 3 mm	12 mm	4.5 mm	12.76 (≅ 4.5 mm)	Left		

L.10 Wire Comb Binding

Wire comb binding uses twenty three holes for pages of A4 size, and twenty one holes for pages of letter size

Table L.10: Hole Details for W Series

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
W2_1i-round-0t	2:1, round, no thumbcut MIB: 8 = twentyTwo-HoleUS A4: 23 Hole Letter: 21 Hole	●	0.2 - 0.32" std: 1/4" Europe typ: 6 or 6.4 mm	1/2"	3 mm + ½ diameter 0.318 - 0.438" Europe: 6 - 6.2 mm	17.50 (≅ 0.243")	Left		Printer Finishing MIB
W2_1i-square-0t	2:1, square, no thumbcut A4: 23 Hole Letter: 21 Hole	■	0.2 - 0.32" std: 1/4" Europe typ: 6 or 6.4 mm	1/2"	3 mm + ½ diameter 0.318 - 0.438" Europe: 6 - 6.2 mm	17.50 (≅ 0.243")	Left		
W3_1i-square-0t	3:1, square, no thumbcuts A4: 34 Hole A5: 24 Hole Letter: 32 Hole	■	5/32 x 5/32" (4x4 mm)	1/3"	0.2"	14.40 (≅ 0.2")	Left		

L.11 Coil and Spiral Binding

Table L.11: Hole Details for C Series

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
C9.5m-round-0t	9.5 mm, round, no thumbcut MIB: 17 - metric26Hole and 18 - metric30Hole A4: 30 Hole A3: 30 Hole JIS: 26 Hole B4: 26 Hole B5: 26 Hole	●	5 - 8 mm	9.5 mm	4.5 - 13 mm	31.18 (≅ 11 mm)	Left for A4/ JIS B5 Top for A3/ JIS B4		Printer Finishing MIB

L.12 Special Binding

Table L.12: Hole Details for S Series

HOLE PATTERN ID	DESCRIPTION	SHAPE	HOLE EXTENT	PATTERN GEOMETRY	PATTERN AXIS OFFSET	JDF DEFAULT PATTERN AXIS OFFSET	EDGE	NOTE	SOURCE
S-generic	Generic request of a hole pattern with an arbitrary or unknown number of holes (e.g., an inline shotgun).	●	5 - 13 mm 0.2-0.51”	N/A	N/A	N/A	Any		N/A
S1-generic	Generic request of a hole pattern with 1 hole.	●	5 - 13 mm 0.2-0.51”	N/A	N/A	N/A	Any		N/A

Appendix M

FileSpec Use Cases

New in JDF 1.2

The purpose of this appendix is to give a series of use cases with examples for the use of the Attributes of ***FileSpec***: @*MimeType*, @*URL*, @*Compression* and the ***FileSpec***/***Container*** Subelement. These use cases include container packaging files, such as tar, zip and Multipart/Related files and container compression and encoding files, each of which require one or more ***Container*** Subelements to link one ***FileSpec*** with its container ***FileSpec***.

M.1 Examples of Attribute Values of FileSpec

‣ Table M.1 Use Cases showing MimeType, URL and Compression Attribute Values shows a number of use cases and the corresponding values for the @*MimeType*, @*URL* and @*Compression* Attributes. Each ***Container*** Element points to the ***FileSpec*** shown on the next row in the table. The use cases are arranged in order of increasing complexity.

Note: All of the @*URL* examples in this appendix for ***FileSpec*** resources that are not contained in other files are Absolute URIs, so that the complication of resolving ***FileSpec***/@*URI* with ***RunList***/@*Directory* is not considered. Of course, the @*URL* examples for ***FileSpec*** resources that are contained in other files SHALL all be Relative URIs (relative to the Base URI that is defined to be the Absolute URI of where the **JDF** consumer extracted the container file) as the **JDF** spec requires (see the @*URL* description at ‣ Section 8.58 FileSpec).

Table M.1: Use Cases showing MimeType, URL and Compression Attribute Values (Sheet 1 of 2)

DESCRIPTION OF USE CASE	MIME TYPE	URL	COMPRESSION
1.) Single a.pdf PDF file, no compression	application/pdf	ftp://www.any.com/share/a.pdf	
2.) Single a.pdf PDF file, with Gzip compression	application/pdf	a.pdf	Gzip
Container FileSpec	Gzip	ftp://www.any.com/a.gz	
3.) Single a.pdf PDF file, no compression, but Base64 encoded	application/pdf	a.pdf	Base64
Container FileSpec	Base64	ftp://www.any.com/a.mme	
4.) Single PDF file, no compression, but BinHex encoded into a BinHex file	application/pdf	a.pdf	BinHex
Container FileSpec	application/mac-binhex40	ftp://www.any.com/a.hqx	
5.) Single a.pdf PDF file with ZLIB compression in b.zip ZIP file (containing one or more files)	application/pdf	a.pdf	ZLIB
Container FileSpec	application/zip	ftp://www.any.com/b.zip	
6.) Single a.pdf PDF file compressed by Deflate in a b.zip with one or more files, and the b.zip packaging file itself is Base64 encoded as b.mme. To read, unencode, then uncompress. To write, compress, then encode.	application/pdf	a.pdf	Deflate
Container FileSpec	application/zip	b.zip	Base64
Container FileSpec	Base64	ftp://www.any.com/b.mme	

Table M.1: Use Cases showing MimeType, URL and Compression Attribute Values (Sheet 2 of 2)

DESCRIPTION OF USE CASE	MIME TYPE	URL	COMPRESSION
7.) Single myFiles/myPicture.jpg file in myNestedZip.zip file with one or many files, but the myNestedZip.zip is itself zipped into c.zip.	image/jpeg	myFiles/myPicture.jpg	Deflate
Container FileSpec	application/zip	myNestedZip.zip	Deflate
Container FileSpec	application/zip	ftp://www.any.com/c.zip	
8.) Single a.pdf PDF file which is ZLIB compressed, in a c.zip with one or many files which is contained in a tar file and compressed with ZLIB into a.tar.gz file.	application/pdf	a.pdf	ZLIB
Container FileSpec	application/zip	c.zip	
Container FileSpec	Tar[a]	d.tar	ZLIB
Container FileSpec	GZip	ftp://www.any.com/d.tar.gz	

a. The UNIX Tar file packaging format is not registered with IANA as a MIME media type, so CIP4 has assigned the "Tar" file type to it for use in the ***FileSpec***/@*MimeType* Attribute.

M.2 Corresponding XML examples

The above use case examples are represented in XML as follows:

Example M.1: FileSpec #1

Single a.pdf PDF file, No Compression:

```
<FileSpec Class="Parameter" ID="F1" Status="Available"
       MimeType="application/pdf" URL="ftp://www.any.com/share/a.pdf"/>
```

Example M.2: FileSpec #2

Single a.pdf PDF file, with Gzip compression:

```
<FileSpec Class="Parameter" ID="F1" Status="Available"
       Compression="Gzip" MimeType="application/pdf" URL="a.pdf">
    <Container>
       <FileSpec MimeType="Gzip" URL="ftp://www.any.com/a.gz"/>
    </Container>
</FileSpec>
```

Example M.3: FileSpec #3

Single a.pdf PDF file, no compression, but Base64 encoded:

```
<FileSpec Class="Parameter" ID="F1" Status="Available"
       Compression="Base64" MimeType="application/pdf" URL="a.pdf">
    <Container>
       <FileSpec MimeType="Base64" URL="ftp://www.any.com/a.mme"/>
    </Container>
</FileSpec>
```

Example M.4: FileSpec #4

Single PDF file, no compression, but BinHex encoded:

```
<FileSpec Class="Parameter" ID="F1" Status="Available"
       Compression="BinHex" MimeType="application/pdf" URL="a.pdf">
    <Container>
       <FileSpec MimeType="application/mac-binhex40"
              URL="ftp://www.any.com/a.hqx"/>
    </Container>
</FileSpec>
```

Example M.5: FileSpec #5

Single a.pdf PDF file, in b.zip ZIP file containing one or more files:

```
<FileSpec Class="Parameter" ID="F1" Status="Available"
       Compression="ZLIB" MimeType="application/pdf" URL="a.pdf">
    <Container>
       <FileSpec MimeType="application/zip" URL="ftp://www.any.com/b.zip"/>
    </Container>
</FileSpec>
```

Example M.6: FileSpec #6

Single a.pdf PDF file, in a b.zip with one or more files, and the b.zip packaging file itself is Base64 encoded as b.mme. To read, decode, then decompress. To write, compress, then encode.

```
<FileSpec Class="Parameter" ID="F1" Status="Available"
       Compression="Deflate" MimeType="application/pdf" URL="a.pdf">
    <Container>
       <FileSpec Compression="Base64" MimeType="application/zip" URL="b.zip">
          <Container>
             <FileSpec MimeType="Base64" URL="ftp://www.any.com/b.mme"/>
          </Container>
       </FileSpec>
    </Container>
</FileSpec>
```

Example M.7: FileSpec #7

Single myFiles/myPicture.jpg file in myNestedZip.zip file with one or many files, but the myNestedZip.zip is itself zipped into c.zip

```
<FileSpec Class="Parameter" ID="F1" Status="Available"
       Compression="Deflate" MimeType="image/jpeg" URL="myFiles/myPicture.jpg">
    <Container>
       <FileSpec Compression="Deflate" MimeType="application/zip"
              URL="myNestedZip.zip">
          <Container>
             <FileSpec MimeType="application/zip"
                    URL="ftp://www.any.com/c.zip"/>
          </Container>
       </FileSpec>
    </Container>
</FileSpec>
```

Example M.8: FileSpec #8

Single a.pdf PDF file, which is ZLIB compressed in a c.zip with one or many files which is contained in a tar file and compressed with ZLIB into a.tar.gz file.:

```
<FileSpec Class="Parameter" ID="F1" Status="Available"
       Compression="ZLIB" MimeType="application/pdf" URL="a.pdf">
    <Container>
       <FileSpec MimeType="application/zip" URL="c.zip">
          <Container>
             <FileSpec Compression="ZLIB" MimeType="Tar" URL="d.tar">
                <Container>
                   <FileSpec MimeType="GZip"
                      URL="ftp://www.any.com/d.tar.gz"/>
                </Container>
             </FileSpec>
          </Container>
       </FileSpec>
    </Container>
</FileSpec>
```

M.3 Additional examples showing Partitioning of FileSpec

This section has additional examples of container files and various schemes of Partitioning.

Example M.9: FileSpec #9

Package b.zip contains multiple pdf files a.pdf, b.pdf etc.

```
<FileSpec Class="Parameter" Status="Available" ID="ID_002"
          MimeType="application/zip"
          URL="ftp://www.any.com/b.zip"/>
<FileSpec Class="Parameter" Status="Available" ID="A_FILE"
       Compression="Deflate" MimeType="application/pdf"
       URL="a.pdf">
    <Container>
       <FileSpecRef rRef="ID_002"/>
    </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="B_FILE"
       Compression="Deflate" MimeType="application/pdf"
       URL="b.pdf">
    <Container>
       <FileSpecRef rRef="ID_002"/>
    </Container>
</FileSpec>
```

Example M.10: FileSpec #10

Package b.zip contains two pdf files a.pdf, b.pdf and a tiff, c.tiff used by a Partitioned Resource

```
<FileSpec Class="Parameter" Status="Available" ID="ID_003"
       MimeType="application/zip" URL="ftp://www.any.com/b.zip"/>
<FileSpec Class="Parameter" Status="Available" ID="ALL_FILES"
       Compression="Deflate" MimeType="application/pdf"
       PartIDKeys="PartVersion">
    <Container>
       <FileSpecRef rRef="ID_003"/>
    </Container>
    <FileSpec PartVersion="English" URL="a.pdf"/>
    <FileSpec PartVersion="French" URL="b.pdf"/>
    <FileSpec MimeType="application/tif" PartVersion="German" URL="c.tif"/>
</FileSpec>
```

Example M.11: FileSpec #11

Single a.pdf PDF file, in b.zip which is contained in c.tar file:

```
<FileSpec Class="Parameter" Status="Available" ID="ID_004_TAR" MimeType="Tar"
      URL="ftp://www.any.com/c.tar"/>
<FileSpec Class="Parameter" Status="Available" ID="ID_004_ZIP"
      MimeType="application/zip" URL="b.zip">
   <Container>
      <FileSpecRef rRef="ID_004_TAR"/>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="C_FILE"
      Compression="Deflate" MimeType="application/pdf" URL="a.pdf">
   <Container>
      <FileSpecRef rRef="ID_004_ZIP"/>
   </Container>
</FileSpec>
```

Example M.12: FileSpec #11.1 — No Partitioning

Multiple files in several zip's contained in a tar file, various examples with and without Partitioning,

So the file layout looks like:

```
e.tar
c.zip
a.pdf
b.pdf
d.zip
a.pdf
b.pdf
```

```
<FileSpec Class="Parameter" Status="Available" ID="ID_005_TAR"
       MimeType="Tar" URL="ftp://www.any.com/e.tar"/>
<FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_C"
       MimeType="application/zip" URL="c.zip">
   <Container>
      <FileSpecRef rRef="ID_005_TAR"/>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_D"
       MimeType="application/zip" URL="d.zip">
   <Container>
      <FileSpecRef rRef="ID_005_TAR"/>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="A_ENGLISH_FILE"
       Compression="Deflate" MimeType="application/pdf"
       URL="a.pdf">
   <Container>
      <FileSpecRef rRef="ID_005_ZIP_C"/>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="B_ENGLISH_FILE"
       Compression="Deflate" MimeType="application/pdf"
       URL="b.pdf">
   <Container>
      <FileSpecRef rRef="ID_005_ZIP_C"/>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="A_GERMAN_FILE"
       Compression="Deflate" MimeType="application/pdf"
       URL="a.pdf">
   <Container>
      <FileSpecRef rRef="ID_005_ZIP_D"/>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="B_GERMAN_FILE"
       Compression="Deflate" MimeType="application/pdf"
       URL="b.pdf">
   <Container>
      <FileSpecRef rRef="ID_005_ZIP_D"/>
   </Container>
</FileSpec>
```

Example M.13: FileSpec #11.2 — Intermediate container Partitioned

```
<FileSpec Class="Parameter" Status="Available" ID="ID_005_TAR" MimeType="Tar"
       URL="ftp://www.any.com/e.tar"/>
<FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIPS"
       MimeType="application/zip" PartIDKeys="PartVersion">
   <Container>
      <FileSpecRef rRef="ID_005_TAR"/>
   </Container>
   <FileSpec PartVersion="English" URL="c.zip"/>
   <FileSpec PartVersion="German" URL="d.zip"/>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="A_ENGLISH_FILE"
       Compression="Deflate" MimeType="application/pdf"
       URL="a.pdf">
   <Container>
      <FileSpecRef rRef="ID_005_ZIPS">
         <Part PartVersion="English"/>
      </FileSpecRef>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="B_ENGLISH_FILE"
       Compression="Deflate" MimeType="application/pdf"
       URL="b.pdf">
   <Container>
      <FileSpecRef rRef="ID_005_ZIPS">
         <Part PartVersion="English"/>
      </FileSpecRef>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="A_GERMAN_FILE"
       Compression="Deflate" MimeType="application/pdf"
       URL="a.pdf">
   <Container>
      <FileSpecRef rRef="ID_005_ZIPS">
         <Part PartVersion="German"/>
      </FileSpecRef>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="B_GERMAN_FILE"
       Compression="Deflate" MimeType="application/pdf"
       URL="b.pdf">
   <Container>
      <FileSpecRef rRef="ID_005_ZIPS">
         <Part PartVersion="German"/>
      </FileSpecRef>
   </Container>
</FileSpec>
```

Example M.14: FileSpec #11.3 — the pdf is Partitioned

```
<FileSpec Class="Parameter" Status="Available" ID="ID_005_TAR"
      MimeType="Tar" URL="ftp://www.any.com/e.tar"/>
<FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_C"
      MimeType="application/zip" URL="c.zip">
   <Container>
      <FileSpecRef rRef="ID_005_TAR"/>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_D"
      MimeType="application/zip" URL="d.zip">
   <Container>
      <FileSpecRef rRef="ID_005_TAR"/>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="ALL_FILES"
      Compression="Deflate" PartIDKeys="PartVersion DocIndex">
   <!-- English Files -->
   <FileSpec PartVersion="English">
      <Container>
         <FileSpecRef rRef="ID_005_ZIP_C"/>
      </Container>
      <!-- English File A -->
      <FileSpec DocIndex="1" MimeType="application/pdf" URL="a.pdf"/>
      <!-- English File B -->
      <FileSpec DocIndex="2" MimeType="application/pdf" URL="b.pdf"/>
   </FileSpec>
   <!-- German Files -->
   <FileSpec PartVersion="German">
      <Container>
         <FileSpecRef rRef="ID_005_ZIP_D"/>
      </Container>
      <!-- German File A -->
      <FileSpec DocIndex="1" MimeType="application/pdf" URL="a.pdf"/>
      <!-- German File B -->
      <FileSpec DocIndex="2" MimeType="application/pdf" URL="b.pdf"/>
   </FileSpec>
</FileSpec>
```

Example M.15: FileSpec #11.3a — the pdf is Partitioned, Different File Layout

As above but the file layout is not reflected in the container structure, the files are intermingled

```
<FileSpec Class="Parameter" Status="Available" ID="ID_005_TAR"
       MimeType="Tar" URL="ftp://www.any.com/e.tar"/>
<FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_C"
       MimeType="application/zip" URL="c.zip">
   <Container>
      <FileSpecRef rRef="ID_005_TAR"/>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_D"
       MimeType="application/zip" URL="d.zip">
   <Container>
      <FileSpecRef rRef="ID_005_TAR"/>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="ALL_FILES"
       Compression="Deflate" MimeType="application/pdf"
       PartIDKeys="PartVersion DocIndex">
   <!-- English Files -->
   <FileSpec PartVersion="English">
      <!-- English File A -->
      <FileSpec DocIndex="1" URL="a.pdf">
         <Container>
            <FileSpecRef rRef="ID_005_ZIP_C"/>
         </Container>
      </FileSpec>
      <!-- English File B -->
      <FileSpec DocIndex="2" URL="a.pdf">
         <Container>
            <FileSpecRef rRef="ID_005_ZIP_D"/>
         </Container>
      </FileSpec>
   </FileSpec>
   <!-- German Files -->
   <FileSpec PartVersion="German">
      <!-- German File A -->
      <FileSpec DocIndex="1" URL="b.pdf">
         <Container>
            <FileSpecRef rRef="ID_005_ZIP_C"/>
         </Container>
      </FileSpec>
      <!-- German File B -->
      <FileSpec DocIndex="2" URL="b.pdf">
         <Container>
            <FileSpecRef rRef="ID_005_ZIP_D"/>
         </Container>
      </FileSpec>
   </FileSpec>
</FileSpec>
```

Example M.16: FileSpec #11.4 — Both Partitioned

```
<FileSpec Class="Parameter" Status="Available" ID="ID_005_TAR" MimeType="Tar"
      URL="ftp://www.any.com/e.tar"/>
<FileSpec ID="ID_005_ZIPS" MimeType="application/zip"
      PartIDKeys="PartVersion">
   <Container>
      <FileSpecRef rRef="ID_005_TAR"/>
   </Container>
   <FileSpec PartVersion="English" URL="c.zip"/>
   <FileSpec PartVersion="German" URL="d.zip"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="ALL_FILES"
      PartIDKeys="PartVersion DocIndex">
   <!-- English Files -->
   <FileSpec PartVersion="English">
      <Container>
         <FileSpecRef rRef="ID_005_ZIPS">
            <Part PartVersion="English"/>
         </FileSpecRef>
      </Container>
      <!-- English File A -->
      <FileSpec DocIndex="1" MimeType="application/pdf" URL="a.pdf"/>
      <!-- English File B -->
      <FileSpec DocIndex="2" MimeType="application/pdf" URL="b.pdf"/>
   </FileSpec>
   <!-- German Files -->
   <FileSpec PartVersion="German">
      <Container>
         <FileSpecRef rRef="ID_005_ZIPS">
            <Part PartVersion="German"/>
         </FileSpecRef>
      </Container>
      <!-- German File A -->
      <FileSpec DocIndex="1" MimeType="application/pdf" URL="a.pdf"/>
      <!-- German File B -->
      <FileSpec DocIndex="2" MimeType="application/pdf" URL="b.pdf"/>
   </FileSpec>
</FileSpec>
```

Example M.17: FileSpec #12

Multiple PDF and TIFF files in several zip's contained in a tar file. Use all PDF files in c.zip, using the ***FileSpec***/@*FileFormat* mechanism and just Pictures/TIFS/a.pdf in d.zip. File layout looks like:

```
e.tar
c.zip
a.pdf
a.tif
b.pdf
b.tif
d.zip
PDFS/a.pdf
PDFS/b.pdf
Pictures/TIFS/a.pdf
Pictures/TIFS/b.pdf
```

```
<FileSpec Class="Parameter" Status="Available" ID="ID_005_TAR" MimeType="Tar"
      URL="ftp://www.any.com/e.tar"/>
<FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_C"
      MimeType="application/zip" URL="c.zip">
   <Container>
      <FileSpecRef rRef="ID_005_TAR"/>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_D"
      MimeType="application/zip" URL="d.zip">
   <Container>
      <FileSpecRef rRef="ID_005_TAR"/>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="PDF_FILES"
      Compression="Deflate" FileFormat="%s.pdf" FileTemplate="all"
      MimeType="application/pdf">
   <Container>
      <FileSpecRef rRef="ID_005_ZIP_C"/>
   </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="Pictures"
      Compression="Deflate" URL="Pictures/TIFS/a.pdf">
   <Container>
      <FileSpecRef rRef="ID_005_ZIP_D"/>
   </Container>
</FileSpec>
```

M.4 Example of an Intent Job Ticket with a doubly nested ZIP packaging file

Here is a complete example of an intent Job ticket using ***ArtDeliveryIntent*** with a doubly nested packaging file. The example shows a myPictures.jpg file that is contained in myNestedZip.zip file which is contained in myZip.zip file:

Example M.18: Intent Job Ticket

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="FileSpecProposal01" JobID="bookJob"
       JobPartID="bookJob-1" Status="Waiting" Type="Product" Version="1.4">
    <ResourcePool>
       <ArtDeliveryIntent Class="Intent" ID="FileSpecProposal02" Status="Draft">
          <ArtDelivery ArtDeliveryType="DigitalMedia">
             <RunListRef rRef="FileSpecProposal05"/>
          </ArtDelivery>
       </ArtDeliveryIntent>
       <RunList ID="FileSpecProposal05" Class="Parameter" Status="Available">
          <LayoutElement>
             <FileSpec Compression="Deflate" MimeType="image/jpeg"
                    URL="myFiles/myPicture.jpg">
                <Container>
                   <FileSpecRef rRef="ID_002"/>
                </Container>
             </FileSpec>
          </LayoutElement>
       </RunList>
       <Component Amount="100" Class="Quantity" ComponentType="FinalProduct"
             DescriptiveName="FileSpec Test" ID="FileSpecProposal03"
             Status="Unavailable"/>
       <FileSpec Class="Parameter" Status="Available" ID="ID_001"
             MimeType="application/zip" URL="http://www.CIP4.org/myZip.zip"/>
       <FileSpec Class="Parameter" Status="Available" Compression="Deflate"
             ID="ID_002" MimeType="application/zip" URL="myNestedZip.zip">
          <Container>
             <FileSpecRef rRef="ID_001"/>
          </Container>
       </FileSpec>
    </ResourcePool>
    <ResourceLinkPool>
       <ComponentLink Amount="100" Usage="Output" rRef="FileSpecProposal03"/>
       <ArtDeliveryIntentLink Usage="Input" rRef="FileSpecProposal02"/>
    </ResourceLinkPool>
</JDF>
```

M.5 AppOS and OSVersion Attributes

New in JDF 1.2

This section lists examples values for the following Attributes of the ***FileSpec*** Resource: *@AppOS* and *@OSVersion*. The listing is intended to be exhaustive for the most likely operating systems that are routinely used in **JDF** applications. However, other operating systems and combinations MAY be used as well. When operating systems have new versions, they can be used and SHOULD follow the patterns established in this the following table.

Table M.2: AppOS and OSVersion Examples (Sheet 1 of 2)

APPOS	OSVERSION	DESCRIPTION
Linux	2.2	Linux operation system
Mac	10.2.4	Macintosh operation system
Solaris	4.0	Sun Solaris operation system
UNIX	BSD	Berkeley UNIX
UNIX	V	System V UNIX
UNIX	V.1	System V UNIX
UNIX	V.2	System V UNIX
UNIX	V.3	System V UNIX
UNIX	PC	UNIX for the PC

Table M.2: AppOS and OSVersion Examples (Sheet 2 of 2)

APPOS	OSVERSION	DESCRIPTION
Windows	95	Windows 95
Windows	98	Windows 98
Windows	NT	Windows NT
Windows	NT-5	Windows 2000
Windows	NT-5.1	XP [not yet registered by Microsoft with IANA]

Appendix N

Deprecated Items

Processes and Resources that have been deprecated in their entirety have been moved to this appendix. The name of the deprecated Process or Resource remains in those chapters along with directions from CIP4 working groups on the preferred method of handling Job data in the latest version of **JDF**. The original Processes and Resources are provided here only for users and developers of **JDF** solutions who require this information to solve backwards compatibility issues; however, we strongly encourage that the use of these deprecated Resources and Process be eliminated from your **JDF** environment to reduce complexity.

Note: Deprecated attributes and elements within process and resources which themselves have not been entirely deprecated remain in the main body of this standard, and this appendix is not meant to be an exhaustive catalog of all deprecated items within **JDF**.

N.1 Deprecated Structures of JDF Nodes and Jobs

N.1.1 PlaceHolder Resource

Deprecated in JDF 1.5

PlaceHolder Resources, unlike ***PhysicalResource***s, do not describe any logical or physical entity. Rather, they define Process linking and help to define Process ordering when the exact nature of interchange Resources is still unknown. In essence, they serve as placeholders that stand in for defined Resources. Using ***PlaceHolder Resource***s, a processing skeleton can be constructed that gives a basic shape to a Job. The appropriate Resources can be substituted for ***PlaceHolder Resource***s when they become known.

This kind of Resource SHOULD only be used to link Nodes of @*Type* = "ProcessGroup", since Process leaf Nodes have well defined Resources that SHOULD be used in preference. The only Resource whose @*Class* = "PlaceHolder" is called ***PlaceHolderResource***.

Like ***ImplementationResource***s, ***PlaceHolder Resource***s contain no Attributes besides those contained in the Abstract Resource Element.

N.1.2 ResourceUpdate

New in JDF 1.1,

Deprecated in JDF 1.3

ResourceUpdate Elements are an Abstract Element class that optionally contains any of the Attributes and Elements valid for the **Resource** that they reside in. The naming convention for ***ResourceUpdate*** Elements is to add the suffix ***"Update"*** to the Resource name. REQUIRED Attributes and Elements of Resources are optional in the respective ***ResourceUpdate***. In addition, a ***ResourceUpdate*** defined within a **Resource** SHALL contain a unique @*UpdateID* of type NMTOKEN. Only Devices that process the Resource as input can reference the @*UpdateID* of a ***ResourceUpdate***. Such references to ***ResourceUpdate*** Elements SHALL update the current state of the Device.

When a ***ResourceUpdate*** is referenced from a Device (e.g., from a PPML TicketRef element ▸ [PPML]), said Device will update ONLY those Elements that are explicitly specified within the ***ResourceUpdate***. No Attributes are inherited from the **Resource** that contains the ***ResourceUpdate***.

ResourceUpdate Elements are useful for Process Input Resources only and SHALL NOT be applied to Intent Resources.

Functionality similar to that of ***ResourceUpdate*** is provided by Partitioned Resources and it is RECOMMENDED to reference Partitions instead of ***ResourceUpdate*** Elements.

Table N.1: Contents of the Abstract ResourceUpdate Element

NAME	DATA TYPE	DESCRIPTION
UpdateID New in JDF 1.1 Deprecated in JDF 1.3	NMTOKEN	Unique ID that identifies the ***ResourceUpdate***. Note that only one ***Resource***, ***Resource*** Partition, or ***ResourceUpdate*** with a given value of @*UpdateID* may occur per **JDF** document, even though the scope of the ***ResourceUpdate*** is local to the Resource that it is defined in.

N.1.3 StatusPool

Deprecated in JDF 1.3.

In **JDF** 1.3 and beyond, ***StatusPool*** has been replaced by a Partitioned ***NodeInfo*** Resource.

The ***StatusPool*** describes the @*Status* of a **JDF** Node that processes Partitioned Resources. ***StatusPool*** Elements are only valid if the Node's @*Status* = "Pool", otherwise the Node's @*Status* is valid for all parts, regardless of the contents of ***StatusPool***. It MAY contain ***PartStatus*** Elements that define the Node's status with respect to specific Partitions. It is an error to define ***PartStatus*** Elements that reference identical or overlapping parts within one ***StatusPool***. Partitioned Resources are described in ‣ Section 3.10.5 Description of Partitioned Resources.

Table N.2: Contents of the StatusPool Element

NAME	DATA TYPE	DESCRIPTION
Status ?	enumeration	Identifies the @*Status* of the Node when ***JDF***/@*Status* = "Pool". Individual ***PartStatus*** Elements MAY override this value for the Partitions they represent. @*Status* applies to all Partitions of the Node except where it is overridden by ***PartStatus***/@*Status*. **Allowed values are from:** ***JDF***/@*Status* (except "Pool")
StatusDetails ? New in JDF 1.2	string	Identifies the @*StatusDetails* of the Node when ***JDF***/@*Status* = "Pool". Individual ***PartStatus*** Elements MAY override this value for the Partitions they represent. @*StatusDetails* applies to all Partitions of the Node except where it is overridden by ***PartStatus***/@*StatusDetails*. **Values include those from:** ‣ Section C.11 StatusDetails Supported Strings.
PartStatus *	element	Element that defines the Node's status for a set of parts.

N.1.3.1 PartStatus

The following table describes the ***PartStatus*** Element.

Table N.3: Contents of the PartStatus Element

NAME	DATA TYPE	DESCRIPTION
Status ?	enumeration	Identifies the status of an individual part of the Node. If not specified, defaults to ***StatusPool/***@*Status*. In **JDF** 1.3 and beyond, @*Status* has been replaced by ***NodeInfo***/@*NodeStatus*. **Allowed values are from:** ***JDF***/@*Status*.
StatusDetails ? New in JDF 1.2	string	Description of the status that provides details beyond the enumerative values given by the @*Status* Attribute. If not specified, defaults to ***StatusPool/***@*StatusDetails*. In **JDF** 1.3 and beyond, @*StatusDetails* has been replaced by ***NodeInfo***/@*NodeStatus*. **Values include those from:** ‣ Section C.11 StatusDetails Supported Strings.
Part Modified in JDF 1.2	element	Specifies the selected part that the ***PartStatus*** is valid for. This SHALL be a leaf or intermediate Partition of the Node's Output Resource. Thus, if the Node's Output Resource is Partitioned by "Side" and "Separation", the ***Part*** may contain either "Side" only or "Side" and "Separation", but not "Separation" only. See ‣ Table 3.26 Part Element for details of the ***Part*** Element. For details on Partitioned Resources, see ‣ Section 3.10.5 Description of Partitioned Resources. **Note:** The cardinality of ***Part*** has been changed from * to none (i.e., exactly one Element) in version 1.1 of the **JDF** specification

N.2 Lot

In the case of identifying resources that are planned to be consumed, ***Lot*** elements for each unique resource lot are placed in the associated ***ResourceLink*** or a ***PartAmount*** element, See ‣ Section 3.9.2 ResourceLink and ‣ Section 3.9.3.2.2 PartAmount.

Table N.4: Lot Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ActualAmount ?	double	Total amount of the resource that has been consumed from this resource lot. The sum of all values of @*ActualAmount* for all ***Lot*** elements SHOULD equal the @*ActualAmount* specified in the parent ***ResourceLink*** of the ***Lot*** elements.

Table N.4: Lot Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Amount ?	double	Total amount of the resource that is planned to be consumed from this resource lot. The sum of all values of @*Amount* for all ***Lot*** elements SHOULD equal the @*Amount* specified in the parent ***ResourceLink*** of the ***Lot*** elements.
LotID	string	Unique identifier related to this resource lot. The identifier SHALL be unique within the scope of all resource lots for the related @*ProductID*. An MIS that uses multiple identifiers to identify a resource lot SHALL assign a single unique ID to each lot, and SHALL map this single unique ID to the appropriate set of multiple identifiers.
Consumption ?	enumeration	Used for indicating level of consumption for the ***Lot***. This attribute SHALL NOT be specified for resources that are produced. It MAY only be specified for resources that are partially or fully consumed. This information is used by readers for auditing ***Consumable Resource***s to identify shortages and overages. For example, a roll of paper that was supposed to have 10,000 feet on it may be marked as fully consumed, yet only 9,400 feet of paper were consumed. **Allowed values are:** Full Partial

In the case of identifying resources after they have been consumed, ***Lot*** elements are specified within the first ***ResourceLink*** in the ***ResourceAudit***, or in the ***AmountPool*** that can appear inside the ***ResourceLink***. See ▸ Section 3.11.4.8 ResourceAudit for the structure of the ***ResourceAudit*** element.

Example N.1: MediaLink with Lot

The following is an example of a ***ResourceLink*** used to report that a substitute resource was used:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" Type="ConventionalPrinting"
     Status="Completed" ID="ID100" JobPartID="ID345" Version="1.4">
   <ResourcePool>
      <Media ID="RI007" Class="Consumable" ProductID="3002" Status="Unavailable"
             Brand="Coated Roll Stock" Dimension="2520 8640000"
             MediaType="Paper" Thickness="36"/>
      <ConventionalPrintingParams ID="RI008" Class="Parameter" Status="Available"/>
      <Component ID="RI009" Class="Quantity" Status="Unavailable"
             ComponentType="Sheet"/>
   </ResourcePool>
   <ResourceLinkPool>
      <MediaLink rRef="RI007" Amount="9800" ActualAmount="9703" Usage="Input">
         <Lot ActualAmount="5250" Consumption="Full"
              LotID="LN1040788312RN2005091-04"/>
         <Lot ActualAmount="4453" Consumption="Partial"
              LotID="LN1040788339RN2005091-01"/>
      </MediaLink>
      <ConventionalPrintingParamsLink rRef="RI008" Usage="Input"/>
      <ComponentLink rRef="RI009" Usage="Output"/>
   </ResourceLinkPool>
</JDF>
```

N.3 Life Cycle of JDF

N.3.1 Case 5: Spawning and Merging of Independent Jobs

Deprecated in JDF 1.5

Compatibility Warning. Note that Spawning and Merging of Independent Jobs is under development and subject to major changes in a future release

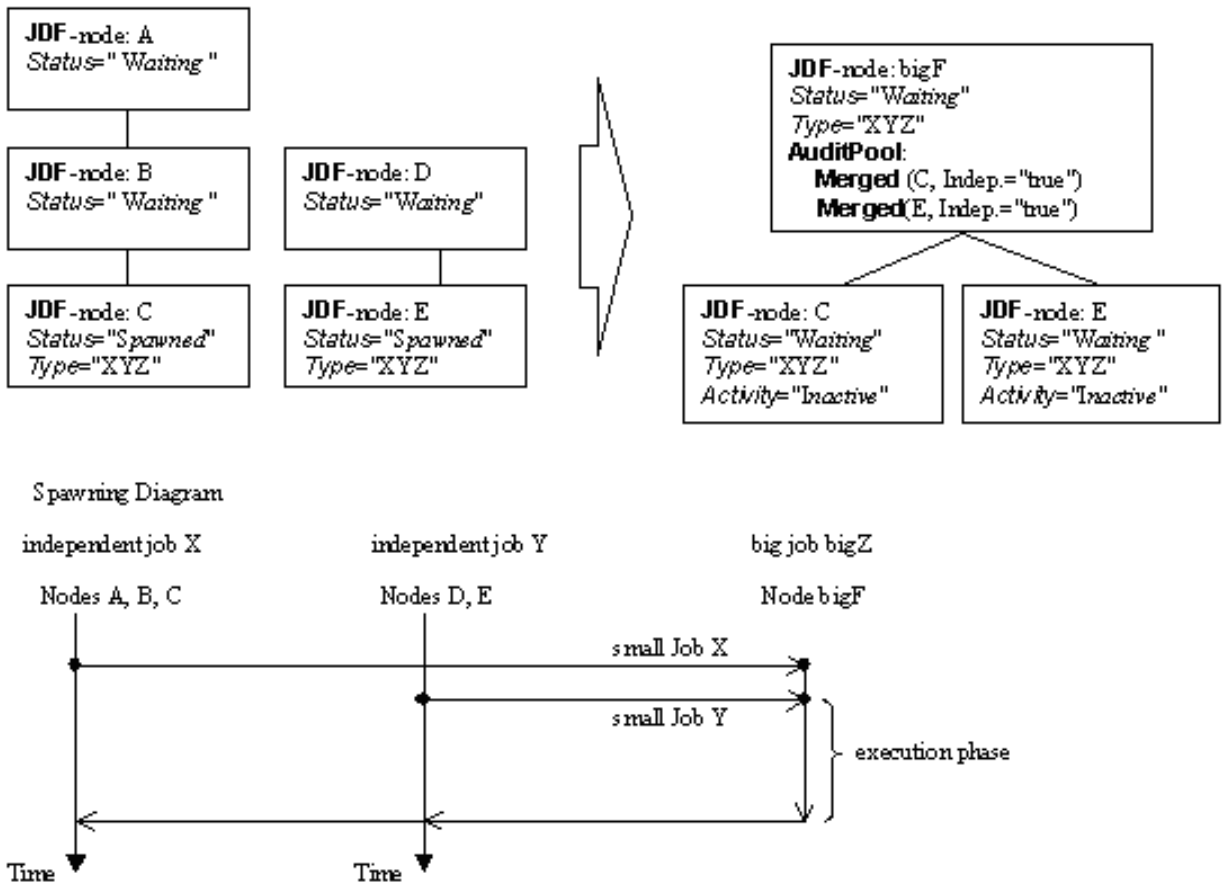
It is useful to spawn and merge independent Jobs in situations where the execution of separate, independent Small Jobs is not efficient in a commercial sense. Business cards for individual customers that are printed on one set of Sheets and

subsequently cut are an example of this kind of situation. In cases such as these, Small Jobs can be collected in order to form a Big Job that can then be executed as a whole. This allows Job aspects such as production, equipment load, and balancing of ***ImplementationResource***s to be performed more efficiently.

Note that production Devices will generally require their Resources to unambiguously define the production details. Thus a **JDF** Agent SHALL prepare the Resources in a way that the exact positioning of the contents of individual Small Jobs is specified. It is therefore RECOMMENDED to use the procedure that is described in this section for Product Intent Nodes only.

In this example, diagrammed in ▸ Figure N-1: Example of the spawning and merging of independent Jobs, Nodes C and E represent Small Jobs of identical type. Node bigF represents a Big Job, which might exist already or which might have been created for the purposes of this spawning-and-merging Process. Once Nodes C and E are gathered beneath Node bigF, as described below, a Big Job can then be executed as a whole for the sake of efficiency. When the Big Job is executed, the Small Jobs are effectively executed simultaneously. Nodes A, B and D are provided to demonstrate that spawned Nodes in this example might be related to other Nodes in various ways.

Figure N-1: Example of the spawning and merging of independent Jobs



Spawning

Spawning begins as it did in Case 1 or Case 2. Then, the Process to be spawned (Job C in ▸ Figure N-1: Example of the spawning and merging of independent Jobs) is copied into a newly created or already existing Big Job (Big Job bigZ in ▸ Figure N-1: Example of the spawning and merging of independent Jobs). The Process type of the Root Node of the Big Job SHALL be identical to that of the spawned Processes. The *@Activation* state of the spawned Processes is set to "Inactive", and an ***AncestorPool*** Element is added to the inactive spawned **JDF** Node to define the ancestry (as was described above). A ***Merged*** Element containing information about the spawned independent Jobs and when they have been received is added to the Big Job.

In the original Jobs, the *@Status* of the Process is designated as "Spawned", and a ***Spawned*** Element with the OPTIONAL *@jRefDestination* Attribute specified is added to the parent of the original Job. The *@jRefDestination* Attribute contains the ID of the Big Job beneath which the spawned Process has been placed. The changes in the parent are the equivalent of those described in Case 1 except for the specification of the *@jRefDestination* Attribute in the ***Spawned*** Element.

Where necessary, Resource instances SHALL be copied and logged as in Case 2 by appending the IDs to the appropriate Attribute (*@rRefsROCopied* or *@rRefsRWCopied*) of the ***Spawned*** Element in the parent of the original Job. This is REQUIRED in single spawning and merging. Furthermore, the ***ResourceLink*** Elements of the spawned Process SHALL be copied to the ***ResourceLinkPool*** of the active, big Process Node. In this way, the Input Resources and the Resources to be produced are linked to the Big Job.

Merging

For each of the spawned Small Jobs, the return procedure is performed as it was in the preceding cases. Once the Process explained in Case 1 is performed, the completed Job is copied back to its original location and the @*Activation* Attribute is restored by setting it to the activation of the Big Job Node after completion.

Eventually, copied Resources SHALL be purged and handled just as they were in Case 2. Then, the merging SHALL be logged by appending the ***Merged*** Element to the ***AuditPool*** container of the parent of the original Node. In independent spawning and merging, the @*jRefSource* Attribute SHALL be specified in the appropriate ***Merged*** Element.

If the Big Job is retained, a ***Spawned*** Element with the Attribute @*Independent* = "true" SHALL be appended to the ***AuditPool*** of the Big Job. For instance, saving the finished Big Job might be desirable if the audit information contained in the Big Job SHOULD be available for individual invoicing. Finally, the newly created big **JDF** node SHOULD be deleted to avoid the double existence of Nodes.

N.4 JMF Messaging Elements

N.4.1 Signal

N.4.1.1 Trigger

The following 3 Elements were deprecated from ***Signal/Trigger*** in **JDF** 1.2.

N.4.1.2 Added

Deprecated in JDF 1.2

Table N.5: Added Element

NAME	DATA TYPE	DESCRIPTION
AddedElement *	element	If the appending of an Element like a service, Controller, Device or Message triggered this signal, this Element describes which service, Controller, Device, Message, etc. has been added. This is an Abstract Element. It is a placeholder for a ***ResponseTypeObj*** like ***NotificationDef***, a ***JDFController***, a ***Device***, a ***JDFService*** or a ***MessageService***. For details on these Elements see ▸ Section 5.8 Messages for Events and Capabilities.

N.4.1.3 ChangedAttribute

Deprecated in JDF 1.2

Table N.6: ChangedAttribute Element

NAME	DATA TYPE	DESCRIPTION
AttributeName	NMTOKEN	Name of the Attribute that changed.
ElementID ?	NMTOKEN	ID of the Element that changed. Used only in conjunction with a change of a certain Resource or Node which cannot uniquely be addressed by the other Attributes of this Element.
ElementType	NMTOKEN	Name of the Element which contains the changed Attribute.
OldValue	string	Old value. The string SHALL be cast to the appropriate data type that depends on the Attribute's data type.
NewValue	string	New value of the Attribute.

N.4.1.4 Removed

Deprecated in JDF 1.2

Table N.7: Removed Element

NAME	DATA TYPE	DESCRIPTION
RemovedElement *	element	If the removal of an Element like a service, Controller, Device or Message triggered this signal, this Element describes the service, Controller, Device, Message, etc. that has been removed. This is an Abstract Element. It is a placeholder for a ***ResponseTypeObj*** like ***NotificationDef***, a ***JDFController***, a ***Device***, a ***JDFService*** or a ***MessageService***. For details on these Elements see ▸ Section 5.8 Messages for Events and Capabilities.

N.4.2 Events

Deprecated in JDF 1.5

The ***Events*** Message type is intended to be used to query for supported ***Signal*** Messages and to subscribe for asynchronous, randomly occurring ***Signal*** Messages of a Device or Controller. These events are described in ▸ Section 4.6.1 Classification of Notifications and can only be transmitted via ***Signal*** Messages. If the ***Query*** Message contains a ***Subscription*** Element, a ***NotificationFilter*** Element is combined by a logical AND operation with the ***Subscription*** Element for selective subscriptions. An empty ***Events*** Message (without a ***Subscription*** and ***NotificationFilter*** Element) can be used to query for all events as described in ▸ Section 5.34 Notification, which are supported by a Device or Controller. If all signals are requested, a ***NotificationFilter*** with @*SignalTypes* = "All" SHALL be included in the ***Query*** Message.

The Controller that subscribes for ***Events*** Messages receives ***Signal*** Messages. In **JDF** 1.2, the ***Events*** Message was enhanced to subscribe for all types of ***Signal*** Messages, not only ***Notification*** Signals. The event type and values of ***Notification*** Messages are provided by specifying a @*Type* Attribute and an ***Abstract NotificationDetails*** Element in the ***Notification*** Element, as described in ▸ Section 3.11.4.5 Notification. Possible ***NotificationDetails*** Elements are defined in ▸ Section C.8 NotificationDetails.

Table N.8: Events Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***NotificationFilter*** ?	Refines the list of events queried.
ResponseTypeObj	***NotificationDef*** *	List of Notification types that match ***NotificationFilter***.

Example N.2: Query with Subscription to All Events

Example of a ***Subscription*** of all ***Events*** and the ***Response*** Message, including the feature of subscribing for all Messages by setting ***NotificationFilter***/@*SignalTypes* = "All":

```
<Query ID="M170" Type="Events" xsi:type="QueryEvents">
<Subscription URL="http://www.anycompany.com/MIS/JMF/JobTracker"/>
<NotificationFilter Classes="Event Warning Error Fatal" SignalTypes="All"/>
</Query>
```

Example N.3: Response for Subscription to All Events

The ***Response*** Message to the previous ***Query*** Message:

```
<Response ID="M1001" refID="M170" Type="Events" xsi:type="ResponseEvents"
xmlns:anycompany="http://www.anycompany.com">
<NotificationDef Classes="Warning Error Fatal" Type="Error"/>
<NotificationDef Classes="Event" Type="FCNKey"/>
<NotificationDef Classes="Event Error" Type="Barcode"/>
<NotificationDef Classes="Event" Type="SystemTimeSet"/>
<NotificationDef Classes="Event" Type="anycompany:PrivateEvent_1"/>
<NotificationDef Classes="Event" Type="anycompany:PrivateEvent_2"/>
<NotificationDef Classes="Event" Type="anycompany:PrivateEvent_2"/>
<NotificationDef SignalType="Status"/>
<NotificationDef SignalType="Resource"/>
</Response>
```

N.4.2.1 NotificationDef

Table N.9: NotificationDef Element

NAME	DATA TYPE	DESCRIPTION
Classes ? Modified in JDF 1.2	enumerations	***Notification***/@*Class* of the Notification in a ***Signal***. **Allowed values are:** Event Information Warning Error Fatal **Constraint:** @*Classes* SHALL NOT be specified unless @*SignalType* = "Notification". For details, see ▸ Section 4.6.1 Classification of Notifications.
SignalType = "Notification" New in JDF 1.2	NMTOKEN	***Signal***/@*Type* value of the subscribed Message. **Values include those from:** ***Message***/@*Type*. **Note:** The values are limited to Signal Messages
Type ? Modified in JDF 1.2	NMTOKEN	Notification type, that is the name of the Element derived from the ***Abstract NotificationDetails*** Element. **Constraint:** @*Type* SHALL NOT be specified unless @*SignalType* = "Notification". **Values include those from:** ▸ Table C.14 List of NotificationDetails Elements.

N.4.3 KnownControllers

Deprecated in JDF 1.5

The ***KnownControllers*** Query Message requests information about the Controllers and/or Devices that are known to the Controller that is queried and can be directly accessed by **JMF** messaging. ***KnownControllers*** is intended to be used with a "registration" server. A processor that needs information about its system environment can query a registration server for a list of known Controllers and/or Devices. A single Controller or Device that supports multiple URLs or protocols is defined using multiple ***JDFController*** Elements with the same @*ControllerID* Attribute. This list can subsequently be iterated using the other Process registration queries in this section. The URL of the master registration server SHALL be defined using a method outside of **JDF**.

Table N.10: KnownControllers Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***ControllerFilter*** New in JDF 1.4	Allows filtering the Response.
ResponseTypeObj	***JDFController*** *	Known Controllers.

N.4.3.1 ControllerFilter

New in JDF 1.4

Table N.11: ControllerFilter Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ControllerID ?	string	Only Controllers whose @*ControllerID* or Devices whose @*DeviceID* matches this @*ControllerID* should be returned in the Response. If this Attribute is not specified, the Response should contain ***JDFController*** Elements for all known Controllers and/or Devices.

Table N.11: ControllerFilter Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
URLTypes ?	enumerations	Only URL's whose ***JDFController***/@*URLType* Attribute in the Response match one of the values in this @*URLTypes* Attribute should be returned in the Response. If this Attribute is not specified, the Response should contain ***JDFController*** Elements with URLs of any type. **Allowed values are from:** ***JDFController***/@*URLType*.

N.4.3.2 JDFController

Table N.12: JDFController Element

NAME	DATA TYPE	DESCRIPTION
ControllerID ? New in JDF 1.2	string	String that identifies the Controller or Device. The @*ControllerID* is used as the @*SenderID* of **JMF** Messages that are produced by this Controller or Device. A **JMF** Message that is intended for a specific Controller SHOULD specify the Controller's @*ControllerID* value or the Device's @*DeviceID* in ***JMF***/@*DeviceID*.
URL	URL	URL of the Controller or Device. If the URL scheme is "file:", @*URL* SHALL specify the directory where the **JMF** Messages are to be deposited.
URLType ? New in JDF 1.4	enumeration	Identifies the purpose of this URL. **Allowed values are:** JDFError JDFInput JDFOutput JMF SecureJMF

Example N.4: KnownControllers Query

```
<Query ID="Q1" Type="KnownControllers" SenderID="MIS"
xsi:type="QueryKnownControllers">
<ControllerFilter ControllerID="PrintController1" URLTypes="JMF SecureJMF"/>
</Query>
```

Example N.5: KnownControllers Response

```
<Response ID="M1" Type="KnownControllers" xsi:type="ResponseKnownControllers"
refID="Q1" SenderID="RegistrationServer">
<JDFController ControllerID="PrintController1"
DescriptiveName="Printer Controller"
URL="http://www.anycompany.com/controller"
URLType="JMF"/>
<JDFController ControllerID="PrintController1"
DescriptiveName="Printer Controller"
URL="https://www.anycompany.com/controller/secure"
URLType="SecureJMF"/>
</Response>
```

N.4.4 RepeatMessages

Deprecated in JDF 1.5

The ***RepeatMessages*** Query Message returns a list of Messages that have been previously sent by the Controller. The OPTIONAL ***MsgFilter*** Element allows the list to be filtered. The list of **JMF** Messages that fulfill the filter criteria can be sorted by time, with the most recent listed first. This specification places no requirements on the size of the Message buffer of a Controller that supports ***RepeatMessages***.

Table N.13: RepeatMessages Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***MsgFilter*** ?	A filter for the Messages to be repeated. For details, see ▸ Section N.4.2 Events.
ResponseTypeObj	***Message*** *	The recent Messages queried.

N.4.4.1 MsgFilter

If the returned list is incomplete because the parameters supplied in the ***MsgFilter*** Element cannot be fulfilled by the application, the @*ReturnCode* is either 108 (empty list) or 109 (incomplete list) and SHOULD be flagged as a warning with ***Notification***[@*Class* = "Warning" and @*Type* = "Error"].

Table N.14: MsgFilter Element

NAME	DATA TYPE	DESCRIPTION
After ?	dateTime	Messages sent only after a certain time.
Before ?	dateTime	Messages sent only before a certain time.
Count ?	integer	Maximum number of Messages, most recent first.
DeviceID ?	string	ID of the Device whose Messages are requested.
Family ? Modified in JDF 1.3	enumeration	Filter for Message Family. **Allowed values are:** Acknowledge Command – Repeat ***Command*** Messages that are triggered by a ***Registration*** Message. New in JDF 1.3 Response Signal All – ***Response***, ***Signal*** and ***Acknowledge*** Messages are queried. Deprecated in JDF 1.2.
JobID ? New in JDF 1.2	string	@*JobID* of the Job whose Messages are queried/subscribed.
JobPartID ? New in JDF 1.2	string	@*JobPartID* of the Job whose Messages are queried/subscribed.
MessageRefID ?	NMTOKEN	The @*refID* Attribute SHALL match the value of @*MessageRefID*.
MessageID ?	NMTOKEN	The @*ID* Attribute SHALL match the value of @*MessageID*.
MessageType ?	NMTOKEN	@*Type* Attribute of the requested Messages. **Values include those from:** ***Message***/@*Type*.
QueueEntryID ? New in JDF 1.2	string	@*QueueEntryID* of the Job whose Messages are queried/subscribed. If @*QueueEntryID* is specified, @*JobID*, @*JobPartID* and ***Part*** are ignored. If none of @*JobID*, @*JobPartID*, ***Part*** or @*QueueEntryID* are specified, ***MsgFilter*** applies to all Jobs that will be processed by the receiver.
ReceiverURL ?	URL	URL for which the Messages are intended.
Part * New in JDF 1.2	element	***Part*** of the Job whose Messages are queried/subscribed. For details on Node Partitions, see ▸ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources.

Example N.6: RepeatMessages Response

The following is an example of a ***Response*** Message to a ***RepeatMessages*** Query Message. Note the nesting of ***Response*** Messages, where the first layer is the response to the ***RepeatMessages*** Query Message and its contents are the repeated Messages.

```
<Response ID="RepMsg" Type="RepeatMessages" xsi:type="ResponseRepeatMessages">
```

```
<Response ID="R1" Time="2000-06-14T11:00:01+02:00" Type="Status"
xsi:type="ResponseStatus"/>
<Response ID="R2" Time="2000-06-14T10:50:22+02:00" Type="Occupation"
xsi:type="ResponseOccupation"/>
<Signal ID="R3" Time="2000-06-14T08:20:23+02:00" Type="Resource"
xsi:type="SignalResource"/>
<Signal ID="R4" Time="2000-06-14T03:01:22+02:00" Type="Notification"
xsi:type="SignalNotification"/>
</Response>
```

N.4.5 NodeInfo

New in JDF 1.2

Deprecated in JDF 1.3

The ***NodeInfo*** Message can be used as a Command Message or a Query Message to modify or to query **JDF** ***NodeInfo*** Elements. The Query Message simply retrieves information about the ***NodeInfo*** without modifying it, while the command modifies those settings within the ***NodeInfo*** that are specified. Settings that are not specified remain unchanged.

N.4.5.1 NodeInfo Query

The ***NodeInfo*** Query Message is made selective by specifying a ***NodeInfoQuParams*** Element. The query's Response Message returns a list of ***NodeInfoResp*** Elements that contains the queried information concerning the ***NodeInfo*** Elements. If the list is empty because the selective query parameters of the ***NodeInfoQuParams*** lead to a null selection, then the @*ReturnCode* is 103 (@*JobID* unknown), 104 (@*JobPartID* unknown) or 108 (empty list) and SHOULD be flagged as a warning with ***Notification*** [@*Class* = "Warning" and @*Type* ="Error"].

Table N.15: NodeInfo Query Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***NodeInfoQuParams***	Specifies the Node queried.
ResponseTypeObj	***NodeInfoResp*** *	Details of the ***NodeInfo*** Elements

N.4.5.1.1 NodeInfoQuParams

Table N.16: NodeInfoQuParams Element

NAME	DATA TYPE	DESCRIPTION
JobID	string	Job ID of the **JDF** Node that is being queried.
JobPartID ?	string	Job Part ID of the **JDF** Node that is being queried.
QueueEntryID ?	string	@*QueueEntryID* of the Job that is currently being executed. If @*QueueEntryID* is specified, @*JobID*, @*JobPartID*, and ***Part*** are ignored. If none of @*JobID*, @*JobPartID*, ***Part***, or @*QueueEntryID* are specified, ***ResourceQuParams*** applies to all Jobs.
Part *	element	***Part*** Elements that describe the Partition of the Job whose ***NodeInfo*** is modified. For details on Node Partitions, see ▸ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources.

N.4.5.2 NodeInfo Command

The ***NodeInfo*** Command Message is used to modify the ***NodeInfo*** – generally scheduling information – of a submitted **JDF** Node. It is made selective by specifying the OPTIONAL Attributes in the ***NodeInfoCmdParams*** Element.

The Response Message contains a list of ***NodeInfoResp*** Elements with a copy of ***NodeInfo*** after the changes have been applied. If the ***NodeInfo*** Command Message was successful, the value of the @*ReturnCode* Attribute is "0". If it is not successful, the value of @*ReturnCode* might be one of those described in the above section about the ***NodeInfo*** Query Message; it might also be "200" (invalid parameters) or "201" (insufficient parameters). Partial application of the ***NodeInfo*** SHOULD

also be flagged as a warning. If the value of @*ReturnCode* is larger than "0", the Controller that issued the command can evaluate the returned ***NodeInfo*** in order to find the setting that could not be applied.

Table N.17: NodeInfo Command Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
CommandTypeObj	***NodeInfoCmdParams***	Specifies the ***NodeInfo*** Elements to be modified.
ResponseTypeObj	***NodeInfoResp*** *	Contains information about the ***NodeInfo*** and the ***NodeInfo*** after modification.

N.4.5.2.1 NodeInfoCmdParams

Table N.18: NodeInfoCmdParams Element

NAME	DATA TYPE	DESCRIPTION
JobID	string	Job ID of the **JDF** Node that is being modified.
JobPartID ?	string	Job Part ID of the **JDF** Node that is being modified.
QueueEntryID ?	string	@*QueueEntryID* of the Job that is currently being executed. If @*QueueEntryID* is specified, @*JobID*, @*JobPartID*, and ***Part*** are ignored. If none of @*JobID*, @*JobPartID*, ***Part***, or @*QueueEntryID* are specified, ***NodeInfoCmdParams*** applies to all Jobs.
UpdateMethod = "Complete"	enumeration	Method how ***NodeInfo*** is applied to the **JDF**. **Allowed values are:** Complete – The ***NodeInfo*** in the JDF is completely overwritten by ***NodeInfo*** in this Message. Incremental – The ***NodeInfo*** in the JDF is incrementally updated by the values that are explicitly set in ***NodeInfo*** in this Message.
Part *	element	***Part*** Elements that describe the Partition of the Job whose ***NodeInfo*** is modified. For details on Node Partitions, see ▸ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources.
NodeInfo ?	element	***NodeInfo*** to be uploaded to the Device.

N.4.5.2.2 NodeInfoResp

Table N.19: NodeInfoResp Element

NAME	DATA TYPE	DESCRIPTION
JobID	string	Job ID of the **JDF** Node that is being modified.
JobPartID ?	string	Job Part ID of the **JDF** Node that is being modified.
QueueEntryID ?	string	@*QueueEntryID* of the Job that is currently being executed. If @*QueueEntryID* is specified, @*JobID*, @*JobPartID*, and ***Part*** are ignored. If none of @*JobID*, @*JobPartID*, ***Part***, or @*QueueEntryID* are specified, ***NodeInfoResp*** applies to all Jobs.
Part *	element	***Part*** Elements that describe the Partition of the Job ***NodeInfo*** is modified. For details on Node Partitions, see ▸ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources.
NodeInfo ?	element	***NodeInfo*** after uploading to the Controller.

The following is an example for retrieving ***NodeInfo*** settings:

```
<Query ID="Q1" Type="NodeInfo">
<NodeInfoQuParams JobID="J1"/>
```

```
</Query>
```

The following is a possible Response Message to the Query Message above:

```
<Response ID="M1" Type="NodeInfo" refID="Q1">
<NodeInfoResp JobID="J1" JobPartID="P1">
<NodeInfo/>
</NodeInfoResp>
<NodeInfoResp JobID="J1" JobPartID="P2">
<NodeInfo/>
</NodeInfoResp>
</Response>
```

N.4.6 KnownJDFServices

Deprecated in JDF 1.2

In **JDF** 1.2 and beyond, ***KnownJDFServices*** has been replaced with ***KnownDevices*** and @*DeviceDetails* = "Capabilities".

Table N.20: KnownJDFServices Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	—	—
ResponseTypeObj	***JDFService*** *	Processes that the Controller or Device can execute.

The ***KnownJDFServices*** Query Message returns a list of services that are defined in the **JDF** specification, such as **ConventionalPrinting**, **RIPing**, or **EndSheetGluing**. It allows a Controller to publish the services that the Devices it controls are capable of providing. The response is a list of ***JDFService*** Elements, one for each supported Process type.

N.4.6.1 JDFService

JDFService Elements define the Node types that can be processed by the Controller. A **JDF** processor SHOULD be capable of processing Combined Process Nodes of any of the individual ***JDFService*** Elements that are specified. It is therefore not necessary to define every permutation of allowed combinations. It need not be able to process individual Nodes with a type defined in the @*Types* Attribute of a "Combined" ***JDFService*** Element.

Table N.21: JDFService Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
CombinedMethod ? New in JDF 1.1	enumeration	Specifies how the Processes specified in @*Types* may be specified. **Allowed values are:** Combined – The list of Processes in @*Types* SHALL be specified as a Combined Process. ProcessGroup – The list of Processes in @*Types* SHALL be specified as a "ProcessGroup" of individual Processes. CombinedProcessGroup – The list of Processes in @*Types* may be specified either as a Combined Process or as a "ProcessGroup" of individual Processes. None – No support for "Combined" or "ProcessGroup". Only the individual Process type defined in @*Types* is supported. The default.
Type	NMTOKEN	**JDF** @*Type* Attribute of the supported Process. Extension types may be specified by stating the namespace in the value. **Values include those from:** ***JDF***/@*Type*
TypeOrder ? New in JDF 1.1	enumeration	Ordering restriction for Combined Process Nodes. **Allowed values are:** Fixed – The order of Process types specified in the @*Types* Attribute is ordered and each type can be specified only once (e.g., for Cutting and Folding, order does matter). The default. Unordered – The order of Process types specified in the @*Types* Attribute is unordered and each type can be specified only once (e.g., for **DigitalPrinting**, **Screening** and **Trapping**, order does not matter). Unrestricted – The order of Process types specified in the @*Types* Attribute is unordered and each type can be specified multiply (e.g., Cutting, Folding, where the Device can do both Processes, in any order and multiple times).

Table N.21: JDFService Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Types ?	NMTOKENS	If @*Type* = "Combined", or @*Type* = "ProcessGroup" this Attribute represents the list of Combined Processes. If any of the services are in a namespace other than **JDF**, the namespace prefix SHOULD be included in this list. For details, see ▸ Section 3.3.3 Combined Process Nodes. **Values include those from:** *JDF*/@*Types*

The following is an example of a Response Message to a ***KnownJDFServices*** Query Message:

```
<Response ID="M1" refID="Q1" Type="KnownJDFServices">
<JDFService Type="Rendering" />
<JDFService Type="Folding" />
<JDFService Type="Combined" Types="Gathering Stitching"/>
<JDFService Type="AnyCompaniesNamespace:MyFolding" />
…
</Response>
```

N.4.7 Occupation

Deprecated in JDF 1.5

Occupation queries the occupation status of an employee. No Job context is needed to issue an ***Occupation*** Message.

Table N.22: Occupation Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***EmployeeDef*** *	Defines the employees queried.
ResponseTypeObj	***Occupation*** *	The occupation status of the employees.

N.4.7.1 EmployeeDef

The ***Occupation*** Query Message might be focused to certain employees specifying a ***EmployeeDef*** Element. If no ***EmployeeDef*** Element is specified, a list of all known employees is returned.

Table N.23: EmployeeDef Element

NAME	DATA TYPE	DESCRIPTION
PersonalID ?	string	@*PersonalID* of the employee being tracked.

N.4.7.2 Occupation

The response returns a list of ***Occupation*** Elements for the queried employees. These Elements consist of one entry for every Job that is currently being executed. The list format accommodates both employees that service multiple Jobs or Job Parts in parallel and multiple employees working on one Job.

Table N.24: Occupation Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Busy = "100"	double	Busy state of the employee in percentage. A value of 100 means that the employee is fully occupied with this task. The sum of all @*Busy* values of one employee SHOULD NOT exceed 100.
JobID ?	string	@*JobID* of the **JDF** node that the employee is assigned to. If no @*JobID* is specified but Devices are, the employee is performing tasks not related to a Job.
JobPartID ?	string	Job Part ID of the **JDF** node that is currently being executed.

Table N.24: Occupation Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
QueueEntryID ? New in JDF 1.2	string	@*QueueEntryID* of the Job that is currently being executed. If @*QueueEntryID* is specified, @*JobID*, @*JobPartID* and ***Part*** are ignored. If none of @*JobID*, @*JobPartID*, ***Part*** or @*QueueEntryID* are specified, ***Occupation*** applies to all Jobs.
Device *	element	Devices that the employee is currently assigned to. The data type of ***Device*** is ResourceElement. See ▸ Section 3.10.1 ResourceElement – Subelement of a Resource.
Employee	element	Description of the employee being tracked. The data type of ***Employee*** is ResourceElement. See ▸ Section 3.10.1 ResourceElement – Subelement of a Resource.
Part * New in JDF 1.2	element	***Part*** Elements that describe the Partition of the that is being executed. For details on Node Partitions, see ▸ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources

Example N.7: Occupation Response

The following is an example of Response Message to an ***Occupation*** Query Message:

```
<Response ID="M1" Type="Occupation" xsi:type="ResponseOccupation" refID="Q1">
<!--Two Jobs on one Device with one operator-->
<Occupation Busy="30" JobID="J1">
<Employee PersonalID="P1234"/>
<Device DeviceID="Press1"/>
</Occupation>
<Occupation Busy="70" JobID="J2">
<Employee PersonalID="P1234"/>
<Device DeviceID="Press1"/>
</Occupation>
<!--Another operator on Job j2 -->
<Occupation Busy="50" JobID="J2">
<Employee PersonalID="P4321"/>
<Device DeviceID="Press1"/>
</Occupation>
<!--No Job context -->
<Occupation Busy="0">
<Device DeviceID="Press2"/>
<Employee PersonalID="P5678"/>
</Occupation>
</Response>
```

N.4.8 Track

Deprecated in JDF 1.5

The ***Track*** Query Message requests information about the location of Jobs that are known by a Controller. If a high level Controller controls lower level Controllers, it SHOULD also list the Jobs that are controlled by these. The Response Message is a list of ***TrackResult*** Elements.

Table N.25: Track Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj	***TrackFilter*** ?	Refines the ***Track*** Query Message.
ResponseTypeObj	***TrackResult*** *	Details of the tracked Jobs.

N.4.8.1 TrackFilter

The ***TrackFilter*** Element refines the list of ***TrackResult*** Elements that are to be returned. Only Jobs that match all parameters specified are included.

Table N.26: TrackFilter Element

NAME	DATA TYPE	DESCRIPTION
JobID ?	string	@*JobID* of the **JDF** node that is being tracked. Defaults to list ***JobPhase*** Elements of all known Nodes.
JobPartID ?	string	@*JobPartID* of the **JDF** node that is being tracked.
ProjectID ? New in JDF 1.2	string	@*ProjectID* of the **JDF** node that is being tracked.
QueueEntryID ? New in JDF 1.2	string	@*QueueEntryID* of the Job that is currently being executed. If @*QueueEntryID* is specified, @*JobID*, @*JobPartID* and ***Part*** are ignored. If none of @*JobID*, @*JobPartID*, ***Part***, @*ProjectID* or @*QueueEntryID* are specified, ***TrackFilter*** applies to all Jobs.
Status ?	enumerations	The ***JDF***/@*Status* of the Jobs being tracked. The value of this attribute is a list of any combination of values. **Default value is:** all enumerations, if not known or specified. **Allowed values are from:** ***JDF***/@*Status* (▸ Table 3.4 JDF).
Part * New in JDF 1.2	element	***Part*** Elements that describe the Partition of the Job that is being tracked. For details on Node Partitions, see ▸ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources.

N.4.8.2 TrackResult

One ***TrackResult*** is returned for each known **JDF** or spawned **JDF** part. ***TrackResult*** Elements contain information about the location of distributed Jobs.

Table N.27: TrackResult Element

NAME	DATA TYPE	DESCRIPTION
JobID	string	@*JobID* of the **JDF** node that is being tracked.
JobPartID ?	string	@*JobPartID* of the highest level node of the **JDF** that is being tracked.
ProjectID ? New in JDF 1.2	string	@*ProjectID* of the highest level node of the **JDF** node that is being tracked.
QueueEntryID ? New in JDF 1.2	string	@*QueueEntryID* of the Job that is currently being tracked.
URL	URL	URL of the Controller that owns this Job.
IsDevice	boolean	If "true", the Controller that emitted this Message is the Device that has access to the Job and can be queried for details of the Job.
Part * New in JDF 1.2	element	***Part*** Elements that describe the Partition of the Job that is being tracked. For details on Node Partitions, see ▸ Section 4.3.2 Partial Processing of Nodes with Partitioned Resources.

Example N.8: Track Response

The following is an example of a ***Response*** Message on a ***Track*** Message:

```
<Response ID="M1" Type="Track" xsi:type="ResponseTrack" refID="Q1">
<TrackResult IsDevice="true" JobID="1" JobPartID="42"
URL="http://www.anycompany.com/controller"/>
</Response>
```

N.4.9 QueueEntryStatus

Deprecated in JDF 1.2

In **JDF** 1.2 and beyond, use ***QueueStatus*** with an appropriate ***QueueFilter*** instead of ***QueueEntryStatus***.

Table N.28: QueueEntryStatus Message

OBJECT TYPE	ELEMENT NAME	DESCRIPTION
QueryTypeObj Modified in JDF 1.1A	*QueueEntryDefList*	Defines the addressed queue entries. Note that this Element was *QueueEntryDef* * prior to **JDF** 1.1A.
ResponseTypeObj	*QueueEntry* *	Describes the status of the queried queue entries.

For the definition of the Elements above see ▸ Section 5.14 Elements for Queues.

The *QueueEntryStatus* Message returns queue entry descriptions. The *QueueEntryDef* Elements specify the queue entries to be queried. If no *QueueEntryDef* Element is specified, the query returns a list of *QueueEntry* Elements, one for each entry in the queue. If no *QueueEntryDef* is specified and the query defines a persistent channel, a *Signal* is emitted for any entry whose status changes. This includes changes as a result of modifications of the queue status, such as hold or resume.

N.4.9.1 QueueEntryDefList

New in JDF 1.1A

Deprecated in JDF 1.2

The *QueryTypeObj* of *QueueEntryStatus* has been modified from *QueueEntryDef* * to *QueueEntryDefList* because of a type collision in the XML Schema. *QueueEntryDef* had been used both as a *QueryTypeObj* and as a *CommandTypeObj*.

Table N.29: QueueEntryDefList Element

NAME	DATA TYPE	DESCRIPTION
QueueEntryDef *	element	Defines the addressed queue entries.

N.5 Deprecated Processes

N.5.1 DBDocTemplateLayout

Deprecated in JDF 1.5

This Process specifies the creation of a master document template that is used as an Input Resource for the *DBTemplateMerging* Process. It is similar to the Process except that the output is a set of document templates. Document template are represented in **JDF** as *LayoutElement* Resources with @*Template* = "true".

Table N.30: DBDocTemplateLayout – Input Resources

NAME	DESCRIPTION
LayoutElement *	Page elements without links to a database.
DBRules	Description of the rules that are to be applied to database records in order to generate graphic output.
DBSchema	Database schema that describe the structure of data in the database.

Table N.31: DBDocTemplateLayout – Output Resources

NAME	DESCRIPTION
LayoutElement *	The document template is a *LayoutElement* with links to a database. These links are proprietary to the linking application and are not described in **JDF**. The @*Template* Attribute SHALL be "true".

N.5.2 DBTemplateMerging

Deprecated in JDF 1.5

This Process specifies the creation of personalized PDL Instance Documents by combining a document template and instance data records from a database. The resulting Instance Documents will generally be consumed by an **Imposition**, a **RIPing** and ultimately, by a **DigitalPrinting** Process.

Table N.32: DBTemplateMerging – Input Resources

NAME	DESCRIPTION
DBMergeParams	Parameters of the merge Process.
DBSelection	Instance database records to be merged into the document.
LayoutElement *	Document template page element with internal links to a database.

Table N.33: DBTemplateMerging – Output Resources

NAME	DESCRIPTION
RunList	Page element without links to a database. This Element usually contains a printable ***LayoutElement*** Resource such as PPML, PDF or even plain ASCII.

N.5.3 FormatConversion

New in JDF 1.1

Deprecated in JDF 1.5

The **FormatConversion** Process controls the conversion from ***ByteMap*** to an external file raster format. The ***FormatConversionParams*** Resource defines the type and parameters to control the output file specified by the output ***RunList***.
Deprecation note: Starting with **JDF** 1.5, use a Combined Process of **RasterReading** and **Rendering**.

Table N.34: FormatConversion – Input Resources

NAME	DESCRIPTION
FormatConversionParams	Parameters that control the operation of the Process that produces the resulting image file pages.
RunList	List of ***ByteMap*** Resources to be converted to raster file format.

Table N.35: FormatConversion – Output Resources

NAME	DESCRIPTION
RunList	This Resource identifies the location of the resulting raster files. If the ***FileSpec***/@*MimeType* of this Resource is specified, then it SHALL match the input ***FormatConversionParams***/@*MimeType*. If ***FileSpec***/@*MimeType* is not specified, then ***FormatConversionParams***/@*MimeType* is used to update the Output Resource.

N.5.4 Ordering

Deprecated in JDF 1.5

This Process can be used to describe the **Ordering** (requisition) of a ***Resource*** Element. Orders can be placed internally (i.e., within the company or externally).

Table N.36: Ordering – Input Resources

NAME	DESCRIPTION
OrderingParams	Necessary information about the items to be ordered (e.g., the supplier address, item quantity or unit type).

Table N.37: Ordering – Output Resources

NAME	DESCRIPTION
Resource + Modified in JDF 1.1	All kinds of ***PhysicalResource***s can be ordered.

N.5.5 Packing

Deprecated in JDF 1.1

This Process can be used to describe the **Packing** of a ***PhysicalResource*** Element for transport purposes. The **Packing** Process has been deprecated in version 1.1 and beyond. It is replaced by the individual Processes defined in ▸ Section 6.6.5 Packaging Processes.

Table N.38: Packing – Input Resources

NAME	DESCRIPTION
PackingParams	Necessary information about the **Packing** Process.
PhysicalResource	All kinds of ***PhysicalResource***s can be packed.

Table N.39: Packing – Output Resources

NAME	DESCRIPTION
PhysicalResource	The packaged ***PhysicalResource***s. Note that @*Amount* Attributes referring to this Resource still refer to individual products and not to boxes, cartons or pallets.

N.5.6 FilmToPlateCopying

Deprecated in JDF 1.1

FilmToPlateCopying has been replaced by the more generic **ContactCopying**.

FilmToPlateCopying is the Process of making an analog copy of a film onto a printing plate.

Table N.40: FilmToPlateCopying – Input Resources

NAME	DESCRIPTION
DevelopingParams ?	Controls the physical and chemical specifics of the media development Process.
ExposedMedia	The film or films to be copied onto the plate.
Media	The unexposed plate.
PlateCopyParams	The settings of the exposure task.

Table N.41: FilmToPlateCopying – Output Resources

NAME	DESCRIPTION
ExposedMedia	The resulting exposed plate.

N.5.7 PreflightAnalysis

Deprecated in JDF 1.2

This Resource was deprecated as a result of a major revision to the **Preflight** Process and its associated Resources.

PreflightAnalysis Resources record the results of a **Preflight** Process. The semantics for results are specific to the ***FileType*** of the file. The Elements in this Resource, detailed in the table below, place the results in specific categories. The value for each of these Elements is an array of ***PreflightResultsDetail*** and ***PreflightInstance*** Subelements. Within the ***PreflightInstance*** Subelements, results are further broken down into ***PreflightInstanceDetails***.

Each ***PreflightResultsDetail and*** ***PreflightInstance*** Subelement in the ***PreflightAnalysis*** hierarchy describes the results of a comparison of the properties of the file against one ***PreflightConstraint*** in the ***PreflightProfile***.

Resource Properties

Resource Class: Parameter

Output of Processes: **Preflight**

Table N.42: PreflightAnalysis Resource

NAME	DATA TYPE	DESCRIPTION
ColorsResultsPool ?	element	A pool of ***PreflightDetail*** and ***PreflightInstance*** Subelements that provides analysis about color.
DocumentResult-sPool ?	element	A pool of ***PreflightDetail*** and ***PreflightInstance*** Subelements that provides analysis about documents.
FontsResultsPool ?	element	A pool of ***PreflightDetail*** and ***PreflightInstance*** Subelements that provides analysis about fonts.
FileTypeResult-sPool ?	element	A pool of ***PreflightDetail*** and ***PreflightInstance*** Subelements that provides analysis about file types.
ImagesResultsPool ?	element	A pool of ***PreflightDetail*** and ***PreflightInstance*** Subelements that provides analysis about images.
PagesResultsPool ?	element	A pool of ***PreflightDetail*** and ***PreflightInstance*** Subelements that provides analysis about finished pages.

N.5.7.1 PreflightDetail

PreflightDetail Subelements are used to describe one property within the ***PreflightAnalysis*** category in which they occur. This Subelement is also used by ***PreflightInventory*** Resource.

Table N.43: ***PreflightDetail Element***

NAME	DATA TYPE	DESCRIPTION
PageRefs	IntegerRangeList	Identifies the set of pages in a ***RunList*** Resource that exhibit the characteristic identified by the combination of the @*Property* Attribute and the ***Value*** Element.
Property ?	string	Identifies the property described by this Element.
Status ?	enumeration	**Allowed values are:** Error – Value violates the ***ConstraintValue*** specified in the associated ***PreflightConstraint*** Element. The constraint was flagged as an Error in the profile. Warning – Value violates the ***ConstraintValue*** specified in the associated ***PreflightConstraint*** Element. The constraint was flagged as a Warning in the profile. Ignore – The constraint is ignored, and no ***PreflightDetail*** or ***PreflightInstanceDetail*** Elements are created for this constraint. IgnoreValue – No comparison was made against a ***ConstraintValue***. In other words, either the @*Status* for the ***PreflightConstraint*** was "Ignore" or "IgnoreValue", or this ***PreflightDetail*** is part of a ***PreflightInventory*** hierarchy.
Value ?	element	Identifies the value of the property. The semantics are PDL-specific.

N.5.7.2 PreflightInstance

PreflightInstance Subelements are used to collect ***PreflightInstanceDetail*** Elements for one instance of some object which occurs in the PDL files referenced by a run list. For example, there might be one ***PreflightInstance*** Element for each font that occurs in the pages of a run list. This Subelement is also used by ***PreflightInventory*** Resources.

Table N.44: PreflightInstance Element

NAME	DATA TYPE	DESCRIPTION
Identifier ?	string	Identifies the instance this Element collects ***PreflightInstanceDetail*** Elements.
PageRefs Modified in JDF 1.1	IntegerRangeList	Identifies the set of finished pages in a ***RunList*** on which the instance occurs.
PreflightInstanceDetail * Modified in JDF 1.1	element	A pool of ***PreflightInstanceDetail*** Elements that describe the properties for this instance

N.5.7.3 PreflightInstanceDetail

PreflightInstanceDetail Subelements describe one property of one instance of some object type that occurs in a PDL file. For example, several ***PreflightInstanceDetail*** Elements might describe the properties of a single font. This Subelement is also used by ***PreflightInventory*** Resources

Table N.45: PreflightInstanceDetail Element

NAME	DATA TYPE	DESCRIPTION
Status ?	enumeration	Specifies the results of the comparison between the value of the property for this instance with the ***ConstraintValue*** for the associated ***PreflightConstraint*** Element. **Allowed values are:** Error – Value violates the ***ConstraintValue*** specified. The constraint was flagged as an Error in the profile. Warning – Value violates the ***ConstraintValue*** specified. The constraint was flagged as a Warning in the profile. IgnoreValue – No comparison was made against a ***ConstraintValue***. In other words, either the @*Status* for the Constraint was "Ignore" or "IgnoreValue", or this ***PreflightInstanceDetail*** is part of a ***PreflightInventory*** hierarchy.
Property ?	string	Identifies the property described by this Element.
Value ?	element	Identifies the value of the property. The semantics are PDL-specific.

N.5.8 PreflightInventory

Deprecated in JDF 1.2

This Resource was deprecated as a result of a major revision to the **Preflight** Process and its associated Resources.

PreflightInventory Resources, like ***PreflightAnalysis*** Resources, record the results of a **Preflight** Process. The semantics for results are specific to the ***FileType*** of the for the file. The Elements in this Resource, detailed in the table below, place the results in specific categories. The value of each of these Elements is an array of ***PreflightResultsDetail*** and ***PreflightInstance*** Subelements. Within the ***PreflightInstance*** Subelements, results are further broken down into ***PreflightInstanceDetail***.

Each ***PreflightResultsDetail*** or ***PreflightInstance*** Subelement in the ***PreflightInventory*** hierarchy describes the results of a comparison of the properties of the file against one ***PreflightConstraint*** in the ***PreflightProfile***.

Resource Properties

Resource Class: Parameter

Input of Processes: **Preflight**

Output of Processes: **Preflight**

Table N.46: PreflightInventory Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
ColorsResultsPool ?	element	A pool of ***PreflightDetail*** and ***PreflightInstance*** Subelements that provides a color inventory.
DocumentResultsPool ?	element	A pool of ***PreflightDetail*** and ***PreflightInstance*** Subelements that provides a document inventory.

Table N.46: PreflightInventory Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
FontsResultsPool ?	element	A pool of ***PreflightDetail*** and ***PreflightInstance*** Subelements that provides a font inventory.
FileTypeResult-sPool ?	element	A ***PreflightDetail*** and ***PreflightInstance*** Subelement that provides a file-type inventory.
ImagesResultsPool ?	element	A pool of ***PreflightDetail*** and ***PreflightInstance*** Subelements that provides an image inventory.
PagesResultsPool ?	element	A pool of ***PreflightDetail*** and ***PreflightInstance*** Subelements that provides a finished page inventory.

N.5.9 PreflightProfile

Deprecated in JDF 1.2

This Resource was deprecated as a result of a major revision to the **Preflight** Process and its associated Resources.

PreflightProfile Resources specify a set of constraints against which a file may be tested. The semantics for constraints are specific to the ***FileType*** of the for the file. The Elements in this Resource, detailed in the table below, place the results in specific categories. The value for each of these Elements is an array of ***PreflightConstraint*** Subelements. Within the ***PreflightConstraint*** Resources, the ***ConstraintValue*** Element indicates allowable values and the @*Status* Attribute indicates the error level (if any) to be flagged when exceptions to the constraints are identified.

Resource Properties

Resource Class: Parameter

Input of Processes: **Preflight**

Table N.47: PreflightProfile Resource

NAME	DATA TYPE	DESCRIPTION
ColorsContraint-sPool ?	element	A pool of ***PreflightConstraint*** Subelements. Each Element in this pool identifies a specific constraint concerning colors against which to test the file
DocumentCon-straintsPool ?	element	A pool of ***PreflightConstraint*** Subelements. Each Element in this pool identifies a specific constraint concerning documents against which to test the file
FontsConstraint-sPool ?	element	A pool of ***PreflightConstraint*** Subelements. Each Element in this pool identifies a specific constraint concerning fonts against which to test the file
FileTypeConstraint-sPool ?	element	A **Preflight** constraint. The @*Type* Attribute SHALL have a value of "array" and SHALL contain string objects that identify the allowable types of data in the file. The strings in the @*Value* array SHALL be MIME-file types as recorded by the Internet Assigned Numbers Authority (IANA). IANA has procedures for registering new file types if needed.
ImagesConstraint-sPool ?	element	A pool of ***PreflightConstraint*** Subelements. Each Element in this pool identifies a specific constraint concerning images against which to test the file
PagesConstraint-sPool ?	element	A pool of ***PreflightConstraint*** Subelements. Each Element in this pool identifies a specific constraint concerning finished pages against which to test the file

N.5.9.1 PreflightConstraint

*Table N.48: **PreflightConstraint Element** (Sheet 1 of 2)*

NAME	DATA TYPE	DESCRIPTION
AttemptFixupErrors = "false"	boolean	If "true", the Device performing preflight SHOULD attempt to fix errors that are identified during preflight. Errors that are corrected are not given a @*Status* Attribute. Default = "false"
AttemptFixupWarnings = "false"	boolean	If "true", the Device performing preflight SHOULD attempt to fix warnings that are identified during preflight. Warnings that are corrected are not given a @*Status* Attribute. Default = "false"

Table N.48: PreflightConstraint Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Constraint ?	string	Describes the specific file characteristic to be checked.
Status	enumeration	**Allowed values are:** Error – Values that violate the ***ConstraintValue*** specified are flagged as Errors in ***PreflightDetail*** and ***PreflightInstanceDetail*** Elements. Warning – Values that violate the ***ConstraintValue*** specified are flagged as Warnings in ***PreflightDetail*** and ***PreflightInstanceDetail*** Elements. Ignore – The constraint is ignored, and no ***PreflightDetail*** or ***PreflightInstanceDetail*** Elements are created for this constraint. IgnoreValue – No comparison is made against the ***ConstraintValue***.
ConstraintValue ?	element	Provides a value against which to test occurrences of the characteristic in the file. **Note**: The semantics of the ***ConstraintValue*** Element depend on the PDL characteristic in question.

N.5.10 Proofing

Deprecated in JDF 1.2

The **Proofing** Process is deprecated in **JDF**/1.2. Instead, use a Combined Process to produces the hard proof (e.g., one that includes the **ImageSetting**, **ConventionalPrinting**, or **DigitalPrinting** Process). Then input the hard proof to a separate **Approval** Process.

The **Proofing** Process results in the creation of a physical proof, represented by an ***ExposedMedia*** Resource. Proofs can be used to check an imposition or the expected colors for a Job. The inputs of this Process are a ***RunList***, which identifies the pages to proof; the ***ProofingParams*** Resource, which describes the type of proof to be created; and a ***Media*** Resource to describe the physical media that will be used.

Table N.49: Proofing – Input Resources

NAME	DESCRIPTION
ColorantControl ? Modified in JDF 1.1A	Identifies the color model used by the Job.
ColorSpaceConversionParams ?	This Resource provides information needed to convert colorspaces in the pages for proofing. Generally present if a color proof is desired, unless the pages in the ***RunList*** have already been operated on by a previous colorspace conversion process.
Layout ?	REQUIRED if an imposition proof is desired.
Media	This Resource characterizes the output media for the proof.
ProofingParams	This Resource provides the parameters needed to produce the desired proof.
RunList (Document)	Identifies the pages to be proofed. When the ***Layout*** Resource is present in the ***ProofingParams*** Resource, @*Ord* values from ***ContentObject*** Subelements refer to pages in this ***RunList***.
RunList (Marks) ?	Structured list of incoming marks. These are typically printers marks (e.g., fold, cut or punch marks, or color bars). When the ***Layout*** Resource is present in the ***ProofingParams*** Resource, @*Ord* values from ***MarkObject*** Subelements refer to pages in this ***RunList***.

Table N.50: Proofing – Output Resources

NAME	DESCRIPTION
ExposedMedia	The resulting physical proof.

N.5.11 SoftProofing

Deprecated in JDF 1.2

The **SoftProofing** Process is deprecated in **JDF**/1.2. Instead, use a Combined Process to produce the soft proof in which the last Process is the **Approval** Process that approves the soft proof.

SoftProofing is the Process of reviewing final-form output on a monitor rather than in paper form. The inputs are a ***RunList***, which identifies the pages to proof; the ***ProofingParams*** Resource, which describes the type of proof to be created.

Within the ***ProofingParams*** Resource, the proof Device parameter specifies the characterization the monitor on which the proof will be viewed. This processor SHALL create and perform a transformation from the final target Device to the proof Device colors before displaying the document contents.

The soft proofing parameters allow sufficient control to determine whether any images are displayed in the proof. If so, the ability to select low resolution proxies or full resolution images is provided. The mechanism for approving proofs requires the generation of a PDF file containing the proofing parameters and a digital signature noting the acceptance of them. The approval PDF file need not contain any graphical data.

Like all other color manipulation supported in **JDF**, the color conversion controls are based on the use of ICC profiles. While the assumed characterization of input data can take many forms, each can internally be represented as an ICC Profile. In order to perform the transformations, input profiles SHALL be paired with the identified final target Device profile to create the transformation.

Table N.51: SoftProofing – Input Resources

NAME	DESCRIPTION
ColorantControl ? Modified in JDF 1.1A	Identifies the color model used by the Job.
ColorSpaceConversionParams ?	This Resource provides information needed to convert colorspaces in the pages for proofing. Generally present if a color proof is desired, unless the pages in the ***RunList*** have already been operated on by a previous colorspace conversion process.
Layout ?	REQUIRED if an imposition proof is desired.
ProofingParams	Provides the parameters needed to produce the desired proof.
RunList (Document)	Identifies the pages to be proofed. When the ***Layout*** Resource is present in the ***ProofingParams*** Resource, @*Ord* values from ***ContentObject*** Subelements refer to pages in this ***RunList***.
RunList (Marks) ?	Structured list of incoming marks. These are typically printer's marks (e.g., fold marks, cut marks, punch marks, or color bars). When the ***Layout*** Resource is present in the ***ProofingParams*** Resource, @*Ord* values from ***MarkObject*** Subelements refer to pages in this ***RunList***.

Table N.52: SoftProofing – Output Resources

NAME	DESCRIPTION

N.5.12 IDPrinting

Deprecated in JDF 1.1

The **IDPrinting** Process was deprecated in **JDF**/1.1. Instead, implementations SHOULD use a Combined Process with the **DigitalPrinting** Process and other Processes, thus improving interoperability by reducing one of the combinations of Processes. Also the **IDPrinting** Process defined a number of Resources and Subelements which are deprecated since they duplicate other Resources.

IDPrinting, which stands for Integrated Digital Printing, is a specific form of digital printing. It combines functionality that might be represented by the **Interpreting**, **Rendering**, **Screening**, and **DigitalPrinting** Processes in a single Process. In addition, Devices which support **IDPrinting** frequently provide some degree of finishing capabilities, such as collating and stapling, as well as some automated layout capabilities, such as N-up and duplex printing.

Controls for **IDPrinting** are provided in the ***IDPrintingParams*** Resource. These controls are intended to be somewhat limited in their scope. If greater control over various aspects of the printing Process is needed, **IDPrinting** SHOULD NOT be used. Ultimately, the controls specified for **IDPrinting** can be used to generate an Internet Printing Protocol (IPP) Job. See **JDF**/1.0 Appendix F for a mapping between **JDF** **IDPrinting** and IPP. **IDPrinting** may be combined with other Processes, such as **Trapping** or **ColorSpaceConversion**.

Table N.53: IDPrinting – Input Resources

NAME	DESCRIPTION
ColorantControl ?	The ***ColorantControl*** Resources that define the ordering and usage of inks in print modules.
Component (Cover) ?	A finished cover may be combined with the pages that will be output by this Process.
Component (Input) ?	Various components can be used in **IDPrinting** instead of ***Media***. Examples include waste, precut ***Media***, or a set of preprinted Sheets or webs.
Component (Proof) ?	A Proof component is used if a proof was produced during an earlier **ConventionalPrinting** Process.
ExposedMedia ?	A Proof is useful for comparisons (completeness, color accuracy) with the print out of the **IDPrinting** Process.
FontPolicy ?	Describes the behavior of the font machinery in absence of requested fonts.
Ink ?	Ink or toner and information about it is needed for **IDPrinting**.
InterpretingParams *	A set of Resources that specify how the Device SHOULD interpret the PDL files which are referenced by the ***RunList*** for the Process. Note that ***InterpretingParams*** is an Abstract Resource. Instances are PDL-specific.
IDPrintingParams ?	Specific parameters to set up the machinery.
Media ?	The physical ***Media*** and information about the ***Media***, such as thickness, type, and size, are used to set up paper travel in the press. This has to be present if no preprinted ***Component*** (Input) Resource is present. Note: Printing a Job on more than one Web or Sheet at the same time is parallel processing.
RenderingParams ?	This Resource describes the format of the ***ByteMap*** Elements to be created.
RunList	The set of pages to be printed.
ScreeningParams ?	Parameters specifying which halftone mechanism is to be applied and with what specific controls.
TransferFunctionControl ?	Controls whether the Device performs transfer functions and what values are used when doing so.

Table N.54: IDPrinting – Output Resources

NAME	DESCRIPTION
Component (Good)	Components are produced for other printing Processes or postpress Processes. Note that the *@Amount* Attribute of the ***ResourceLink*** to this Resource indicates the number of copies which will be produced.
Component (Waste) ?	Produced waste, may be used by other Processes.

N.5.13 AdhesiveBinding

Deprecated in JDF 1.1

The **AdhesiveBinding** has been split into the following individual Processes:

- **CoverApplication**,
- **Gluing**,
- **SpinePreparation**,
- **SpineTaping**.

Note that the parameters of the ***GlueApplication ABOperations*** have been moved into ***CoverApplicationParams*** and ***SpineTapingParams*** as ***GlueApplication refelements***. The generic ***GlueApplication ABOperation*** is now described by the **Gluing** Process.

N.5.14 Dividing

Deprecated in JDF 1.1.

Dividing has been replaced by **Cutting**. In-line finishing of Web Presses often includes equipment for cutting the ribbon(s) in cross direction. This operation can be described with the **Dividing** Process. **Dividing** in cross direction is likely to happen after former folding, which is a **LongitudinalRibbonOperations** Process. It may affect one or more ribbons at the same time that are all part of one ***Component***.

Table N.55: Dividing – Input Resources

NAME	DESCRIPTION
Component	The **Dividing** Process consumes one ***Component*****:** the Web(s) or ribbon(s) entering the crosscutting machinery. The substrate might have been treated with **LongitudinalRibbonOperations** and may be folded with a former fold.
DividingParams	Specific parameters to set up the machinery.

Table N.56: Dividing – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: either the divided Web or ribbon.

N.5.15 LongitudinalRibbonOperations

Deprecated in JDF 1.1

In-line finishing within Web Printing presses can include folding, perforating, or applying a line of glue on the ribbon while it is traveling in longitudinal direction. In version 1.1 of **JDF** and beyond, in-line finishing is described using the "standard" finishing Processes (e.g., **Creasing**, **Cutting**, **Folding** or in a Combined Process Node with **ConventionalPrinting**).

Table N.57: LongitudinalRibbonOperations – Input Resources

NAME	DESCRIPTION
Component	The ***Component*** can consist of more than one Web or ribbon that has been combined with the **Gathering** Process.
LongitudinalRibbonOperationParams	Specific parameters to set up the machinery tools for the **LongitudinalRibbonOperations** Process.

Table N.58: LongitudinalRibbonOperations – Output Resources

NAME	DESCRIPTION
Component +	A ribbon is produced that is used in other postpress Processes. If the **LongitudinalRibbonOperations** Process was slitting, more than one ***Component*** is produced.

N.5.16 Numbering

Deprecated in JDF 1.5

Numbering is the Process of stamping or applying variable marks in order to produce unique components for items such as lottery notes or currency. No database access is needed, and the counters automatically increase incrementally. **Numbering** is also used for alphanumeric, automatic and unique marking.

Table N.59: Numbering – Input Resources (Sheet 1 of 2)

NAME	DESCRIPTION
Component	One ***Component*** (e.g., a printed Sheet or a pile of Sheets) are modified in the **Numbering** Process.

Table N.59: Numbering – Input Resources (Sheet 2 of 2)

NAME	DESCRIPTION
NumberingParams	Specific parameters to set up the machinery.

Table N.60: Numbering – Output Resources

NAME	DESCRIPTION
Component	One ***Component*** is produced: the numbered Sheet.

N.5.17 SaddleStitching

Deprecated in JDF 1.1

In **SaddleStitching**, Signatures are collected so that all sections have a common spine, and then stitched with staples through the spine. **SaddleStitching** has been replaced by **Stitching** in **JDF** 1.1.

Table N.61: SaddleStitching – Input Resources

NAME	DESCRIPTION
Component	The only REQUIRED ***Component*** is the collected pile.
SaddleStitchingParams	Specific parameters to set up the machinery.

Table N.62: SaddleStitching – Output Resources

NAME	DESCRIPTION
Component	The stitched-together components.

N.5.18 SideSewing

Deprecated in JDF 1.1

Replaced by **ThreadSewing**.

This is a binding technique resulting in robust products that have a significant loss of inner margin space and poor handling characteristics. For these reasons, other binding techniques are used more often. In **SideSewing**, the first step is to create the holes in the book block and inject the glue (see ▸ Section 6.6.2 HoleMaking). Then the entire book is sewn at once with a *@ThreadMaterial* such as "Cotton" or "Polyester". If the book block is rather thick, a **Stitching** Process using wire might be performed before **SideSewing**.

Table N.63: SideSewing – Input Resources

NAME	DESCRIPTION
Component	The only REQUIRED ***Component*** is the gathered Sheets.
SideSewingParams	Specific parameters to set up the machinery.

Table N.64: SideSewing – Output Resources

NAME	DESCRIPTION
Component	The ***Component*** is produced.

N.6 Deprecated Intents

N.6.1 BindingIntent Deprecated Subelements

Note: ***BindingIntent*** is still a valid Resource. The following sections from within ***BindingIntent*** were deprecated and were deemed large enough to warrant moving them to this section.

N.6.1.1 AdhesiveBinding

Deprecated in JDF 1.1

Table N.65: AdhesiveBinding Element

NAME	DATA TYPE	DESCRIPTION
Scoring ?	EnumerationSpan	Scoring option for ***AdhesiveBinding***. **Allowed values are:** TwiceScored QuadScored None **Note:** Values are based on viewing the cover in its flat pre-binding state.
SpineGlue ?	EnumerationSpan	Glue type used to define ***AdhesiveBinding*** procedures. **Allowed values are:** ColdGlue Hotmelt PUR – Polyurethane Rubber
TapeBinding ?	OptionSpan	If "true", a cloth tape which has been pre-glued with hot-melt adhesive is used in **AdhesiveBinding** the unmilled block (e.g., FastBack or DocuTech binding).

N.6.1.2 BookCase

Deprecated in JDF 1.1

This Subelements contains details of the book case for hard-cover book binding. The actual binding parameters are set in the appropriate ***AdhesiveBinding***, ***ThreadSewing***, or ***ThreadSealing*** Elements.

Table N.66: BookCase Element

NAME	DATA TYPE	DESCRIPTION
HeadBands ?	OptionSpan	The following **CaseBinding** choice specifies the use of headbands on a case bound book. If "true", headbands are inserted both top and bottom.
Shape ?	EnumerationSpan	Indicates the shape of the "back" or spine of a case bound book. **Allowed values are:** RoundedBack SquareBack
Thickness ?	NumberSpan	Specifies thickness of board which is wrapped as front and back covers of a case bound book, in points.

N.6.2 DeliveryIntent Deprecated Subelements

Note: ***DeliveryIntent*** is still a valid Resource. The following sections from within ***DeliveryIntent*** were deprecated and were deemed large enough to warrant moving them to this section. All Pricing related information has been moved to ▸[PrintTalk].

N.6.2.1 Pricing

Deprecated in JDF 1.3

Table N.67: Pricing Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AdditionalPrice ?	double	Price for ordering the number of copies specified in the @*AdditionalAmount* Attribute as specified in the parent Element of the ***Pricing***.
Currency ?	NMTOKEN	Three digit currency definition according to ▸[ISO4217:2001]. It defaults to the currency defined in the parent quote.

Table N.67: Pricing Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
HasPrice = "true"	boolean	Specifies whether the line item defined by this quote has a price. If "false", the line item is not included in the parent quote, and the price is unknown and SHALL be added. If "true", the line item is included in the parent quote.
Item ?	string	Name of the item that this particular quote element describes. If not specified, ***Pricing*** applies to the entire ***DropItemIntent***.
Price ?	double	Price for ordering the number of copies specified in the @*Amount* Attribute as specified in the parent Element of the ***Pricing***. If not specified, it defaults to the sum of prices of the direct child ***Pricing*** Elements.
Payment ? New in JDF 1.1	element	Details of the payment method.
Pricing *	element	Individual items of the quote. Note that a parent quote defines the complete quote (i.e., including the values defined in the line items of any child quotes but excluding all line items with @*HasPrice* = "false"). The sum of line items need not be identical to the parent quote.

N.6.2.2 Payment

New in JDF 1.1

Deprecated in JDF 1.3

Table N.68: Payment Element

NAME	DATA TYPE	DESCRIPTION
PayTerm ?	text element	Describes the payment terms & conditions.
CreditCard ?	element	Specifies credit card information

N.6.2.3 CreditCard

New in JDF 1.1

Deprecated in JDF 1.3

Table N.69: CreditCard Element

NAME	DATA TYPE	DESCRIPTION
Authorization ?	String	Authorization code for this transaction.
AuthorizationExpires ?	gYearMonth	Expiration date of the @*Authorization*.
Expires	gYearMonth	Expiration date of the credit card.
Number	NMTOKEN	Credit card number. The format is specified without blanks or any other separator characters.
Type	NMTOKEN	Credit card brand. **Values include:** Amex DinersClub Discovery MasterCard – This includes derived brands (e.g., EuroCard). Visa

N.6.3 NumberingIntent

Deprecated in JDF 1.5

This Resource describes the parameters of stamping or applying variable marks in order to produce unique components, for items such as lottery notes or currency.

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***NumberingParams***

Input of Processes: Any Product Intent Node (▸ Section 7.0.1 Product Intent Descriptions)

Table N.70: NumberingIntent Resource

NAME	DATA TYPE	DESCRIPTION
ColorName ?	Enumera-tionSpan	Defines the color of the numbering. **Allowed values are from:** ▸ Table A.22 NamedColor Enumeration Values.
ColorNameDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If ColorNameDetails is supplied, ColorName SHOULD also be supplied.
ColorPool ?	refelement	Additional details about the colors used.
NumberItem +	element	Individual position of the numbers on the finished page.

N.6.3.1 NumberItem

Table N.71: NumberItem Element

NAME	DATA TYPE	DESCRIPTION
ColorName ?	Enumera-tionSpan	Defines the color of the numbering. **Default value is from:** ***NumberingIntent***/@*ColorName*. **Allowed values are from:** ▸ Table A.22 NamedColor Enumeration Values..
ColorNameDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If ***ColorName-Details*** is supplied, ***ColorName*** SHOULD also be supplied.
Orientation?	NumberSpan	Rotation of the numbering machine in degrees. If ***Orientation***/@*Actual* = 0, the top of the numbers is along the leading edge.
StartValue = "1"	string	First value of the numbering machine.
Step = "1"	integer	Number that specifies the difference between two subsequent numbers of the numbering machine.
XPosition ?	NumberSpan	Position of the number in the X direction of the product.
YPosition ?	NumberSpan	Position of the number in the Y direction of the product.
SeparationSpec ?	element	Specifies the name of the ***Color*** in the ***ColorPool*** that is used for Numbering.

N.6.4 SizeIntent

Deprecated in JDF 1.1

SizeIntent has been deprecated in **JDF** 1.1. All contents have been moved to ***LayoutIntent***. This Resource records the size of the finished pages for the product component. It does not, however, specify the size of any intermediate results, such as press Sheets.

Resource Properties

Resource Class: Intent

Process Resource Pairing: ***CutMark***, ***CuttingParams***, ***Layout***, ***LayoutPreparationParams***, ***Sheet***, ***Surface***, ***TrimmingParams***

Example Partition: "Option"

Input of Processes: Any Product Intent Node (▸ Section 7.0.1 Product Intent Descriptions)

Table N.72: SizeIntent Resource

NAME	DATA TYPE	DESCRIPTION
Dimensions	XYPairSpan	Specifies the height and width of the product component in points. **Note:** Height and width are ambiguously specified in **JDF** 1.0.
Pages ?	IntegerSpan	Specifies the number of pages of the product component.
Type = "Folded"	enumeration	Specifies whether the product component referred to is flat or finished. **Allowed values are:** Folded – Size of the product after folding. The default value Flat – Size of the unfolded Sheet. Note that this describes the size of a Sheet that is folded to create a product, not the size of the Sheet in the press.

N.7 Deprecated Parameters

N.7.1 AdhesiveBindingParams

Deprecated in JDF 1.1

This Resource describes the details of the following four subprocesses of the **AdhesiveBinding** Process:

- Back preparation
- Multiple glue applications
- Spine taping
- Cover application

These subprocesses are identified as instances of the Abstract ***ABOperation*** Element. Although a workflow may exist that groups these Processes according to its own capabilities, it is likely that they will be performed in the order presented. A description of each follows the table containing the contents of the ***AdhesiveBindingParams*** Resource.

Resource Properties

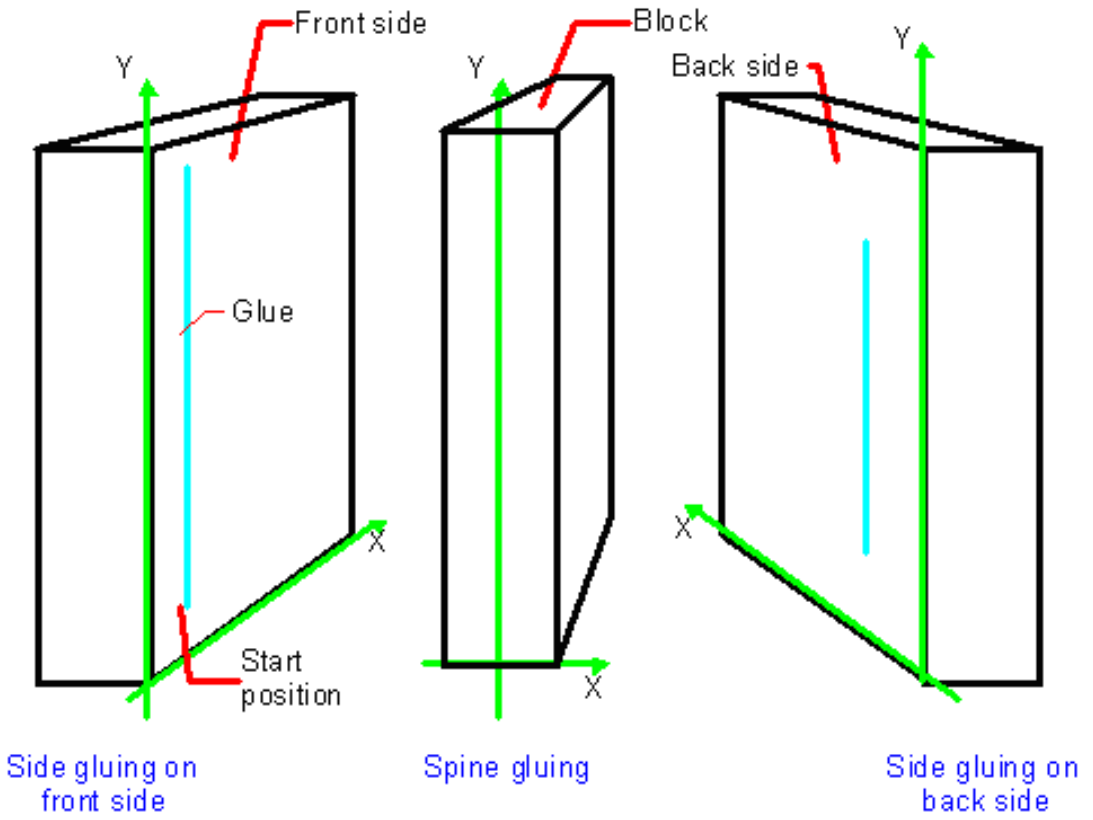
Resource Class: Parameter

Input of Processes: **AdhesiveBinding**

Table N.73: AdhesiveBindingParams Resource

NAME	DATA TYPE	DESCRIPTION
FlexValue ?	double	Flex quality parameter given in [N/cm].
PullOutValue ?	double	Pull out quality parameter given in [N/cm].
ABOperation +	Element	An Abstract Element which is a placeholder for an operation (**SpinePreparation**, ***GlueApplication***, **SpineTaping**, and **CoverApplication**). Each ***ABOperation*** Element describes the parameters of one single operation of the complete **AdhesiveBinding** Process.

Figure N-2: Parameters and coordinate system for glue application



N.7.2 BoxFoldingParams Deprecated Subelements

N.7.2.1 BoxApplication

Deprecated in JDF 1.4

A ***BoxApplication*** describes the application of an external ***Component*** such as a window or handle to a folding box in the box folder-gluer. Note that a short description of the application SHOULD be specified in ***BoxApplication***/ *@DescriptiveName*. Application of an external ***Component*** SHOULD be described with a combined **Inserting** process.

Table N.74: BoxApplication Element

NAME	DATA TYPE	DESCRIPTION
ApplicationArea ?	rectangle	Area in the current coordinate system of the folder gluer where the ***Component*** is applied. **Note:** A single point is specified by X0 = X1 and Y0 = Y1 of the rectangle and a line is specified by X0 = X1 or Y0 = Y1.
Component	refelement	Reference to a ***Component*** that is applied. This ***Component*** SHALL also be specified as in input ***Component*** to the **BoxFolding** Process with *@ProcessUsage* = "Application"
GlueLine *	element	Specification of a glue lines needed to glue the ***Component*** described in this ***BoxApplication***. The glue lines are applied to the ***Component*** in the coordinate system of the ***BoxApplication***/***Component***. The glue lines applied to the blank are specified in ***BoxFoldingParams Deprecated Subelements***/***GlueLine***.

N.7.3 CustomerMessage

Deprecated in JDF 1.5

CustomerMessage is an abstract definition of messages to the customer. Formatting and details of the content generation of the message are system dependent.

Table N.75: CustomerMessage Element

NAME	DATA TYPE	DESCRIPTION
Language ?	language	Language to be used for the ***CustomerMessage***.
MessageEvents	NMTOKENS	Defines the set of events that trigger a message that is defined or specified by the system. **Values include those from:** ▸ Table C.12 MessageEvents ContentStatus and MilestoneType Values.
ShowList ? Modified in JDF 1.4	NMTOKENS	List of parameters to display in the ***CustomerMessage***. **Values include those from:** ▸ Table H.1 Predefined variables used in @XXX-Template. New in JDF 1.4 **Modification note:** Starting with **JDF** 1.4, the values come from a common list rather than a list that is custom to this Element.
ComChannel *	element	Communication channel for the desired ***CustomerMessage***. In case it is not specified, the ***CustomerMessage*** will be provided according to system pre-defined information. If multiple ***ComChannel*** Elements are specified, the ***CustomerMessage*** SHOULD be sent to all specified communication channels.

N.7.4 DBMergeParams

Deprecated in JDF 1.5

This Resource specifies the parameters of the **DBTemplateMerging** Process.

Resource Properties

Resource Class: Parameter

Table N.76: DBMergeParams Resource

NAME	DATA TYPE	DESCRIPTION
FileSpec ?FileSpecRef	refelement	URL of the generated destination file. This is most often a printable file type (e.g., PDF of PPML). If ***FileSpec*** is not specified, ***DBMergeParams*** SHALL be a Pipe Resource.
SplitDocuments ?	integer	Indicates how often to split documents to create a new file.

N.7.5 DBRules

Deprecated in JDF 1.5

This Resource specifies the rules that are to be applied to convert a database record into a graphic element. It is described by a text element with a human-readable description of the selection rules. For example:

```
insert the "Age" field behind the birthday;
if income>100,000 use Porsche.gif, else use bicycle.jpeg for image #2.
```

The internal representation of the mapping of database fields to graphic content within the document template is implementation-dependent. It can vary from fully variable, multi-page, automated document layout to simply inserting some line-feed characters between database records in an address field. Therefore, ***DBRules*** is defined as a simple human-readable text element.

Resource Properties

Resource Class: Parameter

Table N.77: DBRules Resource

NAME	DATA TYPE	DESCRIPTION
Comment +	element	Human-readable description of the database rules that map database fields to image or text content.

N.7.6 DBSchema

Deprecated in JDF 1.5

This Resource specifies the formal structure of a database record, regardless of type. It is encoded as a text element with a human-readable description of the database schema.

Resource Properties

Resource Class: Parameter

Table N.78: DBSchema Resource

NAME	DATA TYPE	DESCRIPTION
DBSchemaType	enumeration	Database type. **Allowed values are:** CommaDelimited SQL XML
Comment +	element	Human-readable description of the database schema.

N.7.7 DBSelection

Deprecated in JDF 1.5

This Resource specifies a selection of records from a database.

Resource Properties

Resource Class: Parameter

Table N.79: DBSelection Resource

NAME	DATA TYPE	DESCRIPTION
DataBase	URL	URL of the database
Records ?	IntegerRangeList	The indices of the database records.
Select ?	string	Database selection criteria in the native language of the database (e.g., SQL).

N.7.8 DividingParams

Deprecated in JDF 1.1.

Since the **Dividing** Process has been replaced by **Cutting**, this Resource is no longer REQUIRED. This Resource contains Attributes and Elements used in executing the **Dividing** Process.

Resource Properties

Resource Class: Parameter

Example Partition: "RibbonName", "SheetName", "SignatureName", "WebName"

Input of Processes: **Dividing**

Table N.80: DividingParams Resource

NAME	DATA TYPE	DESCRIPTION
DividePositions	DoubleList	Array containing the cross cut positions in y-direction (direction of Web traveling).

N.7.9 FormatConversionParams

New in JDF 1.1

Deprecated in JDF 1.5

This Resource defines the parameters needed for generic **FormatConversion** of digital files.

Resource Properties

Resource Class: Parameter

Example Partition: "DocIndex", "RunIndex", "RunTags"

Input of Processes: **FormatConversion**

Table N.81: FormatConversionParams Resource

NAME	DATA TYPE	DESCRIPTION
ColorPool ? New in JDF 1.2	refelement	Additional detail about the colors used in the file to be converted.
FileSpec (InputFormat) ? Deprecated in JDF 1.2	refelement	The format of the original file is specified in a ***FileSpec*** with @*ResourceUsage* = "InputFormat". A URL SHOULD NOT be specified because the list of files is given by the input ***RunList*** of the **FormatConversion** Process. The purpose of this Element in **JDF** 1.1 and earlier was to provide the MIME type of the file to be created. This is now defined directly using the ***FileSpec*** of the input ***RunList*** of the **FormatConversion** Process.
FileSpec (OutputFormat) ? Deprecated in JDF 1.2	refelement	The format of the converted file is specified in a ***FileSpec*** with @*ResourceUsage* = "OutputFormat". A URL SHOULD NOT be specified because the list of files is given by the output ***RunList*** of the **FormatConversion** Process. The purpose of this Element in **JDF** 1.1 and earlier was to provide the MIME type of the file to be created. This is now defined directly using the ***FileSpec*** of the output ***RunList***.
ImageCompressionParams ? New in JDF 1.2	refelement	Provides a set of controls that determines how images will be down-sampled and compressed in the converted documents
TIFFFormatParams ? New in JDF 1.2	element	Parameters specific to conversion of rasters to TIFF files. (See below.) **FormatConversion** SHOULD NOT be used to convert non-raster files to TIFF. The appropriate **Interpreting** and **Rendering** Processes SHOULD be used first.

To control the creation of files in formats other than TIFF, equivalent Subelements to ***TIFFFormatParams*** may be defined. It is possible to use ***ImageCompressionParams*** to request de-screening of 1-bit per channel rasters to contone rasters (usually accompanied by a reduction in resolution). Additional data regarding the screens used in the original rasters MAY be provided as a ***ScreeningParams*** Resource supplied in a ***LayoutElement*** as part of the input ***RunList***.

N.7.10 IDPrintingParams

Deprecated in JDF 1.1

This Resource contains the parameters needed to control the **IDPrinting** Process.

Resource Properties

Resource Class: Parameter

Example Partition: "DocIndex", "DocRunIndex", "DocSheetIndex", "PartVersion", "Run", "RunIndex", "RunTags", "SheetIndex", "SheetName", "Side"

Input of Processes: **IDPrinting**

Table N.82: IDPrintingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
AttributesNaturalLang = "US English"	language	Language selected for communicating Attributes. Default = "US English"
IDPAttributeFidelity = "false"	boolean	Indicates whether or not the Device SHALL reject the Job if there are Attribute Values or Elements that it does not support. Default = "false"
IPPJobPriority = "50"	integer	The scheduling priority for the Job where 100 is the highest and 1 is the lowest. Amongst the Jobs that can be printed, all higher priority Jobs SHALL be printed before any lower priority ones. Default = 50
IPPVersion ?	XYPair	A pair of numbers indicating the version of the IPP protocol to use when communicating to IPP Devices. The X value is the major version number.
OutputBin ?	NMTOKEN	Specifies the bin to which the finished document is to be output. **Values include those from:** ▸ Table N.83 OutputBin Attribute Values.

Table N.82: IDPrintingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
PageDelivery ?	enumeration	Indicates how pages are to be delivered to the output bin or finisher. **Allowed values are:** SameOrderFaceUp – Order as defined by the ***RunList***, with the "front" sides of the media up. SameOrderFaceDown – Order as defined by the ***RunList***, with the "front" sides of the media up. ReverseOrderFaceUp – Order reversed, as defined by the ***RunList***, with the "front" sides of the media up. ReverseOrderFaceDown – Order reversed, as defined by the ***RunList***, with the "front" sides of the media down.
PrintQuality ?	enumeration	Indicates how pages are to be delivered to the output bin or finisher. **Allowed values are:** High – Highest quality available on the printer. Normal – The default quality provided by the printer. Draft – Lowest quality available on the printer.
SheetCollate ?	boolean	Determines whether the sequencing of the leaves in the output of the Job. If "true", Sheets for each copy of the document are sequenced together, followed by the Sheets for the next copy. If "false", all copies of the first Sheet are sequenced, followed by the second and subsequent Sheet. @*SheetCollate* describes the order of the final Sheet, but does not prescribe the order in which they are produced.
Cover *	element	0, 1 or 2 ***Cover*** Elements. The default instance is that there is no cover.
IDPFinishing ?	refelement	This Element provides the details of how media for each Instance Document is to be finished.
IDPLayout ?	refelement	This Element provides the details of how the contents the finished pages will be imaged onto media.
JobSheet *	element	A set of Sheets which SHALL be produced with the Job. The default case is that no Job Sheets are produced
MediaIntent ?	refelement	A ***MediaIntent*** Element. This Element is ignored if a ***MediaSource*** Resource is present and can be honored for the **IDPrinting** Process. If ***MediaSource*** is absent or cannot be honored, this Element describes the intended media for the Job to allow the Device to select from among the available media.
MediaSource ?	refelement	Describes the source and physical orientation of the media to be used.

N.7.11 OutputBin Attribute Values

Table N.83: OutputBin Attribute Values

VALUE	DESCRIPTION
Top	The bin that, when facing the Device, can best be identified as “top”.
Middle	The bin that, when facing the Device, can best be identified as “middle”.
Bottom	The bin that, when facing the Device, can best be identified as “bottom”.
Side	The bin that, when facing the Device, can best be identified as “side”.
Left	The bin that, when facing the Device, can best be identified as “left”.
Right	The bin that, when facing the Device, can best be identified as “right”.
Center	The bin that, when facing the Device, can best be identified as “center”.
Rear	The bin that, when facing the Device, can best be identified as “rear”.
FaceUp	The bin that can best be identified as “face up” with respect to the Device.
FaceDown	The bin that can best be identified as “face down” with respect to the Device.
FitMedia	Requests the Device to select a bin based on the size of the media.
LargeCapacity	The bin that can best be identified as the “large capacity” bin (in terms of the number of Sheets) with respect to the Device.
Mailbox-N	The Job will be output to the bin that is best identified as “Mailbox-1”, “Mailbox-2”...etc.
Stacker-N	The Job will be output to the bin that is best identified as “Stacker-1’, ‘Stacker-2” ...etc.
Tray-N	The Job will be output to the tray that is best identified as “Tray-1”, “Tray-2” ... etc.

N.7.11.1 Cover

Deprecated in JDF 1.1

This Element describes the cover requested for the Job. Covers may be applied to the whole Job, or to each Instance Document in the Job. Note that front and back covers may be specified.

Table N.84: Cover Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BackSide = "false"	boolean	The next page from the ***RunList*** is imaged onto the back of this cover. This would be the inside of a "Front" cover and outside of a "Back" cover. Default = "false"
CoverType = "Front"	enumeration	Specifies whether this ***Cover*** Element specifies the front or back cover. **Allowed values are:** Front – The front cover. Back – The back cover.
FrontSide = "false"	boolean	The next page from the ***RunList*** is imaged onto the front of this cover. This would be the outside of a "Front" cover and inside of a "Back" cover. Default = "false"
IDPFinishing ?	refelement	An ***IDPFinishing*** Element that describes the finishing options for the cover.
IDPLayout ?	element	This Element provides the details of how page contents will be imaged onto the cover.
MediaIntent ?	refelement	A ***MediaIntent*** Element. This Element describes the media to be used for the Job. This Element is ignored if a ***MediaSource*** Resource is present and can be honored for the **IDPrinting** Process. If ***MediaSource*** is absent or cannot be honored, this Element describes the intended media for the Job to allow the Device to select from among the available media.

Table N.84: Cover Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
MediaSource ?	refelement	Describes the source and physical orientation of the media to be used.

N.7.11.2 IDPFinishing

Deprecated in JDF 1.1

IDPFinishing Elements describe finishing operations that are to be applied to sets of Sheets that are output by the **IDPrinting** Process. The finishings are applied to the entire Job when there are no Instance Documents. Otherwise, each Instance Document is finished separately. Operation-specific Subelements may also be present when a Device provides controls for a finishing operation. Additional Subelements are expected to be defined over time. Also, more detail will be added to the currently defined Elements .

Table N.85: IDPFinishing Element

NAME	DATA TYPE	DESCRIPTION
Finishings ?	IntegerList	A set of finishing operations to apply to the Job. The operations are encoded as an enumeration. **Values include those from:** ▸ Table N.86 Finishings Attribute Values.
IDPFolding ?	refelement	Provides details of how to fold the set of pages (or document). When this Element is present, @*Finishings* is ignored.
IDPHoleMaking ?	refelement	Provides details of how to punch holes in the set of pages (or document). When this Element is present, @*Finishings* is ignored.
IDPStitching ?	refelement	Provides details of how to stitch the set of pages (or document). When this Element is present, @*Finishings* is ignored.
IDPTrimming ?	refelement	Provides details of how to trim the set of pages (or document). When this Element is present, @*Finishings* is ignored.

N.7.11.2.1 Finishings Attribute Values

Table N.86: Finishings Attribute Values

VALUE	DESCRIPTION
3	(none) Perform no finishing
4	(staple) Bind the document(s) with one or more staples. The exact number and placement of the staples is site-defined.
5	(punch) This value indicates that holes are REQUIRED in the finished document. The exact number and placement of the holes is site-defined. The punch specification may be satisfied (in a site- and implementation-specific manner) either by drilling/punching, or by substituting predrilled media.
6	(cover) This value is specified when it is desired to select a non-printed (or preprinted) cover for the document. This does not supplant the specification of a printed cover (on cover stock medium) by the document itself.
7	(bind) This value indicates that a binding is to be applied to the document; the type and placement of the binding is site-defined.
8	(saddle-stitch) Bind the document(s) with one or more staples (wire stitches) along the middle fold. The exact number and placement of the staples and the middle fold is implementation and/or site-defined.
9	(edge-stitch) Bind the document(s) with one or more staples (wire stitches) along one edge. The exact number and placement of the staples is implementation and/or site-defined.
10	(fold) Fold the document(s) with one or more folds. The exact number and orientations of the folds is implementation and/or site-defined.
11	(trim) Trim the document(s) on one or more edges. The exact number of edges and the amount to be trimmed is implementation and/or site-defined.
12	(bale) Bale the document(s). The type of baling is implementation and/or site-defined.
13	(booklet-maker) Deliver the document(s) to the Signature booklet maker. This value is a short cut for specifying a Job that is to be folded, trimmed and then saddle-stitched.
14	(jog-offset) Shift each copy of an output document from the previous copy by a small amount which is Device dependent. This value has no effect on the "Job-Sheet." This value SHOULD NOT have an effect if each copy of the Job consists of one Sheet.
50	(bind-left) Bind the document(s) along the left edge. The type of the binding is site-defined.
51	(bind-top) Bind the document(s) along the top edge. The type of the binding is site-defined.
52	(bind-right) Bind the document(s) along the right edge. The type of the binding is site-defined.
53	(bind-bottom) Bind the document(s) along the bottom edge. The type of the binding is site-defined.

N.7.11.3 IDPFolding

Deprecated in JDF 1.1

This Element describes the folding requested for a set of pages in the document.

Table N.87: IDPFolding Element

NAME	DATA TYPE	DESCRIPTION
FoldingParams ?	refelement	Describes the details of how to fold the media.

N.7.11.4 IDPHoleMaking

Deprecated in JDF 1.1

This Element describes the hole making requested for a set of pages in the document.

Table N.88: IDPHoleMaking Element

NAME	DATA TYPE	DESCRIPTION
HoleMakingParams ?	refelement	Describes the details of the holes to be punched into the Media.

N.7.11.5 IDPLayout

Deprecated in JDF 1.1

Table N.89: IDPLayout Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Border = "0"	number	A real number that indicates the width of a border, in points, which will be drawn around the page images on the media. Default = "0" (i.e., no border will be drawn).
FinishedPageOrientation = "Portrait"	enumeration	Indicates the desired orientation of the finished page. This value is used with @*PresentationDirection* to determine how pages will be imaged onto the media. **Allowed values are:** Portrait – The short edges of the media are the top and bottom. Landscape – The long edges of the media are the top and bottom.
ForceFrontSide ?	NumberRangeList	A set of numbers which identify a set of finished pages in the ***RunList*** that are always to be imaged on the front side of a piece of media.
ImageShift ?	element	Element which describes how page images are to be placed onto the media. When @*NumberUp* is present and is not "1,1", @*NumberUp* is applied before the ***ImageShift***, and all contents for each surface are shifted the same amount.
NumberUp ?	XYPair	The number of pages to impose onto a single side of media. The way in which the pages are to be imaged onto the media is determined by the values of @*FinishedPageOrientation* and @*PresentationDirection*. @*FinishedPageOrientation* indicates how the page will be oriented, and @*PresentationDirection* indicates how page images will be distributed, given that orientation.
PresentationDirection ?	enumeration	Indicates the order in which the requested @*NumberUp* pages will be imaged onto the media. The value of @*FinishedPageOrientation* is used to define "top", "left", "right" and "bottom" for the media. **Allowed values are:** ToBottomToRight – Pages are imaged in successive columns, from left to right, starting at the top of each column. ToBottomToLeft – Pages are imaged in successive columns, from right to left, starting at the top of each column. ToTopToRight – Pages are imaged in successive columns, from left to right, starting at the bottom of each column. ToTopToLeft – Pages are imaged in successive columns, from right to left, starting at the bottom of each column. ToRightToBottom – Pages are imaged in successive rows, from top to bottom, starting at the left of each row. ToRightToTop – Pages are imaged in successive rows, from bottom to top, starting at the left of each row. ToLeftToBottom – Pages are imaged in successive rows, from top to bottom, starting at the right of each row. ToLeftToTop – Pages are imaged in successive rows, from bottom to top, starting at the right of each row.

Table N.89: IDPLayout Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Rotate = "0"	number	A number of degrees which the page contents are to be rotated prior to being imaged onto page contents. A positive value is taken to mean an counter-clockwise rotation. The page contents will be scaled to fit the printable area of the media after the rotation. Note: Text will be reflowed in cases where the PDL for the page allows reflow by the Device. Default = "0"
Sides = "OneSided"	enumeration	Indicates how pages are to be imposed onto sides of the medium. **Allowed values are:** OneSided – Page contents will only be imaged on one side of the media. The default. TwoSidedLongEdge – Impose pages upon the front and back sides of media Sheets so that the orientation of the pages on each side is appropriate for binding along the long edge. Equivalent to "*work-and-turn*". TwoSidedShortEdge – Impose pages upon the front and back sides of media Sheets so that the orientation of the pages on each side is appropriate for binding along the short edge. Equivalent to "*work-and-tumble*".

N.7.11.6 IDPStitching

Deprecated in JDF 1.1

This Element describes the stitching requested for a set of pages in the document

Table N.90: IDPStitching Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
StitchingPosition ?	enumeration	**Specifies the location for stitching. All locations are interpreted as if the document were a portrait document. Ignored if *StitchingParams* is present.** **Allowed values are:** None – The document is not to be stitched. TopLeft – Bind the document with one or more staples in the top left corner. BottomLeft – Bind the document with one or more staples in the Bottom left corner. TopRight – Bind the document with one or more staples in the top right corner. BottomRight – Bind the document with one or more staples in the bottom right corner. LeftEdge – Bind the document with one or more staples across the left edge. TopEdge – Bind the document with one or more staples across the top edge. RightEdge – Bind the document with one or more staples across the right edge. BottomEdge – Bind the document with one or more staples across the bottom edge. DualLeftEdge – Bind the document with two staples across the left edge. DualTopEdge – Bind the document with two staples across the top edge. DualRightEdge – Bind the document with two staples across the right edge. DualBottomEdge – Bind the document with two staples across the bottom edge.
StitchingReferenceEdge ?	enumeration	The edge of the output media relative to which the stapling or stitching SHALL be applied. If *StitchingParams* is present, @*StitchingReferenceEdge* defines the @*BindingEdge*. **Allowed values are:** Bottom – The bottom edge coincides with the x-axis of the coordinate system. Top – The top edge is opposite and parallel to the bottom edge. Left – The left edge coincides with the y-axis of the coordinate system. Right – The right edge is opposite and parallel to the left edge.

Table N.90: IDPStitching Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
StitchingParams ?	refelement	A ***StitchingParams*** Element which provides detailed control of the stitching. @*StitchingReferenceEdge* SHALL be present if ***StitchingParams*** is provided.

N.7.11.7 IDPTrimming

Deprecated in JDF 1.1

This Element describes the trimming requested for a set of pages in the document.

Table N.91: IDPTrimming Element

NAME	DATA TYPE	DESCRIPTION
TrimmingParams ?	refelement	Describes the details of how to trim the media.

N.7.11.8 ImageShift

Deprecated in JDF 1.1

ImageShift Elements describe how finished page contents will be imaged onto media. All Attributes refer to positioning along the "X" or "Y" axis. The "X" dimension is the first number of the ***Media*** @*Dimension* Attribute; "Y" is the second number

*Table N.92: **ImageShift** Element*

NAME	DATA TYPE	DESCRIPTION
PositionX = "None"	enumeration	Indicates how finished page images are to be positioned horizontally on the surface. Shifts are applied after positioning. **Allowed values are:** Center – Center the page images horizontally on the surface without regard to limitations of the printable area. Left – Position the left edge of the page images so they is coincident with the left edge of the printable area of the surface. None – Place the page images wherever the print data specifies (the default). Right – Position the right edge of the page images so they is coincident with the right edge of the printable area of the surface.
PositionY = "None"	enumeration	Indicates how finished page images are to be positioned vertically on the surface. Shifts are applied after positioning. **Allowed values are:** Bottom – Position the bottom edge of the page images so they is coincident with the bottom edge of the printable area of the surface. Center – Center the page images horizontally on the surface without regard to limitations of the printable area. None – Place the page images wherever the print data specifies (the default). Top – Position the top edge of the page images so they is coincident with the top edge of the printable area of the surface.
ShiftX ?	integer	The image is to be shifted along the x axis on both sides of the media.
ShiftY ?	integer	The image is to be shifted along the y axis on both sides of the media.
ShiftXSide1 ?	integer	The image is to be shifted along the x axis on the front side of the media.
ShiftXSide2 ?	integer	The image is to be shifted along the x axis on the back side of the media.
ShiftYSide1 ?	integer	The image is to be shifted along the y axis on the front side of the media.
ShiftYSide2 ?	integer	The image is to be shifted along the y axis on the back side of the media.

N.7.11.9 JobSheet

Deprecated in JDF 1.1

This Element describes a Job Sheet which may be produced along with the Job. Job Sheets include separators, Sheets, and error Sheets. The information provided on the Sheet depends on the type of Sheet. In addition, any Sheet type may include an optional Message as a comment Subelement for the Sheet Element. Such a Message comment SHALL have a @*Name* Attribute with the value 'SheetMessage'.

Table N.93: JobSheet Element

NAME	DATA TYPE	DESCRIPTION
SheetFormat = "Standard"	NMTOKEN	Identifies the format of the ***JobSheet***. One one value is defined here, but site-specific values may be defined. **Values include:** Standard
SheetOccurrence	enumeration	Indicates when the Sheet is to be produced and inserted into the set of output pages. **Allowed values are:** Always – Valid for "ErrorSheet" or "AccountingSheet". The Sheet is always produced at the end of the Job. End – Valid for "JobSheet" or "SeparatorSheet". The Sheet is produced at the end of the Job (for "JobSheet") or at the end of each copy of each Instance Document (for "SeparatorSheet"). OnError – Valid for "ErrorSheet". The Sheet is produced at the end of the Job when an error or warning occurs. Slip – Valid for "SeparatorSheet". The Sheet is produced between each copy of each Instance Document. Start – Valid for "JobSheet" or "SeparatorSheet". The Sheet is produced at the start of the Job (for "JobSheet") or at the start of each copy of each Instance Document (for "SeparatorSheet"). Both – Valid for "JobSheet" or "SeparatorSheet". The Sheet is produced at the beginning and end of the Job (for "JobSheet"s) or at the beginning and end of each copy of each Instance Document (for "SeparatorSheet"s). None – Valid for any @*SheetType*.
SheetType	enumeration	Identifies the type of Sheet. **Allowed values are:** AccountingSheet – A Sheet that reports accounting information for the Job. ErrorSheet – A Sheet that reports errors for the Job. JobSheet – A Sheet that delimits the Job. SeparatorSheet – A Sheet that delimits one copy (set) of the Job.
IDPFinishing ?	refelement	An ***IDPFinishing*** Element that describes the finishing options for the Job Sheet.
IDPLayout ?	element	This Element provides the details of how page contents will be imaged onto the Job Sheet.
MediaIntent ?	refelement	A ***MediaIntent*** Element. This Element describes the media to be used for the Job Sheets. This Element is ignored if a ***MediaSource*** Resource is present and can be honored. If ***MediaSource*** is absent or cannot be honored, this Element describes the intended media for the Job Sheets to allow the Device to select from among the available media.
MediaSource ?	refelement	Describes the source and physical orientation of the media to be used.

Overriding IDPrintingParams using Partitioning

IDPrintingParams MAY be overridden using Partitioning mechanisms as described in ▸ Section 3.10.5 Description of Partitioned Resources. Overrides MAY apply to a set of Instance Documents, set of copies of Instance Documents, or to a set of finished pages, output surfaces, Sheets of media in a personalized printing Job, or header or trailer insert Sheets added by a ***RunList***.

Note: If more than one override refers to the same content, the lowest level override takes precedence. The following list defines Partitioning precedence, from lowest to highest (i.e., the lower entries in the list take precedence):

- Job level Partitioning (*lowest priority*):
- "PartVersion", "Run", "SheetName", "Side", "RunTags"
- Page level Partitioning:
- "RunIndex"

- "SheetIndex"
- Instance Document level Partitioning (*highest priority*):
- "DocCopies"
- "DocIndex"
- "DocSheetIndex"
- "DocRunIndex"

Note: It is strongly discouraged to mix page-level Partitions and Instance Document-level Partitions. ***Cover*** Elements in ***IDPrintingParams*** are counted when calculating @*DocSheetIndex* or @*DocRunIndex*.

Example of a Partitioned IDPrinting Node

The following example shows how Partitioning can be used to describe a fairly complex example. Three color models (**ColorantControl** Partitions) are applied to a set of Sheets using the @*DocSheetIndex* key;

1 DeviceN:*DocSheetIndex* = "0" defines the cover;

2 ***DeviceCMYK*** @*DocSheetIndex* = "1" defines the first Sheet (non cover);

3 DeviceGray:*DocSheetIndex* = "2 ~ -1" defines all other Sheets;

The cover is selected from a different input tray using the @*Location* key. The same key is used to describe the **Media** in each tray.

```
<?xml version='1.0' encoding='utf-8' ?>
<JDF ID="HDM20010402140111" Type="IDPrinting" JobID="HDM20010402140111"
Status="Waiting" Version="1.2">
<ResourcePool>
<Media ID="Link0003" Class="Consumable" Locked="false" Status="Available"
Dimension="700 900" MediaType="Paper" PartIDKeys="Location">
<Media Weight="90" Location="Tray 1"/>
<Media Weight="120" Location="Tray 2"/>
</Media>
<RunList ID="Link0004" Class="Parameter" Locked="false" Status="Available"
PartIDKeys="Run">
<RunList Run="Run0005" Pages="0">
<LayoutElement>
<FileSpec URL="Cover.pdf"/>
</LayoutElement>
</RunList>
<RunList Run="Run0006" Pages="0 ~ 7">
<LayoutElement>
<FileSpec URL="File2.pdf"/>
</LayoutElement>
</RunList>
</RunList>
<IDPrintingParams ID="Link0008" Class="Parameter" Locked="false"
Status="Available">
<IDPLayout NumberUp="2 2"/>
<MediaSource MediaLocation="Tray 1">
<MediaRef rRef="Link0003"/>
</MediaSource>
<Cover CoverType="Front" FrontSide="true">
<IDPLayout NumberUp="1 1"/>
<MediaSource MediaLocation="Tray 2">
<MediaRef rRef="Link0003"/>
</MediaSource>
</Cover>
</IDPrintingParams>
<ColorantControl ID="Link0009" Class="Parameter" Locked="false"
Status="Available" PartIDKeys="DocSheetIndex">
<ColorantControl DocSheetIndex="0" ProcessColorModel="DeviceN"/>
<ColorantControl DocSheetIndex="1" ProcessColorModel="DeviceCMYK"/>
<ColorantControl DocSheetIndex="2 ~ -1" ProcessColorModel="DeviceGray"/>
</ColorantControl>
</ResourcePool>
<ResourceLinkPool>
<MediaLink rRef="Link0003" Usage="Input"/>
<RunListLink rRef="Link0004" Usage="Input"/>
<IDPrintingParamsLink rRef="Link0008" Usage="Input"/>
 <ColorantControlLink rRef="Link0009" Usage="Input"/>
</ResourceLinkPool>
```

</JDF>

N.7.12 Layout Deprecated Subelement

Note: ***Layout*** is still a valid Resource. The following sections from within ***Layout*** were deprecated and were deemed large enough to warrant moving them to this section.

N.7.12.1 Signature

Deprecated in JDF 1.3

This Element groups individual ***Sheet*** Resources into one ***Signature*** Subelement. In **JDF** 1.3 and beyond, ***Signature*** is represented as a Partition of ***Layout*** with ***Layout***/@*PartIDKeys* "SignatureName" set.

Table N.94: Signature Element

NAME	DATA TYPE	DESCRIPTION
Name ?	string	Unique name of the Signature. @*Name* is used for external reference to a Signature, as in a ***Part*** Element.
InsertSheet *	refelement	Specifies how to complete a Signature in an automated printing environment.
Media ? New in JDF 1.1	refelement	Describes the media to be used. Defaults to ***Layout***/***Media***.
MediaSource ? Deprecated in JDF 1.1	refelement	Describes the media to be used. Replaced by ***Media*** in **JDF** 1.1.
Sheet *	refelement	Resources that comprise the ***Signature***.

N.7.13 LongitudinalRibbonOperationParams

Deprecated in JDF 1.1.

This Resource provides the parameters of the **LongitudinalRibbonOperations** Process. It is defined as a list of Abstract @*LROperation* Elements.

Resource Properties

Resource Class: Parameter

Example Partition: "RibbonName", "SheetName", "SignatureName", "WebName"

Input of Processes: **LongitudinalRibbonOperations**

Table N.95: LongitudinalRibbonOperationParams Resource

NAME	DATA TYPE	DESCRIPTION
LROperation +	element	Abstract Element which is a placeholder for a longitudinal ribbon operation.

N.7.13.1 LROperation

Deprecated in JDF 1.1.

LROperation is an Abstract Element that describes the **LongitudinalRibbonOperations** Process. The defined instances (subclasses) of ***LROperation*** are ***LongFold***, ***LongGlue***, ***LongPerforate***, and ***LongSlit***. All instances of ***LROperation*** have the following common contents.

Table N.96: LROperation Element

NAME	DATA TYPE	DESCRIPTION
WorkingList = "0 1000000000"	NumberList	List of lengths of the @*Operation* to be performed in point. Entries with an odd position (first, third, etc.) in the list define an offset where the tool is inactive. Entries with an even position define a working length where the tool is on. The start position is the leading edge of the plate. If the sum of all entries is higher than the circumference of the press cylinder, the values exceeding the circumference are cropped. Counting always restarts at the leading edge. Default = "0 1000000000" (i.e., always on).
XOffset	double	Position of the tool for longitudinal action along the cylinder axis.

N.7.13.2 LongFold

Deprecated in JDF 1.1.

LongFold is derived from the Abstract Element ***LROperation*** and describes a longitudinal fold operation and has no further contents in addition to those of ***LROperation***.

N.7.13.3 LongGlue

Deprecated in JDF 1.1.

LongGlue is derived from the Abstract Element ***LROperation*** and describes a longitudinal gluing operation and has the following contents in addition to those of ***LROperation.***

Table N.97: LongGlue Element

NAME	DATA TYPE	DESCRIPTION
GlueBrand ?	string	Glue brand. Use only when @*Operation* = "Glue".
GlueType ?	Enumeration	If @*Operation* = "Glue", the listed values can be used: **Allowed values are:** ColdGlue Hotmelt PUR – Polyurethane
LineWidth ?	double	Width of the @*Operation* line.
MeltingTemperature ?	integer	Temperature needed for melting the glue (in degrees centigrade). Use only when @*GlueType* = "Hotmelt" and @*Operation* = "Glue".

N.7.13.4 LongPerforate

Deprecated in JDF 1.1.

LongPerforate is derived from the Abstract Element ***LROperation*** and describes a longitudinal gluing operation and has the following contents in addition to those of ***LROperation.***

Table N.98: LongPerforate Element

NAME	DATA TYPE	DESCRIPTION
TeethPerDimension ?	integer	If @*Operation* = "Perforate", the number of teeth in a given perforation extent is defined in teeth/point. *MicroPerforation* is defined by specifying a large number of teeth (n>1000).

N.7.13.5 LongSlit

Deprecated in JDF 1.1.

LongSlit is derived from the Abstract Element ***LROperation*** and describes a longitudinal cut operation and has no further contents in addition to those of ***LROperation.***

N.7.14 MediaSource

Deprecated in JDF 1.1

This Resource describes the source and physical orientation of the media to be used in **DigitalPrinting** or **IDPrinting**.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***DigitalPrintingParams***, ***IDPrintingParams***, ***InsertSheet***, ***Layout***, ***Sheet***, ***Tile***

Input of Processes: **DigitalPrinting**, **IDPrinting**

Table N.99: MediaSource Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
LeadingEdge ?	number	Specifies the size, in points, of the edge of the media that represents the scanline direction. If this Attribute is absent, the scanline direction is assumed to be along the x-axis of the "Dimension" parameter for the ***Media***.

Table N.99: MediaSource Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
MediaLocation ?	string	Identifies the location, such as a slot name or ID, of the media in the Device. If the media Resource is Partitioned by @*Location* (see also ▸ Section 3.10.6.4 Locations of PhysicalResources) there SHOULD be a match between one @*Location* Partition Key and this @*MediaLocation* value.
ManualFeed = "false"	boolean	Indicates whether the media will be fed manually. Default = "false"
Media ?	refelement	A ***Media*** Resource which identifies the media to be used. Only one of ***Component*** or ***Media*** SHOULD be specified.

N.7.15 NumberingParams

Deprecated in JDF 1.5

This Resource describes the parameters of stamping or applying variable marks in order to produce unique components (e.g., lottery notes, currency). One ***NumberingParams*** Element SHALL be defined per numbering machine.

Resource Properties

Resource Class: Parameter

Input of Processes: **Numbering**

Table N.100: NumberingParams Resource

NAME	DATA TYPE	DESCRIPTION
NumberingParam *	element	Set of parameters for one numbering machine

N.7.16 NumberingParam

*Table N.101: **NumberingParam** Element*

NAME	DATA TYPE	DESCRIPTION
Orientation	double	Rotation of the numbering machine in degrees. If @*Orientation* = "0", the top of the numbers is along the leading edge.
StartValue ?	string	First value of the numbering machine.
Step = "1"	integer	Number that specifies the difference between two subsequent numbers of the numbering machine.
XPosition	double	Position of the numbering machine along the printer axis.
YPosition	DoubleList	List of stamp positions, in points, starting from the leading edge.

N.7.17 OrderingParams

Deprecated in JDF 1.5

Attributes of the **Ordering** Process, which results in an acquisition.

Resource Properties

Resource Class: Parameter

Input of Processes: **Ordering**

Table N.102: OrderingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Amount	double	Amount of the ordered Resource.
Contact * New in JDF 1.1	refelement	Address and further information of the ***Contact*** responsible for this order.
Unit	string	Unit of measurement for @*Amount*.

Table N.102: OrderingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
Comment	element	***OrderingParams*** require a ***Comment*** element that contains a human-readable description of what to order.
Company ? Deprecated in JDF 1.1	refelement	Address and further information of the ***Company*** responsible for this order. Replaced with ***Contact***/***Company*** in **JDF** 1.1.

N.7.18 PackingParams

Deprecated in JDF 1.1

The ***PackingParams*** Resource has been deprecated in version 1.1 and beyond. It is replaced by the individual Resources used by the Processes defined in ▸ Section 6.6.4 Numbering and ▸ Section 6.6.5 Packaging Processes.

This Resource specifies the box packing parameters for a **JDF** Job, using information that identifies the type of package, the wrapping used, and the shape of the package. Note that this specifies packing for shipping only, not packing of items into custom boxes etc. Boxes are convenience packaging, and are not envisioned to be protection for shipping. Cartons perform this function. All quantities are specified as finished pieces per wrapped/boxed/carton or palletized package.

The model for packaging is that products are *wrapped* together, wrapped packages are placed in *boxes*, boxes are placed in *cartons*, and cartons are stacked on *pallets*.

Resource Properties

Resource Class: Parameter

Input of Processes: **Packing**

Table N.103: PackingParams Resource (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
BoxedQuantity ?	integer	How many units of *product* in a box.
BoxShape ?	shape	Describes the length, width and height of the box in points.
CartonQuantity ?	integer	How many units of *product* in a carton.
CartonShape ?	shape	Describes the length, width and height of the carton in points (e.g., 288 544 1012).
CartonMaxWeight ?	double	Maximum weight of an individual carton in kilograms.
CartonStrength ?	double	Strength of the carton in Newtons per square meter.
PalletQuantity ?	integer	Number of *product* per pallet
PalletSize ?	XYPair	Describes the length and width of the pallet in points (e.g., 3500 3500).
PalletMaxHeight ?	double	Maximum height of a loaded pallet in points.
PalletMaxWeight ?	double	Maximum weight of a loaded pallet in kilograms.
PalletType ?	enumeration	Type of pallet used. **Allowed values are::** 2Way – Two-way entry 4Way – Four-way entry Euro – Standard 1*1 m Euro pallet
PalletWrapping = "None"	enumeration	Wrapping of the completed pallet. **Allowed values are:** StretchWrap Banding None – The default.
WrappedQuantity ?	integer	Number of units of *product* per wrapped package.

Table N.103: PackingParams Resource (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
WrappingMaterial = "None"	name	Examples include: RubberBand ShrinkWrap PaperBand Polyethylene None – The default.

N.7.19 PlaceHolderResource

Deprecated in JDF 1.5

This Resource is used to link Process Group Nodes when the exact nature of interchange Resources is still unknown. In this way, a skeleton of Process networks can be constructed, with the ***PlaceHolderResource*** Resources serving as place holders in lieu of the appropriate Resources. This Resource needs no structure besides that provided in an Abstract Resource Element as it has no inherent value except as a stand-in for other Resources.

Resource Properties

Resource Class: PlaceHolder

Input of Processes: ***Process Group Nodes***

Output of Processes: ***Process Group Nodes***

Resource Structure

The Resource has no additional structure.

N.7.20 PlateCopyParams

Deprecated in JDF 1.1

This Resource specifies the parameters of the **FilmToPlateCopying** Process.

Resource Properties

Resource Class: Parameter

Input of Processes: **FilmToPlateCopying**

Table N.104: PlateCopyParams Resource

NAME	DATA TYPE	DESCRIPTION
Cycle ?	integer	Number of exposure light units to be used. The amount depends on the subject to be exposed.
Diffusion ?	enumeration	The diffusion foil setting. **Allowed values are:** On Off
Vacuum ?	double	Amount of vacuum pressure to be used. Measured in bars.

N.7.21 ProofingParams

Deprecated in JDF 1.2

This Resource specifies the settings needed for all proofing operations, including both “hard” or “soft” proofing, of color and imposition proofs.

Resource Properties

Resource Class: Parameter

Example Partition: "DocIndex", "RunIndex", "RunTags", "SheetName", "Side", "SignatureName"

Input of Processes: **Proofing, SoftProofing**

Table N.105: ProofingParams Resource

NAME	DATA TYPE	DESCRIPTION
ColorType ?	enumeration	Color quality of the proof. **Allowed values are:** Monochrome – Black and white. BasicColor – Color does not match precisely. This implies the absence of a color matching system. MatchedColor – Color is matched to the output of the press using a color matching system.
DisplayTraps = "false"	boolean	If "true", the trap networks are shown in the proof. Default = "false"
HalfTone = "false"	boolean	Specifies whether the proof is to emulate halftone screens. Default = "false"
ImageViewingStrategy = "NoImages"	string	Identifies which images will be displayed during the **SoftProofing** Process. **Allowed values are:** NoImages – Default value. OmitReference – Displays only images actually embedded in the file. UseProxies – Displays images embedded in the file and proxy versions of referenced data. UseReplacements – Displays embedded images plus the full resolution version of referenced images.
ManualFeed = "false" New in JDF 1.1	boolean	Indicates whether the media will be fed manually. Default = "false"
ProofRenderingIntent = "Perceptual" New in JDF 1.1	enumeration	Identifies the rendering intents associated with the proof. Values are ICC-defined rendering intent values: **Allowed values are:** Saturation Perceptual – The default. RelativeColorimetric AbsoluteColorimetric
ProofType = "None"	enumeration	Describes the type of the proof. **Allowed values are:** None – Default value. Not a proof or the type is unknown. Page – Page proof Imposition – Imposition proof.
Resolution ?	XYPair	Resolution of the output.
FileSpec ?	refelement	A **FileSpec** Resource pointing to an ICC profile that describes the proofer Device. The @*ResourceUsage* Attribute of the ***FileSpec*** SHALL be "ProoferProfile".
Media ?	refelement	Describes the media to be used.

N.7.22 RunList Deprecated Subelements

N.7.22.1 DynamicInput

Deprecated in JDF 1.4

Table N.106: DynamicInput Element (Sheet 1 of 2)

NAME	DATA TYPE	DESCRIPTION
Name ?	string	Label that SHALL match the @*ReplaceField* Attribute of the appropriate ***DynamicField*** Element

Table N.106: DynamicInput Element (Sheet 2 of 2)

NAME	DATA TYPE	DESCRIPTION
—	text	Defines the text string that is to be inserted as a replacement for the text defined in @*ReplaceField* of a ***DynamicField*** Element.

N.7.23 SaddleStitchingParams

This Resource provides the parameters of the **SaddleStitching** Process.

Deprecated in JDF 1.1

Resource Properties

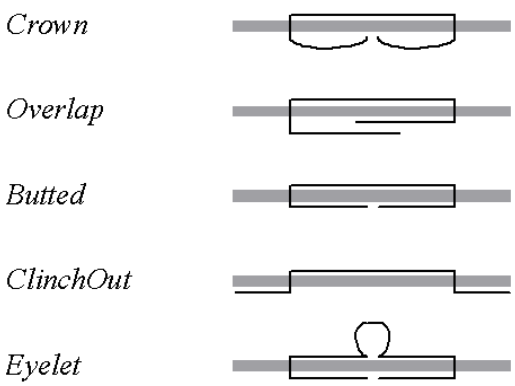
Resource Class: Parameter

Input of Processes: **SaddleStitching**

Table N.107: SaddleStitchingParams Resource

NAME	DATA TYPE	DESCRIPTION
NumberOfStitches	integer	The number of stitches that will be made.
StitchPositions ?	NumberList	Array containing the stitch positions along the saddle. The center of the stitch SHALL be specified, and the number of entries SHALL match the number given in the @*NumberOfStitches* Attribute.
StapleShape ?	enumeration	Shape of staples. **Allowed values are:** Crown Overlap Butted ClinchOut Eyelet **Note:** These values are displayed in ▸ Figure N-3: Staple shapes, below.
StitchWidth ?	double	Width of each stitch.
WireGauge ?	double	Gauge of the wire being used.
WireBrand ?	string	Brand of wire being used.

Figure N-3: Staple shapes



The Process coordinate system is defined as follows — The Y-axis is aligned with the binding edge, and increases from the registered edge to the edge opposite the registered edge. The X-axis, meanwhile, is aligned with the registered edge. It increases from the binding edge to the edge opposite the binding edge, which is the product front edge.

N.7.24 Sheet

Deprecated in JDF 1.3

This Resource provides a description of a Sheet, as well as the marks on that Sheet. In **JDF** 1.3 and beyond, Sheet is represented as a Partition of ***Layout*** with ***Layout***/@*PartIDKeys* "SheetName" set.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***InsertSheet***, ***Layout***

Example Partition: "SheetName".

Table N.108: Sheet Resource

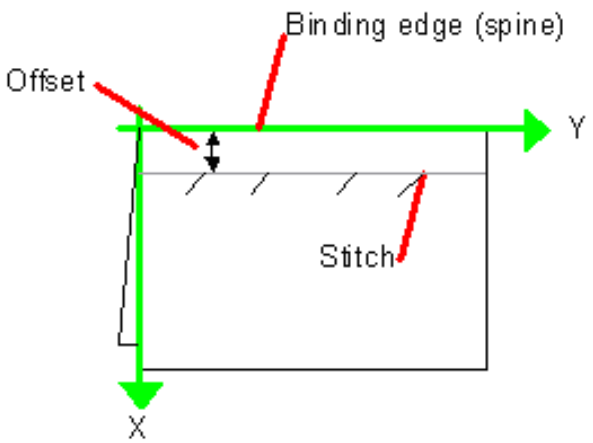
NAME	DATA TYPE	DESCRIPTION
LockOrigins = "false"	boolean	Determines the relationship of the coordinate systems for front and back surfaces. When "false", all contents for all surfaces are transformed into the first quadrant, in which the origin is at the lower left corner of the surface. When "true", contents for the front surface are imaged into the first quadrant (as above), but contents for the back surface are imaged into the second quadrant, in which the origin is at the lower right. This allows the front and back origins to be aligned even if the exact media size is unknown.
Name ? Clarified in JDF 1.2	string	Name of the Sheet. @*Name* SHALL be unique within a given ***Layout***. Name is used for external reference to a Sheet in, for example, a ***Part*** Element.
SurfaceContentsBox ?	rectangle	This box, specified in surface coordinate space, defines the area into which contents and marks will occur for all ***Surfaces*** in the ***Sheet***. CTMs for ***MarkObject*** or ***ContentObject*** Elements transform page contents or marks into this rectangle.
InsertSheet *	refelement	Specifies how to complete a Sheet in an automated printing environment.
Media ? New in JDF 1.1	refelement	Describes the media to be used. Defaults to ***Layout***/***Signature***/***Media***.
MediaSource ? Deprecated in JDF 1.1	refelement	Describes the media to be used. Replaced by ***Media*** in **JDF** 1.1.
Surface (Front) ?	refelement	Describes the front surface to be used. Two surfaces may be attached: one front surface and one back surface. The surface is defined by the @*Side* Attribute of the ***Surface*** Resource. ***Surface***/@*Side* SHALL be "Front".
Surface (Back) ?	refelement	Describes the back surface to be used. ***Surface***/@*Side* SHALL be "Back".

N.7.25 SideSewingParams

Deprecated in JDF 1.1

This Resource provides the parameters for the **SideSewing** Process. **SideSewing** is a special case of **ThreadSewing**. The Process coordinate system is defined in the following way: the Y-axis is aligned with the binding edge. It then increases from the registered edge to the edge opposite to the registered edge. The X-axis is aligned with the registered edge, which then increases from the binding edge to the edge opposite to the binding edge (i.e., the product front edge).

Figure N-4: Parameters and coordinate system used for side sewing



Resource Properties

Resource Class: Parameter

Input of Processes: **SideSewing**

Table N.109: SideSewingParams Resource

NAME	DATA TYPE	DESCRIPTION
NumberOfNeedles	integer	Specifies the number of needles to be used.
NeedlePositions ?	NumberList	Array containing the Y-coordinates of the needle positions. The number of entries SHALL match the number given in @*NumberOfNeedles*.
Offset	double	Specifies the distance between the stitch and the binding edge.
SewingPattern ?	enumeration	Specifies the sewing pattern to be used. **Allowed values are:** Normal Staggered CombinedStaggered
ThreadMaterial ?	enumeration	Specifies the thread material to be used. **Allowed values are:** Cotton Nylon Polyester
ThreadThickness ?	double	The thickness of the thread to be used.
ThreadBrand ?	string	The brand of thread to be used.

N.7.26 Surface

Deprecated in JDF 1.3

This Resource describes the marks on a Sheet surface. Up to two ***Surface*** Resources may be defined for a ***Sheet***. In **JDF** 1.3 and beyond, ***Surface*** is represented as a Partition of ***Layout*** with ***Layout***/@*PartIDKeys* "Side" set.

Resource Properties

Resource Class: Parameter

Resource referenced by: ***Sheet***

Example Partition: "Side". Otherwise it is strongly discouraged to Partition the ***Layout*** tree, including ***Surface***.

Table N.110: Surface Resource

NAME	DATA TYPE	DESCRIPTION
Side	enumeration	The side of the ***Sheet*** that the ***Surface*** describes. **Allowed values are:** Front Back
SurfaceContentsBox ?	rectangle	This rectangle provides the region of the surface into which the contents of ***ContentObject*** Elements and ***MarkObject***s are to be imaged. **Note:** The @*SurfaceContentsBox* also provides a translation for an object's @*CTM*.
PlacedObject *	element	Provides a list of the ***ContentObject*** and ***MarkObject*** Elements to be placed on to the surface. Contains the marks on the surface in rendering order. See the description that follows. **Note:** ***PlacedObject*** is not a container but an Abstract type.

List of Figures

Figure 1-1 Handling of Default Values of JDF Attributes. . . . 14
Figure 2-1 Example of JDF and JMF workflow interactions . . . 24
Figure 2-2 JDF tree structure . . . 25
Figure 2-3 Example of a hierarchical tree structure of JDF Nodes . . . 27
Figure 2-4 Example of a Process chain linked by Input Resources and Output Resources . . . 27
Figure 2-5 Standard coordinate system . . . 29
Figure 2-6 Relation between Resource and process coordinate systems . . . 29
Figure 2-7 Relation between Resource and process coordinate systems . . . 29
Figure 2-8 Layout of simple saddle stitched brochure (product example) . . . 33
Figure 2-9 Equation for Surface Coordinate System Transformations . . . 33
Figure 2-10 Surface coordinate system . . . 34
Figure 2-11 Press coordinate system used for Sheet-Fed Printing . . . 34
Figure 2-12 Press coordinate system used for Web Printing . . . 35
Figure 2-13 Coordinate systems after Folding (product example) . . . 35
Figure 2-14 Coordinate systems after Collecting (product example) . . . 36
Figure 2-15 Examples of Transformations and Coordinate Systems in JDF. . . . 37
Figure 2-16 Transforming a point (example) . . . 38
Figure 3-1 Any-Element (generic content) – a diagram of its structure . . . 43
Figure 3-2 JDF Node – a Diagram of its Structure . . . 49
Figure 3-3 Job hierarchy with Process, Process Group and Product Intent Nodes . . . 50
Figure 3-4 Combined Process Node dependencies . . . 55
Figure 3-5 ResourcePool and Abstract Resource Element – a diagram of the structure . . . 61
Figure 3-6 Nodes linked by a Resource . . . 67
Figure 3-7 ResourceLink Elements and ResourceRef Elements . . . 67
Figure 3-8 ResourceLinkPool and ResourceLink Element – a diagram of the structure . . . 69
Figure 3-9 AmountPool – a Diagram of its Structure . . . 74
Figure 3-10 Amount handling . . . 82
Figure 3-11 Workflow for splitting shared Input Resources . . . 103
Figure 3-12 Workflow for combining shared Output Resources . . . 103
Figure 3-13 Workflow for splitting independent Input Resources . . . 103
Figure 3-14 Workflow for combining independent Output Resources . . . 103
Figure 3-15 AuditPool and Abstract Audit Element – a diagram of the structure . . . 105
Figure 3-16 Specific Audit - a Diagram of its Structure . . . 107
Figure 4-1 Life Cycle of a JDF Node . . . 128
Figure 4-2 Example of a simple Process chain linked by Resources . . . 129
Figure 4-3 Example of a simple Process chain linked by Resources . . . 129
Figure 4-4 Example of a Pipe Resource Linking Two Processes via Pull . . . 132
Figure 4-5 Example of a Pipe Resource Linking Two Processes via Push . . . 132
Figure 4-6 Example of status transitions in case of overlapping Processing . . . 133
Figure 4-7 The spawning and merging mechanism and its phases . . . 137
Figure 4-8 JDF Node structure that requires Resource copying during spawning and merging . . . 139

Figure 4-9 Example for a JDF Node structure with nested spawning . . . 140
Figure 5-1 JMF Root Element – a diagram of its structure . . . 147
Figure 5-2 Interaction of Messages with a Subscription . . . 148
Figure 5-3 Interaction of Command and Acknowledge Messages . . . 154
Figure 5-4 Example of Reliable Signaling . . . 155
Figure 5-5 JMF QueueEntry Status Transition Diagram . . . 167
Figure 5-6 Effects of the global queue Messages on the queue Status . . . 168
Figure 5-7 Mechanism of a PipePull Message . . . 198
Figure 5-8 Mechanism of a PipePush Message . . . 198
Figure 5-9 Without UpdateJDF Message . . . 238
Figure 5-10 With UpdateJDF Message . . . 238
Figure 6-1 Imposition for Cut and Stack . . . 267
Figure 6-2 Worst case scenario for area coverage calculation . . . 273
Figure 6-3 Overview of Web Printing . . . 284
Figure 6-4 Bundle Creation . . . 289
Figure 6-5 Bundle Transport . . . 289
Figure 6-6 Combined Process with Feeding Process . . . 295
Figure 6-7 Input Components . . . 295
Figure 6-8 Output Component . . . 296
Figure 6-9 Gathering . . . 297
Figure 6-10 Print Roll . . . 303
Figure 6-11 Packaging Process Coordinate System . . . 313
Figure 7-1 Roll-up Display . . . 327
Figure 7-2 Structure of a normal hardcover book . . . 335
Figure 7-3 Structure of a padded hardcover book . . . 335
Figure 7-4 Structure of a book with GlueProcedure = "SideOnly" (Layflat) . . . 339
Figure 8-1 BinderySignature Trims . . . 383
Figure 8-2 WebCellAlignment, Example 1 . . . 384
Figure 8-3 WebCellAlignment Example 2 . . . 384
Figure 8-4 WebCellAlignment Example 3 . . . 385
Figure 8-5 Definition of margins in SignatureCell . . . 389
Figure 8-6 Tightbacking for Block Preparation . . . 389
Figure 8-7 Rounding and Backing for Block Preparation . . . 389
Figure 8-8 Folding examples for some values of BoxFoldAction/@Action . . . 393
Figure 8-9 BoxFoldingType Attribute for values of Type00, Type01 and Type02 . . . 393
Figure 8-10 BoxFoldingType Attribute for values of Type03, Type04 and Type10 . . . 393
Figure 8-11 BoxFoldingType Attribute for values of Type 11, Type12 and Type13 . . . 394
Figure 8-12 BoxFoldingType Attribute for values of Type15 and Type20 . . . 394
Figure 8-13 Box packing . . . 396
Figure 8-14 BundlingParams Coordinate System . . . 400
Figure 8-15 CaseMakingParams . . . 402
Figure 8-16 Parameters and coordinate system for CasingIn . . . 403
Figure 8-17 Parameters used for channel binding . . . 404

Figure 8-18 Coordinate systems used for collecting . . . 406
Figure 8-19 Component – terms and definitions . . . 426
Figure 8-20 Orientation of the Finished Product on the Roll . . . 430
Figure 8-21 Parameters and coordinate system for cover application . . . 443
Figure 8-22 Parameters and coordinate system for glue application . . . 443
Figure 8-23 Definition of the PlatePosition Attribute on a newspaper-Web Press . . . 448
Figure 8-24 Example of a single physical section of eight pages . . . 449
Figure 8-25 Basic Shape for RepeatDesc/@LayoutStyle Examples . . . 462
Figure 8-26 RepeatDesc/@LayoutStyle = "StraightNest" . . . 462
Figure 8-27 RepeatDesc/@LayoutStyle = "Reverse2ndRow" . . . 463
Figure 8-28 RepeatDesc/@LayoutStyle = "Reverse2ndRowAligned" . . . 463
Figure 8-29 RepeatDesc/@LayoutStyle = "Reverse2ndColumn" . . . 464
Figure 8-30 RepeatDesc/@LayoutStyle = "Reverse2ndColumnAligned" . . . 464
Figure 8-31 RepeatDesc/@GutterX2 and @GutterY2: Secondary Gutters . . . 465
Figure 8-32 Parameters and coordinate system used for end-Sheet gluing . . . 473
Figure 8-33 Names of the reference edges of a Sheet in the FoldingParams Resource . . . 483
Figure 8-34 Fold catalog part 1 . . . 485
Figure 8-35 Fold catalog part 2 . . . 486
Figure 8-36 Coordinate system used for Gathering . . . 488
Figure 8-37 Parameters and coordinate system for glue application . . . 489
Figure 8-38 Parameters and coordinate system used for Inserting . . . 509
Figure 8-39 Setup of the Jacketing Machinery . . . 515
Figure 8-40 Parameters and coordinate system for jacketing . . . 515
Figure 8-41 Shingling for Stripping . . . 521
Figure 8-42 Shingling for Stripping – Details . . . 521
Figure 8-43 RelativeBox including margins . . . 536
Figure 8-45 Anchor with 1.5 Scaling and Rotation of 90° Clockwise . . . 541
Figure 8-44 Anchor with No Scaling and Rotation of 90° Clockwise . . . 541
Figure 8-46 Diagram of a 4-up cross-folded saddle-stitch imposition with vertical gutter creep . . . 568
Figure 8-47 Diagram of a step-and-repeat 2-up saddle-stitch imposition with vertical spine gutter creep . . . 568
Figure 8-48 Parameters used for channel binding . . . 574
Figure 8-49 Paper Roll with some Roll-specific Information . . . 585
Figure 8-50 Relief and Floor Thickness for a Flexo Plate or Flexo Sleeve . . . 585
Figure 8-51 Types of Interlocks for Flexo Sleeve . . . 585
Figure 8-52 TabSetCollationOrder Attribute Values . . . 586
Figure 8-53 Diagram of a Single Bank of Tabs . . . 587
Figure 8-54 Inside Loss, Outside Gain . . . 588
Figure 8-55 PRGroup – a diagram of its structure . . . 617
Figure 8-56 PrintRollingParams Coordinate System . . . 627
Figure 8-57 ShapeTemplate Example 1 . . . 656
Figure 8-58 ShapeTemplate Example 2 . . . 657
Figure 8-59 ShapeTemplate Example 3 . . . 657

Figure 8-60 Parameters and coordinate systems for the SpinePreparation Process 662
Figure 8-61 Parameters and coordinate system for the SpineTaping Process 663
Figure 8-62 Stacking Layers 663
Figure 8-63 Pile Patterns 663
Figure 8-64 Odd count handling for a Bundle 664
Figure 8-65 Odd count handling for a Layer 664
Figure 8-66 Staple shapes 668
Figure 8-67 Parameters and coordinate system used for saddle stitching 668
Figure 8-68 Parameters and coordinate system used for Stitching 669
Figure 8-69 Stitching Coordinate System for StitchOrigin Values 669
Figure 8-70 Strapped Bundle 672
Figure 8-71 Strapped Bundle with Sub-bundles 672
Figure 8-72 Shingling for Stripping 675
Figure 8-73 Shingling for Stripping – Details 675
Figure 8-74 RelativeBox including margins 676
Figure 8-75 Definition of margins in StripCellParams 678
Figure 8-76 Parameters and coordinate system used for thread sewing 683
Figure 8-77 Parameters and coordinate system used for side sewing 683
Figure 8-78 Roll Stand 685
Figure 8-79 Parameters and coordinate system used for trimming 692
Figure 9-1 Anchor with No Scaling and No Rotation 715
Figure 9-2 Anchor with No Scaling and Rotation of 90° Clockwise 716
Figure 9-3 Anchor with 1.5 Scaling and Rotation of 90° Clockwise 716
Figure 9-4 Hole line parameters 726
Figure 9-5 Line hole punching for multiple webs 726
Figure 9-6 RegisterRibbon lengths and coordinate system for BlockPreparation 750
Figure 10-1 Parameter space in Device capabilities 755
Figure 10-2 DeviceCap – a diagram of its structure 757
Figure 10-3 Abstract State Element – a diagram of its structure 765
Figure 10-4 macro Element – a diagram of its structure 778
Figure 10-5 Abstract Term Element – a diagram of its structure 782
Figure 10-6 Abstract Evaluation Element – a diagram of its structure 785
Figure 11-1 Life Cycle of a Process and Queue Entry 818
Figure 11-2 Example of NodeStatus and resource amount in case of Overlapping Processing 819
Figure 11-3 Example of Exchange of Certificates 824
Figure 11-4 Interaction of Messages with a Subscription 825
Figure 11-5 Example of Reliable Signaling 826
Figure C-1 Fold Catalog Part 1 862
Figure C-2 Fold Catalog Part 2 863
Figure J-1 Legend for Interpreting Diagrams 899
Figure O-1 Example of the spawning and merging of independent Jobs 977
Figure O-2 Parameters and coordinate system for glue application 1001
Figure O-3 Staple shapes 1019

Figure 0-4 Parameters and coordinate system used for side sewing 1020

Appendix P

List of Tables

Table 4.1 Callout Icon Usage xxix
Table 1.1 CIP4 XJDF/XJMF Schema Locations 3
Table 1.2 Modification Notes 6
Table 1.3 Cardinality Symbols 6
Table 1.4 Template for Element Descriptions 8
Table 1.5 Glossary 8
Table 1.6 Conformance Terminology 13
Table 1.7 JDF Data Types 16
Table 1.8 Units Used in JDF XJDF 18
Table 2.1 Information contained in JDF Nodes, arranged numerically 25
Table 2.2 Information contained in JDF Nodes, arranged by group 26
Table 2.3 Data types for specifying coordinates and transformation 30
Table 2.4 Matrices and Orientation values for describing the orientation of a Component 32
Table 2.5 JDF Processes used for the production of the simple brochure 33
Table 3.1 Any Element (generic content) 41
Table 3.2 Definition of "XXX" 43
Table 3.3 Behavior for Activation Values in ▸ Table 3.4 JDF XJDF Node 44
Table 3.4 JDF XJDF Node 44
Table 3.5 AncestorPool Element 56
Table 3.6 Ancestor Element 56
Table 3.7 ResourcePool Element 58
Table 3.8 Abstract Resource Element 58
Table 3.9 SourceResource Element 60
Table 3.10 ProductList Element 61
Table 3.11 ResourceSet Element 62
Table 3.12 Dependent Element 63
Table 3.13 Abstract Parameter Resource Element 63
Table 3.14 Abstract PhysicalResource Element 64
Table 3.15 Location Element 65
Table 3.16 ResourceLinkPool Element 66
Table 3.17 ResourceLink Element 69
Table 3.18 ProcessUsage Attribute Values 72
Table 3.19 AmountPool Element 73
Table 3.20 PartAmount Element 74
Table 3.21 Lot Element 77
Table 3.22 Abstract ResourceElement 78
Table 3.23 Abstract ResourceRef Element 78

Table 3.24 Example of actual amount and amount handling . . . 82
Table 3.25 Identical Element . . . 88
Table 3.26 Partitionable Resource Element . . . 91
Table 3.27 Part Element . . . 92
Table 3.28 Condition Attribute Values . . . 97
Table 3.29 PartUsage Attribute examples . . . 100
Table 3.30 AuditPool Element . . . 104
Table 3.31 Abstract Audit Element . . . 105
Table 3.32 List of Audit Elements . . . 106
Table 3.33 AuditCreated Audit Element . . . 107
Table 3.34 Deleted Audit Element . . . 108
Table 3.35 Merged Audit Element . . . 108
Table 3.36 Modified Audit Element . . . 108
Table 3.37 AuditNotification Element . . . 109
Table 3.38 Notification Audit Element . . . 109
Table 3.39 PhaseTime Audit Element . . . 110
Table 3.40 Activity Element . . . 111
Table 3.41 ModulePhase Element . . . 111
Table 3.42 AuditProcessRun Element . . . 113
Table 3.43 ProcessRun Audit Element . . . 113
Table 3.44 ProcessRun Element . . . 113
Table 3.45 AuditResource Element . . . 114
Table 3.46 ResourceAudit Audit Element . . . 114
Table 3.47 Spawned Audit Element . . . 118
Table 3.48 AuditStatus Element . . . 119
Table 3.49 Excerpt from TrappingParams . . . 122
Table 4.1 Examples of Resource and Process states in the case of simple Process routing . . . 130
Table 4.2 Examples of Partitioning across multiple Resources . . . 130
Table 4.3 Actions generated when a dynamic-pipe buffer passes various levels . . . 134
Table 4.4 Event Sequence in Digital Finishing . . . 134
Table 5.1 JMF XJMF Element . . . 143
Table 5.2 Message Element . . . 145
Table 5.3 Header . . . 145
Table 5.4 Query Message Element . . . 148
Table 5.5 Subscription Element . . . 149
Table 5.6 Command Message Element Family . . . 150
Table 5.7 Signal Message Element Family . . . 151
Table 5.8 Trigger Element . . . 151
Table 5.9 ChangedPath Element . . . 152
Table 5.10 Response Message Element Family . . . 153
Table 5.11 Acknowledge Message Element . . . 154
Table 5.12 Registration Message Element . . . 155
Table 5.13 Subscription Element . . . 156

Table 5.14 ObservationTarget Element . . . 157
Table 5.15 Template for Message tables . . . 159
Table 5.16 List of JMF Messages . . . 159
Table 5.17 List of XJMF Messages . . . 161
Table 5.18 Messages for events and capabilities . . . 162
Table 5.19 Messages to query/affect a Job, Device or Controller . . . 163
Table 5.20 Messages for Control of Dynamic Pipes . . . 164
Table 5.21 PipeParams Element . . . 164
Table 5.22 Messages for queue entry handling . . . 165
Table 5.23 Status Transitions for QueueEntry Handling Messages . . . 166
Table 5.24 Messages for global handling of queues . . . 168
Table 5.25 Definition of the Queue Status Attribute Values . . . 168
Table 5.26 Queue Element . . . 169
Table 5.27 QueueEntry Element . . . 170
Table 5.28 QueueEntryDef Element . . . 171
Table 5.29 QueueFilter Element . . . 171
Table 5.30 Messages for Gang Jobs . . . 173
Table 5.31 AbortQueueEntry Message . . . 174
Table 5.32 AbortQueueEntryParams Element . . . 175
Table 5.33 CloseQueue Message . . . 175
Table 5.34 FlushQueue Command Message . . . 176
Table 5.35 FlushQueueParams Element . . . 176
Table 5.36 FlushQueue Query Message . . . 176
Table 5.37 FlushQueueInfo Element . . . 176
Table 5.38 FlushResources Command . . . 177
Table 5.39 FlushResources Query . . . 177
Table 5.40 FlushResourceParams Element . . . 177
Table 5.41 Contents of the ForceGang Command Message . . . 178
Table 5.42 CommandForceGang Message . . . 178
Table 5.43 GangCmdFilter Element . . . 178
Table 5.44 ResponseForceGang Message . . . 178
Table 5.45 GangStatus Message . . . 178
Table 5.46 CommandGangStatus Message . . . 178
Table 5.47 GangQuFilter Element . . . 179
Table 5.48 ResponseGangStatus Message . . . 179
Table 5.49 GangInfo Element . . . 179
Table 5.50 HoldQueue Message . . . 179
Table 5.51 HoldQueueEntry Message . . . 180
Table 5.52 HoldQueueEntryParams Element . . . 180
Table 5.53 KnownDevices Message . . . 180
Table 5.54 QueryKnownDevices Message . . . 181
Table 5.55 DeviceFilter Element . . . 181
Table 5.56 ResponseKnownDevices Message . . . 182

Table 5.57 DeviceList Element . . . 182
Table 5.58 SignalKnownDevices Message . . . 182
Table 5.59 KnownMessages Message . . . 182
Table 5.60 QueryKnownMessages Message . . . 183
Table 5.61 KnownMsgQuParams Element . . . 183
Table 5.62 ResponseKnownMessages Message . . . 183
Table 5.63 MessageService Element . . . 183
Table 5.64 KnownSubscriptions Message . . . 185
Table 5.65 QueryKnownSubscriptions Message . . . 185
Table 5.66 SubscriptionFilter Element . . . 185
Table 5.67 ResponseKnownSubscriptions Message . . . 186
Table 5.68 SubscriptionInfo Element . . . 186
Table 5.69 SignalKnownSubscriptions Message . . . 187
Table 5.70 ModifyNode Command . . . 187
Table 5.71 ModifyNode Signal . . . 187
Table 5.72 ModifyNodeCmdParams Element . . . 188
Table 5.73 NewComment Element . . . 188
Table 5.74 CommandModifyQueueEntry Message . . . 188
Table 5.75 ModifyQueueEntryParams Element . . . 189
Table 5.76 Operation Attribute Values . . . 189
Table 5.77 ResponseModifyQueueEntry Message . . . 190
Table 5.78 NewJDF Query Message . . . 191
Table 5.79 NewJDFQuParams Element . . . 191
Table 5.80 NewJDF Command Message . . . 191
Table 5.81 NewJDFCmdParams Element . . . 191
Table 5.82 IDInfo Element . . . 191
Table 5.83 Notification Signal . . . 192
Table 5.84 QueryNotification Message . . . 192
Table 5.85 NotificationFilter Element . . . 193
Table 5.86 ResponseNotification Message . . . 193
Table 5.87 SignalNotification Message . . . 194
Table 5.88 Notification Element . . . 194
Table 5.89 Event Element . . . 195
Table 5.90 Milestone Element . . . 195
Table 5.91 OpenQueue Message . . . 195
Table 5.92 PipeClose Message . . . 196
Table 5.93 CommandPipeControl Message . . . 196
Table 5.94 PipeParams Element . . . 196
Table 5.95 ResponsePipeControl Message . . . 197
Table 5.96 PipePause Message . . . 197
Table 5.97 PipePull Message . . . 197
Table 5.98 PipePush Message . . . 198
Table 5.99 Messages for global handling of queues . . . 199

Table 5.100 Definition of the Queue Status Attribute Values . . . 199
Table 5.101 QueueStatus Message . . . 199
Table 5.102 QueryQueueStatus Message . . . 199
Table 5.103 QueueStatusParams Element . . . 199
Table 5.104 ResponseQueueStatus Message . . . 200
Table 5.105 SignalQueueStatus Message . . . 200
Table 5.106 Queue Element . . . 200
Table 5.107 RemoveQueueEntry Message . . . 201
Table 5.108 RemoveQueueEntryParams Element . . . 201
Table 5.109 RequestForAuthentication Command Message . . . 202
Table 5.110 AuthenticationCmdParams Element . . . 202
Table 5.111 Certificate Element . . . 202
Table 5.112 AuthenticationResp Element . . . 203
Table 5.113 RequestForAuthentication Query Message . . . 203
Table 5.114 AuthenticationQuParams Element . . . 203
Table 5.115 RequestQueueEntry Message . . . 205
Table 5.116 CommandRequestQueueEntry Message . . . 205
Table 5.117 RequestQueueEntryParams Element . . . 205
Table 5.118 ResponseRequestQueueEntry Message . . . 206
Table 5.119 Resource Query Message . . . 206
Table 5.120 QueryResource Message . . . 206
Table 5.121 ResourceQuParams Element . . . 206
Table 5.122 Resource Command Message . . . 210
Table 5.123 CommandResource Message . . . 210
Table 5.124 ResourceCmdParams Element . . . 211
Table 5.125 ResponseResource Message . . . 213
Table 5.126 ResourceInfo Element . . . 214
Table 5.127 SignalResource Message . . . 217
Table 5.128 ResourcePull Message . . . 217
Table 5.129 ResourcePullParams Element . . . 218
Table 5.130 Status Transitions for ResubmitQueueEntry Handling Messages . . . 219
Table 5.131 ResubmitQueueEntry Message . . . 219
Table 5.132 CommandResubmitQueueEntry Message . . . 219
Table 5.133 ResubmissionParams Element . . . 220
Table 5.134 ResponseResubmitQueueEntry Message . . . 220
Table 5.135 ResumeQueue Message . . . 220
Table 5.136 ResumeQueueEntry Message . . . 221
Table 5.137 ResumeQueueEntryParams Element . . . 221
Table 5.138 ReturnQueueEntry Message . . . 221
Table 5.139 CommandReturnQueueEntry Message . . . 221
Table 5.140 ReturnQueueEntryParams Element . . . 222
Table 5.141 ResponseReturnQueueEntry Message . . . 222
Table 5.142 SetQueueEntryPosition Message . . . 222

Table 5.143 QueueEntryPosParams Element . . . 222
Table 5.144 SetQueueEntryPriority Message . . . 223
Table 5.145 QueueEntryPriParams Element . . . 223
Table 5.146 ShutDown Message . . . 223
Table 5.147 CommandShutDown Message . . . 224
Table 5.148 ShutDownCmdParams Element . . . 224
Table 5.149 ResponseShutDown Message . . . 224
Table 5.150 Status Message . . . 224
Table 5.151 QueryStatus Message . . . 224
Table 5.152 StatusQuParams Element . . . 225
Table 5.153 ResponseStatus Message . . . 226
Table 5.154 DeviceInfo Element . . . 226
Table 5.155 Activity Element . . . 228
Table 5.156 JobPhase Element . . . 228
Table 5.157 ModuleStatus Element . . . 230
Table 5.158 SignalStatus Message . . . 231
Table 5.159 StopPersistentChannel Message . . . 232
Table 5.160 CommandStopPersistentChannel Message . . . 232
Table 5.161 StopPersChParams Element . . . 232
Table 5.162 ResponseStopPersistentChannel Message . . . 233
Table 5.163 SubmissionMethods Message . . . 233
Table 5.164 SubmissionMethods Element . . . 233
Table 5.165 Status Transitions for QueueEntry Handling Messages . . . 234
Table 5.166 SubmitQueueEntry Message . . . 234
Table 5.167 CommandSubmitQueueEntry Message . . . 235
Table 5.168 QueueSubmissionParams Element . . . 235
Table 5.169 ResponseSubmitQueueEntry Message . . . 237
Table 5.170 SuspendQueueEntry Message . . . 237
Table 5.171 SuspendQueueEntryParams Element . . . 237
Table 5.172 UpdateJDF Command . . . 238
Table 5.173 UpdateJDF Signal . . . 238
Table 5.174 UpdateJDFCmdParams Element . . . 239
Table 5.175 CreateLink Element . . . 239
Table 5.176 CreateResource Element . . . 239
Table 5.177 MoveResource Element . . . 239
Table 5.178 RemoveLink Element . . . 240
Table 5.179 WakeUp Message . . . 240
Table 5.180 CommandWakeUp Message . . . 241
Table 5.181 WakeUpCmdParams Element . . . 241
Table 5.182 ResponseWakeUp Message . . . 241
Table 6.1 Template for Input Resources . . . 244
Table 6.2 Generic Input Resources . . . 244
Table 6.3 Template for Output Resources . . . 245

Table 6.4 Approval – Input Resources . . . 245
Table 6.5 Approval – Output Resources . . . 245
Table 6.6 Buffer – Input Resources . . . 246
Table 6.7 Buffer – Output Resources . . . 246
Table 6.8 Combine – Input Resources . . . 246
Table 6.9 Combine – Output Resources . . . 246
Table 6.10 Delivery – Input Resources . . . 247
Table 6.11 Delivery – Output Resources . . . 247
Table 6.12 ManualLabor – Input Resources . . . 247
Table 6.13 ManualLabor – Output Resources . . . 247
Table 6.14 QualityControl – Input Resources . . . 248
Table 6.15 QualityControl – Output Resources . . . 248
Table 6.16 ResourceDefinition – Input Resources . . . 248
Table 6.17 ResourceDefinition – Output Resources . . . 248
Table 6.18 Split – Input Resources . . . 248
Table 6.19 Split – Output Resources . . . 248
Table 6.20 Verification – Input Resources . . . 249
Table 6.21 Verification – Output Resources . . . 249
Table 6.22 AssetListCreation – Input Resources . . . 250
Table 6.23 AssetListCreation – Output Resources . . . 250
Table 6.24 Bending – Input Resources . . . 250
Table 6.25 Bending – Output Resources . . . 250
Table 6.26 ColorCorrection – Input Resources . . . 251
Table 6.27 ColorCorrection – Output Resources . . . 251
Table 6.28 ColorSpaceConversion – Input Resources . . . 251
Table 6.29 ColorSpaceConversion – Output Resources . . . 251
Table 6.30 ContactCopying – Input Resources . . . 252
Table 6.31 ContactCopying – Output Resources . . . 252
Table 6.32 ContoneCalibration – Input Resources . . . 252
Table 6.33 ContoneCalibration – Output Resources . . . 252
Table 6.34 CylinderLayoutPreparation – Input Resources . . . 253
Table 6.35 CylinderLayoutPreparation – Output Resources . . . 253
Table 6.36 DieDesign – Input Resources . . . 253
Table 6.37 DieDesign – Output Resources . . . 253
Table 6.38 DieLayoutProduction – Input Resources . . . 254
Table 6.39 DieLayoutProduction – Output Resources . . . 254
Table 6.40 DigitalDelivery – Input Resources . . . 257
Table 6.41 DigitalDelivery – Output Resources . . . 258
Table 6.42 ImageEnhancement – Input Resources . . . 258
Table 6.43 ImageEnhancement – Output Resources . . . 258
Table 6.44 ImageReplacement – Input Resources . . . 258
Table 6.45 ImageReplacement – Output Resources . . . 258
Table 6.46 ImageSetting – Input Resources . . . 259

Table 6.47 ImageSetting – Output Resources . . . 259
Table 6.48 Imposition – Input Resources . . . 259
Table 6.49 Imposition – Output Resources . . . 260
Table 6.50 Glossary for Automated Imposition . . . 260
Table 6.51 Variables for Automated Imposition . . . 263
Table 6.52 InkZoneCalculation – Input Resources . . . 269
Table 6.53 InkZoneCalculation – Output Resources . . . 269
Table 6.54 Interpreting – Input Resources . . . 269
Table 6.55 Interpreting – Output Resources . . . 270
Table 6.56 LayoutElementProduction – Input Resources . . . 270
Table 6.57 LayoutElementProduction – Output Resources . . . 270
Table 6.58 LayoutPreparation – Input Resources . . . 270
Table 6.59 LayoutPreparation – Output Resources . . . 271
Table 6.60 LayoutShifting – Input Resources . . . 271
Table 6.61 LayoutShifting – Output Resources . . . 271
Table 6.62 PageAssigning – Input Resources . . . 271
Table 6.63 PageAssigning – Output Resources . . . 271
Table 6.64 PDFToPSConversion – Input Resources . . . 272
Table 6.65 PDFToPSConversion – Output Resources . . . 272
Table 6.66 PDLCreation – Input Resources . . . 272
Table 6.67 PDLCreation – Output Resources . . . 272
Table 6.68 Preflight – Input Resources . . . 272
Table 6.69 Preflight – Output Resources . . . 273
Table 6.70 PreviewGeneration – Input Resources . . . 274
Table 6.71 PreviewGeneration – Output Resources . . . 274
Table 6.72 PSToPDFConversion – Input Resources . . . 275
Table 6.73 PSToPDFConversion – Output Resources . . . 275
Table 6.74 RasterReading – Input Resources . . . 275
Table 6.75 RasterReading – Output Resources . . . 275
Table 6.76 Rendering – Input Resources . . . 276
Table 6.77 Rendering – Output Resources . . . 276
Table 6.78 Scanning – Input Resources . . . 277
Table 6.79 Scanning – Output Resources . . . 277
Table 6.80 Screening – Input Resources . . . 277
Table 6.81 Screening – Output Resources . . . 277
Table 6.82 Separation – Input Resources . . . 278
Table 6.83 Separation – Output Resources . . . 278
Table 6.84 ShapeDefProduction – Input Resources . . . 278
Table 6.85 ShapeDefProduction – Output Resources . . . 278
Table 6.86 SheetOptimizing – Input Resources . . . 279
Table 6.87 SheetOptimizing – Output Resources . . . 279
Table 6.88 Stripping – Input Resources . . . 280
Table 6.89 Stripping – Output Resources . . . 280

Table 6.90 Tiling – Input Resources. . . . 282
Table 6.91 Tiling – Output Resources . . . 282
Table 6.92 Trapping – Input Resources . . . 283
Table 6.93 Trapping – Output Resources . . . 283
Table 6.94 ConventionalPrinting – Input Resources. . . . 284
Table 6.95 ConventionalPrinting – Output Resources . . . 285
Table 6.96 DigitalPrinting – Input Resources . . . 286
Table 6.97 DigitalPrinting – Output Resources . . . 286
Table 6.98 Varnishing – Input Resources . . . 287
Table 6.99 Varnishing – Output Resources . . . 287
Table 6.100 BlockPreparation – Input Resources . . . 288
Table 6.101 BlockPreparation – Output Resources . . . 288
Table 6.102 BoxFolding – Input Resources . . . 288
Table 6.103 BoxFolding – Output Resources . . . 288
Table 6.104 BoxPacking – Input Resources . . . 288
Table 6.105 BoxPacking – Output Resources . . . 289
Table 6.106 Bundling – Input Resources . . . 289
Table 6.107 Bundling – Output Resources. . . . 290
Table 6.108 CaseMaking – Input Resources . . . 290
Table 6.109 CaseMaking – Output Resources . . . 290
Table 6.110 CasingIn – Input Resources . . . 290
Table 6.111 CasingIn – Output Resources . . . 291
Table 6.112 ChannelBinding – Input Resources . . . 291
Table 6.113 ChannelBinding – Output Resources . . . 291
Table 6.114 CoilBinding – Input Resources . . . 291
Table 6.115 CoilBinding – Output Resources. . . . 291
Table 6.116 Collecting – Input Resources . . . 292
Table 6.117 Collecting – Output Resources . . . 292
Table 6.118 CoverApplication – Input Resources . . . 292
Table 6.119 CoverApplication – Output Resources. . . . 292
Table 6.120 Creasing – Input Resources. . . . 293
Table 6.121 Creasing – Output Resources . . . 293
Table 6.122 Cutting – Input Resources . . . 293
Table 6.123 Cutting – Output Resources . . . 293
Table 6.124 DieMaking – Input Resources. . . . 294
Table 6.125 DieMaking – Output Resources . . . 294
Table 6.126 Embossing – Input Resources . . . 294
Table 6.127 Embossing – Output Resources. . . . 294
Table 6.128 EndSheetGluing – Input Resources . . . 295
Table 6.129 EndSheetGluing – Output Resources . . . 295
Table 6.130 Feeding – Input Resources . . . 296
Table 6.131 Feeding – Output Resources . . . 296
Table 6.132 Folding – Input Resources . . . 296

Table 6.133 Folding – Output Resources . . . 296
Table 6.134 Gathering – Input Resources . . . 297
Table 6.135 Gathering – Output Resources . . . 297
Table 6.136 Gluing – Input Resources . . . 297
Table 6.137 Gluing – Output Resources . . . 297
Table 6.138 HeadBandApplication – Input Resources . . . 298
Table 6.139 HeadBandApplication – Output Resources . . . 298
Table 6.140 HoleMaking – Input Resources . . . 298
Table 6.141 HoleMaking – Output Resources . . . 298
Table 6.142 Inserting – Input Resources . . . 299
Table 6.143 Inserting – Output Resources . . . 299
Table 6.144 Jacketing – Input Resources . . . 299
Table 6.145 Jacketing – Output Resources . . . 299
Table 6.146 Labeling – Input Resources . . . 300
Table 6.147 Labeling – Output Resources . . . 300
Table 6.148 Laminating – Input Resources . . . 300
Table 6.149 Laminating – Output Resources . . . 300
Table 6.150 LooseBinding – Input Resources . . . 301
Table 6.151 LooseBinding – Output Resources . . . 301
Table 6.152 Palletizing – Input Resources . . . 302
Table 6.153 Palletizing – Output Resources . . . 302
Table 6.154 Perforating – Input Resources . . . 302
Table 6.155 Perforating – Output Resources . . . 302
Table 6.156 PlasticCombBinding – Input Resources . . . 303
Table 6.157 PlasticCombBinding – Output Resources . . . 303
Table 6.158 PrintRolling – Input Resources . . . 303
Table 6.159 PrintRolling – Output Resources . . . 303
Table 6.160 RingBinding – Input Resources . . . 304
Table 6.161 RingBinding – Output Resources . . . 304
Table 6.162 ShapeCutting – Input Resources . . . 304
Table 6.163 ShapeCutting – Output Resources . . . 304
Table 6.164 Shrinking – Input Resources . . . 304
Table 6.165 Shrinking – Output Resources . . . 305
Table 6.166 SpinePreparation – Input Resources . . . 305
Table 6.167 SpinePreparation – Output Resources . . . 305
Table 6.168 SpineTaping – Input Resources . . . 305
Table 6.169 SpineTaping – Output Resources . . . 306
Table 6.170 Stacking – Input Resources . . . 306
Table 6.171 Stacking – Output Resources . . . 306
Table 6.172 StaticBlocking – Input Resources . . . 306
Table 6.173 StaticBlocking – Output Resources . . . 306
Table 6.174 Stitching – Input Resources . . . 307
Table 6.175 Stitching – Output Resources . . . 307

Table 6.176 Strapping – Input Resources . . . 307
Table 6.177 Strapping – Output Resources . . . 308
Table 6.178 StripBinding – Input Resources . . . 308
Table 6.179 StripBinding – Output Resources . . . 308
Table 6.180 ThreadSealing – Input Resources . . . 308
Table 6.181 ThreadSealing – Output Resources . . . 308
Table 6.182 ThreadSewing – Input Resources . . . 309
Table 6.183 ThreadSewing – Output Resources . . . 309
Table 6.184 Trimming – Input Resources . . . 309
Table 6.185 Trimming – Output Resources . . . 309
Table 6.186 WebInlineFinishing – Input Resources . . . 309
Table 6.187 WebInlineFinishing – Output Resources . . . 310
Table 6.188 Winding – Input Resources . . . 310
Table 6.189 Winding – Output Resources . . . 310
Table 6.190 WireCombBinding – Input Resources . . . 310
Table 6.191 WireCombBinding – Output Resources . . . 311
Table 6.192 Wrapping – Input Resources . . . 311
Table 6.193 Wrapping – Output Resources . . . 311
Table 7.1 Product Element . . . 315
Table 7.2 Intent Element . . . 316
Table 7.3 Product Intent – Input Resources Elements . . . 317
Table 7.4 Product Intent – Output Resources . . . 318
Table 7.5 Template for Intent Resources . . . 318
Table 7.6 Abstract Span Element . . . 319
Table 7.7 List of Span Elements . . . 319
Table 7.8 DurationSpan Element . . . 320
Table 7.9 EnumerationSpan Element . . . 320
Table 7.10 IntegerSpan Element . . . 320
Table 7.11 NameSpan Element . . . 321
Table 7.12 NumberSpan Element . . . 321
Table 7.13 OptionSpan Element . . . 321
Table 7.14 ShapeSpan Element . . . 322
Table 7.15 StringSpan Element . . . 322
Table 7.16 TimeSpan Element . . . 322
Table 7.17 XYPairSpan Element . . . 323
Table 7.18 ArtDeliveryIntent Resource . . . 323
Table 7.19 ArtDelivery Element . . . 325
Table 7.20 AssemblingIntent Element . . . 326
Table 7.21 AssemblyItem Element . . . 327
Table 7.22 BindIn Element . . . 327
Table 7.23 BlowIn Element . . . 327
Table 7.24 StickOn Element . . . 328
Table 7.25 BindingIntent Resource Element . . . 328

Table 7.26 BindingType Attribute Values . . . 330
Table 7.27 AdhesiveNote Element . . . 331
Table 7.28 BindList Element . . . 331
Table 7.29 BindItem Element . . . 331
Table 7.30 ChannelBinding Element . . . 332
Table 7.31 CoilBinding Element . . . 333
Table 7.32 EdgeGluing Element . . . 333
Table 7.33 HardCoverBinding Element . . . 333
Table 7.34 LooseBinding Element . . . 336
Table 7.35 PlasticCombBinding Element . . . 336
Table 7.36 RingBinding Element . . . 337
Table 7.37 SaddleStitching Element . . . 338
Table 7.38 SideSewing Element . . . 338
Table 7.39 SideStitching Element . . . 338
Table 7.40 SoftCoverBinding Element . . . 338
Table 7.41 StripBinding Element . . . 339
Table 7.42 Tabs Element . . . 340
Table 7.43 Tape Element . . . 340
Table 7.44 ThreadSealing Element . . . 341
Table 7.45 ThreadSewing Element . . . 341
Table 7.46 WireCombBinding Element . . . 341
Table 7.47 ColorIntent Resource Element . . . 342
Table 7.48 SurfaceColor Element . . . 343
Table 7.49 ColorStandard Attribute Values . . . 343
Table 7.50 ColorsUsed Element . . . 344
Table 7.51 ContentCheckIntent Element . . . 344
Table 7.52 PreflightItem Element . . . 344
Table 7.53 ProofItem Element . . . 344
Table 7.54 DeliveryIntent Resource . . . 345
Table 7.55 DropIntent Element . . . 347
Table 7.56 DropItemIntent Element . . . 348
Table 7.57 EmbossingIntent Resource Element . . . 349
Table 7.58 EmbossingItem Element . . . 349
Table 7.59 FoldingIntent Resource Element . . . 350
Table 7.60 HoleMakingIntent Resource Element . . . 351
Table 7.61 InsertingIntent Resource . . . 352
Table 7.62 InsertList Element . . . 352
Table 7.63 Insert Element . . . 352
Table 7.64 LaminatingIntent Resource Element . . . 353
Table 7.65 LayoutIntent Resource Element . . . 354
Table 7.66 MediaIntent Resource Element . . . 356
Table 7.67 PackingIntent Resource . . . 360
Table 7.68 ProductionIntent Resource Element . . . 361

Table 7.69 ProofingIntent Resource Element . . . 361
Table 7.70 PreflightItem Element . . . 362
Table 7.71 ProofItem Element . . . 362
Table 7.72 PublishingIntent Resource Element . . . 363
Table 7.73 ScreeningIntent Resource . . . 363
Table 7.74 ShapeCuttingIntent Resource Element . . . 364
Table 7.75 ShapeCut Element . . . 364
Table 7.76 VariableIntent Element . . . 365
Table 8.1 Resource Element . . . 367
Table 8.2 AmountPool Element . . . 368
Table 8.3 PartAmount Element . . . 368
Table 8.4 PartWaste Element . . . 369
Table 8.5 WasteDetails Attribute Values . . . 369
Table 8.6 Part Element . . . 370
Table 8.7 ApprovalParams Resource . . . 373
Table 8.8 ApprovalPerson Element . . . 374
Table 8.9 ApprovalSuccess Resource . . . 374
Table 8.10 ApprovalDetails Element Resource . . . 375
Table 8.11 Assembly Resource . . . 375
Table 8.12 AssemblySection Element . . . 376
Table 8.13 PageAssignedList Element . . . 377
Table 8.14 AssetListCreationParams Resource . . . 378
Table 8.15 BendingParams Resource . . . 379
Table 8.16 BinderySignature Resource . . . 380
Table 8.17 SignatureCell Element . . . 386
Table 8.18 BlockPreparationParams Resource . . . 390
Table 8.19 BoxFoldingParams Resource . . . 390
Table 8.20 BoxFoldAction Element . . . 391
Table 8.21 Action Attribute Values . . . 392
Table 8.22 BoxPackingParams Resource . . . 395
Table 8.23 BufferParams Resource . . . 396
Table 8.24 Bundle Resource . . . 396
Table 8.25 BundleItem Element . . . 397
Table 8.26 BundlingParams Resource . . . 400
Table 8.27 ByteMap Resource . . . 400
Table 8.28 Band Element . . . 401
Table 8.29 PixelColorant Element . . . 401
Table 8.30 CaseMakingParams Resource . . . 402
Table 8.31 CasingInParams Resource . . . 403
Table 8.32 ChannelBindingParams Resource . . . 404
Table 8.33 CoilBindingParams Resource . . . 404
Table 8.34 CollectingParams Resource . . . 405
Table 8.35 Color Resource . . . 407

Table 8.36 DeviceNColor Element . . . 409
Table 8.37 Diecutting Data (DDES3) . . . 409
Table 8.38 PrintConditionColor Element . . . 410
Table 8.39 ColorantControl Resource . . . 414
Table 8.40 ColorantAlias Element . . . 416
Table 8.41 ColorantConvertProcess Element . . . 417
Table 8.42 ColorantOrder Element . . . 417
Table 8.43 ColorantParams Element . . . 417
Table 8.44 DeviceColorantOrder Element . . . 417
Table 8.45 ColorSpaceSubstitute Element . . . 417
Table 8.46 Sample output for different values of ProcessColorModel, ColorantParams, ColorantOrder, ColorantControlLink and DeviceColorantOrder Elements. . . . 418
Table 8.47 Sample output for different values of ProcessColorModel, ColorantParams and ColorantOrder elements.. . . . 418
Table 8.48 DeviceNSpace Element . . . 419
Table 8.49 ColorCorrectionParams Resource . . . 420
Table 8.50 ColorCorrectionOp Element . . . 420
Table 8.51 ColorPool Resource . . . 421
Table 8.52 ColorSpaceConversionParams Resource . . . 421
Table 8.53 ColorSpaceConversionOp Element . . . 422
Table 8.54 SourceCS Attribute Values . . . 424
Table 8.55 Mapping of SourceCS enumeration values to color spaces in the most common input file formats 424
Table 8.56 Glossary – Component . . . 426
Table 8.57 Component Resource . . . 427
Table 8.58 ProductType Attribute Values . . . 430
Table 8.59 Contact Resource . . . 431
Table 8.60 ComChannel Element . . . 432
Table 8.61 Company Element . . . 433
Table 8.62 Person Element . . . 433
Table 8.63 ContactCopyParams Resource . . . 434
Table 8.64 ContentList Content Resource . . . 434
Table 8.65 ContentData Resource Element . . . 435
Table 8.66 ContentType Attribute Values . . . 436
Table 8.67 BarcodeProductionParams Element . . . 436
Table 8.68 ContentMetadata Element . . . 437
Table 8.69 PositionObj Element . . . 437
Table 8.70 ConventionalPrintingParams Resource . . . 440
Table 8.71 CoverApplicationParams Resource . . . 442
Table 8.72 Score Element . . . 442
Table 8.73 CreasingParams Resource. . . . 444
Table 8.74 CustomerInfo Resource . . . 444
Table 8.75 CutBlock Resource . . . 445
Table 8.76 CutMark Resource . . . 445

Table 8.77 Cut mark types as specified by CutMark/@MarkType . . . 446
Table 8.78 CuttingParams Resource . . . 447
Table 8.79 CutBlock Element . . . 447
Table 8.80 CylinderLayout Resource . . . 448
Table 8.81 CylinderPosition Element . . . 448
Table 8.82 CylinderLayoutPreparationParams Resource . . . 451
Table 8.83 DeliveryParams Resource . . . 451
Table 8.84 Drop Element . . . 453
Table 8.85 DropItem Element . . . 454
Table 8.86 DevelopingParams Resource . . . 455
Table 8.87 Device Resource . . . 455
Table 8.88 IconList Element . . . 458
Table 8.89 Icon Element . . . 458
Table 8.90 Module Element . . . 458
Table 8.91 DieLayout Resource . . . 459
Table 8.92 RuleLength Element . . . 460
Table 8.93 Station Element . . . 460
Table 8.94 DieLayoutProductionParams Resource . . . 461
Table 8.95 RepeatDesc Element . . . 461
Table 8.96 DigitalDeliveryParams Resource . . . 465
Table 8.97 DigitalMedia Resource . . . 466
Table 8.98 DigitalPrintingParams Resource . . . 467
Table 8.99 ElementColorParams Resource . . . 469
Table 8.100 EmbossingParams Resource . . . 470
Table 8.101 Emboss Element . . . 471
Table 8.102 Employee Resource . . . 472
Table 8.103 EndSheetGluingParams Resource . . . 472
Table 8.104 EndSheet Element . . . 472
Table 8.105 ExposedMedia Resource . . . 474
Table 8.106 ExternalImpositionTemplate Resource . . . 475
Table 8.107 FeedingParams Resource . . . 475
Table 8.108 Feeder Element . . . 475
Table 8.109 FeederQualityParams Element . . . 476
Table 8.110 CollatingItem Element . . . 477
Table 8.111 FileSpec Resource . . . 478
Table 8.112 ResourceUsage Attribute Values . . . 481
Table 8.113 Container Element . . . 482
Table 8.114 FileAlias Element . . . 482
Table 8.115 FoldingParams Resource . . . 483
Table 8.116 FontParams Resource . . . 487
Table 8.117 FontPolicy Resource . . . 487
Table 8.118 GatheringParams Resource . . . 488
Table 8.119 GlueApplication Resource . . . 489

Table 8.120 GluingParams Resource . . . 489
Table 8.121 Glue Element . . . 490
Table 8.122 HeadBandApplicationParams Resource . . . 490
Table 8.123 HoleList Resource . . . 491
Table 8.124 HoleMakingParams Resource . . . 491
Table 8.125 IdentificationField Resource . . . 493
Table 8.126 EncodingDetails Attribute Values . . . 494
Table 8.127 BarcodeDetails Element . . . 495
Table 8.128 ExtraValues Element . . . 496
Table 8.129 Usage of Barcode Attributes for Certain Barcode Types . . . 496
Table 8.130 BarcodeVersion Values – for HIBC_DATAMATRIX . . . 497
Table 8.131 BarcodeVersion Values – for QR barcodes . . . 497
Table 8.132 ImageCompressionParams Resource . . . 498
Table 8.133 ImageCompression Element . . . 498
Table 8.134 CCITTFaxParams Element . . . 500
Table 8.135 DCTParams Element . . . 501
Table 8.136 FlateParams Element . . . 501
Table 8.137 JBIG2Params Element . . . 502
Table 8.138 JPEG2000Params Element . . . 502
Table 8.139 LZWParams Element . . . 503
Table 8.140 ImageEnhancementParams Resource . . . 503
Table 8.141 ImageEnhancementOp Element . . . 503
Table 8.142 ImageReplacementParams Resource . . . 504
Table 8.143 ImageSetterParams Resource . . . 505
Table 8.144 Sides Attribute Values . . . 506
Table 8.145 Ink Resource . . . 507
Table 8.146 InkZoneCalculationParams Resource . . . 507
Table 8.147 InkZoneProfile Resource . . . 508
Table 8.148 InsertingParams Resource . . . 509
Table 8.149 Location of Inserts . . . 510
Table 8.150 InterpretingParams Resource . . . 510
Table 8.151 InterpretingDetails Element . . . 512
Table 8.152 PDFInterpretingParams Element . . . 512
Table 8.153 OCGControl Element . . . 513
Table 8.154 ReferenceXObjParams Element . . . 514
Table 8.155 JacketingParams Resource . . . 515
Table 8.156 LabelingParams Resource . . . 516
Table 8.157 LaminatingParams Resource . . . 516
Table 8.158 Layout Resource . . . 517
Table 8.159 ExternalImpositionTemplate Element . . . 522
Table 8.160 LayerList Element . . . 522
Table 8.161 LayerDetails Element . . . 522
Table 8.162 LogicalStackParams Element . . . 522

Table 8.163 Stack Element . . . 522
Table 8.164 PageCondition Element . . . 523
Table 8.165 Abstract PlacedObject Element . . . 524
Table 8.166 ContentObject Element . . . 526
Table 8.167 PageActivation . . . 527
Table 8.168 Condition . . . 527
Table 8.169 PageCondition Element . . . 528
Table 8.170 MarkObject Element . . . 528
Table 8.171 ColorControlStrip Element . . . 530
Table 8.172 CIELABMeasuringField Element . . . 530
Table 8.173 DensityMeasuringField Element . . . 531
Table 8.174 CutMark Resource . . . 532
Table 8.175 Cut mark types as specified by CutMark/@MarkType . . . 532
Table 8.176 DynamicField Element . . . 533
Table 8.177 FillMark Element . . . 534
Table 8.178 MarkColor Element . . . 534
Table 8.179 JobField Element . . . 535
Table 8.180 MarkActivation Element . . . 535
Table 8.181 RegisterMark Resource . . . 535
Table 8.182 ScavengerArea Resource . . . 536
Table 8.183 Position Element . . . 537
Table 8.184 SheetConditionActivation Element . . . 538
Table 8.185 StripMark Element . . . 538
Table 8.186 MarkName Attribute Values . . . 540
Table 8.187 Example (1) of Ord Attribute in PlacedObject Elements . . . 544
Table 8.188 Example (2) of Ord Attribute in PlacedObject Elements . . . 545
Table 8.189 Example Calculating Ord values in automated Imposition . . . 545
Table 8.190 OrdExpresson for Varying Page Length Booklet . . . 546
Table 8.191 DocOrd Usage . . . 546
Table 8.192 LayoutElement Resource . . . 547
Table 8.193 ElementType Attribute Values . . . 548
Table 8.194 Dependencies Element . . . 549
Table 8.195 LayoutElementProductionParams Resource . . . 549
Table 8.196 LayoutElementPart Element . . . 552
Table 8.197 BarcodeProductionParams Element . . . 553
Table 8.198 PositionObj Element . . . 553
Table 8.199 LayoutPreparationParams Resource . . . 560
Table 8.200 PageDistributionScheme Attribute Values . . . 565
Table 8.201 PageCell Element . . . 566
Table 8.202 ImageShift Element . . . 567
Table 8.203 LayoutShift Resource . . . 570
Table 8.204 ShiftPoint Element . . . 570
Table 8.205 LooseBindingParams Resource . . . 574

Table 8.206 ChannelBindingDetails Element . . . 575
Table 8.207 CoilBindingDetails Element . . . 575
Table 8.208 CombBindingDetails Element . . . 575
Table 8.209 RingBindingDetails Element . . . 575
Table 8.210 StripBindingDetails Element . . . 576
Table 8.211 ManualLaborParams Resource . . . 576
Table 8.212 Media Resource . . . 577
Table 8.213 MediaTypeDetails Attribute Values . . . 583
Table 8.214 TabDimensions Element . . . 586
Table 8.215 MiscConsumable Resource . . . 592
Table 8.216 NodeInfo Resource . . . 593
Table 8.217 NodeStatus Attribute Values . . . 595
Table 8.218 PageAssignParams Resource . . . 596
Table 8.219 PageList Resource . . . 597
Table 8.220 PageData Element . . . 598
Table 8.221 PageElement Element . . . 599
Table 8.222 Pallet Resource . . . 600
Table 8.223 PalletizingParams Resource . . . 600
Table 8.224 PDFToPSConversionParams Resource . . . 601
Table 8.225 PDLCreationParams Resource . . . 603
Table 8.226 FontParams Element . . . 604
Table 8.227 PDFToPSConversionParams Resource PSCreationDetails Element . . . 604
Table 8.228 PSToPDFConversionParams Resource PDFCreationDetails Element . . . 607
Table 8.229 AdvancedParams Element . . . 608
Table 8.230 PDFXParams Element . . . 609
Table 8.231 ThinPDFParams Element . . . 610
Table 8.232 PDLResourceAlias Resource . . . 610
Table 8.233 PerforatingParams Resource . . . 610
Table 8.234 PlasticCombBindingParams Resource . . . 611
Table 8.235 PreflightParams Resource . . . 612
Table 8.236 PreflightTest Element . . . 612
Table 8.237 PreflightAction Element . . . 613
Table 8.238 BasicPreflightTest Element . . . 613
Table 8.239 PreflightArgument Element . . . 614
Table 8.240 BoxArgument Element . . . 614
Table 8.241 Box Attribute Values . . . 614
Table 8.242 BoxToBoxDifference Element . . . 615
Table 8.243 PreflightReport Resource . . . 615
Table 8.244 PreflightCheck Element . . . 616
Table 8.245 PRItem Element . . . 616
Table 8.246 PRError Element . . . 617
Table 8.247 PRGroup Element . . . 617
Table 8.248 Abstract PRGroupOccurrenceBase Element . . . 619

Table 8.249 List of PRGroupOccurrenceBase Elements . . . 620
Table 8.250 ArgumentValue Element . . . 620
Table 8.251 PRGroupOccurrence Element . . . 620
Table 8.252 StringListValue Element . . . 620
Table 8.253 PROccurrence Element . . . 620
Table 8.254 PreflightReportRulePool Resource . . . 621
Table 8.255 PRRule Element . . . 621
Table 8.256 PRRuleAttr Element . . . 621
Table 8.257 ReportAttr Attribute Values . . . 622
Table 8.258 Contingent Report Behavior . . . 622
Table 8.259 Preview Resource . . . 623
Table 8.260 PreviewGenerationParams Resource . . . 624
Table 8.261 PrintCondition Resource . . . 626
Table 8.262 PrintRollingParams Resource . . . 626
Table 8.263 ProductionPath Resource . . . 627
Table 8.264 FolderSuperstructureWebPath Element . . . 627
Table 8.265 PostPressComponentPath Element . . . 628
Table 8.266 PrintingUnitWebPath Element . . . 628
Table 8.267 PSToPDFConversionParams Resource . . . 629
Table 8.268 AdvancedParams Element . . . 630
Table 8.269 PDFXParams Element . . . 631
Table 8.270 PDFXOutputIntentProfile Attribute Values . . . 632
Table 8.271 ThinPDFParams Element . . . 632
Table 8.272 QualityControlParams Resource . . . 633
Table 8.273 BindingQualityParams Element . . . 633
Table 8.274 QualityControlResult Resource . . . 633
Table 8.275 QualityMeasurement Element . . . 634
Table 8.276 BindingQualityMeasurement Element . . . 634
Table 8.277 RasterReadingParams Resource . . . 635
Table 8.278 RegisterMark Resource . . . 636
Table 8.279 RenderingParams Resource . . . 636
Table 8.280 TIFFFormatParams Element . . . 637
Table 8.281 TIFFtag Element . . . 638
Table 8.282 TIFFEmbeddedFile Element . . . 638
Table 8.283 ResourceDefinitionParams Resource . . . 639
Table 8.284 ResourceParam Element . . . 639
Table 8.285 RingBindingParams Resource . . . 639
Table 8.286 RollStand Resource . . . 640
Table 8.287 RunList Resource . . . 641
Table 8.288 Pages, Documents and Sets for common PDL types . . . 645
Table 8.289 ByteMap Element . . . 646
Table 8.290 Band Element . . . 647
Table 8.291 ElementType Attribute Values . . . 647

Table 8.292 ScanParams Resource . . . 649
Table 8.293 ScavengerArea Resource . . . 650
Table 8.294 ScreeningParams Resource . . . 651
Table 8.295 SeparationControlParams Resource . . . 651
Table 8.296 Shape Resource . . . 652
Table 8.297 ShapeCuttingParams Resource . . . 652
Table 8.298 ShapeDef Resource . . . 653
Table 8.299 CutLines Element . . . 654
Table 8.300 ShapeDefProductionParams Resource . . . 655
Table 8.301 ObjectModel Element . . . 655
Table 8.302 ShapeTemplate Element . . . 655
Table 8.303 ShapeDimension Element . . . 656
Table 8.304 SheetOptimizingParams Resource . . . 658
Table 8.305 GangElement Element . . . 658
Table 8.306 SeparationListBack Element . . . 660
Table 8.307 SeparationListFront Element . . . 660
Table 8.308 ShrinkingParams Resource . . . 660
Table 8.309 SpinePreparationParams Resource . . . 661
Table 8.310 Operations Attribute Values . . . 661
Table 8.311 SpineTapingParams Resource . . . 662
Table 8.312 Parameters in Stacking . . . 664
Table 8.313 StackingParams Resource . . . 665
Table 8.314 Disjointing Element . . . 666
Table 8.315 InsertSheet Element . . . 666
Table 8.316 SheetUsage Attribute Values . . . 667
Table 8.317 StaticBlockingParams Resource . . . 667
Table 8.318 StitchingParams Resource . . . 670
Table 8.319 Strap Resource . . . 671
Table 8.320 StrappingParams Resource . . . 672
Table 8.321 StripBindingParams Resource . . . 672
Table 8.322 StrippingParams Resource . . . 673
Table 8.323 Position Element . . . 676
Table 8.324 StripCellParams Element . . . 677
Table 8.325 StripMark Element . . . 678
Table 8.326 MarkName Attribute Values . . . 681
Table 8.327 MarkSide Attribute Values . . . 681
Table 8.328 ThreadSealingParams Resource . . . 682
Table 8.329 ThreadSewingParams Resource . . . 683
Table 8.330 Tile Resource . . . 684
Table 8.331 Tool Resource . . . 685
Table 8.332 TransferCurve Resource . . . 686
Table 8.333 TransferCurvePool Resource . . . 687
Table 8.334 TransferCurveSet Element . . . 687

Table 8.335 TransferFunctionControl Resource . . . 688
Table 8.336 TrappingDetails Resource . . . 688
Table 8.337 TrappingOrder Element . . . 689
Table 8.338 TrappingParams Resource . . . 689
Table 8.339 ColorantZoneDetails Element . . . 691
Table 8.340 TrapRegion Resource . . . 691
Table 8.341 TrimmingParams Resource . . . 692
Table 8.342 UsageCounter Resource . . . 693
Table 8.343 VarnishingParams Resource . . . 693
Table 8.344 VerificationParams Resource . . . 694
Table 8.345 WebInlineFinishingParams Resource . . . 695
Table 8.346 FolderProduction Element . . . 695
Table 8.347 WindingParams Resource . . . 695
Table 8.348 WireCombBindingParams Resource . . . 696
Table 8.349 WrappingParams Resource . . . 697
Table 9.1 Address Element . . . 699
Table 9.2 ApprovalPerson Element . . . 699
Table 9.3 AutomatedOverPrintParams Element . . . 700
Table 9.4 BarcodeCompParams Element . . . 700
Table 9.5 BarcodeReproParams Element . . . 701
Table 9.6 Certification Element . . . 701
Table 9.7 ColorantAlias Element . . . 702
Table 9.8 ColorCorrectionOp Element . . . 702
Table 9.9 ColorMeasurementConditions Resource . . . 703
Table 9.10 ColorSpaceConversionOp Element . . . 704
Table 9.11 SourceCS Attribute Values . . . 706
Table 9.12 Mapping of SourceCS enumeration values to color spaces in the most common input file formats 707
Table 9.13 ComChannel Element . . . 709
Table 9.14 ChannelTypeDetails Attribute – predefined values for certain ChannelType values . . . 710
Table 9.15 Comment Element . . . 710
Table 9.16 ConvertingConfig Element . . . 712
Table 9.17 CostCenter Element . . . 712
Table 9.18 Crease Element . . . 713
Table 9.19 Cut Element . . . 713
Table 9.20 DeviceMark Element . . . 714
Table 9.21 DeviceNSpace Element . . . 717
Table 9.22 Disjointing Element . . . 717
Table 9.23 Disposition Element . . . 718
Table 9.24 FileSpec Element . . . 719
Table 9.25 Disposition Element . . . 720
Table 9.26 FitPolicy Element . . . 721
Table 9.27 Fold Element . . . 722

Table 9.28 GangSource Element . . . 722
Table 9.29 GeneralID Element . . . 723
Table 9.30 GlueLine Element . . . 723
Table 9.31 HolePattern Element . . . 725
Table 9.32 HoleLine Element . . . 726
Table 9.33 IdentificationField Element . . . 727
Table 9.34 EncodingDetails Attribute Values . . . 728
Table 9.35 BarcodeDetails Element . . . 729
Table 9.36 ExtraValues Element . . . 729
Table 9.37 Usage of Barcode Attributes for Certain Barcode Types . . . 730
Table 9.38 BarcodeVersion Values – for HIBC_DATAMATRIX . . . 730
Table 9.39 BarcodeVersion Values – for QR barcodes . . . 731
Table 9.40 ImageCompression Element . . . 731
Table 9.41 CCITTFaxParams Element . . . 733
Table 9.42 DCTParams Element . . . 733
Table 9.43 FlateParams Element . . . 734
Table 9.44 JBIG2Params Element . . . 734
Table 9.45 JPEG2000Params Element . . . 734
Table 9.46 LZWParams Element . . . 735
Table 9.47 InsertSheet Element . . . 736
Table 9.48 SheetUsage Attribute Values . . . 737
Table 9.49 JobField Element . . . 738
Table 9.50 MarkColor Element . . . 738
Table 9.51 MediaLayers Element . . . 739
Table 9.52 MetadataMap Element . . . 739
Table 9.53 Expr Element . . . 741
Table 9.54 MetadataMap: Setting Attributes . . . 741
Table 9.55 MISDetails Element . . . 744
Table 9.56 Module Element . . . 744
Table 9.57 ObjectResolution Element . . . 745
Table 9.58 OCGControl Element . . . 746
Table 9.59 Perforate Element . . . 746
Table 9.60 Person Element . . . 747
Table 9.61 QueueEntry Element . . . 747
Table 9.62 QueueFilter Element . . . 748
Table 9.63 RefAnchor Element . . . 749
Table 9.64 RegisterRibbon Element . . . 750
Table 9.65 ScreenSelector Element . . . 751
Table 9.66 SeparationSpec Element . . . 752
Table 9.67 Shape Resource . . . 753
Table 10.1 DeviceCap Element . . . 758
Table 10.2 ActionPool Element . . . 759
Table 10.3 Action Element . . . 760

Table 10.4 DevCapPool Element . . . 760
Table 10.5 ModulePool Element . . . 760
Table 10.6 ModuleCap Element . . . 760
Table 10.7 DevCaps Element . . . 761
Table 10.8 Loc Element . . . 763
Table 10.9 DevCap Element . . . 763
Table 10.10 Abstract State Element . . . 766
Table 10.11 ListType Attribute Values . . . 767
Table 10.12 List of State Elements . . . 768
Table 10.13 BooleanState Element . . . 768
Table 10.14 ValueLoc Element . . . 769
Table 10.15 DateTimeState Element . . . 769
Table 10.16 DurationState Element . . . 769
Table 10.17 EnumerationState Element . . . 770
Table 10.18 IntegerState Element . . . 770
Table 10.19 MatrixState Element . . . 771
Table 10.20 MatrixState/Value Element . . . 772
Table 10.21 NameState Element . . . 772
Table 10.22 NumberState Element . . . 773
Table 10.23 PDFPathState Element . . . 773
Table 10.24 PDFPathState/Value Element . . . 774
Table 10.25 RectangleState Element . . . 774
Table 10.26 ShapeState Element . . . 774
Table 10.27 StringState Element . . . 775
Table 10.28 StringState/Value Element . . . 776
Table 10.29 XYPairState Element . . . 776
Table 10.30 DisplayGroupPool Element . . . 777
Table 10.31 DisplayGroup Element . . . 777
Table 10.32 FeaturePool Element . . . 778
Table 10.33 MacroPool Element . . . 778
Table 10.34 macro Element . . . 779
Table 10.35 choice Element . . . 779
Table 10.36 otherwise Element . . . 779
Table 10.37 when Element . . . 779
Table 10.38 set Element . . . 780
Table 10.39 FeatureAttribute Element . . . 780
Table 10.40 call Element . . . 780
Table 10.41 Performance Element . . . 780
Table 10.42 TestPool Element . . . 781
Table 10.43 Test Element . . . 781
Table 10.44 List of Term Elements . . . 782
Table 10.45 and Element . . . 783
Table 10.46 or Element . . . 783

Table 10.47 xor Element . . . 783
Table 10.48 not Element . . . 784
Table 10.49 TestRef Element . . . 784
Table 10.50 Abstract Evaluation Element . . . 785
Table 10.51 List of Evaluation Elements . . . 786
Table 10.52 Mapping of Evaluation Element to State Element . . . 786
Table 10.53 BooleanEvaluation Element . . . 787
Table 10.54 DateTimeEvaluation Element . . . 787
Table 10.55 DurationEvaluation Element . . . 787
Table 10.56 EnumerationEvaluation Element . . . 788
Table 10.57 IntegerEvaluation Element . . . 788
Table 10.58 IsPresentEvaluation Element . . . 788
Table 10.59 MatrixEvaluation Element . . . 788
Table 10.60 MatrixEvaluation/Value Element . . . 789
Table 10.61 NameEvaluation Element . . . 789
Table 10.62 NumberEvaluation Element . . . 789
Table 10.63 PDFPathEvaluation Element . . . 789
Table 10.64 PDFPathEvaluation/Value Element . . . 789
Table 10.65 RectangleEvaluation Element . . . 790
Table 10.66 ShapeEvaluation Element . . . 790
Table 10.67 StringEvaluation Element . . . 790
Table 10.68 StringEvaluation/Value Element . . . 790
Table 10.69 XYPairEvaluation Element . . . 791
Table 10.70 Object Classes for a Document . . . 798
Table 10.71 Properties Implemented by each Class of Object . . . 799
Table 10.72 Mapping between property types (in the preflight spec) and evaluations . . . 800
Table 10.73 List of Properties Categories . . . 801
Table 10.74 Annotation Properties . . . 802
Table 10.75 AnnotationType Attribute Values . . . 802
Table 10.76 Box Properties . . . 802
Table 10.77 Class Properties . . . 803
Table 10.78 ClassName Attribute Values . . . 803
Table 10.79 PropertyList Attribute Values . . . 803
Table 10.80 Colorant Properties . . . 804
Table 10.81 Document Properties . . . 804
Table 10.82 Fill Properties . . . 807
Table 10.83 FillColorType Attribute Values . . . 807
Table 10.84 Font Properties . . . 807
Table 10.85 FontType Attribute Values . . . 808
Table 10.86 Graphic Properties . . . 808
Table 10.87 Image Properties . . . 810
Table 10.88 Logical Properties . . . 811
Table 10.89 PageBox Properties . . . 811

Table 10.90 Pages Properties. . . . 812
Table 10.91 PDLObject Properties. . . . 813
Table 10.92 Reference Properties 813
Table 10.93 Shading Properties. . . . 813
Table 10.94 Stroke Properties 813
Table 10.95 Text Properties. . . . 814
Table 10.96 Vector Properties 815
Table 11.1 Event Sequence in Digital Finishing 820
Table 11.2 MIME Types and File Extensions 828
Table 11.3 MIME Content-Types 828
Table 11.4 Modifying Job Parameters 832
Table A.1 Activation Enumeration Values 847
Table A.2 Anchor Enumeration Values 847
Table A.3 847
Table A.4 Face Enumeration Values 848
Table A.5 ISOPaperSubstrate Enumeration Values 848
Table A.6 JDFJMFVersion Enumeration Values 848
Table A.7 MediaDirection Enumeration Values 848
Table A.8 NamedColor Enumeration Values. . . . 849
Table A.9 NodeStatus Enumeration Values 849
Table A.10 Orientation Enumeration Values. . . . 850
Table A.11 Polarity Enumeration Values. . . . 850
Table A.12 Policy Enumeration Values 850
Table A.13 851
Table A.14 SheeLay Enumeration Values 851
Table A.15 Sides Enumeration Values 851
Table A.16 Scope Enumeration Values 851
Table A.17 SourceObjects. . . . 852
Table A.18 WorkStyle Enumeration Values 852
Table A.19 XYRelation Enumeration Values 852
Table C.1 CombShape Names 857
Table C.2 Device Class Values 857
Table C.3 ModuleType Attribute Values for Conventional Printing Devices 858
Table C.4 ModuleType Attribute Values for Postpress. . . . 858
Table C.5 ModuleType Attribute Values for DigitalPrinting 859
Table C.6 ModuleType Attribute Values for PrintingUnitWebPath Modules of Web Printing Devices . . . 860
Table C.7 ModuleType Attribute Values for Folding FolderSuperstructureWebPath Modules of Web Printing Devices 860
Table C.8 ModuleType Attribute Values for Packing PostPressComponentPath Modules of Web Printing Devices 861
Table C.9 Coating Names. . . . 863
Table C.10 Input Tray and Output Bin Names 864
Table C.11 Media Coating Names 865

Table C.12 MessageEvents ContentStatus and MilestoneType Values . . . 867
Table C.13 Abstract NotificationDetails . . . 868
Table C.14 List of NotificationDetails Elements . . . 868
Table C.15 Barcode Element . . . 868
Table C.16 FCNKey Element . . . 869
Table C.17 SystemTimeSet Element . . . 869
Table C.18 CounterReset Element . . . 869
Table C.19 Error Element . . . 869
Table C.20 ErrorData Element . . . 869
Table C.21 Event Element . . . 870
Table C.22 Milestone Element . . . 870
Table C.23 PrintStandard Values . . . 871
Table C.24 ProductType Attribute Values . . . 871
Table C.25 StatusDetails Mapping for Generic Devices . . . 872
Table C.26 StatusDetails Mapping for Printing Devices . . . 874
Table C.27 StatusDetails Mapping for Postpress Devices . . . 875
Table D.1 Return codes for JMF XJMF . . . 877
Table E.1 Attributes for Color Space Adjustment . . . 881
Table F.1 Translation of Paper grades between [ISO12647-2:2004] and [ISO12647-2:2013] . . . 883
Table F.2 Conversion Factor from USWeight (lbs) to Weight (g/m2) . . . 883
Table F.3 Grammage Equivalents for Common (US) Basis Weights . . . 884
Table F.4 Japanese Media Weight . . . 885
Table G.1 Media Sizes . . . 887
Table H.1 MimeType Attribute Values (IANA Registered) . . . 891
Table H.2 MimeType and File Type Combinations . . . 892
Table I.1 Predefined variables used in @XXXTemplate and @ShowList . . . 895
Table J.1 Schemes Names for Binding Orientations . . . 901
Table J.2 Transformations for each Scheme . . . 901
Table J.3 Original Diagram . . . 903
Table J.4 Horizontal Binding Edges . . . 903
Table J.5 Original Diagram . . . 904
Table J.6 Vertical Binding Edges . . . 904
Table J.7 Pagination Diagrams . . . 905
Table L.1 References . . . 939
Table M.1 Naming Scheme for Hole Patterns . . . 951
Table M.2 Hole Details for R2 Series . . . 951
Table M.3 Hole Details for R3 and R4 Series . . . 953
Table M.4 Hole Details for R5 and R6 Series . . . 954
Table M.5 Hole Details for R7 and R11 Series . . . 956
Table M.6 Hole Details for P, W, C and S Series . . . 958
Table N.1 Use Cases showing MimeType, URL and Compression Attribute Values . . . 961
Table N.2 AppOS and OSVersion Examples . . . 972
Table O.1 Contents of the Abstract ResourceUpdate Element . . . 975

Table 0.2 Contents of the StatusPool Element . . . 976
Table 0.3 Contents of the PartStatus Element . . . 976
Table 0.4 Added Element . . . 978
Table 0.5 ChangedAttribute Element . . . 978
Table 0.6 Removed Element . . . 978
Table 0.7 Events Message . . . 979
Table 0.8 NotificationDef Element . . . 979
Table 0.9 KnownControllers Message . . . 980
Table 0.10 ControllerFilter Element . . . 980
Table 0.11 JDFController Element . . . 980
Table 0.12 RepeatMessages Message . . . 981
Table 0.13 MsgFilter Element . . . 981
Table 0.14 NodeInfo Query Message . . . 982
Table 0.15 NodeInfoQuParams Element . . . 982
Table 0.16 NodeInfo Command Message . . . 983
Table 0.17 NodeInfoCmdParams Element . . . 983
Table 0.18 NodeInfoResp Element . . . 983
Table 0.19 KnownJDFServices Message . . . 984
Table 0.20 JDFService Element . . . 984
Table 0.21 Occupation Message . . . 985
Table 0.22 EmployeeDef Element . . . 985
Table 0.23 Occupation Element . . . 985
Table 0.24 Track Message . . . 986
Table 0.25 TrackFilter Element . . . 986
Table 0.26 TrackResult Element . . . 986
Table 0.27 QueueEntryStatus Message . . . 987
Table 0.28 QueueEntryDefList Element . . . 987
Table 0.29 DBDocTemplateLayout – Input Resources . . . 988
Table 0.30 DBDocTemplateLayout – Output Resources . . . 988
Table 0.31 DBTemplateMerging – Input Resources . . . 988
Table 0.32 DBTemplateMerging – Output Resources . . . 988
Table 0.33 FormatConversion – Input Resources . . . 988
Table 0.34 FormatConversion – Output Resources . . . 988
Table 0.35 Ordering – Input Resources . . . 989
Table 0.36 Ordering – Output Resources . . . 989
Table 0.37 Packing – Input Resources . . . 989
Table 0.38 Packing – Output Resources . . . 989
Table 0.39 FilmToPlateCopying – Input Resources . . . 989
Table 0.40 FilmToPlateCopying – Output Resources . . . 989
Table 0.41 PreflightAnalysis Resource . . . 990
Table 0.42 PreflightDetail Element . . . 990
Table 0.43 PreflightInstance Element . . . 991
Table 0.44 PreflightInstanceDetail Element . . . 991

Table 0.45 PreflightInventory Resource . . . 991
Table 0.46 PreflightProfile Resource . . . 992
Table 0.47 PreflightConstraint Element . . . 992
Table 0.48 Proofing – Input Resources . . . 993
Table 0.49 Proofing – Output Resources . . . 993
Table 0.50 SoftProofing – Input Resources . . . 994
Table 0.51 SoftProofing – Output Resources . . . 994
Table 0.52 IDPrinting – Input Resources . . . 994
Table 0.53 IDPrinting – Output Resources . . . 995
Table 0.54 Dividing – Input Resources . . . 995
Table 0.55 Dividing – Output Resources . . . 995
Table 0.56 LongitudinalRibbonOperations – Input Resources . . . 996
Table 0.57 LongitudinalRibbonOperations – Output Resources . . . 996
Table 0.58 Numbering – Input Resources . . . 996
Table 0.59 Numbering – Output Resources . . . 996
Table 0.60 SaddleStitching – Input Resources . . . 996
Table 0.61 SaddleStitching – Output Resources . . . 996
Table 0.62 SideSewing – Input Resources . . . 997
Table 0.63 SideSewing – Output Resources . . . 997
Table 0.64 AdhesiveBinding Element . . . 997
Table 0.65 BookCase Element . . . 997
Table 0.66 Pricing Element . . . 998
Table 0.67 Payment Element . . . 998
Table 0.68 CreditCard Element . . . 998
Table 0.69 NumberingIntent Resource . . . 999
Table 0.70 NumberItem Element . . . 999
Table 0.71 SizeIntent Resource . . . 1000
Table 0.72 AdhesiveBindingParams Resource . . . 1000
Table 0.73 BoxApplication Element . . . 1001
Table 0.74 CustomerMessage Element . . . 1002
Table 0.75 DBMergeParams Resource . . . 1002
Table 0.76 DBRules Resource . . . 1003
Table 0.77 DBSchema Resource . . . 1003
Table 0.78 DBSelection Resource . . . 1003
Table 0.79 DividingParams Resource . . . 1004
Table 0.80 FormatConversionParams Resource . . . 1004
Table 0.81 IDPrintingParams Resource . . . 1005
Table 0.82 OutputBin Attribute Values . . . 1005
Table 0.83 Cover Element . . . 1006
Table 0.84 IDPFinishing Element . . . 1007
Table 0.85 Finishings Attribute Values . . . 1007
Table 0.86 IDPFolding Element . . . 1008
Table 0.87 IDPHoleMaking Element . . . 1008

Table 0.88 IDPLayout Element 1008
Table 0.89 IDPStitching Element 1009
Table 0.90 IDPTrimming Element 1010
Table 0.91 ImageShift Element 1010
Table 0.92 JobSheet Element 1011
Table 0.93 Signature Element 1013
Table 0.94 LongitudinalRibbonOperationParams Resource 1013
Table 0.95 LROperation Element 1013
Table 0.96 LongGlue Element 1014
Table 0.97 LongPerforate Element 1014
Table 0.98 MediaSource Resource 1014
Table 0.99 NumberingParams Resource 1015
Table 0.100 NumberingParam Element 1015
Table 0.101 OrderingParams Resource 1015
Table 0.102 PackingParams Resource 1016
Table 0.103 PlateCopyParams Resource 1017
Table 0.104 ProofingParams Resource 1017
Table 0.105 DynamicInput Element 1018
Table 0.106 SaddleStitchingParams Resource 1019
Table 0.107 Sheet Resource 1020
Table 0.108 SideSewingParams Resource 1021
Table 0.109 Surface Resource 1021

Adobe
Kodak
HEIDELBERG
MULLER MARTINI
RICOH
xerox