

JDF Specification

Release 1.3



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 © 2000-2005, 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 must 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 must include the following short form Copyright Notice: Copyright © 2000-2005, 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 Description Format, JDF 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 must 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

Front Matter

Legal Notice	i
Table of Contents	iii
JDF Preface and User Overview	xxvii
Chapter 1 Introduction	1
1.1 Background on JDF	1
1.2 Document References	1
1.3 Conventions Used in This Specification	2
1.3.1 Text Styles	2
1.3.2 XPath Notation Used in this Specification	3
1.3.3 Callouts	3
1.3.4 Specification of Cardinality	4
1.4 Glossary	4
1.4.1 Conformance Terminology	7
1.4.2 Conformance Requirements for JDF Entities	8
1.4.2.1 Conformance Requirements for Support of Attributes and Attribute Values	8
1.4.2.2 Conformance Requirements for Support of Elements	9
1.4.2.3 Conformance Requirements for Support of Processes	9
1.4.2.4 Conformance Requirements for Support of Combined Processes	9
1.4.3 Conformance to SettingsPolicy	10
1.5 Data Structures	10
1.6 Units	12
1.6.1 Counting in JDF	13
Chapter 2 Overview of JDF	15
2.1 System Components	15
2.1.1 Job Components	15
2.1.1.1 Jobs and Nodes	15
2.1.1.2 Elements	15
2.1.1.3 Attributes	15
2.1.1.4 Relationships	15
2.1.1.5 Links	16
2.1.2 Workflow Component Roles	16
2.1.2.1 Machines	16
2.1.2.2 Devices	16
2.1.2.3 Agents	16

2.1.2.4	Controllers	17
2.1.2.5	Management Information Systems—MIS	17
2.1.2.6	System Interaction	17
2.2	JDF Workflow	18
2.2.1	Job Structure.....	19
2.3	Hierarchical Tree Structure and Networks in JDF	21
2.4	Role of Messaging in JDF	22
2.5	Coordinate Systems in JDF	23
2.5.1	Introduction	23
2.5.1.1	Source Coordinate Systems	24
2.5.2	Coordinates and Transformations.....	24
2.5.3	Coordinate Systems of Resources and Processes.....	25
2.5.3.1	Coordinate Systems of Combined Processes.....	25
2.5.3.2	Coordinate System Transformations	25
2.5.4	Product Example: Simple Brochure	27
2.5.5	General Rules.....	32
2.5.6	Homogeneous Coordinates	32
Chapter 3	Structure of JDF Nodes and Jobs	35
3.1	JDF Nodes	37
3.1.1	Generic Contents of JDF Elements	37
3.1.2	JDF Node Attributes and Elements.....	40
3.2	Common Node Types	46
3.2.1	Product Intent Nodes	47
3.2.2	Process Group Nodes.....	47
3.2.2.1	Use of the Types attribute in ProcessGroup nodes – Gray Boxes	47
3.2.2.2	Use of the NamedFeatures attribute in Product and ProcessGroup nodes.....	48
3.2.2.3	ResourceLink Structure in ProcessGroup nodes	48
3.2.3	Combined Process Nodes	49
3.2.3.1	Combined Process Nodes with Multiple Processes of the Same Type.....	50
3.2.3.2	Examples of Combined Process Nodes	50
3.2.3.3	Specifying non-linear dependencies in a Combined node.....	51
3.2.4	Process Nodes.....	52
3.3	AncestorPool	52
3.4	CustomerInfo	54
3.5	NodeInfo	54
3.6	StatusPool	54
3.7	ResourcePool and its Resource Children	54
3.7.1	Abstract Resource	54

3.7.2 Resource Classes.....	60
3.7.2.1 Parameter Resources.....	60
3.7.2.2 Intent Resources.....	60
3.7.2.3 Implementation Resources.....	60
3.7.2.4 Physical Resources (Consumable, Quantity, Handling).....	61
3.7.2.5 Placeholder Resources	62
3.7.3 Position of Resources within JDF Nodes.....	62
3.7.4 Pipe Resources.....	63
3.7.5 ResourceUpdate Elements.....	63
3.8 ResourceLinkPool and ResourceLink	63
3.8.1 Abstract ResourceLink element.....	63
3.8.1.1 AmountPool and PartAmount.....	68
3.8.2 Links to Parameter Resources.....	69
3.8.3 Links to Implementation Resources.....	69
3.8.4 Links to Physical Resources	70
3.8.4.1 Identification of Physical Resources.....	72
3.8.5 Links to Placeholder Resources.....	73
3.8.6 Links to Intent Resources	73
3.9 ResourcePool and ResourceLinkPool – Deep Structure	74
3.9.1 ResourceElement – Subelement of a Resource.....	74
3.9.2 ResourceRef – Element for Inter-Resource Linking and refElement.....	74
3.9.2.1 Status of Resources That Contain rRef References	75
3.9.2.2 Alignment of ResourceLink and ResourceRef.....	75
3.9.3 Set of Resources and Partitioned Subsets Thereof.....	76
3.9.4 Resource Amount	76
3.9.4.1 Evaluating and Updating Amount related attributes in a Device	77
3.9.4.2 Specifying Amount for a partially completed process.....	77
3.9.5 Description of Partitioned Resources.....	79
3.9.5.1 Subelements in Partitioned Resources.....	80
3.9.5.2 Amount in Partitioned Resources	80
3.9.5.3 Relating PartIDKeys and Partitions.....	81
3.9.5.3.1 Incomplete Partitions	81
3.9.5.3.2 Number of Partition Keys per Partitioned Leaf or Node	81
3.9.5.3.3 Degenerate Partitions	82
3.9.5.4 Partitioning of Resource sub-Elements.....	82
3.9.5.5 Logical partitions and the Identical element.....	83
3.9.5.5.1 Restrictions when using Identical elements	84
3.9.6 PartIDKeys Attribute and Partition Keys	85
3.9.6.1 Options in Intent Resources.....	94
3.9.6.2 Locations of Physical Resources	94
3.9.7 Linking to Subsets of Resources	95
3.9.7.1 Handling Amount in a ResourceLink to a Partitioned Resource.....	95
3.9.7.2 Implicit, Sparse and Explicit PartUsage in Partitioned Resources	96

3.9.7.3 Referencing Partitioned Resources from Nodes That Allow Multiple ResourceLink elements.....	97
3.9.8 Splitting and Combining Resources.....	98
3.10 AuditPool	99
3.10.1 Audit Elements.....	102
3.10.1.1 ProcessRun.....	102
3.10.1.2 Notification.....	103
3.10.1.2.1 NotificationDetails	104
3.10.1.3 PhaseTime.....	104
3.10.1.4 ResourceAudit	107
3.10.1.4.1 Logging Machine Data by Using the ResourceAudit	107
3.10.1.4.2 Logging Changes in Product Descriptions by Using the ResourceAudit	108
3.10.1.5 Created.....	109
3.10.1.6 Deleted.....	109
3.10.1.7 Modified.....	109
3.10.1.8 Spawned.....	109
3.10.1.9 Merged.....	110
3.11 JDF Extensibility	111
3.11.1 Namespaces in XML.....	111
3.11.1.1 JDF Namespace	111
3.11.1.2 JDF Extension Namespace	111
3.11.2 Extending Process Types	111
3.11.3 Extending Existing Resources	112
3.11.4 Extending NMTOKEN Lists.....	112
3.11.5 Creating New Resources	113
3.11.6 Future JDF Extensions	113
3.11.7 Maintaining Extensions.....	113
3.11.8 Processing Unknown Extensions.....	113
3.11.9 Derivation of Types in XML Schema.....	114
3.12 JDF Versioning	114
3.12.1 JDF Versioning Requirements.....	114
3.12.2 JDF Version Definition	114
3.12.3 JDF Version Policies.....	114
3.12.3.1 JDF Specification Version Policies	114
3.12.3.2 JDF Schema Version Policies.....	115
3.12.3.3 JDF Application Version Policies.....	115
3.12.3.3.1 JDF Agent Version Policies	115
3.12.3.3.2 JDF Device/Controller Version Policies	116
Chapter 4 Life Cycle of JDF	117
4.1 Creation and Modification	117
4.1.1 Product Intent Constructs	117
4.1.1.1 Representation of Product Intent	118

4.1.1.2 Representation of Product Binding.....	118
4.1.2 Defining Business Objects Using Intent Resources.....	118
4.1.3 Specification of Delivery of End Products.....	121
4.1.4 Specification of Process Specifics for Product Intent Nodes.....	121
4.2 Process Routing	121
4.2.1 Determining Executable Nodes.....	123
4.2.2 Distributing Processing to Work Centers or Devices.....	123
4.2.3 Device / Controller Selection.....	124
4.3 Execution Model	124
4.3.1 Serial Processing.....	124
4.3.2 Partial Processing of Nodes with Partitioned Resources.....	126
4.3.3 Overlapping Processing Using Pipes.....	127
4.3.3.1 Pipes of Partitionable Resources.....	129
4.3.3.2 Dynamic Pipes.....	130
4.3.3.3 Comparison of Non-Dynamic and Dynamic Pipes.....	131
4.3.4 Parallel Processing.....	131
4.3.5 Iterative Processing.....	131
4.3.5.1 Informal Iterative Processing.....	132
4.3.5.2 Formal Iterative Processing.....	132
4.3.6 Approval, Quality Control and Verification.....	132
4.4 Spawning and Merging	132
4.4.1 Case 1: Standard Spawning and Merging.....	134
4.4.2 Case 2: Spawning and Merging with Resource Copying.....	135
4.4.2.1 Spawning of Resources with Inter-Resource Links.....	135
4.4.3 Case 3: Parallel Spawning and Merging of Partitioned Resources.....	136
4.4.4 Case 4: Nested Spawning and Merging in Reverse Sequence.....	136
4.4.5 Case 5: Spawning and Merging of Independent Jobs.....	137
4.4.6 Case 6: Simultaneous Spawning and Merging of Multiple Nodes.....	139
4.5 Node and Resource IDs	139
4.6 Error Handling	140
4.6.1 Classification of Notifications.....	140
4.6.2 Event Description.....	140
4.6.3 Error Logging in the JDF File.....	140
4.6.4 Error Handling via Messaging (JMF).....	140
4.7 Test Running	140
4.7.1 Resource Status During Testrun.....	141
4.8 Capability and Constraint Definitions	142

Chapter 5 JDF Messaging with the Job Messaging Format	143
5.1 JMF Root	143
5.2 JMF Message Families	147
5.2.1 Query	147
5.2.2 Response	148
5.2.3 Signal	149
5.2.4 Command	151
5.2.5 Acknowledge	152
5.2.6 Registration	154
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 Persistent Channels	154
5.3.3.1 Persistent Channels for Signals	154
5.3.3.2 Persistent Channels for Commands	154
5.3.4 Subscription Element	154
5.3.5 Creating Persistent Channels in a JDF Node	156
5.3.6 Deleting Persistent Channels	156
5.4 JMF Messaging Levels	156
5.5 Error and Event Messages	156
5.5.1 Pure Event Messages	157
5.6 Message Template	157
5.7 Messages for Events and Capabilities	158
5.7.1 Events	158
5.7.2 KnownControllers	161
5.7.3 KnownDevices	161
5.7.4 KnownJDFServices	163
5.7.5 KnownMessages	163
5.7.6 RepeatMessages	165
5.7.7 StopPersistentChannel	166
5.8 Messages to Query/Command a Job, Device or Controller	167
5.8.1 FlushResources	168
5.8.2 ModifyNode	168
5.8.3 NewJDF	169
5.8.3.1 NewJDF Query	170
5.8.3.2 NewJDF Command	170
5.8.4 NodeInfo	171
5.8.5 Occupation	171
5.8.6 Resource	172

5.8.6.1 Resource Query.....	173
5.8.6.2 Resource Command.....	175
5.8.7 ResourcePull.....	179
5.8.8 ShutDown	181
5.8.9 Status.....	182
5.8.10 Track.....	188
5.8.11 UpdateJDF.....	190
5.8.12 WakeUp.....	192
5.9 Messages for Pipe Control	193
5.9.1 PipeClose.....	193
5.9.2 PipePull.....	194
5.9.3 PipePush.....	195
5.9.4 PipePause.....	196
5.10 Queue Support	196
5.10.1 Queue Entry ID Generation	197
5.10.2 Use of QueueFilter in Queue Entry Handling commands	197
5.11 Messages for Queue Entry Handling	197
5.11.1 AbortQueueEntry	199
5.11.2 HoldQueueEntry	199
5.11.3 RemoveQueueEntry	200
5.11.4 RequestQueueEntry	200
5.11.5 ResubmitQueueEntry.....	201
5.11.6 ResumeQueueEntry	201
5.11.7 ReturnQueueEntry.....	202
5.11.8 SetQueueEntryPosition.....	202
5.11.9 SetQueueEntryPriority	203
5.11.10 SubmitQueueEntry.....	204
5.11.11 SuspendQueueEntry.....	206
5.12 Messages for Global Handling of Queues	206
5.12.1 CloseQueue.....	207
5.12.2 FlushQueue	208
5.12.2.1 FlushQueue Command	208
5.12.2.2 FlushQueue Query	209
5.12.3 HoldQueue.....	209
5.12.4 OpenQueue	209
5.12.5 QueueEntryStatus.....	210
5.12.6 QueueStatus.....	210
5.12.7 ResumeQueue.....	210
5.12.8 SubmissionMethods.....	210

5.13 Elements for Queues	211
5.13.1 Queue	211
5.13.2 QueueEntry	213
5.13.3 QueueEntryDef	214
5.13.4 QueueFilter	214
5.14 Gang Jobs	215
5.14.1 ForceGang	215
5.14.2 GangStatus	216
5.15 Extending Messages	216
5.15.1 IfraTrack Support	217
Chapter 6 Processes	219
6.1 Process Template	219
6.2 General Processes	220
6.2.1 Approval	220
6.2.2 Buffer	221
6.2.3 Combine	221
6.2.4 Delivery	221
6.2.5 ManualLabor	222
6.2.6 Ordering	222
6.2.7 Packing	222
6.2.8 QualityControl	222
6.2.9 ResourceDefinition	223
6.2.10 Split	223
6.2.11 Verification	223
6.3 Product Intent Descriptions	224
6.4 Prepress Processes	225
6.4.1 AssetListCreation	225
6.4.2 Bending	226
6.4.3 ColorCorrection	226
6.4.4 ColorSpaceConversion	227
6.4.5 ContactCopying	227
6.4.6 ContoneCalibration	227
6.4.7 CylinderLayoutPreparation	228
6.4.8 DBDocTemplateLayout	228
6.4.9 DBTemplateMerging	229
6.4.10 DigitalDelivery	229
6.4.11 FilmToPlateCopying	230
6.4.12 FormatConversion	230

6.4.13 ImageReplacement.....	230
6.4.14 ImageSetting.....	230
6.4.15 Imposition.....	231
6.4.16 InkZoneCalculation	232
6.4.17 Interpreting.....	233
6.4.18 LayoutElementProduction.....	233
6.4.19 LayoutPreparation.....	234
6.4.20 PDFToPSConversion.....	234
6.4.21 PDLCreation	235
6.4.22 Preflight.....	235
6.4.23 PreviewGeneration	236
6.4.24 Proofing.....	238
6.4.25 PSToPDFConversion.....	238
6.4.26 RasterReading	238
6.4.27 Rendering	239
6.4.28 RIPing	239
6.4.29 Scanning	240
6.4.30 Screening.....	240
6.4.31 Separation.....	241
6.4.32 SoftProofing	241
6.4.33 Stripping.....	241
6.4.34 Tiling	243
6.4.35 Trapping.....	243
6.5 Press Processes	244
6.5.1 ConventionalPrinting.....	244
6.5.2 DigitalPrinting.....	246
6.5.3 IDPrinting	248
6.6 Postpress Processes	248
6.6.1 AdhesiveBinding	248
6.6.2 BlockPreparation.....	248
6.6.3 BoxFolding	249
6.6.4 BoxPacking	249
6.6.5 Bundling.....	250
6.6.6 CaseMaking	250
6.6.7 CasingIn.....	251
6.6.8 ChannelBinding.....	251
6.6.9 CoilBinding.....	252
6.6.10 Collecting	252
6.6.11 CoverApplication.....	253
6.6.12 Creasing.....	253

6.6.13 Cutting.....	253
6.6.14 Dividing	254
6.6.15 Embossing	254
6.6.16 EndSheetGluing.....	254
6.6.17 Feeding	255
6.6.18 Folding	256
6.6.19 Gathering	256
6.6.20 Gluing.....	257
6.6.21 HeadBandApplication	257
6.6.22 HoleMaking	258
6.6.23 Inserting	258
6.6.24 Jacketing.....	258
6.6.25 Labeling	259
6.6.26 Laminating	259
6.6.27 LongitudinalRibbonOperations.....	259
6.6.28 Numbering	260
6.6.29 Palletizing.....	260
6.6.30 Perforating	260
6.6.31 PlasticCombBinding.....	260
6.6.32 PrintRolling.....	261
6.6.33 RingBinding.....	261
6.6.34 SaddleStitching	262
6.6.35 ShapeCutting	262
6.6.36 Shrinking	262
6.6.37 SideSewing	263
6.6.38 SpinePreparation	263
6.6.39 SpineTaping.....	263
6.6.40 Stacking	263
6.6.41 Stitching	264
6.6.42 Strapping.....	264
6.6.43 StripBinding.....	264
6.6.44 ThreadSealing.....	265
6.6.45 ThreadSewing.....	265
6.6.46 Trimming	265
6.6.47 WebInlineFinishing.....	266
6.6.48 WireCombBinding	266
6.6.49 Wrapping.....	267
6.7 Postpress Processes Structure	267
6.7.1 Block Production	267
6.7.1.1 Block Compiling.....	267

6.7.1.2 Block Joining	267
6.7.1.2.1 Single-Leaf Binding Methods	268
6.7.1.2.2 Loose-Leaf Binding Method	268
6.7.1.2.3 Mechanical Binding Methods	268
6.7.2 HoleMaking	268
6.7.3 Laminating	268
6.7.4 Numbering	268
6.7.5 Packaging Processes	268
6.7.6 Processes in Hardcover Book Production	269
6.7.7 Sheet Processes.....	269
6.7.8 Tip-on/in	270
6.7.9 Trimming	270
6.7.10 Web Processes.....	270
Chapter 7 Resources	271
7.1 Intent Resources	271
7.1.1 Span Subelements of an Intent Resource	272
7.1.1.1 Abstract Span Element.....	273
7.1.1.2 DurationSpan Element.....	273
7.1.1.3 EnumerationSpan Element	274
7.1.1.4 IntegerSpan Element.....	274
7.1.1.5 NameSpan Element.....	274
7.1.1.5.1 Specifying New Values in a NameSpan Subelement	275
7.1.1.6 NumberSpan Element.....	275
7.1.1.7 OptionSpan Element	275
7.1.1.8 ShapeSpan Element	275
7.1.1.9 StringSpan Element	276
7.1.1.10 TimeSpan Element.....	276
7.1.1.11 XYPairSpan Element.....	276
7.1.2 ArtDeliveryIntent	277
7.1.3 BindingIntent	282
7.1.4 ColorIntent	294
7.1.5 DeliveryIntent.....	297
7.1.6 EmbossingIntent	302
7.1.7 FoldingIntent	304
7.1.8 HoleMakingIntent	304
7.1.9 InsertingIntent	305
7.1.10 LaminatingIntent	307
7.1.11 LayoutIntent	307
7.1.12 MediaIntent	310
7.1.13 NumberingIntent	316
7.1.14 PackingIntent	316

7.1.15 ProductionIntent.....	317
7.1.16 ProofingIntent.....	318
7.1.17 PublishingIntent	320
7.1.18 ScreeningIntent.....	320
7.1.19 ShapeCuttingIntent	321
7.1.20 SizeIntent.....	322
7.2 Process Resources	322
7.2.1 Process Resource Template.....	322
7.2.2 Address.....	324
7.2.3 AdhesiveBindingParams.....	324
7.2.4 ApprovalParams	324
7.2.5 ApprovalSuccess	325
7.2.6 Assembly	326
7.2.7 AssetListCreationParams	329
7.2.8 AutomatedOverPrintParams	329
7.2.9 BarcodeCompParams.....	330
7.2.10 BarcodeReproParams	331
7.2.11 BendingParams	331
7.2.12 BinderySignature	332
7.2.13 BlockPreparationParams	336
7.2.14 BoxFoldingParams.....	337
7.2.15 BoxPackingParams.....	342
7.2.16 BufferParams	343
7.2.17 Bundle.....	344
7.2.18 BundlingParams.....	346
7.2.19 ByteMap.....	346
7.2.20 CaseMakingParams.....	348
7.2.21 CasingInParams	349
7.2.22 ChannelBindingParams	350
7.2.23 CIELABMeasuringField.....	350
7.2.24 CoilBindingParams	351
7.2.25 CollectingParams.....	352
7.2.26 Color	353
7.2.27 ColorantAlias.....	359
7.2.28 ColorantControl.....	359
7.2.29 ColorControlStrip	363
7.2.30 ColorCorrectionParams	364
7.2.31 ColorMeasurementConditions	366
7.2.32 ColorPool	367
7.2.33 ColorSpaceConversionParams.....	368

7.2.34 ColorSpaceConversionOp	370
7.2.35 ComChannel	379
7.2.36 Company.....	380
7.2.37 Component	381
7.2.38 Contact.....	386
7.2.39 ContactCopyParams.....	387
7.2.40 ContentList.....	387
7.2.41 ConventionalPrintingParams	389
7.2.42 CostCenter	392
7.2.43 CoverApplicationParams	392
7.2.44 CreasingParams	393
7.2.45 CustomerInfo	395
7.2.46 CutBlock.....	396
7.2.47 CutMark	397
7.2.48 CuttingParams	399
7.2.49 CylinderLayout	400
7.2.50 CylinderLayoutPreparationParams	403
7.2.51 DBMergeParams	403
7.2.52 DBRules.....	403
7.2.53 DBSchema.....	404
7.2.54 DBSelection	404
7.2.55 DeliveryParams.....	405
7.2.56 DensityMeasuringField	408
7.2.57 DevelopingParams.....	409
7.2.58 Device	409
7.2.59 DeviceMark	413
7.2.60 DeviceNSpace	413
7.2.61 DieLayout.....	414
7.2.62 DigitalDeliveryParams.....	415
7.2.63 DigitalMedia	416
7.2.64 DigitalPrintingParams	416
7.2.64.1 Coordinate systems in DigitalPrinting.....	416
7.2.65 Disjointing	419
7.2.66 Disposition	420
7.2.67 DividingParams.....	421
7.2.68 ElementColorParams.....	421
7.2.69 EmbossingParams.....	422
7.2.70 Employee	423
7.2.71 EndSheetGluingParams	424
7.2.72 ExposedMedia	425

7.2.73 ExternalImpositionTemplate	426
7.2.74 FeedingParams.....	427
7.2.75 FileSpec.....	430
7.2.76 FitPolicy	436
7.2.77 Fold.....	437
7.2.78 FoldingParams.....	438
7.2.79 FontParams	442
7.2.80 FontPolicy	442
7.2.81 FormatConversionParams	443
7.2.82 GatheringParams.....	446
7.2.83 GeneralID.....	447
7.2.84 GlueApplication.....	447
7.2.85 GlueLine.....	448
7.2.86 GluingParams	449
7.2.87 HeadBandApplicationParams	450
7.2.88 Hole.....	451
7.2.89 HoleLine.....	451
7.2.90 HoleList.....	452
7.2.91 HoleMakingParams.....	453
7.2.92 IdentificationField	455
7.2.93 IDPrintingParams.....	460
7.2.94 ImageCompressionParams	460
7.2.95 ImageReplacementParams	467
7.2.96 ImageSetterParams.....	468
7.2.97 Ink	471
7.2.98 InkZoneCalculationParams.....	472
7.2.99 InkZoneProfile.....	472
7.2.100 InsertingParams.....	473
7.2.101 InsertSheet.....	475
7.2.102 InterpretedPDLData	479
7.2.103 InterpretingParams	479
7.2.103.1 PDF Optional Content Groups.....	483
7.2.104 JacketingParams	483
7.2.105 JobField	484
7.2.106 LabelingParams.....	486
7.2.107 LaminatingParams.....	486
7.2.108 Layout	487
7.2.108.1 Migrating from a Pre-JDF 1.3 Layout to a Partitioned Layout	492
7.2.108.1.1 Partition Key restrictions:	492
7.2.108.1.2 Position of PlacedObject elements in Layout	492
7.2.108.2 CTM Definitions.....	493

7.2.108.3 Finding the Trim Box of an Object.....	493
7.2.108.4 Using Ord to Reference Elements in RunList Resources.....	493
7.2.108.5 Using Expressions in the OrdExpression Attribute.....	495
7.2.108.6 Dynamic marks.....	496
7.2.108.7 DynamicField Subelement Properties.....	497
7.2.109 LayoutElement.....	498
7.2.110 LayoutElementProductionParams.....	501
7.2.111 LayoutPreparationParams.....	502
7.2.112 LongitudinalRibbonOperationParams.....	514
7.2.113 ManualLaborParams.....	514
7.2.114 Media.....	514
7.2.114.1 Corrugated Media:.....	521
7.2.114.2 Self adhesive Media.....	522
7.2.115 MediaSource.....	522
7.2.116 MiscConsumable.....	523
7.2.117 MISDetails.....	523
7.2.118 NodeInfo.....	524
7.2.119 NumberingParams.....	526
7.2.120 ObjectResolution.....	527
7.2.121 OrderingParams.....	528
7.2.122 PackingParams.....	528
7.2.123 PageList.....	528
7.2.124 Pallet.....	532
7.2.125 PalletizingParams.....	532
7.2.126 PDFToPSConversionParams.....	533
7.2.127 PDLCreationParams.....	536
7.2.128 PDLResourceAlias.....	536
7.2.129 PerforatingParams.....	537
7.2.130 Person.....	538
7.2.131 PlaceholderResource.....	539
7.2.132 PlasticCombBindingParams.....	539
7.2.133 PlateCopyParams.....	540
7.2.134 PreflightAnalysis.....	540
7.2.135 PreflightInventory.....	540
7.2.136 PreflightParams.....	540
7.2.137 PreflightProfile.....	542
7.2.138 PreflightReport.....	542
7.2.139 PreflightReportRulePool.....	547
7.2.140 Preview.....	549
7.2.141 PreviewGenerationParams.....	551
7.2.142 PrintCondition.....	552

7.2.143 PrintRollingParams	553
7.2.144 ProductionPath	554
7.2.145 ProofingParams	555
7.2.146 PSToPDFConversionParams	555
7.2.147 QualityControlParams	561
7.2.148 QualityControlResult	561
7.2.149 RasterReadingParams.....	562
7.2.150 RegisterMark.....	563
7.2.151 RegisterRibbon	564
7.2.152 RenderingParams	565
7.2.153 ResourceDefinitionParams	566
7.2.154 RingBindingParams	567
7.2.155 RollStand	568
7.2.156 RunList.....	568
7.2.157 SaddleStitchingParams.....	574
7.2.158 ScanParams	574
7.2.159 ScavengerArea	576
7.2.160 ScreeningParams	576
7.2.161 SeparationControlParams.....	578
7.2.162 SeparationSpec	579
7.2.163 ShapeCuttingParams.....	579
7.2.164 Sheet.....	580
7.2.165 ShrinkingParams.....	580
7.2.166 SideSewingParams.....	581
7.2.167 SpinePreparationParams.....	581
7.2.168 SpineTapingParams	582
7.2.169 StackingParams.....	584
7.2.170 StitchingParams.....	586
7.2.171 Strap	589
7.2.172 StrappingParams	589
7.2.173 StripBindingParams	590
7.2.174 StrippingParams	591
7.2.175 Surface.....	597
7.2.176 ThreadSealingParams	597
7.2.177 ThreadSewingParams	598
7.2.178 Tile	599
7.2.179 Tool.....	600
7.2.180 TransferCurve.....	601
7.2.181 TransferCurvePool.....	601
7.2.182 TransferFunctionControl	602

7.2.183 TrappingDetails.....	603
7.2.184 TrappingParams	604
7.2.185 TrapRegion	608
7.2.186 TrimmingParams.....	608
7.2.187 UsageCounter.....	609
7.2.188 VerificationParams.....	610
7.2.189 WebInlineFinishingParams	611
7.2.190 WireCombBindingParams.....	612
7.2.191 WrappingParams	613
7.3 Device Capability Definitions	613
7.3.1 DeviceCap	613
7.3.2 ActionPool.....	617
7.3.3 DevCapPool.....	618
7.3.4 ModulePool	618
7.3.5 DevCaps	618
7.3.6 DevCap	621
7.3.7 State.....	623
7.3.8 DisplayGroupPool	638
7.3.9 FeaturePool	639
7.3.10 MacroPool.....	639
7.3.11 Performance	642
7.3.12 TestPool.....	642
7.3.13 Term.....	643
7.3.13.1 Boolean Operators	645
7.3.13.2 Evaluation	646
7.3.13.3 TestRef.....	652
7.3.14 Examples of Device Capabilities.....	652
7.4 Concept of the Preflight Process	659
7.4.1 Object Classes.....	660
7.4.1.1 Checking for the Presence of a Property.....	661
7.4.1.2 Basic tests on set of objects	662
7.4.2 Properties.....	662
7.4.2.1 Annotation Properties	662
7.4.2.2 Box Properties.....	663
7.4.2.3 Class Properties.....	664
7.4.2.4 Colorant Properties	665
7.4.2.5 Document Properties.....	665
7.4.2.6 Fill Properties.....	668
7.4.2.7 Font Properties.....	669
7.4.2.8 Graphic Properties	670
7.4.2.9 Image Properties	672
7.4.2.10 Logical Properties	674

7.4.2.11 PageBox Properties	674
7.4.2.12 Pages Properties	674
7.4.2.13 PDLObject Properties	675
7.4.2.14 Reference Properties	676
7.4.2.15 Shading Properties	676
7.4.2.16 Stroke Properties	676
7.4.2.17 Text Properties	677
7.4.2.18 Vector Properties	678
Chapter 8 Building a System Around JDF	679
8.1 Implementation Considerations and Guidelines	679
8.2 JDF and JMF Interchange Protocol	679
8.2.1 File-Based Protocol (JDF + JMF)	679
8.2.1.1 JMF Transport Using The File Protocol	679
8.2.2 HTTP-Based Protocol (JDF + JMF)	680
8.2.2.1 Protocol Implementation Details	680
8.2.3 HTTPS-Based Protocol – SSL with two-way authentication	680
8.2.3.1 Purpose	680
8.3 JDF Packaging	680
8.3.1 MIME Basics	681
8.3.2 MIME Types and File Extensions	681
8.3.2.1 MIME Headers	681
8.3.2.1.1 Content-Type Header	681
8.3.2.1.2 Content-ID Header	681
8.3.2.1.3 Content-Transfer-Encoding	682
8.3.2.1.4 Content-Disposition Header	682
8.3.2.2 Example Packaging of Individual JDF/JMF files in MIME	682
8.3.2.3 CID URL Scheme	682
8.3.2.4 Ordering of Body Parts in MIME Multipart/Related	683
8.4 MIS Requirements	684
8.5 Interoperability Conformance Specifications	684
Appendix A Encoding	687
A.1 Notes About Encoding	687
A.1.1 List, Range and Range List Data Types	687
A.1.2 Whitespace	687
A.1.3 Infinity Limits	687
A.2 Simple Types — Attribute Values	687
A.2.1 boolean	687
A.2.2 CMYKColor	687
A.2.3 date	688
A.2.4 dateTime	688

A.2.5 DateTimeRange	688
A.2.6 DateTimeRangeList.....	688
A.2.7 double.....	688
A.2.8 DoubleList.....	689
A.2.9 DoubleRange.....	689
A.2.10 DoubleRangeList.....	689
A.2.11 duration.....	689
A.2.12 DurationRange	689
A.2.13 DurationRangeList.....	690
A.2.14 gYearMonth	690
A.2.15 hexBinary.....	690
A.2.16 ID	690
A.2.17 IDREF	690
A.2.18 IDREFS	690
A.2.19 integer.....	691
A.2.20 IntegerList.....	691
A.2.21 IntegerRange.....	691
A.2.22 IntegerRangeList.....	691
A.2.23 LabColor	691
A.2.24 language.....	692
A.2.25 matrix.....	692
A.2.26 NameRange	692
A.2.27 NameRangeList.....	692
A.2.28 NMTOKEN.....	692
A.2.29 NMTOKENS	693
A.2.30 PDFPath	693
A.2.31 rectangle.....	693
A.2.32 RectangleRange.....	693
A.2.33 RectangleRange List.....	694
A.2.34 regExp	694
A.2.35 shape.....	694
A.2.36 ShapeRange.....	694
A.2.37 ShapeRangeList.....	694
A.2.38 sRGBColor	695
A.2.39 string.....	695
A.2.40 TimeRange.....	695
A.2.41 TransferFunction	695
A.2.42 URI	695
A.2.43 URL	696
A.2.44 XYPair	696

A.2.45 XYPairRange.....	696
A.2.46 XYPairRangeList.....	696
A.2.47 XPath.....	697
A.3 Enumerations and Lists	697
A.3.1 enumeration.....	697
A.3.2 enumerations.....	697
A.3.3 Defined JDF enumeration Data Types	697
A.3.3.1 JDFJMFVersion.....	697
A.3.3.2 NamedColor.....	697
A.3.3.3 Orientation	698
A.3.3.4 Side	698
A.3.3.5 WorkStyle	698
A.3.4 XYRelation.....	699
A.4 JDF File Formats	699
A.4.1 PNG Image Format	699
Appendix B Schema	701
B.1 Using xsi:type	701
B.1.1 Using xsi:type with JDF Nodes.....	701
B.1.2 Using xsi:type with JMF Messages	702
Appendix C Supported String and NMTOKEN values.....	703
C.1 StatusDetails Supported Strings	703
C.2 ModuleType Supported Strings	707
C.3 NotificationDetails	709
C.3.1 Predefined NotificationDetails	709
C.3.1.1 Barcode	710
C.3.1.2 FCNKey	710
C.3.1.3 SystemTimeSet	710
C.3.1.4 CounterReset	710
C.3.1.5 Error	710
C.3.1.6 Event	711
C.3.1.7 Milestone.....	711
C.4 MessageEvents and Milestone Values	712
C.5 Input Tray and Output Bin Names	714
Appendix D Supported Error Codes in JMF and Notification elements	717
Appendix E Color Adjustment Attribute Description and Usage . . .	719
E.1 Adjustment Using Direct Attributes	719

E.2 Adjustment using ICC Profile Attributes	720
E.3 Adjustment using an ICC Abstract Profile Attribute	720
E.4 Adjustment using an ICC DeviceLink Profile Attribute	720
Appendix F North American and Japanese Media Weight Explained ..	721
F.1 North American Media Weight	721
F.2 Japanese Media Weight	722
Appendix G Media Sizes	725
Appendix H FileSpec Attribute Examples for mimeType and mimeType-Version Attributes	729
Appendix I FileSpec Attributes and Container Subelement	735
I.1 FileSpec attribute value examples	735
I.2 Corresponding XML examples	736
I.3 Additional examples showing partitioning of FileSpec	737
I.4 Example of an Intent Job Ticket with a doubly nested ZIP packaging file ...	741
I.5 AppOS and OSVersion Attributes	742
Appendix J Generating strings with Format and Template	743
Appendix K Resolving RunList/@Directory and FileSpec/@URL URI references	745
K.1 Semantics of the RunList/@Directory attribute	745
Appendix L References	747
Appendix M JDF/CIP4 Hole Pattern Catalog	761
Appendix N Examples	771
N.1 Brief Example	771
N.1.1 Before Processing	771
N.1.2 After Processing	772
N.2 Product JDF	773
N.3 Spawning and Merging	774
N.3.1 Example 2 Component JDF before Spawning	774
N.3.2 Example 2 Component JDF Parent after spawning the cover node	775
N.3.3 Example 2 Component JDF spawned node	776

N.3.4 Example 2 Component JDF after merging	776
N.3.5 Example of a Partitioned ImageSetting Node before Spawning	777
N.3.6 The Spawned Cyan Partition of the ImageSetting Node.....	778
N.3.7 The Root Partitioned ImageSetting Node after Spawning.....	778
N.3.8 The Merged ImageSetting Node	779
N.4 RunList	780
N.5 Messages	782
N.5.1 Simple KnownMessages	782
N.5.2 Simple persistent channel	782
N.6 Stripping	783
N.6.1 Using Position	783
N.6.2 Multiple Bindery Signatures.....	784
N.6.3 Multisection Bindery Signatures	785
N.6.4 Multiple job parts in one imposition	785
N.6.5 FoldOuts.....	786
N.6.6 Multiple Web Layout.....	786
N.6.7 Stripping Process	788
N.7 DigitalDelivery Examples	790
Appendix O New, Deprecated, Modified, & Removed Items	797
O.1 Compatibility Warnings	797
O.2 New Items	797
O.3 Deprecated Items	809
O.4 Modified Items	811
Appendix P Deprecated Elements, JMF Messages, Processes and Resources	815
P.1 Deprecated Structures of JDF Nodes and Jobs	815
P.1.1 ResourceUpdate Elements.....	815
P.1.2 StatusPool and PartStatus	815
P.2 JMF Messaging Elements	816
P.2.1 Signal.....	816
P.2.2 NodeInfo	817
P.2.2.1 NodeInfo Query	818
P.2.2.2 NodeInfo Command	818
P.2.3 KnownJDFServices	820
P.2.4 QueueEntryStatus	821
P.3 Deprecated Processes	821

P.3.1 Packing.....	821
P.3.2 FilmToPlateCopying	822
P.3.3 PreflightAnalysis	822
P.3.4 PreflightInventory.....	824
P.3.5 PreflightProfile	825
P.3.6 Proofing	826
P.3.7 SoftProofing.....	827
P.3.8 IDPrinting.....	828
P.3.9 AdhesiveBinding.....	829
P.3.10 Dividing.....	829
P.3.11 LongitudinalRibbonOperations	830
P.3.12 SaddleStitching.....	830
P.3.13 SideSewing.....	830
P.4 Deprecated Resources	831
P.4.1 BindingIntent Deprecated Subelements	831
P.4.2 DeliveryIntent Deprecated Subelements	832
P.4.3 SizeIntent.....	833
P.4.4 AdhesiveBindingParams	833
P.4.5 DividingParams	834
P.4.6 IDPrintingParams	835
P.4.7 Layout Deprecated Subelement.....	845
P.4.8 LongitudinalRibbonOperationParams.....	845
P.4.9 MediaSource	846
P.4.10 PackingParams	847
P.4.11 PlateCopyParams.....	848
P.4.12 ProofingParams.....	849
P.4.13 SaddleStitchingParams	850
P.4.14 Sheet	851
P.4.15 SideSewingParams	852
P.4.16 Surface	853
Appendix Q List of Figures	855
Appendix R List of Tables	859




JDF Preface and User Overview

This specification is immense ... there 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:

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 third version of JDF (v.1.2) 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;



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?

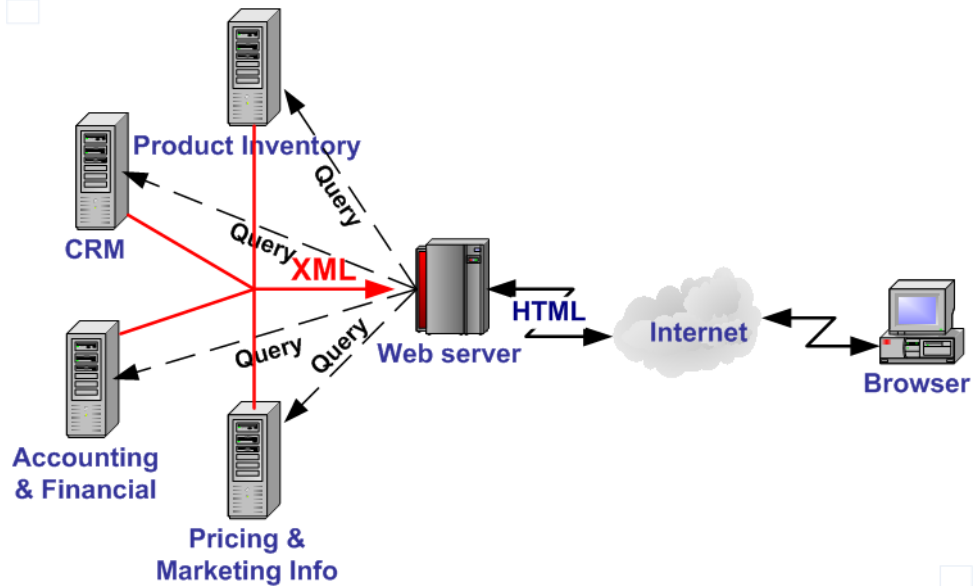
- A standard that can be applied to eliminate wasteful re-keying and redundancy of information; and
- A common computer language for printing and related industries as well as a platform for more effective communication.

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. 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 “middle-ware” 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:

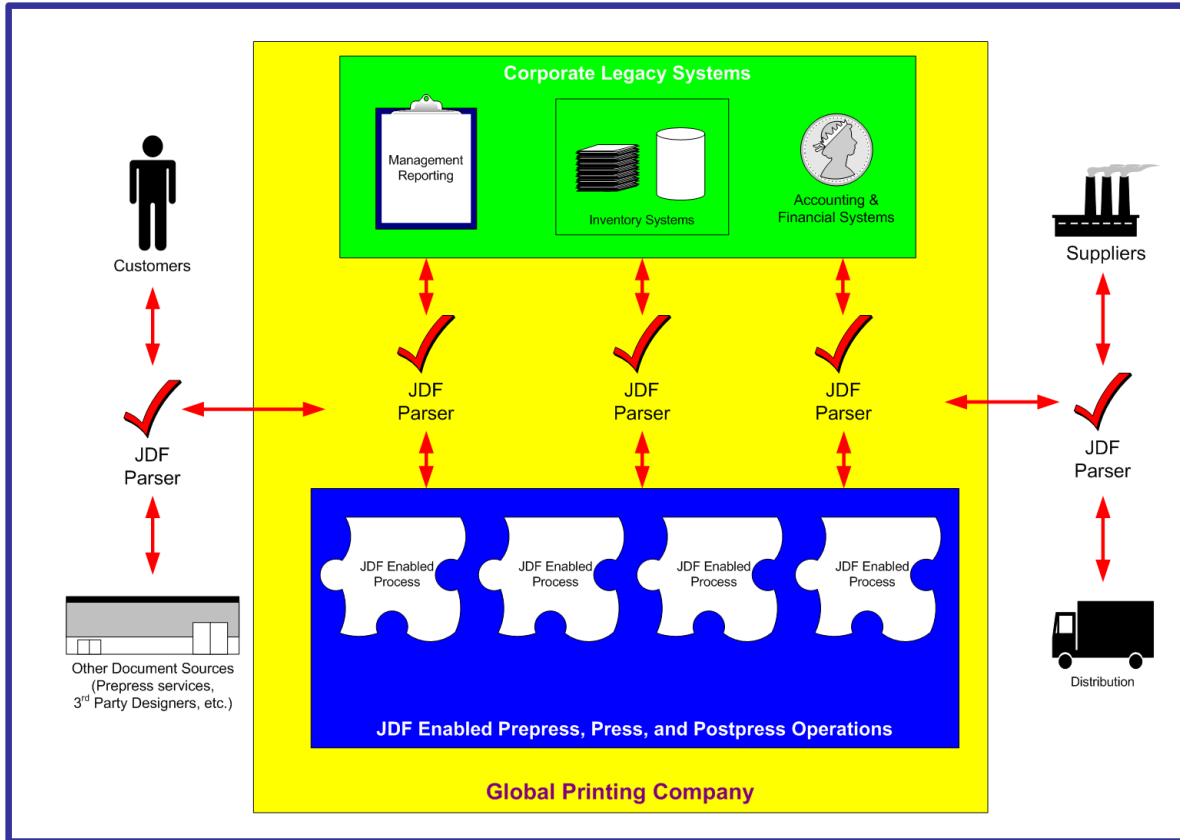
- 1 Acceptance checking of client job tickets;
- 2 Validation of JDF prior to or following transformation of data into and out of databases;
- 3 Ensuring that source job information is collected as a document is created (embedded in document layout software);
- 4 Determining if equipment reads and writes 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:



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/>



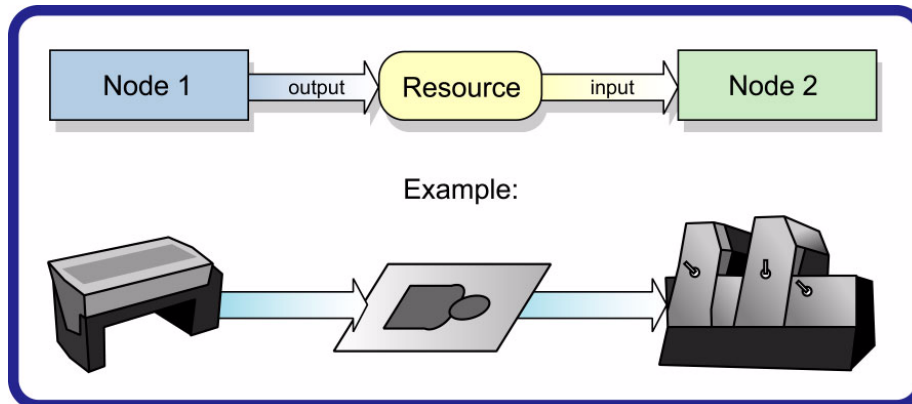
The JDF Concept. The JDF schema is quite complex and detailed—something best left to programmers, MIS personnel and XML experts. But the language and concepts behind JDF are quite simple and straightforward. The schema itself can be downloaded from the CIP4 Website, but is not part of this specification. Instead, this is your “cookbook.” 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 three primary components to JDF:

- 1 JDF itself,
- 2 The Job Messaging Format (JMF) and
- 3 The MIS system.

JDF is simply an exchange format for instructions and job parameters. You can use PDF or its standard variant (PDF/X), to relay production 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. As a matter of convention, processes such as preflighting, scanning, printing, cutting and so on are referred to as process *nodes*. 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.

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 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.



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.

Changes to JDF 1.3

JDF 1.3 includes both some wholly new material, as well as many improvements and refinements to JDF 1.2. A complete catalog of changes can be found in "New, Deprecated, Modified, & Removed Items" on page 797. You will also find [Modified in JDF 1.3](#) and [New in JDF 1.3](#) flags throughout this document. All deprecated Resources, Processes and other major deprecated sections have been removed to Appendix P, 'Deprecated Elements, JMF Messages, Processes and Resources' on page 815 to make the JDF standard easier to read.

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

- Packaging and Label related extensions include support for corrugated media, barcodes, box folder-gluer Devices and die cutters.
- Newspaper workflow and web-press specific extensions have been added.
- **CustomerInfo** and **NodeInfo** are now Resources instead of elements in the JDF. This allows for partitioning of generic information.

- The **Layout** structure that describes imposition has been streamlined to one partitionable Resource which allows for more flexible product specifications.
- Graphical elements can be individually tracked using the new **ContentList** resource.
- The flexibility of content format conversion has been enhanced with the new **PDLCreation** and **RasterReading** processes.
- JMF Message features related to HTTPS security and gang jobs have been added.
- Support for change order management has been added.

For a detailed list of modifications, see Appendix O, 'New, Deprecated, Modified, & Removed Items' on page 797.

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 facilitate 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 basis 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.

Chapter 1 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, how to use this document and some basic document navigational aids.

1.1 Background on JDF

JDF is an extensible, XML-based format built upon the existing technologies of CIP3's Print Production Format [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.

1.2 Document References

Throughout this specification references to other documents are indicated by short symbolic names inside square brackets, (e.g., [ICC.1]). Implementers ought to read and conform to such referenced documents when implementing a part of this specification with such a reference. The reader is directed to "References" on page 747 to find the complete set of JDF references and the full title, date, source and availability of all such references. In addition, this specification assumes that the reader has a basic awareness of or access to, the following documents.

Table 1-1: Basic references (Section 1 of 2)

Term	Definition
[XML]	<i>XML Specification</i> <i>Version 1.0 (Second Edition)</i> Date: 6 October 2000 Produced by: World Wide Web Consortium (W3C) Available at: http://www.w3.org/TR/REC-xml .

Table 1-1: Basic references (Section 2 of 2)

Term	Definition
[XMLNS]	<i>Namespaces in XML</i> Version (W3C Recommendation of 14 January 1999) Date: 14 January 1999 Produced by: World Wide Web Consortium (W3C) Available at: http://www.w3.org/TR/REC-xml-names/
[XPath]	<i>XML Path Language (XPath) Version 1.0</i> Version W3C Recommendation 16 November 1999 Date: 16 November 1999 Produced by: World Wide Web Consortium (W3C) Available at: http://www.w3.org/TR/xpath.html .
[XMLSchema]	<i>XML Schema Part 0+1+2: Primer, Structures and Datatypes</i> Version (W3C Recommendation of 02 May 2001) Date: 02 May 2001 Produced by: World Wide Web Consortium (W3C) XML Schema working group Available at: http://www.w3.org/TR/xmlschema-0/ , http://www.w3.org/TR/xmlschema-1/ and http://www.w3.org/TR/xmlschema-2/ .

1.3 Conventions Used in This Specification

This section contains conventions and notations used within this document.

1.3.1 Text Styles

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

- Elements are written in sans serif. Examples are Comment, CustomerInfo and ResourceLink elements.
- Attributes are written in italic sans serif. Examples are *Status*, *ResourceID* and *ID*.
- Resources are written in bold sans serif. Examples are **Imposition**, **Toner** and **ExposedMedia**.
- Processes are written in bold-italic sans serif. Examples are ***ColorSpaceConversion***, ***Rendering*** and ***Scanning***.
- Enumerative and Boolean values of attributes are written in italics. Examples are *true*, *Waiting*, *Completed* and *Stopped*.
- Standard bold text is used for the following purposes
 - to highlight glossary items. Examples are **device**, **element** and **job**.
 - to highlight defined items inside a table. An example is the data type **NMTOKEN** in the table in Section 1.5, Data Structures.
 - to highlight definitions of local terms. These are terms that are of local importance for a certain chapter or some sections inside a chapter. An example is a **spawned JDF** in Section 4.4, Spawning and Merging.
- For the benefit of those who are reading this document in PDF or online, cross-reference links are denoted by gray text. Examples are "Life Cycle of JDF" on page 117 and "Conventions Used in This Specification" on page 2. To follow a link, click the highlighted text.
- Also for the benefit of online readers, external hyperlinks are graphically designated. An example is <http://www.cip4.org>. To follow a link, click the highlighted text.



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.

1.3.2 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”), and
- 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. //ResourcePool indicates a ResourcePool 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:

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).

1.3.3 Callouts

[New in JDF 1.2](#)

To help the reader familiar with JDF/1.0 or JDF/1.1, this specification indicates additions, deprecations and clarifications using the following callouts. Please note that not all changes are identified with modified callout flags. 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 1-2: Callouts (Section 1 of 2)

Example	Callout Meaning
New in JDF 1.x	New sections, attributes/elements and attribute values.

Table 1-2: Callouts (Section 2 of 2)

Example	Callout Meaning
Deprecated in JDF 1.x	Deprecated sections, attributes/elements and attribute values.
Modified in JDF 1.x	Changed syntax or semantics of sections and attributes/elements, might include clarification as well.

1.3.4 Specification of Cardinality

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

The symbol **T** in the table below represents an attribute, element or resource. The symbol **T** consists of either a single name, such as “**RunList**” or a resource name followed by a parenthesized name, such as “**RunList (Document)**”. The name in parentheses “Document” identifies a particular resource instance when several of the same type exist in some context. For further details, see Section 6.1 “Process Template” and Section 7.2.1 “Process Resource Template”.

Table 1-3: Cardinality symbols

Notation	Description
T	T MUST occur exactly once and represents an attribute, element or resource.
T ?	T is OPTIONAL or is REQUIRED only in the circumstances explained in the description field. T represents an attribute, element or resource. 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 or resource.
T *	T occurs zero or more times, and represents an element or resource.
T = “V”	T is an OPTIONAL attribute, but has the specified default value <i>V</i> 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 MUST 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 “Conformance Requirements for Support of Attributes and Attribute Values” on page 8. This notation is only valid for XML attributes, not XML elements.

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 (Section 1 of 4)

Term	Definition
Abstract Element	Element that defines a data type but is not written as an element with that abstract element's name. For instance PlacedObject is an abstract element of a Layout ; MarkObject and ContentObject elements MAY exist in a Layout .
Agent	The component of a JDF-based workflow that writes JDF.
Attribute	An XML-based syntactic construct describing an unstructured characteristic of a JDF node or element .
Big job	The combined job that independent jobs are merged into in the case of independent spawning and merging.
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 .

Table 1-4: Glossary (Section 2 of 4)

Term	Definition
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 MUST use if an Agent omits an OPTIONAL attribute (as indicated by a “?” or <i>Attribute</i> = “ <i>DefaultValue</i> ” in this specification) from a JDF instance. See Section 1.4.2.1, Conformance Requirements for Support of Attributes and Attribute Values.
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.
Document set	A set of instance documents presumed to be related.
Element	An XML-based syntactic construct describing structured data in JDF.
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. A sheet of paper with no fold inside consists of two finished pages (“recto” and “verso” or front and back side). If there are folds seen in a sheet in the final product, the number of finished pages of one sheet is given by $2*(X+1)*(Y+1)$, where X denotes the number of folds in X direction and Y denotes the number of folds in Y direction, each seen in the completely opened sheet. Examples: One sheet in a book has two finished pages, one front, one back; a brochure with one fold inside has four finished pages.
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.
Gear side	Gear side is the side of a Machine , where the drives and gear are mounted. Gear side is opposite to Operating side
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 .
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.
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.
JMF	Job Messaging Format. A communication format with multi-level capabilities. Structures information between MIS and controllers . There is also the JMF element, which is a top-level element within JDF.
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.
Leaf	Both the recto and verso finished pages on one piece of paper with “leaves” being the plural usage.

Table 1-4: Glossary (Section 3 of 4)

Term	Definition
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.
MIS	Management Information Systems. The functional part of a JDF workflow that oversees all processes and communication between system components and system control.
Node	The JDF element type detailing the resources and process specification needed to produce a final or intermediate product or resource.
Operating side	Operating side is the side of a Machine , where the operator works. Operating side is opposite to Gear side .
Partition	Enumerations of the <i>PartIDKeys</i> attribute of the Resource element used to identify individual physical and logical parts of a job. (See Table 3-25, "Partitionable resource element," on page 86.)
Partitioned resource	Structured resource that represents multiple physical or logical entities, such as 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®.
Process	An individual step in the workflow.
Queue	Entity that accepts job entries via a JMF messaging system.
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).
Resource	A physical or conceptual entity that is modified or used by a node . Examples include paper, images or process parameters.
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.
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.)
Signature	A signature is a set of printed sheets that can be folded or unfolded. 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.
Small job	An independent job that is merged into a big job .
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 either a Sheet or a Signature

Table 1-4: Glossary (Section 4 of 4)

Term	Definition
Tag	A syntactic construct that marks the start or end of an element .
Work center	An organizational unit, such as a department or a subcontracting company, that can accomplish a task.



Getting Pages Straight

The term “page” is very common in everyday conversations regarding printing, but in context of a technical specification for graphic arts it can be misleading. Is page “1” of a document the same as the first page or page one of an imposition or the first page numbered one? The above glossary includes more specific definitions, but, in general, a “reader page” is as the reader sees it in the final product, and a “finished page” is one side of the final cut, folded and bound product. “Recto” and “verso” finished pages describe the forward-facing and away-facing pages of a “leaf,” meaning both recto and verso finished pages of one a piece of paper with “leaves” being the plural of leaf.

A “form” is an imposed (ordered) collection of finished pages set for printing on a “sheet” which is the printer’s roll of paper or paper cut for press size. Sheets 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.) And finally, a “signature” is the printed (folded or yet to be folded) sheet and a “surface” is a single side of either a sheet or a signature.

Finished 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.” It is also a common convention that the page count for a book does not include the cover pages. Hence, a book might be described as a “72-page book, plus four cover pages” or just “plus cover.” Cover pages might be referenced as “cover 1” (front cover), “cover 2” (inside of front cover), “cover 3” (inside of back cover) and “cover 4” (back cover).

Special arrangements, such as over-covers, wraps, and glue on pages applied to covers are treated as inserts and other furnished material that is bound, but not printed, (e.g., treated as separate job parts until bindery).

Where the word “page” is used in this document (as opposed to finished page or reader page), it means “finished page.”

1.4.1 Conformance Terminology

The words “MUST”, “MUST 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 (Section 1 of 2)

Term	Meaning
MUST or REQUIRED	Means that the definition is an absolute requirement of the specification.
MUST 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 must 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.

Table 1-5: Conformance terminology (Section 2 of 2)

Term	Meaning
MAY, NEED NOT or OPTIONAL	<p>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 <i>Attribute</i> = "DefaultValue" in this specification). See Section 1.3.4, Specification of Cardinality</p> <p>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) MUST 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 MUST be prepared to interoperate with an Agent implementation which does not supply the option in the JDF instance.</p>

1.4.2 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.

1.4.2.1 Conformance Requirements for Support of Attributes and Attribute Values

If a JDF Consumer supports an attribute, it MUST 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 MUST 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 1.3.4, 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 1.3.4). 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 MUST supply the Default value when the attribute is omitted from the JDF instance. Such an indicated default value MUST 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 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.

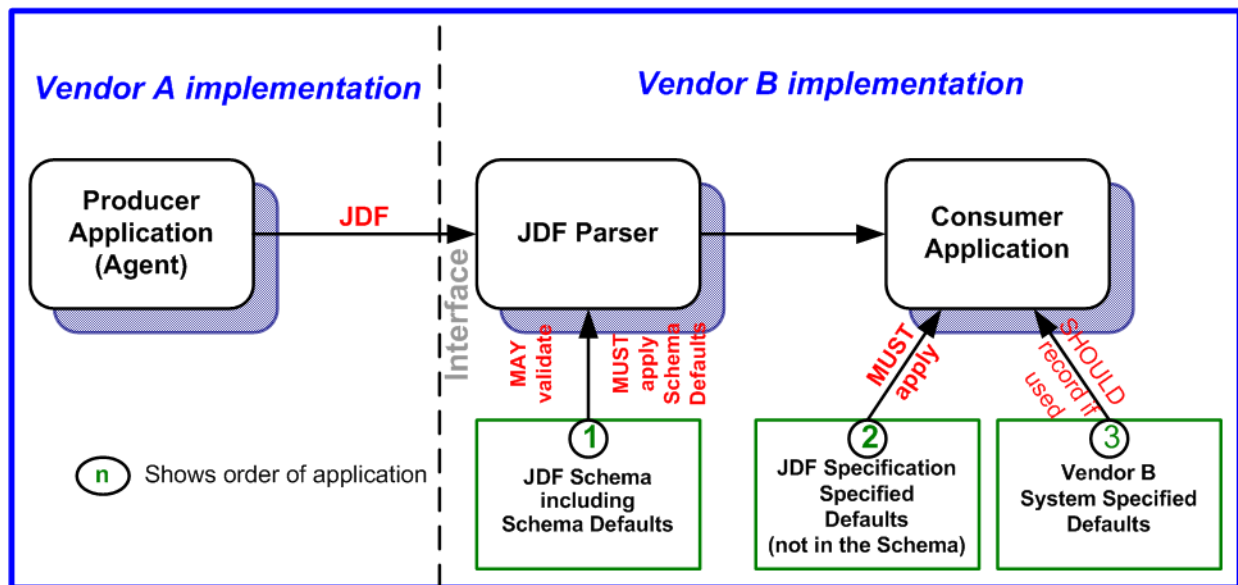


Figure 1-1: Handling of default values of JDF attributes.

1.4.2.2 Conformance Requirements for Support of Elements

If a JDF Consumer supports an element, it

- 1 MUST support all of the attributes (see Section 1.4.2.1) defined for that element that an Agent is REQUIRED to include in the element instance attributes with either no marks or a “+” as defined in Section 1.3.4, and
- 2 SHOULD support the *SettingsPolicy*, *BestEffortExceptions*, *MustHonorExceptions* and *OperatorInterventionExceptions* (see Section 3.1.1, Generic Contents of JDF Elements) attributes and all of their defined values. These attributes control the policy that a JDF Consumer MUST follow when it encounters unsupported settings, (i.e., subelements, attributes or attribute values in the resource.)

1.4.2.3 Conformance Requirements for Support of Processes

All processes are OPTIONAL for a JDF Consumer to support. However, a Device MUST support at least one process or a combined process. If a JDF Consumer supports a process, it

- 1 MUST support all of the input and output ResourceLink elements and referenced resources as described in Section 1.4.2.2 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 1.3.4) 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 MUST ignore all other processes, independent of the *SettingsPolicy* attribute for those other processes.

1.4.2.4 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 1.4.2.3 apply. If a JDF Consumer supports a combined process, it

- 1 MUST support all of the input resources as defined in Section 1.4.2.2 that this specification defines for the *first* process in the combined process node, (i.e., the first process listed in the *Types* attribute),
- 2 MUST support all of the output resources as defined in Section 1.4.2.2 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 node),
- 4 MUST support resources in intermediate process steps that are *not* used as exchange resources between processes in the process chain of the combined process.

1.4.3 Conformance to SettingsPolicy

The *SettingsPolicy*, *BestEffortExceptions*, *MustHonorExceptions* and *OperatorInterventionExceptions* attributes are defined in Table 3-1, “Any element (generic content),” on page 37. 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.

1.5 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 “Encoding” on page 687.

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 “Encoding” on page 687 for all data type definitions.

Table 1-6: JDF data types (Section 1 of 3)

Data Type	Description
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 MUST be a UTC time or a local time that includes the time zone.
DateTimeRange	Two dateTimes 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 DateTimeRanges.
double	Corresponds to [IEEE754] double-precision, 64-bit floating point type, including special tokens <i>INF</i> and <i>-INF</i> . 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 doubles. Note that this data type was named NumberList prior to JDF 1.2.
DoubleRange New in JDF 1.2	Two doubles separated by a “~” (tilde) character that defines the closed interval of the two. Note that this data type was named NumberRange prior to JDF 1.2.

Table 1-6: JDF data types (Section 2 of 3)

Data Type	Description
DoubleRangeList New in JDF 1.2	Whitespace-separated list of double and DoubleRange values. Note that this data type was named NumberRangeList prior to JDF 1.2.
duration	Represents a duration of time.
DurationRange	<i>DurationRange</i> is used to describe 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.
element	Structured data. The specific data type is defined by the element name.
enumeration	Limited set of NMTOKEN (see below).
enumerations	Whitespace-separated list of enumeration data types.
gYearMonth	Represents a specific Gregorian month in a specific Gregorian year.
hexBinary	Represents arbitrary hex encoded binary data.
ID	Unique identifier as defined by [XML] (see Section 1.2, Document References). MUST 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 “~” 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 3.12, JDF Versioning for a discussion of versioning in JDF.
JDFJMFVersions	Whitespace separated list of JDFJMFVersion.
LabColor	Represents a Lab color specification.
language	Represents a language and country code (for example, en-US) for a natural language.
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 numbers representing a coordinate transformation matrix.
NamedColor	Represents a color definition by name. A list of valid NamedColor values is provided in Section A.3.3.2, NamedColor.
NameRange	Two NMTOKEN separated by a “~” (tilde) character that define an interval of NMTOKEN.
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 .
Orientation New in JDF 1.2	Enumeration that specifies named orthogonal two-dimensional orientations.
Orientations New in JDF 1.2	Whitespace separated list of Orientation enumerations that specify named orthogonal two-dimensional orientations.
PDFPath	Whitespace-separated list of path operators as defined in PDF.
rectangle	Whitespace-separated list of four numbers representing a rectangle.

Table 1-6: JDF data types (Section 3 of 3)

Data Type	Description
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 numbers 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.
sRGBColor	Represents an sRGB color specification.
string Modified in JDF 1.2	Character strings without tabs or line feeds. Corresponds to the standard XML normalized-String data type [XMLSchema].
telem	Text elements that contain larger chunks of character data and MAY include line feeds.
text	Text data contained in a telem (text element).
TimeRange	Two dateTimes 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 numbers 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).
XPath	Represents an XPath expression of an XML node set (attributes or elements), boolean, number or string.[XPath]
XYPair	Whitespace-separated list of two numbers. Unless specified otherwise in the attribute Description, these two numbers 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 numbers. One of a set of NMTOKEN values, a list of valid values is provided in "XYRelation" on page 699.

1.6 Units

JDF specifies most values in default units. That means that an implementation **MUST** use the defined default units and **MUST 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 **MUST** 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 in the unit attribute of resources.

Table 1-7: Units used in JDF

Measurement	Unit	Representation	Remarks
Length	point (1/72 inch)	pt	Used for all except microscopic lengths (see below)
	micron	mu	Used for microscopic lengths — where used (instead of points) it will be explicitly stated in the definition of the item. See Media/@Thickness .
Volume	liter	l	—
Weight	gram	g	—
Area	m ²	m2	—
Resolution	dpi	dpi	The dots per inch (dpi) for print output and bitmap image (TIFF, BMP, etc.) file resolution.
Line Screen	lpi	lpi	The lines per inch (lpi) for conventionally screened halftone, screened grayscale and screened monotone bitmap images.
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.
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) are not equivalent.
Paper weight	g/m ²	g/m2	—
Speed	units/hour	*/h	Replace the “*” in the representation with the appropriate unit
Temperature	C° (Celsius)	C	degree centigrade
Angle	degrees°	degree	—
Countable Objects	1	—	Countable objects, such as sheets, have no unit specification.

1.6.1 Counting in JDF

Zero-based indices MUST be used in JDF. Thus the first index is 0, the second index is 1 etc.

Chapter 2 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 3, Structure of JDF Nodes and Jobs.

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. The latter is abbreviated as telem. For more information about elements, see Section 3.1.2, JDF Node Attributes and Elements.

**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 3.1.2, JDF Node Attributes and Elements. Note that an attribute with an empty (zero length) value string **MUST 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/><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/>
    <MoreAnys>
      <Descendent/>
    </MoreAnys>
  </Any>
</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.

2.1.2 Workflow Component Roles

The four components to create, modify, route, interpret and execute a JDF job are known as agents, controllers, 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. The intention is to name the component mechanisms needed for the interaction of actual components in a workflow during the course of a JDF job. In practice, it is very likely that individual system components will include a mixture of the capabilities described in the following sections. For example, many controllers are also agents.

2.1.2.1 Machines

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. Computerized workstations, whether run through automated batch files or controlled by a human worker, are also considered machines if they have no JDF interface.

2.1.2.2 Devices

The most basic function of a device is to execute the information specified by an **agent** and routed by a **controller**. Devices **MUST** 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 controllers.

2.1.2.3 Agents

Agents in a JDF workflow are responsible for writing JDF. An agent has the ability to create 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.



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 systems.

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 Controllers

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, **MUST** have device capability. Therefore, controllers **MUST** be able to work in collaboration with other controllers. In order to communicate with one another, and to communicate with devices, controllers **MUST** 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.

2.1.2.5 Management Information Systems—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 **MUST** 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 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 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 3.10.1, Audit Elements.) 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.

2.1.2.6 System Interaction

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



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.

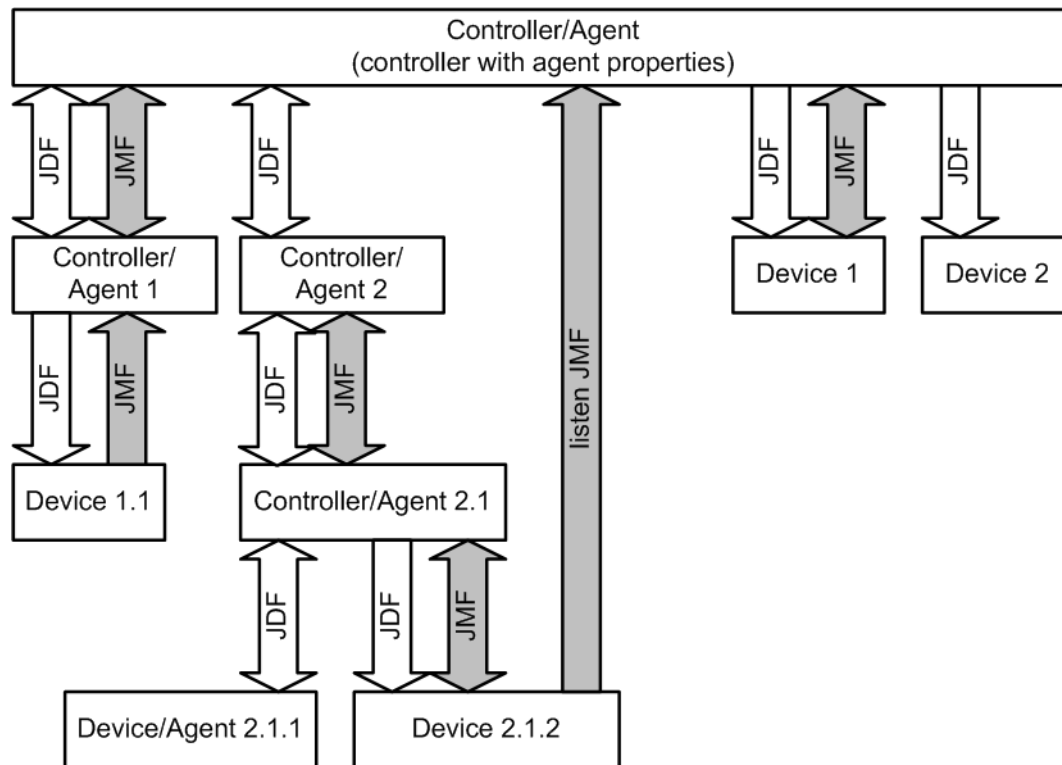


Figure 2-1: Example of JDF and JMF workflow interactions

2.2 JDF Workflow

JDF does not dictate that a workflow be constructed in any pre-specified way for it to be usable. 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, RIPing, 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. Therefore, 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, below, shows a simple example of a tree of nodes.

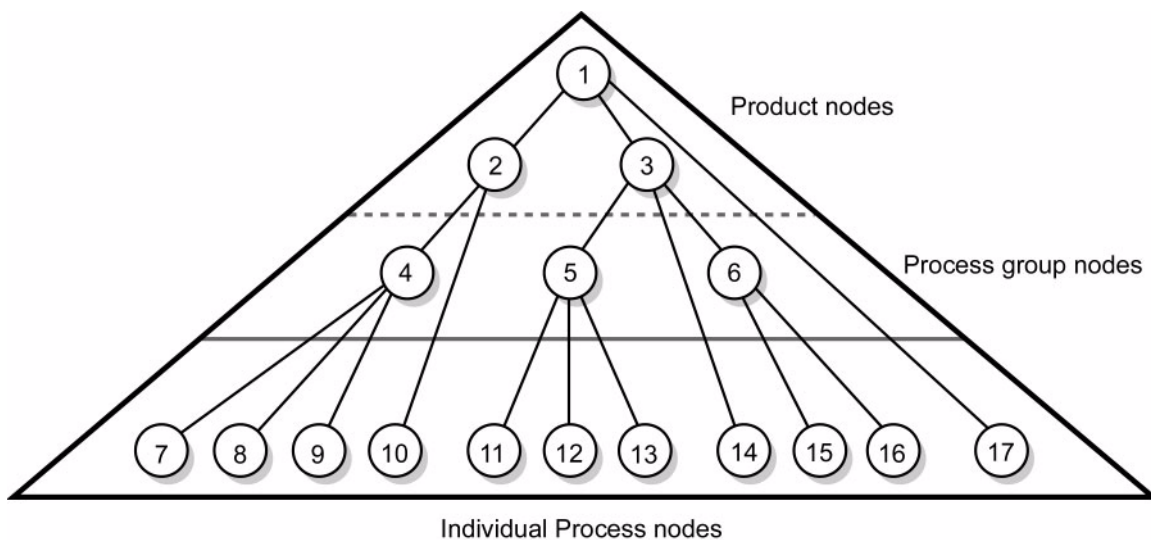


Figure 2-2: JDF tree structure

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 (Section 1 of 2)

Node #	Meaning
1	Entire book
2	Cover
3	Contents
4	Production of cover
5	Production of all color pages

Table 2-1: Information contained in JDF nodes, arranged numerically (Section 2 of 2)

Node #	Meaning
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
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 (Section 1 of 2)

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

Table 2-2: Information contained in JDF nodes, arranged by group (Section 2 of 2)


Process Group	Node #	Meaning
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 **MUST 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 shows an example of a hierarchical structure.



Trees & Nodes

In the real world, if you wanted to scan a photo, you would probably go to the prepress department to find a scanner. JDF uses this same common-sense approach to organization. Processes (nodes) are organized into a hierarchy (tree). Consider your own operations. If you were to group your departments, equipment and processes into an “org chart,” what would it look like?.

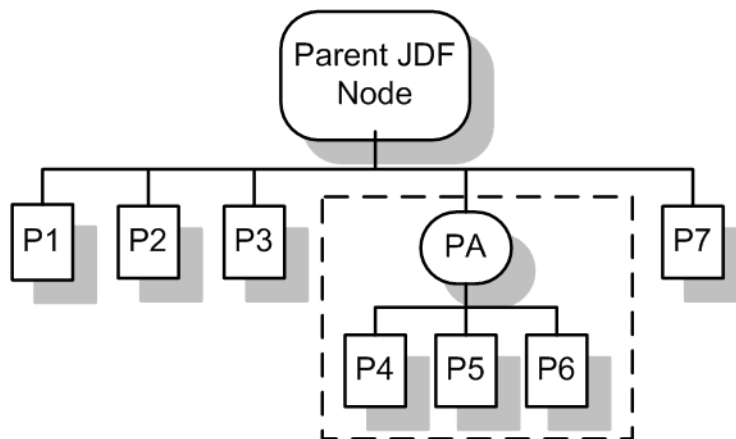


Figure 2-3: Example of a hierarchical tree structure of JDF nodes

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). This interrelationship is known as resource linking.

With resource linking, complex networks of processes can be formed. The Figure 2-4 displays an alternate representation of the process described in Figure 2-3. Whereas Figure 2-3 represents a hierarchical structure, Figure 2-4 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.

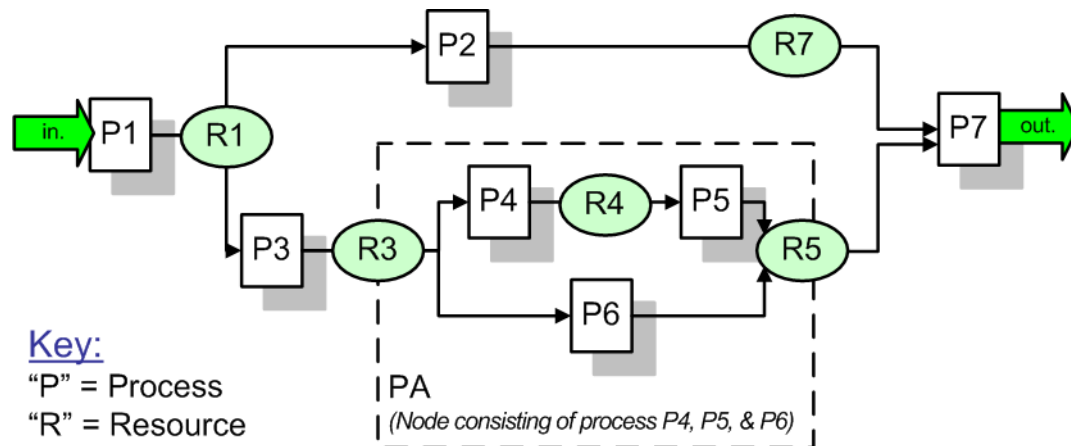


Figure 2-4: Example of a process chain linked by input and output resources

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 **MUST** be familiar with these local constraints. They **MUST** 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 routing 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, JDF Messaging with the Job 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 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`, `Resource` and `Occupation` messages allow the device or work center to communicate quasi real-time¹ 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, JDF Messaging with the Job 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 physical resource 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.)

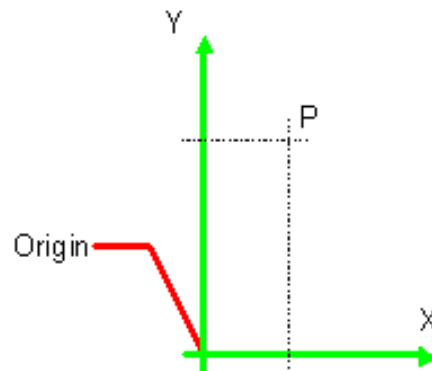


Figure 2-5: Standard coordinate system

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 assuming an idealized Device that is defined in the process coordinate system for each process that the Device implements. A real Device **MUST** 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 2.5.6, *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.

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.

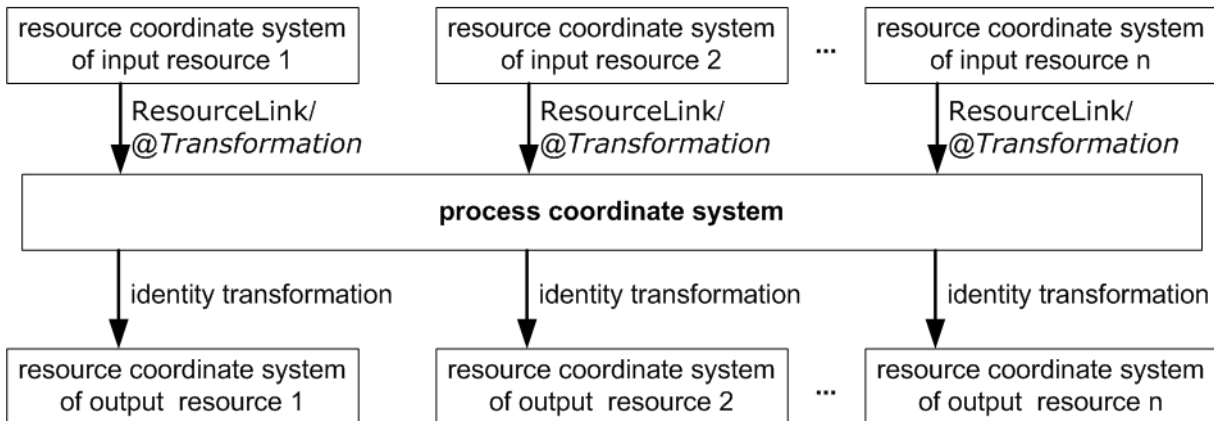


Figure 2-6: Relation between resource and process coordinate systems

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) **MUST** 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 *LayoutPreparation* 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 **MUST** be applied prior to defining the source coordinate system; for instance:

- PDF: the rotation defined by the *Rotate* key **MUST** 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 **MUST** 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 2.5.6, Homogeneous Coordinates)
Orientation	" <i>Rotate180</i> " or " <i>Flip90</i> "

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 the section “**ResourceLinkPool** and **ResourceLink**” on page 63.

2.5.3.1 Coordinate Systems of Combined Processes

[New in JDF 1.2](#)

Combined Processes (See “Combined Process Nodes” on page 49.) combine multiple individual processes and thus also the 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 “Overlapping Processing Using Pipes” on page 127.) 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 MUST be “*Complete*”.

2.5.3.2 Coordinate System Transformations

The following table shows some matrices that can be used to change the orientation of a physical **Resource**. Most of the transformations require the X- (**w**) and the Y-dimension (**h**) of the **Component** as specified in the **Dimension** element. 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
<i>Rotate0</i>		$1\ 0\ 0\ 1\ 0\ 0$ No Action	
<i>Rotate90</i>		$0\ 1\ -1\ 0\ h\ 0$ 90° Counterclockwise Rotation	
<i>Rotate180</i>		$-1\ 0\ 0\ -1\ w\ h$ 180° Rotation	
<i>Rotate270</i>		$0\ -1\ 1\ 0\ 0\ w$ 270° Counterclockwise Rotation	
<i>Flip0</i>		$1\ 0\ 0\ -1\ 0\ h$ Flip around X	
<i>Flip90</i>		$0\ -1\ -1\ 0\ h\ w$ 90° Counterclockwise Rotation + Flip around X	
<i>Flip180</i>		$-1\ 0\ 0\ 1\ w\ 0$ 180° Rotation + Flip around X	
<i>Flip270</i>		$0\ 1\ 1\ 0\ 0\ 0$ 270° Counterclockwise Rotation + Flip around X	

2.5.4 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. 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	ConventionalPrinting	Component
FoldingParams Component	Folding	Component
CollectingParams Component	Collecting	Component
SaddleStitchingParams Component	SaddleStitching	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.)

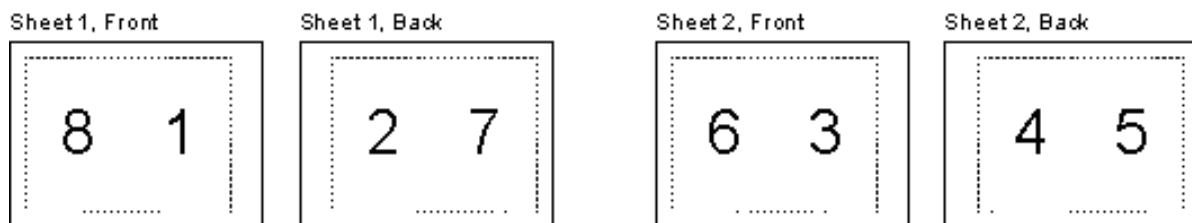


Figure 2-7: Layout of simple saddle stitched brochure (product example)

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:

$$P_{\text{Surface}} = P_{\text{Page}} \times CTM_{\text{Page}} + [\text{SurfaceContentsBox}_{x_{\text{lowerleft}}} \text{SurfaceContentsBox}_{y_{\text{lowerleft}}} 0]$$

Figure 2-8: Equation for Surface Coordinate System Transformations

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

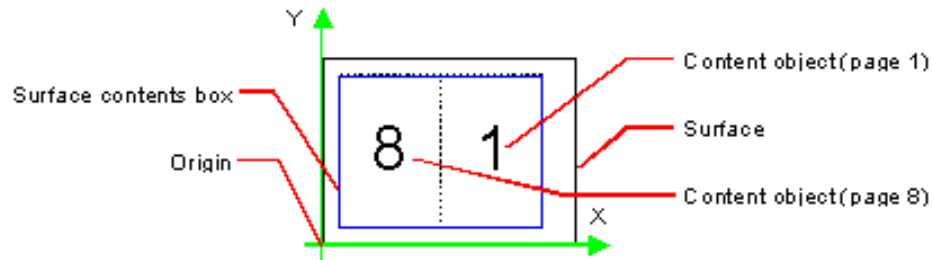


Figure 2-9: Surface coordinate system

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 image setting 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 run list 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*.

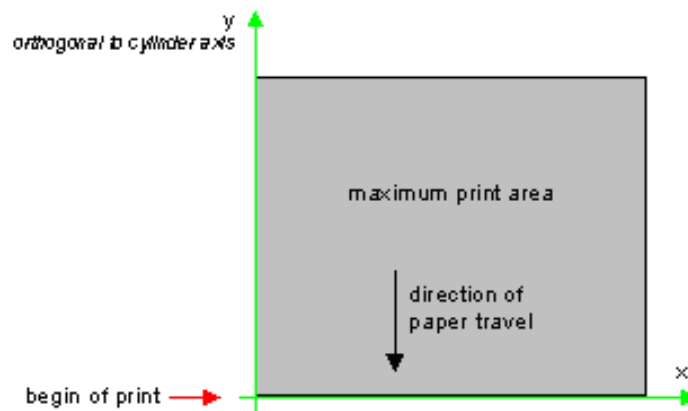


Figure 2-10: Press coordinate system used for sheet-fed printing

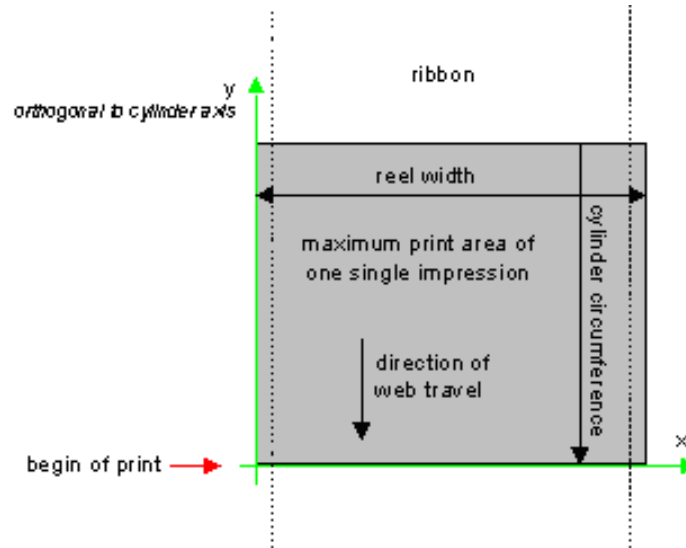


Figure 2-11: Press coordinate system used for web printing

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 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.)

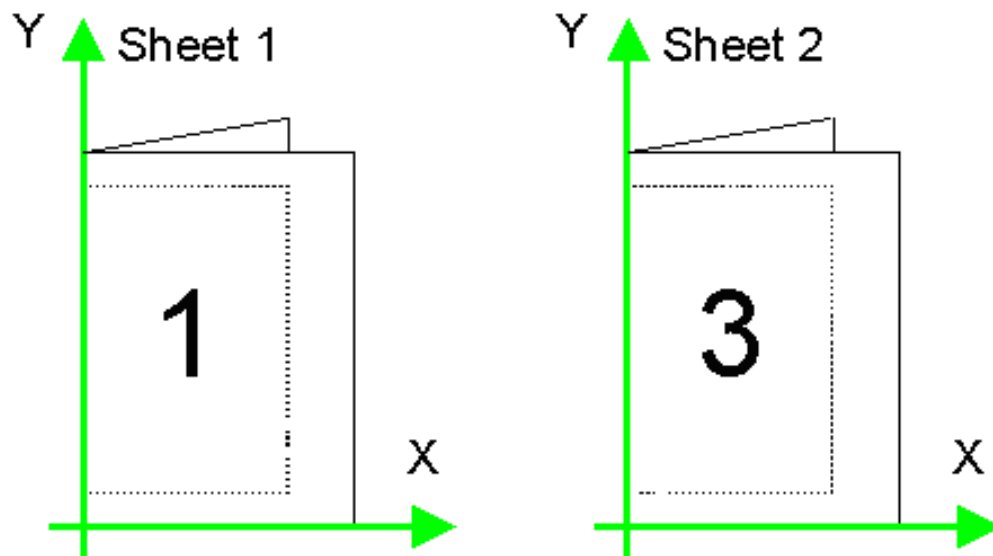


Figure 2-12: Coordinate systems after Folding (product example)

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 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.

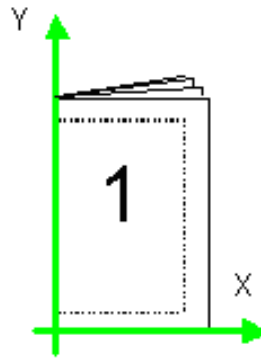


Figure 2-13: Coordinate systems after Collecting (product example)

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 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. 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 components as input and output resources. The *ResourceLink* of each input and output component contains a *Transformation* attribute or an *Orientation* attribute. The *Transformation* attribute is to be used if the width and the height of the component are known or a non-orthogonal rotation is needed. Otherwise the *Orientation* attribute is to 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 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.

2.5.5 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 particular machines 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.

2.5.6 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”.

$$\text{Trf} = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix} \quad \text{would in JDF be written as “a b c d e f”}$$

Some often used transformation matrices are

$$\text{Trf} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{identity transformation}$$

$$\text{Trf} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{bmatrix} \quad \text{translation by dx, dy}$$

$$\text{Trf} = \begin{bmatrix} \cos \varphi & \sin \varphi & 0 \\ -\sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{rotation by } \varphi \text{ degrees counter-clockwise}$$

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

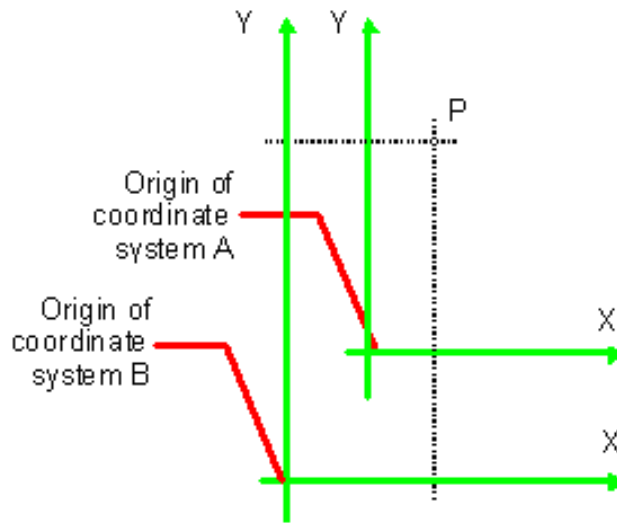


Figure 2-15: Transforming a point (example)

$$P_A = (30, 100)$$

$$P_A = [30 \quad 100 \quad 1]$$

$$P_B = P_A \times \text{Trf}$$

in JDF, *Trf* is written as "1 0 0 1 40 60"

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

$$P_B = (70, 160)$$

$$P_B = [70 \quad 160 \quad 1]$$

Chapter 3 Structure of JDF Nodes and Jobs

Introduction

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 or a process.

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 "Processes" on page 219.

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 1.5, *Data Structures*. Only the data types **element** or **telem** (which is short for 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 elements, which are defined in Section 3.8, *ResourceLinkPool* and *ResourceLink*.

Figure 3-1 is a schematic structure of the JDF node type. In this figure, generic attributes and elements (see Section 3.1.1, *Generic Contents of JDF Elements*) are inserted only in the JDF root node. The element types that are displayed in this figure are described in the subsequent sections. Figure 3-1 and other similar diagrams described 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.

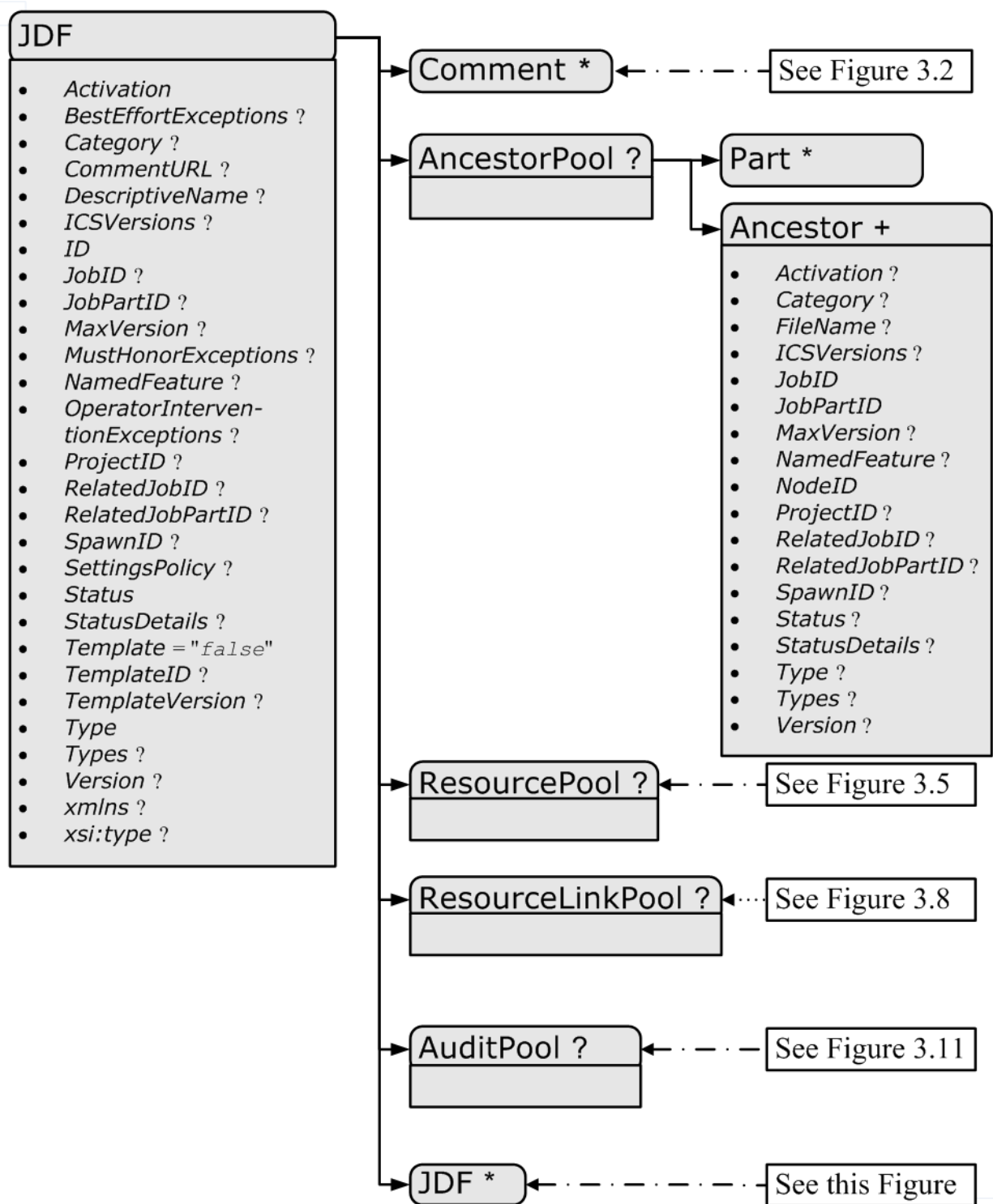


Figure 3-1: JDF node – a diagram of its structure

3.1 JDF Nodes

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.

3.1.1 Generic Contents of JDF 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) (Section 1 of 2)

Name	Data Type	Description
<i>BestEffortExceptions</i> ? New in JDF 1.1	NMTOKENS	The names of the attributes in this element that are to have the best effort policy applied when <i>SettingsPolicy</i> is not <i>BestEffort</i> .
<i>CommentURL</i> ?	URL	URL to an external, human-readable description of the element. Note that <i>CommentURL</i> MAY be specified within a <i>Comment</i> .
<i>DescriptiveName</i> ?	string	Human-readable descriptive name of the JDF element, (e.g., a descriptive name of a resource, process or product). It is strongly RECOMMENDED to supply <i>DescriptiveName</i> in all JDF nodes, <i>Quantity</i> resources (for example: Component resources) and <i>Handling</i> resources (for example, ExposedMedia) for communication from applications to humans in order to reference the process or resource.
<i>MustHonorExceptions</i> ? New in JDF 1.1	NMTOKENS	The names of the attributes in this element that are to have the <i>MustHonor</i> policy applied when <i>SettingsPolicy</i> is not <i>MustHonor</i> .
<i>OperatorInterventionExceptions</i> ? New in JDF 1.1	NMTOKENS	The names of the attributes in this element that are to have the operator intervention policy applied when <i>SettingsPolicy</i> is not <i>OperatorIntervention</i> . If a device has no operator intervention capabilities, <i>OperatorIntervention</i> is treated as <i>MustHonor</i> .

Table 3-1: Any element (generic content) (Section 2 of 2)

Name	Data Type	Description
<i>SettingsPolicy</i> ? New in JDF 1.2	enumeration	<p>The policy for this element indicates what happens when unsupported settings, (i.e., subelements, attributes or attribute values), are present in the element. Possible values are:</p> <p><i>BestEffort</i> – Substitute or ignore unsupported attributes, attribute values, default attribute values or elements, and continue processing the job.</p> <p><i>MustHonor</i> – Reject the job when any unsupported attributes, attribute values or elements are present.</p> <p><i>OperatorIntervention</i> – Pause job and query the operator when any unsupported attributes, attribute values or elements are present. If a device has no operator intervention capabilities, <i>OperatorIntervention</i> is treated as <i>MustHonor</i>.</p> <p>If not specified, <i>SettingsPolicy</i> is inherited from the parent element, and if not specified in the parent element or further superior element, the default value defaults to "<i>BestEffort</i>".</p> <p>For additional details on <i>SettingsPolicy</i>, see "Conformance to SettingsPolicy" on page 10.</p>
Comment *	telem	<p>Any human-readable text. The Comment element is different from an XML comment <code><!-- XML Comment --></code>. 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.</p> <p>Comments MUST NOT be nested within Comment elements.</p>

Table 3-2: Comment element (Section 1 of 2)

Name	Data Type	Description
<i>AgentName</i> ? 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.
<i>AgentVersion</i> ? 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.
<i>Attribute</i> ? New in JDF 1.1	NMTOKEN	<p>Name of the attribute in the parent element of the Comment element that this Comment refers to. <i>Attribute</i> SHOULD include the namespace prefix if the attribute is in a non-JDF namespace. If omitted, the Comment refers to the entire parent element. <i>Attribute</i> 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.</p> <p>Note: <i>Attribute</i> 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.</p>

Table 3-2: Comment element (Section 2 of 2)

Name	Data Type	Description
<i>Author</i> ? New in JDF 1.3	string	Text that identifies the person who created the Comment.
<i>Box</i> ?	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 <i>Path</i> attribute.
<i>ID</i> ? New in JDF 1.3	ID	Identification that is used to reference the Comment.
<i>Language</i> ?	language	Human readable language of the Comment.
<i>Name</i> = "Description" Modified in JDF 1.2	NMTOKEN	<p>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. Predefined values include:</p> <p><i>Description</i> – 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 3-32, "Notification audit element," on page 103).</p> <p><i>Instruction</i> – Message to the operator that contains information regarding the processing of the job. New in JDF 1.2</p> <p><i>JobDescription</i> – Description of the Job. A Comment element that contains <i>Name</i> = "JobDescription" MUST be specified only in a JDF node or CustomerInfo resource. See also CustomerInfo/@CustomerJobName in Section 7.2.45, CustomerInfo. New in JDF 1.2</p> <p><i>OperatorText</i> – Message from the operator that contains information regarding the processing of the job. New in JDF 1.2</p> <p><i>Orientation</i> – Description of the orientation of a physical resource.</p> <p><i>TemplateDescription</i> – Description of the job ticket template. A Comment element that contains <i>Name</i> = "TemplateDescription" MUST be specified only in the root JDF node. New in JDF 1.2</p> <p><i>UserText</i> – Message to a user that contains information regarding the processing of the job. Used in CustomerInfo/CustomerMessage. See "CustomerInfo" on page 54. New in JDF 1.2</p>
<i>Path</i> ?	PDFPath	<p>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 physical resources, Layout resources and resources that are related to Layout, <i>Path</i> 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 <i>Path</i> is defined in.</p> <p>Note that there are cases where a coordinate system is not available and therefore defining <i>Path</i> is NOT RECOMMENDED, (e.g.: CustomerInfo.)</p>
<i>TimeStamp</i> ? 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. Thus carriage returns, line feeds or tabs MAY be lost.

The following figure shows the structure of the generic content defined above.

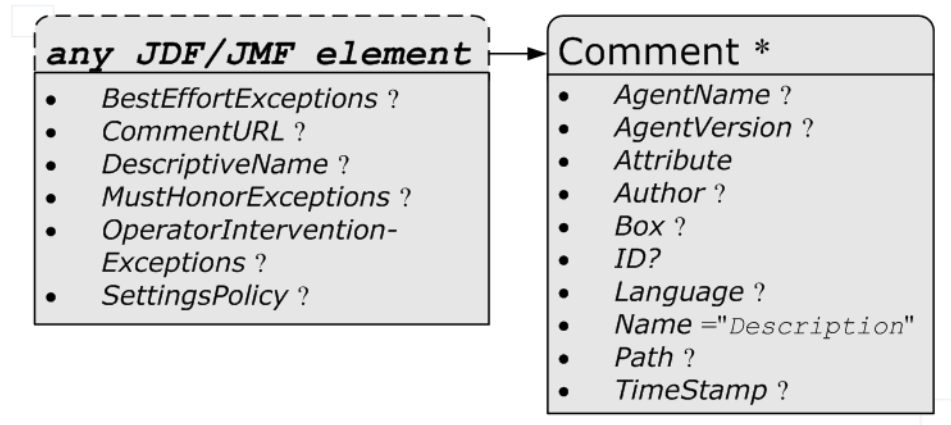


Figure 3-2: Any-element (generic content) – a diagram of its structure

3.1.2 JDF Node Attributes and Elements

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, below, **MUST** 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 or process the JDF node represents. As is detailed in Section 3.2, *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.

The table "JDF node" on page 41 contains a fourth column that 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 heading for this column is "S" which is short for "Scope and Position." The following abbreviations are defined:

Table 3-3: Definition of "Scope" Terms used in Table 3-4 on page 41

Abbreviation	Definition	Description
D	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.
L	Local	The content is only valid locally, within the node where the content is defined.
R	Root	The attribute MUST 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, <i>Spawning and Merging</i> . All attributes and elements listed in subsequent chapters SHOULD be considered local unless otherwise noted.

Table 3-4: JDF node (Section 1 of 5)

Name	Data Type	S Description																					
<p>Activation ? Modified in JDF 1.1</p>	enumeration	<p>D Describes the activation status of the JDF node. Allows for a range of activity, including deactivation and test running. Possible values, in order of involvement from least to most active, are:</p> <p><i>Inactive</i> – The node and all its descendents MUST NOT be executed or tested. This value is set if certain parts of a JDF job MUST NOT be executed or tested.</p> <p><i>Informative</i> – The JDF ticket is for information only. If a job is <i>Informative</i>, it MUST NOT be processed. Jobs with Activation = "<i>Informative</i>" 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 <i>Informative</i>. When a new job ticket with an identical ID attribute and a higher Activation is submitted to a Device, that JDF job ticket MUST replace the JDF job ticket that was submitted to the Device with an Activation of <i>Informative</i>.</p> <p><i>Held</i> – Execution has been held. If a job is <i>Held</i>, it MUST NOT be processed until its Activation is changed to <i>Active</i>.</p> <p><i>TestRun</i> – 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 <i>Active</i>.</p> <p><i>TestRunAndGo</i> – Similar to <i>TestRun</i>, but requests a subsequent automatic start if the test run has been completed successfully.</p> <p><i>Active</i> – The default value if not specified in a parent node – The node MUST be executed according to the steps specified in "Determining Executable Nodes" on page 123.</p> <p>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 <i>Inactive</i>, the node itself is <i>Inactive</i>. If no ancestor is <i>Inactive</i> but any ancestor is <i>Informative</i>, the node is <i>Informative</i> unless the node itself is <i>Inactive</i>. If no ancestor is <i>Informative</i> but any ancestor is <i>TestRun</i>, the node is <i>TestRun</i> unless the node itself is <i>Informative</i>. If no ancestor has a value of <i>Inactive</i> or <i>TestRun</i> and any ancestor has a value of <i>TestRunAndGo</i>, the node has a value of <i>TestRunAndGo</i> unless that node is <i>Inactive</i> or <i>TestRun</i> and so on. The following table illustrates the actions to be applied to a node depending on the value of Activation.</p> <table border="1" data-bbox="651 1625 1398 1898"> <thead> <tr> <th data-bbox="651 1625 1008 1665">Activation</th> <th data-bbox="1008 1625 1219 1665">Test Node</th> <th data-bbox="1219 1625 1398 1665">Execute Node</th> </tr> </thead> <tbody> <tr> <td data-bbox="651 1665 1008 1705"><i>Inactive</i></td> <td data-bbox="1008 1665 1219 1705"><i>false</i></td> <td data-bbox="1219 1665 1398 1705"><i>false</i></td> </tr> <tr> <td data-bbox="651 1705 1008 1745"><i>Informative</i></td> <td data-bbox="1008 1705 1219 1745"><i>false</i></td> <td data-bbox="1219 1705 1398 1745"><i>false</i></td> </tr> <tr> <td data-bbox="651 1745 1008 1785"><i>Held</i></td> <td data-bbox="1008 1745 1219 1785"><i>false</i></td> <td data-bbox="1219 1745 1398 1785"><i>false</i></td> </tr> <tr> <td data-bbox="651 1785 1008 1824"><i>Active</i></td> <td data-bbox="1008 1785 1219 1824"><i>false</i></td> <td data-bbox="1219 1785 1398 1824"><i>true</i></td> </tr> <tr> <td data-bbox="651 1824 1008 1864"><i>TestRun</i></td> <td data-bbox="1008 1824 1219 1864"><i>true</i></td> <td data-bbox="1219 1824 1398 1864"><i>false</i></td> </tr> <tr> <td data-bbox="651 1864 1008 1898"><i>TestRunAndGo</i></td> <td data-bbox="1008 1864 1219 1898"><i>true</i></td> <td data-bbox="1219 1864 1398 1898"><i>true</i></td> </tr> </tbody> </table>	Activation	Test Node	Execute Node	<i>Inactive</i>	<i>false</i>	<i>false</i>	<i>Informative</i>	<i>false</i>	<i>false</i>	<i>Held</i>	<i>false</i>	<i>false</i>	<i>Active</i>	<i>false</i>	<i>true</i>	<i>TestRun</i>	<i>true</i>	<i>false</i>	<i>TestRunAndGo</i>	<i>true</i>	<i>true</i>
Activation	Test Node	Execute Node																					
<i>Inactive</i>	<i>false</i>	<i>false</i>																					
<i>Informative</i>	<i>false</i>	<i>false</i>																					
<i>Held</i>	<i>false</i>	<i>false</i>																					
<i>Active</i>	<i>false</i>	<i>true</i>																					
<i>TestRun</i>	<i>true</i>	<i>false</i>																					
<i>TestRunAndGo</i>	<i>true</i>	<i>true</i>																					

Table 3-4: JDF node (Section 2 of 5)

Name	Data Type	S	Description
Category ? New in JDF 1.2	NMTOKEN	L	<p>Named category of this node. Used when <i>Type</i> = "Combined" or <i>Type</i> = "ProcessGroup" to identify the general node category. This allows processors to identify the general purpose of a node without parsing the <i>Types</i> field. For instance, a RIP for final output and RIP for proof process have identical <i>Types</i> attribute values, but have <i>Category</i> = "ProofRIPing" or <i>Category</i> = "RIPing", respectively. Values include:</p> <p><i>Binding</i> – Binding of a bound product.</p> <p><i>Cutting</i> – Specifies cutting of a Component. New in JDF 1.3</p> <p><i>DigitalPrinting</i> – A RIP and print run on a digital printer that produces final output.</p> <p><i>FinalImaging</i> – A RIP and image that produces final output that is ready for further processing, (e.g., film or plates).</p> <p><i>FinalRIPing</i> – RIP process for generating final output.</p> <p><i>Folding</i> – Folding of a product.</p> <p><i>PostPress</i> – General postpress. Includes <i>Folding</i> and <i>Binding</i>.</p> <p><i>PrePress</i> – General prepress.</p> <p><i>Printing</i> – A press run that produces final output.</p> <p><i>ProofImaging</i> – A RIP that produces proof output.</p> <p><i>ProofRIPing</i> – RIP process for generating a proof. The processes are identical to those in specified for <i>FinalRIPing</i>.</p> <p><i>PublishingPreparation</i> – Preparing an issue of a newspaper or magazine to be published. New in JDF 1.3</p> <p><i>RIPing</i> – General RIP gray box. For details, see Section 6.4.28, RIPing. New in JDF 1.3</p> <p><i>WebPrinting</i> – 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.</p> <p>The value MAY also be the name of a gray box defined by an ICS document or JDF spec. See the ICS documents for the exact names.</p>
ICSVersions ? New in JDF 1.2	NMTOKENS	D	<p>CIP4 Interoperability Conformance Specification (ICS) Versions that this JDF node complies with. The format is <ICSName>_L<ICSLevel>-<ICSVersion>. For instance: MISPRE_L1-1.3 for the MIS to Prepress ICS. If there is a revision to that ICS: "MISPRE_L1-1.3.1". See "Interoperability Conformance Specifications" on page 684 for more information on ICS documents.</p>
<i>ID</i>	ID	L	<p>Unique identifier of a JDF node. This ID is used to refer to the JDF node.</p>
<i>JobID ?</i>	string	D	<p>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.</p>
<i>JobPartID ?</i>	string	D	<p>Identification of a JDF node within a job, used by the application that created the job. Typically, <i>JobPartID</i> 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 <i>JobPartID</i>.</p>

Table 3-4: JDF node (Section 3 of 5)

Name	Data Type	S	Description
MaxVersion ? New in JDF 1.2	JDFJMF-Version	D	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 3.12, JDF Versioning for a discussion of versioning in JDF.
NamedFeatures ? New in JDF 1.2	NMTOKENS	L	<i>NamedFeatures</i> represents an implementation dependent set of parameters for setting up a Device that a Device MUST apply to the JDF ticket. It is formatted as an ordered list of name value pairs with an even number of entries. The <i>NamedFeatures</i> names supported by the Device MAY be specified in DeviceCap elements. See “DeviceCap” on page 613. <i>NamedFeatures</i> MUST be specified only in <i>ProcessGroup</i> nodes, typically with a <i>Types</i> attribute supplied, or <i>Product</i> JDF nodes. See "Use of the NamedFeatures attribute in Product and ProcessGroup nodes" on page 48 for details.
ProjectID ? New in JDF 1.1	string	D	Identification of the project context that this JDF belongs to. Used by the MIS to group a set of JDF jobs.
RelatedJobID ? New in JDF 1.2	string	D	Job identification of a related job. Used to identify the <i>JobID</i> 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	D	Job identification of a related job part. Used to identify the <i>JobPartID</i> 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.
SpawnID ? New in JDF 1.1	NMTOKEN	D	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 "Spawning and Merging" on page 132.
Status Modified in JDF 1.3	enumeration	L	Identifies the status of the node. See Table 3-5 for values of <i>Status</i> . Derivation of the <i>Status</i> of a parent node from the <i>Status</i> of child nodes is non-trivial and implementation-dependent.
StatusDetails ? New in JDF 1.2	string	L	Description of the status phase that provides details beyond the enumerative values given by the Status attribute. For a list of supported values, see "StatusDetails Supported Strings" on page 703.
Template = "false" New in JDF 1.1	boolean	R	Indicates that this JDF node (or instance) is a template that is used to generate JDF elements but MUST NOT be exchanged as a job definition. A Device MUST reject a job ticket that contains <i>Template</i> = "true".
TemplateID ? New in JDF 1.2	string	D	Name or ID that identifies a JDF template. Can be used to differentiate between various templates. If <i>Template</i> = "false", <i>TemplateID</i> identifies the template that was used to generate this JDF.
TemplateVersion ? New in JDF 1.2	string	D	Provides the version of the JDF template. Can be used to differentiate between various template versions. If <i>Template</i> = "false", <i>TemplateVersion</i> identifies the version of the template that was used to generate this JDF.

Table 3-4: JDF node (Section 4 of 5)

Name	Data Type	S	Description
<i>Type</i>	NMTOKEN	L	Identifies the type of the node. Any JDF process name is a valid type. The processes that have been predefined are listed in "Processes" on page 219, 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. <i>Combined</i> <i>ProcessGroup</i> <i>Product</i> – Identifies a product intent node.
<i>Types ?</i> Modified in JDF 1.2	NMTOKENS	L	List of the <i>Type</i> attributes of the nodes that are combined to create this node. This attribute is REQUIRED if <i>Type</i> = " <i>Combined</i> ", OPTIONAL when <i>Type</i> = " <i>ProcessGroup</i> ", and is ignored if <i>Type</i> equals any other value. For details on using <i>Combined</i> nodes, see Section 3.2.3, Combined Process Nodes. If the <i>Types</i> attribute is specified, that JDF node MUST NOT contain child JDF nodes. For details on using <i>ProcessGroup</i> nodes, see Section 3.2.2, Process Group Nodes. If <i>Type</i> = " <i>ProcessGroup</i> ", the tokens MAY also be the name of a gray box that needs expansion. See <i>Category</i> for more details.
<i>Version ?</i> Modified in JDF 1.2	JDFJMF-Version	R D	Text that identifies the version of the JDF node. The <i>Version</i> 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 "JDF Versioning" on page 114.
<i>xmlns ?</i> New in JDF 1.1	URI	R D	JDF supports use of XML namespaces. The namespace MUST be declared in the root JDF element. For details on using namespaces in XML, see[XMLNS]. For version 1.1, 1.2 and 1.3 of JDF, <i>xmlns</i> = " http://www.CIP4.org/JDFSchema_1_1 ".
<i>xsi:type ?</i> New in JDF 1.2	NMTOKEN	L	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 "JDF Nodes" on page 37.
AncestorPool ?	element	R	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 3.3, AncestorPool.
AuditPool ?	element	L	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 3.10, AuditPool.
CustomerInfo ? Deprecated in JDF 1.3	element	D	Container element for customer-specific information. See Section 3.4, CustomerInfo. In JDF 1.3 and beyond, CustomerInfo is a resource that is referenced through a CustomerInfoLink in the ResourceLinkPool.
JDF *	element	L	Child JDF nodes. The nesting of JDF nodes defines the JDF tree.

Table 3-4: JDF node (Section 5 of 5)

Name	Data Type	S	Description
NodeInfo ? Deprecated in JDF 1.3	element	L	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 3.10, AuditPool . In JDF 1.3 and beyond, NodeInfo is a resource that is referenced through a NodeInfoLink in the ResourceLinkPool .
ResourceLinkPool ?	element	L	Container element for ResourceLink elements, which describe the input and output resources of the node. See Section 3.8, ResourceLinkPool and ResourceLink .
ResourcePool ?	element	L a	Container element for resources. See Section 3.7, ResourcePool and its Resource Children .
StatusPool ? Deprecated in JDF 1.3	element	L	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 ". In JDF 1.3 and beyond, StatusPool/PartStatus/@Status is replaced by NodeInfo/@NodeStatus in the respective partition of NodeInfo .

- a. Resources are local in a **ResourcePool** but MAY be referenced from **ResourceLink** elements in descendent nodes. For details see "ResourceLinkPool and ResourceLink" on page 63.

— Attribute: Status

Table 3-5: Status attribute – possible values (Section 1 of 2)

Value	Description
<i>Waiting</i>	The node can be executed, but it has not completed a test run.
<i>TestRunInProgress</i>	The node is currently executing a test run.
<i>Ready</i>	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.
<i>FailedTestRun</i>	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 3.10, AuditPool .
<i>Setup</i>	The process represented by this node is currently being set up.
<i>InProgress</i>	The node is currently executing.
<i>Cleanup</i>	The process represented by this node is currently being cleaned up.
<i>Spawned</i>	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 .
<i>Suspended</i>	Execution has been stopped. If a job is <i>Suspended</i> , running will be resumed later. Unlike <i>Stopped</i> this Status indicates that the job has been taken off the device to execute another job or perform some other action that has is not related to this job. When resumed, the job MAY go into Status = " <i>Setup</i> " before changing to <i>InProgress</i> again. New in JDF 1.3
<i>Stopped</i>	Execution has been stopped. If a job is <i>Stopped</i> , 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.

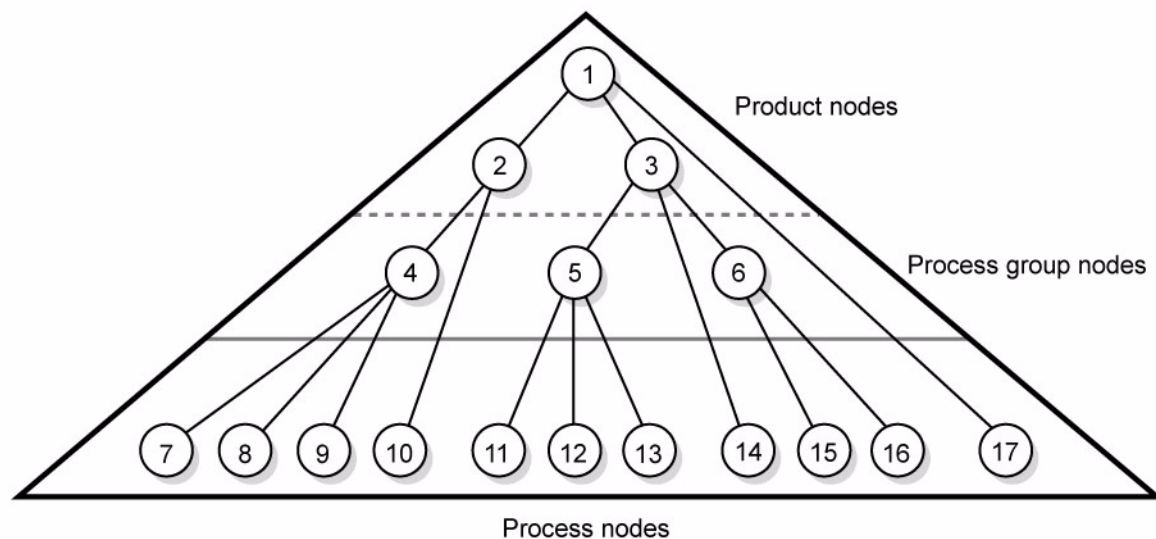
Table 3-5: Status attribute – possible values (Section 2 of 2)

Value	Description
<i>Completed</i>	Indicates that the node has been executed correctly, and is finished.
<i>Aborted</i>	Indicates that the process executing the node has been aborted, which means that execution will not be resumed again.
<i>Part</i>	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. New in JDF 1.3
<i>Pool</i>	Indicates that the node processes partitioned resources and that the <i>Status</i> varies depending on the partition keys. Details are provided in the StatusPool element of the node. Deprecated in JDF 1.3

3.2 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 "Processes" on page 219, 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* nodes, *ProcessGroup* 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**

3.2.1 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 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 **MUST** be specified in *Process* nodes, which exist lower in the job hierarchy.

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 7.1.1, *Span Subelements of an Intent Resource*. 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*.

3.2.2 Process Group Nodes

Intermediate nodes in the JDF job hierarchy, (i.e., nodes 4, 5 and 6 in Figure 3-3), describe groups of processes. The *Type* attribute value of these kinds of nodes is *ProcessGroup*, (and they are referred to as “ProcessGroup 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 can be modeled flexibly. ProcessGroup nodes **MAY** contain further ProcessGroup nodes, individual process nodes or a mixture of both node types. Sequencing of ProcessGroup nodes **SHOULD** be defined by linking resources of the appropriate leaves, or if the nature of the interchange resources is unknown, by linking **PlaceHolder** resources.

The higher the level of the *ProcessGroup* nodes within the hierarchy, the larger the number of processes the group contains. A high level ProcessGroup node (e.g., prepress, finishing or printing processes) might include lower level ProcessGroup 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 *ProcessGroup* node.

3.2.2.1 Use of the Types attribute in ProcessGroup nodes – Gray Boxes

[New in JDF 1.2](#)

ProcessGroup 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. ProcessGroup 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. ProcessGroup nodes with a non-empty *Types* attribute **MUST NOT** be executed. A Device that receives the ProcessGroup node **MUST** define the exact structure of the ProcessGroup node by executing the following steps until the *Types* list referenced by the ProcessGroup 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 ProcessGroup 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 ProcessGroup node to the newly created subordinate JDF node(s). The ResourceLink MUST either be retained or deleted from the ProcessGroup node. If it is retained, the ProcessGroup node MUST NOT be executed before the Resource that is linked by that ResourceLink is available. Otherwise, the ProcessGroup 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 ProcessGroup 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 MUST NOT be linked to the ProcessGroup node but only to the subordinate JDF node created in this process.

An Agent MUST 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 MUST be maintained when instantiating the child nodes. JDF ProcessGroup nodes that contain both a non-empty *Types* attribute and child JDF nodes are *not* supported, although a *ProcessGroup* node MAY contain child *ProcessGroup* nodes that have non-empty *Types* attribute.

3.2.2.2 Use of the NamedFeatures attribute in Product and ProcessGroup nodes

[New in JDF 1.2](#)

ProcessGroup and Product nodes MAY contain a *NamedFeatures* attribute that allows a Controller (e.g., an MIS system) to define a named set of parameters for processes that MUST 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:

- 1 Explicitly supplied parameters
- 2 Parameters supplied by the Device Agent that are associated with the supplied *NamedFeatures* attribute closest to the process.
- 3 Parameters supplied by the Device Agent that are associated with the supplied *NamedFeatures* attribute supplied by the Device agent at node levels closer to the root.

An individual *NamedFeatures* entry is selected by specifying an NMTOKEN pair that matches entries from DeviceCap/FeaturePool/EnumerationState/@Name and DeviceCap/FeaturePool/EnumerationState/@AllowedValueList (See “DeviceCap” on page 613.), where the first and all even (0 based) entries define the name of the parameter set name (e.g., “Screening”), and the second and all odd entries (0 based) define the selected parameter set value, (e.g., “AM_HighRes”). Multiple *NamedFeatures* MAY be selected. Names and values are implementation dependent. Each name MUST occur only once in the *NamedFeatures* list.

Use of *NamedFeatures* is commonly combined with the use of *Types* in ProcessGroup nodes as described in “Use of the Types attribute in ProcessGroup nodes – Gray Boxes” on page 47. *Types* abstractly specifies the set of processes to execute, whereas *NamedFeatures* abstractly specifies the set of Resources for the processes specified in *Types*.

3.2.2.3 ResourceLink Structure in ProcessGroup nodes

[New in JDF 1.2](#)

The contents of the ResourceLinkPool of a ProcessGroup node define the resources that MUST be available for the ProcessGroup Node itself to be executed.

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 resources are assumed to be supplied by the sub-controller that executes the ProcessGroup

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" Version="1.3">
  <!--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 -->
  <ResourceLinkPool>
    <!-- print input media -->
    <MediaLink Usage="Input" rRef="L2"/>
    <!-- gathered output components -->
    <ComponentLink Usage="Output" rRef="L7"/>
  </ResourceLinkPool>
  <JDF ID="J2" Status="Waiting" Type="DigitalPrinting">
    <ResourceLinkPool>
      <!-- digital printing parameters -->
      <DigitalPrintingParamsLink Usage="Input" rRef="L1"/>
      <!-- input sheets -->
      <MediaLink Usage="Input" rRef="L2"/>
      <!-- printed output components -->
      <ComponentLink Usage="Output" rRef="L3"/>
    </ResourceLinkPool>
  </JDF>
  <JDF ID="J3" Status="Waiting" 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" 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>
```

3.2.3 Combined Process Nodes

The processes described in "Processes" on page 219 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 MUST 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 Combined 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 MUST NOT contain further nested JDF nodes.

A device with multiple processing capabilities is able to recognize the Combined Process node as a single unit of work that it can execute. Therefore, all resources for each of the subtasks that define the Combined node and that are explicitly defined as **ResourceLink** elements MUST be available before the node can be executed. In addition, all input and output resources that are consumed and produced externally by the process MUST be specified in the **ResourceLinkPool** element of the node. This includes all REQUIRED Parameter resources as well as the initial input resources and final output resources. Intermediate resources that are internally produced and consumed, on the other hand, need not be specified.

In a Combined Process node, the information defined by the various resources linked as input to the various subprocesses are logically available to all processes of the combined node. In situations where the parameter resource of more than 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 MUST be used, and any other mappings specified by any down-stream subprocess Resource MUST be ignored.

3.2.3.1 Combined Process Nodes with Multiple Processes of the Same Type

A Combined 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**.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting" Type="Combined"
Types="Cutting Folding Cutting" Version="1.3">
  <!--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 third Cutting Process-->
    <CuttingParams Class="Parameter" ID="L3" Status="Available"/>
    <!-- raw input components -->
    <Component Class="Quantity" ID="L4" Status="Available"/>
    <!-- completed output components -->
    <Component Class="Quantity" ID="L5" Status="Unavailable"/>
  </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 first 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>
```

3.2.3.2 Examples of Combined Process Nodes

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 resources and consumable resources 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" Version="1.3">
  <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>
</JDF>
```

3.2.3.3 Specifying non-linear dependencies in a Combined node

A Combined node typically specified a linear execution chain of the individual process steps defined in JDF/@Types. A Device that executes a Combined 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 combined 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 MUST allow execution of the node. The following example specifies an inline combined folder and stitcher

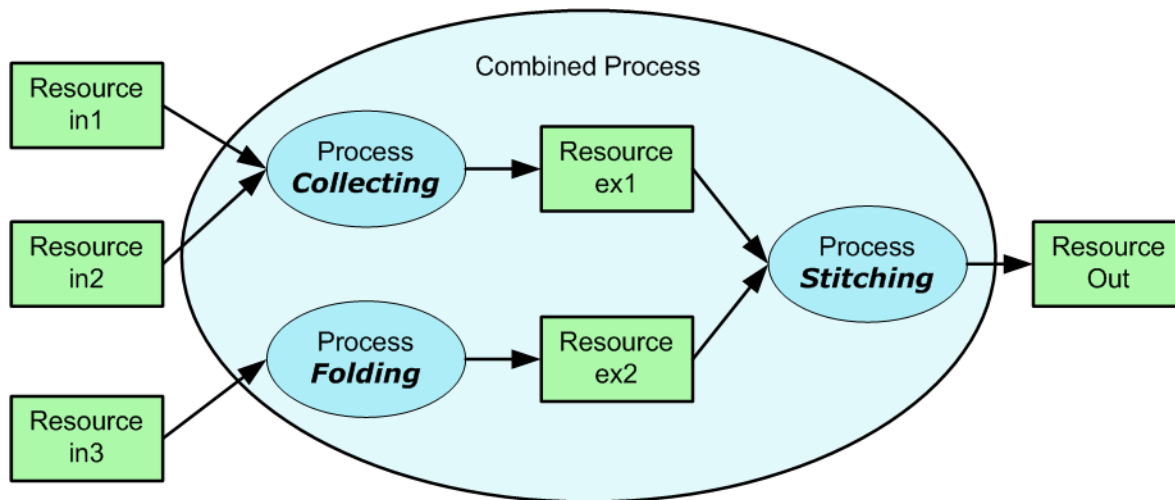


Figure 3-4: Combined node dependencies

```
<JDF ID="ID" xmlns="http://www.CIP4.org/JDFSchema_1_1" Status="Waiting" Type="Combined"
Types="Collecting Stitching Folding" Version="1.3">
<ResourcePool>
  <Component ID="in1"/>
  <Component ID="in2"/>
  <Component ID="in3"/>
</ResourcePool>
```

```

    <Component ID="ex1" PipeProtocol="Internal" PipeID="ex1"/>
    <Component ID="ex2" PipeProtocol="Internal" PipeID="ex2"/>
    <Component ID="Out"/>
</ResourcePool>
<ResourceLinkPool>
    <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="ex2"/>
    <ComponentLink CombinedProcessIndex="2" Usage="Output" rRef="ex1"/>
    <ComponentLink CombinedProcessIndex="1" Usage="Input" rRef="ex1"/>
    <ComponentLink CombinedProcessIndex="1" Usage="Input" rRef="ex2"/>
    <ComponentLink CombinedProcessIndex="1" Usage="Output" rRef="Out"/>
</ResourceLinkPool>
</JDF>

```

3.2.4 Process Nodes

Process nodes represent the very lowest level in a job hierarchy. They MUST 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 below, nodes 7-17 represent process nodes. The various individual process node types are specified in Section 6, Processes.

3.3 AncestorPool

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.



Ancestor Pool

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

Table 3-6: 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 <i>SheetName</i> partition and subsequently spawned with a <i>Separation</i> would contain both <i>SheetName</i> and <i>Separation</i> within Part .

An **Ancestor** element MUST contain read-only copies of all the attributes of the node that it represents with the exception of the *ID* attribute, which MUST 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 3-7: Ancestor element

Name	Data Type	Description
<i>Activation</i> ?	enumeration	Copy of the <i>Activation</i> attribute from the ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
<i>Category</i> ? New in JDF 1.2	NMTOKENS	Copy of the <i>Category</i> attribute from the original ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
<i>FileName</i> ?	URL	The URL of the JDF file where the ancestor node resided prior to spawning.
<i>ICSVersions</i> ? New in JDF 1.2	NMTOKENS	Copy of the <i>ICSVersions</i> attribute from the original ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
<i>JobID</i> ?	NMTOKENS	Copy of the <i>JobID</i> attribute from the original ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
<i>JobPartID</i> ?	string	Copy of the <i>JobPartID</i> attribute from the original ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
<i>MaxVersion</i> ? New in JDF 1.2	JDFJMFVersion	Copy of the <i>MaxVersion</i> attribute from the original ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
<i>NamedFeatures</i> ? New in JDF 1.2	NMTOKENS	Copy of the <i>NamedFeatures</i> attribute from the original ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
<i>NodeID</i>	NMTOKEN ^a	Copy of the <i>ID</i> attribute of the ancestor node.
<i>ProjectID</i> ?	string	Identification of the project context that this JDF belongs to. Used by the application that created the JDF job.
<i>RelatedJobID</i> ? New in JDF 1.2	NMTOKENS	Copy of the <i>RelatedJobID</i> attribute from the original ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
<i>RelatedJobPartID</i> ? New in JDF 1.2	NMTOKENS	Copy of the <i>RelatedJobPartID</i> attribute from the original ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
<i>SpawnID</i> ? New in JDF 1.1	NMTOKEN	Copy of the <i>SpawnID</i> attribute of the ancestor node.
<i>Status</i> ?	enumeration	Copy of the <i>Status</i> attribute from the original ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
<i>StatusDetails</i> ? New in JDF 1.2	string	Copy of the <i>StatusDetails</i> attribute from the original ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
<i>Type</i> ?	NMTOKEN	Copy of the <i>Type</i> attribute from the original ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
<i>Types</i> ?	NMTOKENS	Copy of the <i>Types</i> attribute from the original ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
<i>Version</i> ?	JDFJMFVersion	Copy of the <i>Version</i> attribute from the original ancestor node. For details, see Table 3-4, “JDF node,” on page 41.
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. For details, see Table 3-4, “JDF node,” on page 41. In JDF 1.3 and beyond, CustomerInfo MAY be a resource reference.
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. For details, see Table 3-4, “JDF node,” on page 41. In JDF 1.3 and beyond, NodeInfo MAY be a resource reference.

- a. The data type is NMTOKEN and not IDREF because the ID does not reside in the spawned JDF. The corresponding ID element resides in the original JDF.

3.4 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 3.7, **ResourcePool** and its **Resource Children**).



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?

3.5 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 3.7, **ResourcePool** and its **Resource Children**).

3.6 StatusPool

[Deprecated in JDF 1.3.](#)

Starting with JDF 1.3, **StatusPool** is deprecated and replaced by a partitioned **NodeInfo** resource. For details, see "StatusPool and PartStatus" on page 815.

3.7 ResourcePool and its Resource Children

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

Table 3-8: 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.

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

3.7.1 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 3-9, “Abstract Resource element,” on page 54, below, and are described in greater detail in the sections that follow.

Table 3-9: Abstract Resource element (Section 1 of 4)

Name	Data Type	Description
<i>AgentName</i> ? 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.
<i>AgentVersion</i> ? 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.
<i>Author</i> ? New in JDF 1.2	string	Text that identifies the person who generated the resource.

Table 3-9: Abstract Resource element (Section 2 of 4)

Name	Data Type	Description
<i>CatalogID</i> ?	string	Identification of the resource, (e.g., in a catalog environment). Defaults to the value of <i>ProductID</i> .
<i>CatalogDetails</i> ?	string	Additional details of a resource in a catalog environment.
<i>Class</i>	enumeration	Defines the abstract resource type. For details, see the sections that follow. Possible values are: <i>Consumable</i> <i>Handling</i> <i>Implementation</i> <i>Intent</i> <i>Parameter</i> <i>Placeholder</i> <i>Quantity</i> <i>Class</i> MUST be specified in the resource root, MUST NOT be specified in a resource leaf and SHOULD NOT be specified in an inline resource subelement.
<i>ID</i>	ID	Unique identifier of a resource. <i>ID</i> MUST be specified in the resource root, MUST NOT be overwritten in a resource leaf and SHOULD NOT be specified in an inline resource subelement.
<i>Locked</i> = "false"	boolean	If <i>true</i> , the resource MUST NOT be modified, e.g., because it resides in a spawned ticket that is spawned in read-only mode or referenced by an <i>Audit</i> and MUST NOT be modified without invalidating the <i>Audit</i> .
<i>PartUsage</i> = "Explicit" New in JDF 1.1 Modified in JDF 1.3	enumeration	Description of the interpretation of partitions. One of: <i>Explicit</i> – Require explicit partition matches. All referenced partitions referenced in <i>Part</i> MUST exist, otherwise it is an error. <i>Implicit</i> – 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. <i>Sparse</i> – The closest matching Partition with no non-matching partition keys is returned. If keys with non-matching values exist the link is in error. <i>PartUsage</i> = "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 <i>PartUsage</i> MUST NOT be specified outside of the root of a resource. For details on <i>PartUsage</i> and partitioning, see Section 3.9.7.2, Implicit, Sparse and Explicit PartUsage in Partitioned Resources. <i>PartUsage</i> was moved to this table from Table 3-25 on page 86 in JDF 1.2.
<i>PipeID</i> ?	string	If this attribute exists, the resource is a pipe. <i>PipeID</i> is used by JMF pipe-control messages to identify the pipe. For more information, see "Overlapping Processing Using Pipes" on page 127.

Table 3-9: Abstract Resource element (Section 3 of 4)

Name	Data Type	Description
<i>PipeProtocol</i> ? New in JDF 1.2	NMTOKEN	Defines the protocol use for pipe handling. <i>JMF</i> and <i>Internal</i> 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. Allowed values include: <i>Internal</i> – Internal or virtual pipe used within a combined process. <i>JMF</i> – JMF-based PipePush / PipePull messages. <i>None</i> – No pipe support.
<i>PipeURL</i> ? New in JDF 1.2	URL	Pipe request URL. Dynamic pipe requests to this resource SHOULD be made to this URL. ^a 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 <i>PipeURL</i> , see Section 4.3.3, Overlapping Processing Using Pipes.
<i>ProductID</i> ?	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 resources.
<i>rRefs</i> ? Deprecated in JDF 1.2	IDREFS	Array of <i>IDs</i> of internally referenced resources. In JDF 1.2 and beyond, it is up to the implementation to maintain references.
<i>SpawnIDs</i> ? New in JDF 1.1	NMTOKENS	List of SpawnIDs. This is used as a reference count for how often the resource has been spawned.
<i>SpawnStatus</i> = “NotSpawned”	enumeration	The spawn status of a resource indicates whether or not a resource has been spawned, and under what circumstances. The list of possible values is assumed to be ordered, so that the <i>SpawnStatus</i> of a resource that has ResourceRef elements is defined as the maximum <i>SpawnStatus</i> of all recursively linked resources. Possible values, ordered from lowest to highest are: <i>NotSpawned</i> — Indicates that the resource has not been copied to another process. <i>SpawnedRO</i> – Indicates that the resource has been copied to another process where it cannot be modified. The “RO” stands for read-only. <i>SpawnedRW</i> – Indicates that the resource has been copied to another process where it can be modified. The “RW” stands for read/write.

Table 3-9: Abstract Resource element (Section 4 of 4)

Name	Data Type	Description
Status Modified in JDF 1.2	enumeration	<p>The status of a resource indicates under what circumstances it can be processed or modified. <i>Status</i> MUST be specified in the resource root, MUST NOT be specified in an inline resource subelement and MAY be overwritten in a resource leaf.</p> <p>The values listed below are assumed to be ordered so that the <i>Status</i> of a resource that references further resources can be defined as the minimum <i>Status</i> of all recursively linked resources. Possible values, ordered from lowest to highest, are:</p> <p><i>Incomplete</i> – 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 Section 7 Resources. The structural attributes <i>Class</i> and <i>ID</i> MUST be specified.</p> <p><i>Rejected</i> – Indicates that the resource has been rejected by an Approval process. The metadata is valid. New in JDF 1.2</p> <p><i>Unavailable</i> – Indicates that the resource is not ready to be used or that the resource in the real world represented by the physical resource in JDF is not available for processing. The metadata is valid.</p> <p><i>InUse</i> – Indicates that the resource exists, but is in use by another process. Also used for active pipes (see Section 3.7.4, Pipe Resources and Section 4.3.3, Overlapping Processing Using Pipes).</p> <p><i>Draft</i> – Indicates that the resource exists in a state that is sufficient for setting up the next process but not for production.</p> <p><i>Complete</i> – Indicates that the resource is completely specified and the parameters are valid for usage. A physical resource with <i>Status</i> = "<i>Complete</i>" is not yet available for production, although it is sufficiently specified for a process that references it through a <i>ResourceRef</i> from a parameter resource to commence execution.</p> <p><i>Available</i> – Indicates that the whole resource is available for usage.</p>
UpdateID ? New in JDF 1.1 Deprecated in JDF 1.3	NMTOKEN	<p>Unique ID that identifies the Resource or Resource partition. Note that only one Resource, Resource partition or ResourceUpdate with a given value of <i>UpdateID</i> MAY occur per JDF document, even though the scope of the ResourceUpdate is local to the resource that it is defined in.</p>
GeneralID * New in JDF 1.3	element	<p>Additional identifiers related to the resource.</p>
SourceResource * New in JDF 1.3	element	<p>List of resources that were or SHOULD be taken into account to populate this resource.</p>
QualityControlResult * New in JDF 1.2	refelement	<p>Results of quality measurements which were performed during or after the production of this resource.</p>

- a. Note that in most cases this 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.

Table 3-10: 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 resources 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 MUST NOT be an inline resource.

Figure 3-5 shows the structure of the abstract resource classes defined above. Arrows define inheritance relations and the thin orthogonal lines describe containing relations.

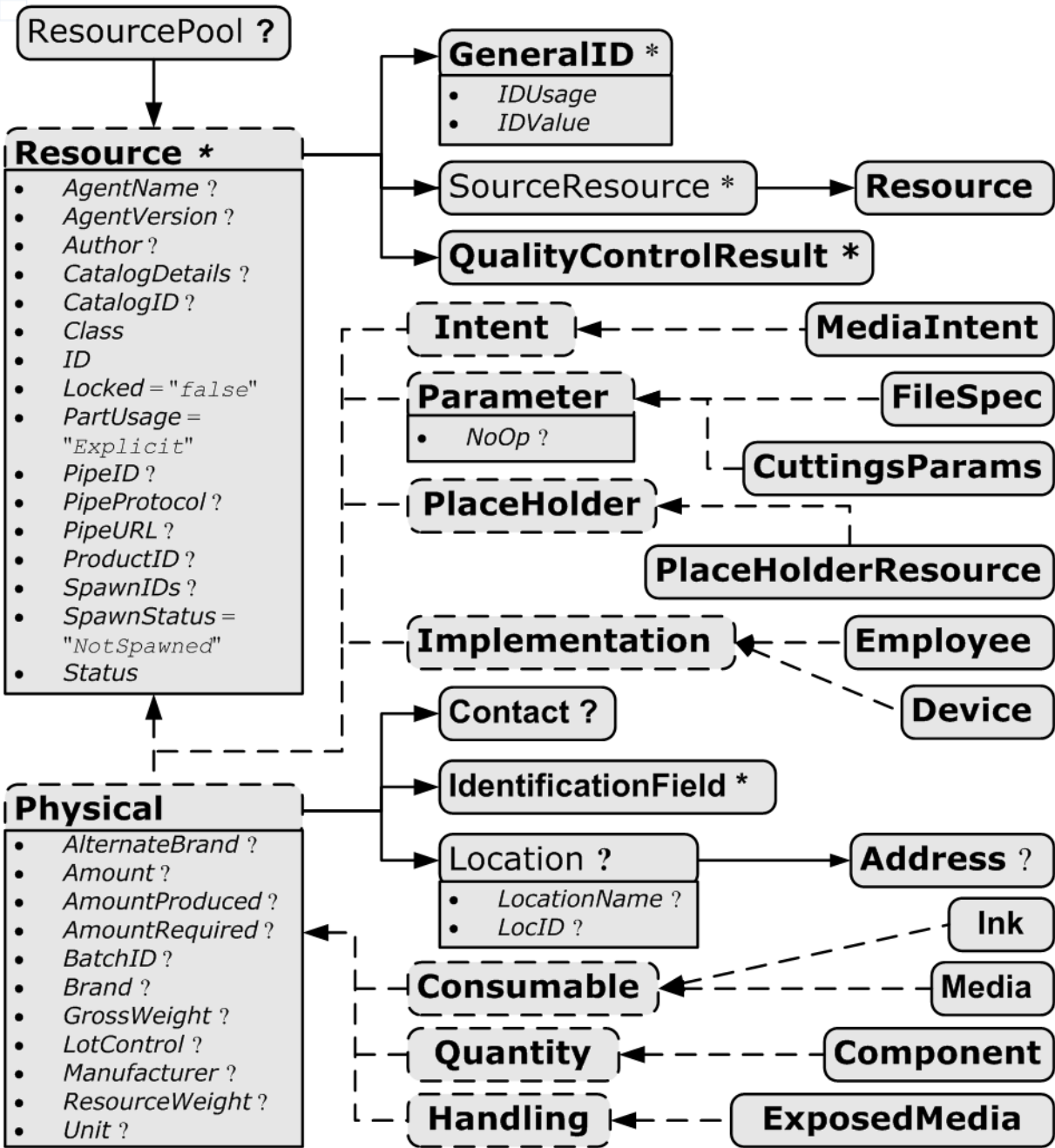


Figure 3-5: Abstract Resource element – a diagram of its structure

3.7.2 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 [Section 7, Resources](#), the class of each resource is indicated in the Resource Properties subheading.

3.7.2.1 Parameter Resources

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 resources contain the suffix “Params” in their titles. Examples of *Parameter* resources include **FoldingParams** and **ConventionalPrintingParams**.



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.

Table 3-11: Abstract Parameter Resource element

Name	Data Type	Description
<i>NoOp</i> = “ <i>false</i> ” New in JDF 1.1	boolean	A value of <i>true</i> indicates that the process step that is parameterized by this resource or resource partition MUST NOT be executed. If <i>false</i> or not specified, the Resource is operational and that the process step that is parameterized by this resource or resource partition MUST be executed. The <i>NoOp</i> attribute MUST only be used for processes that input and output exchange resources of identical resource types, (e.g., RunList or Component).

3.7.2.2 Intent Resources

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 intent resources are described in [Section 7.1, Intent Resources](#). The abstract *Intent* resource element contains no attributes or elements besides those contained in the abstract **Resource** element.

3.7.2.3 Implementation Resources

Implementation resources define the devices and operators that execute a given node. Only two implementation resource types are defined: **Employee** (see [Section 7.2.70, Employee](#)) and **Device**, each of which is described in greater detail in [Section 7, Resources](#).

Implementation resources can only be used as input resources and MAY be linked to any process. The abstract *Implementation* resource element contains no attributes or elements besides those contained in the abstract **Resource** element. An example demonstrating how to use implementation resources is provided in [Section 3.8.3, Links to Implementation Resources](#).

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.

3.7.2.4 Physical Resources (Consumable, Quantity, Handling)

Any resource whose *Class* is *Consumable*, *Quantity* or *Handling* is considered a physical resource. They are defined as follows:

- *Consumable* resources are consumed during a process. Examples include **Ink** and **Media**. They are the unmodified inputs in a process chain.
- *Quantity* resources have 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. **Component** resources are an example of *Quantity* resources.
- *Handling* resources are used during a process, but are 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.

Table 3-12, “Abstract Physical Resource element,” on page 61, defines the additional attributes and elements that can be defined for physical resources. The processes that consume physical resources—any kind of physical resource—have the option of using these attributes and elements to determine in what way the resources are to be consumed. Table 3-13 then describes the contents of the **Location** subelement of physical resource elements.

Table 3-12: Abstract Physical Resource element (Section 1 of 2)

Name	Data Type	Description
<i>AlternateBrand</i> ?	string	Information, such as the manufacturer or type, about a resource compatible to that specified by the <i>Brand</i> attribute, which is described below.
<i>Amount</i> ?	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 3.9.4, Resource Amount.
<i>AmountProduced</i> ? 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 <i>ActualAmount</i> values of output ResourceLink elements of leaf JDF nodes with <i>Status</i> = “ <i>Completed</i> ” that reference this resource
<i>AmountRequired</i> ?	double	Total amount of the resource that is referenced by all nodes that will consume this resource. This corresponds to the sum of all <i>Amount</i> values of input ResourceLink elements of all processes that consume this resource. In case the Resource is the last resource in a process chain, <i>AmountRequired</i> specifies the sum of all <i>Amount</i> values of all output ResourceLink elements that produce this resource.
<i>BatchID</i> ?	string	ID of a specific batch of the physical resource
<i>Brand</i> ? 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. <ul style="list-style-type: none"> • 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, <i>Brand</i> included details of the <i>Manufacturer</i> , which SHOULD be specified in <i>Manufacturer</i> .



AUTOMATING INVENTORY MANAGEMENT

JDF's handling of physical resources 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 3-12: Abstract Physical Resource element (Section 2 of 2)

Name	Data Type	Description
<i>GrossWeight</i> ? New in JDF 1.3	double	Gross weight of a single resource, as counted in <i>Amount</i> , in grams.
<i>LotControl</i> ? New in JDF 1.3	enumeration	Specifies whether the resource is lot controlled. Values are: <i>Controlled</i> - Resource is lot controlled, lot usage SHOULD be reported in ResourceAudit elements. <i>NotControlled</i> - Resource is not lot controlled.
<i>Manufacturer</i> ? New in JDF 1.3	string	Specifies the manufacturer of the resource.
<i>ResourceWeight</i> ? New in JDF 1.1	double	Net weight of a single resource, as counted in <i>Amount</i> , in grams.
<i>Unit</i> ?	NMTOKEN	Unit of measurement for the values of <i>Amount</i> and <i>AmountRequired</i> . Note that it is strongly discouraged to specify units other than those that are defined in Section 1.6, Units.
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 physical resource.
Location ?	element	Description of details of the resource location. Note, in order to describe multiple locations, resources MAY be partitioned by the <i>Location</i> partition key as described in Section 3.9.5, Description of Partitioned Resources.

Structure of Location Subelement**Table 3-13: Location element**

Name	Data Type	Description
<i>LocationName</i> ? New in JDF 1.1	string	Name of the location, (e.g., in MIS). This allows the user to describe distributed resources.
<i>LocID</i> ?	string	Location identifier, (e.g., within a warehouse system).
Address ?	refelement	Address of the storage facility. For more information, see Section 7.2.2, Address.

3.7.2.5 Placeholder Resources

Placeholder resources, unlike physical resources, 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* resources, a processing skeleton can be constructed that gives a basic shape to a job. The appropriate resources can be substituted for *Placeholder* resources 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 *Implementation* resources, *Placeholder* resources contain no attributes besides those contained in the abstract **Resource** element.

3.7.3 Position of Resources within JDF Nodes

Resources MAY exist in any JDF node, but JDF nodes MUST reference only local or global resources. In other words, JDF nodes MUST 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-fed 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 **ProcessGroup** 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 MUST infer this information by explicitly linking to the appropriate parameter resource.

3.7.4 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 MUST 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 4.3.3.2, Dynamic Pipes](#).

3.7.5 ResourceUpdate Elements

[New in JDF 1.1.](#)

[Deprecated in JDF 1.3](#)

For details of the deprecated ResourceUpdate element, see "ResourceUpdate Elements" on page 815.

3.8 ResourceLinkPool and ResourceLink

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 3-14: 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.

3.8.1 Abstract ResourceLink element

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 MUST 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.

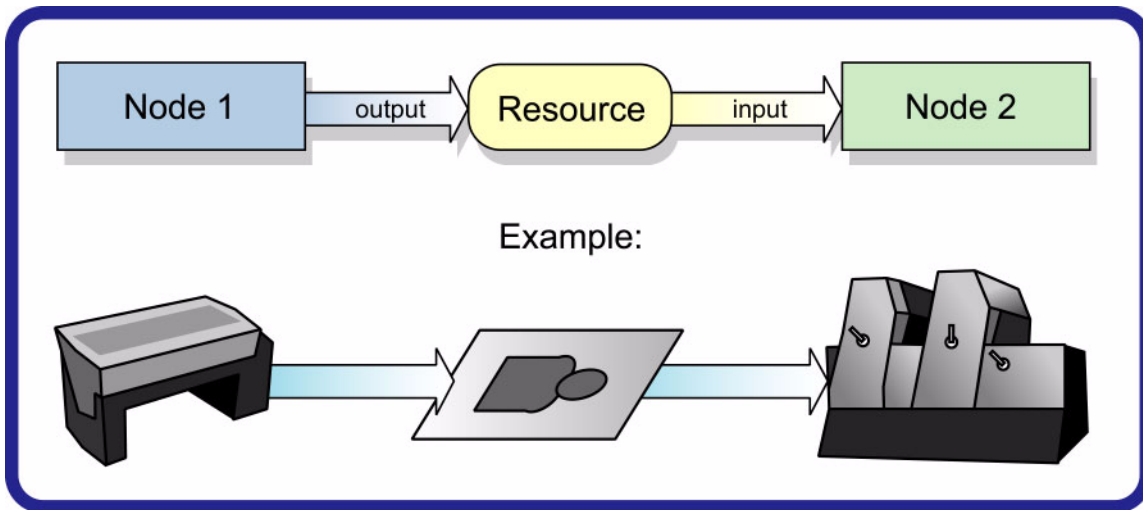


Figure 3-6: Nodes linked by a resource

`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 3.9.2, `ResourceRef` – Element for Inter-Resource Linking and `refElement`).

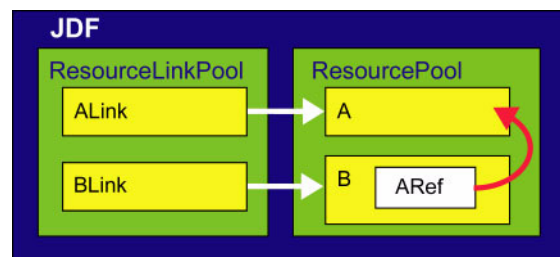


Figure 3-7: ResourceLink elements and ResourceRef elements

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 3.9.5, *Description of Partitioned Resources*.

`ProcessGroup` and `Product` 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

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*.

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 Section 6, Processes and Section 7, Resources).

- Product intent and process parameters are generally input resources, except when one process defines the parameters of a subsequent process.
- *Consumable* resources MUST always be input resources.
- *Quantity* and *Handling* resources are used both as input 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 MUST be defined as an input parameter of the first process where it is referenced. This means that a device MUST 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 MUST be defined as an input parameter of the **Media** of the printing process.
- Any resource parameter that is used MUST 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 MUST define the output resource of its parent process.
- In case of parallel processing, the sum of the outputs of all parallel subnodes MUST 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**. The following seven abstract **ResourceLink** classes exist:

- **ConsumableLink**
- **HandlingLink**
- **ImplementationLink**
- **IntentLink**
- **ParameterLink**
- **PlaceholderLink**
- **QuantityLink**
- Each listed class name is described in greater detail in the sections that follow. Figure Section 3-8, Abstract **ResourceLink** element – a diagram of its structure shows the abstract types derived from the **ResourceLink** type.

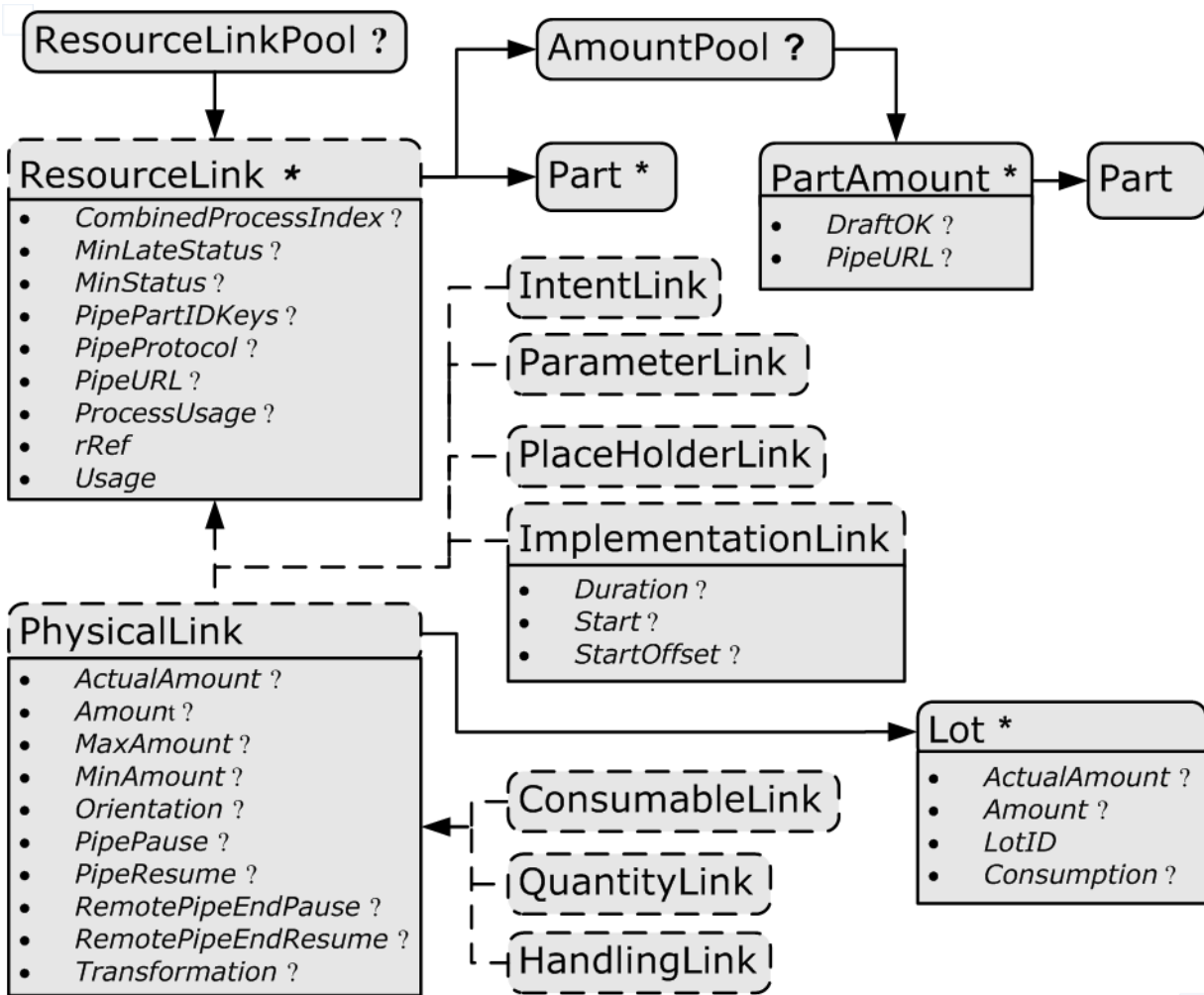


Figure 3-8: Abstract ResourceLink element – a diagram of its structure

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

Table 3-15: Abstract ResourceLink element (Section 1 of 3)

Name	Data Type	Description
<i>CombinedProcessIndex</i> ? New in JDF 1.1	IntegerList	<i>Combined</i> and <i>ProcessGroup</i> nodes MAY contain resources from multiple process nodes. The <i>CombinedProcessIndex</i> attribute specifies the indices of individual processes in the <i>Types</i> attribute to which a <i>ResourceLink</i> in a <i>Combined</i> or <i>ProcessGroup</i> node belongs. Multiple entries in <i>CombinedProcessIndex</i> specify that the <i>ResourceLink</i> is used by the respective multiple processes in the <i>Combined</i> node. It MUST be specified when multiple resources of the same Resource/@Type and <i>ResourceLink/@Usage</i> are specified in one JDF node. If <i>CombinedProcessIndex</i> is not specified, even though multiple processes in the <i>Combined</i> or <i>ProcessGroup</i> node MAY link to the Resource, the <i>ResourceLink</i> applies to all of these processes.

Table 3-15: Abstract ResourceLink element (Section 2 of 3)

Name	Data Type	Description
<i>CombinedProcessType</i> ? Deprecated in JDF 1.1	NMTOKEN	<i>Combined</i> nodes contain input resources from multiple process nodes. The <i>CombinedProcessType</i> attribute specifies the name individual process to which a ResourceLink in a <i>Combined</i> node belongs. It MUST match one of the entries in the <i>Types</i> attribute of the node. It has been replaced by <i>CombinedProcessIndex</i> in JDF 1.1.
<i>DraftOK</i> ? Deprecated in JDF 1.3	boolean	If <i>true</i> , the process can commence with a draft resource. Default = " <i>false</i> ". Replaced with <i>MinLateStatus</i> and <i>MinStatus</i> in JDF 1.3 and beyond.
<i>MinLateStatus</i> ? 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. If not specified, defaults to the value of @MinStatus . Possible values are defined in Resource/@Status .
<i>MinStatus</i> ? New in JDF 1.3	enumeration	Minimum value of Resource/@Status for the execution of this node to commence. If not specified, a default of " <i>Available</i> " is assumed if @Usage="Input" and a default of " <i>Unavailable</i> " is assumed if @Usage="Output" . Possible values are defined in Resource/@Status .
<i>PipePartIDKeys</i> ?	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 = " <i>SheetName Side Separation</i> ") and if the ResourceLink/ @PipePartIDKeys = " <i>SheetName Side</i> ", then pipe requests would be issued only once per surface. The contents of <i>PipePartIDKeys</i> MUST be a subset of the <i>PartIDKeys</i> attribute of the resource that is linked by this ResourceLink. If <i>PipePartIDKeys</i> is not specified, it defaults to the implied or explicit value of <i>PipePartIDKeys</i> of the referenced resource.
<i>PipeProtocol</i> ? New in JDF 1.1 Modified in JDF 1.2	NMTOKEN	Defines the protocol use for pipe handling. <i>JMF</i> and <i>Internal</i> 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. Allowed values include: <i>Internal</i> – Internal or virtual pipe used within a combined process. New in JDF 1.2 <i>JMF</i> – JMF-based PipePush / PipePull messages. <i>None</i> – No pipe support. If <i>PipeURL</i> is specified and <i>PipeProtocol</i> is not specified, <i>JMF</i> is assumed. If not specified, defaults to the value of the referenced Resource/@PipeProtocol .

Table 3-15: Abstract ResourceLink element (Section 3 of 3)

Name	Data Type	Description
<i>PipeURL</i> ? Modified in JDF 1.2	URL	Pipe request URL. Dynamic pipe requests from this end of a pipe SHOULD be made <i>to</i> this URL. ^a If not specified, defaults to the value of the referenced Resource/@PipeURL . 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 <i>PipeURL</i> , see Section 4.3.3, Overlapping Processing Using Pipes.
<i>ProcessUsage</i> ?	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 . The allowed values of <i>ProcessUsage</i> are defined in the appropriate process descriptions in Section 6, Processes and the parenthesized notation (e.g., Component (Cover)) for denoting the value of <i>ProcessUsage</i> is defined in Section 6.1, Process Template.
<i>rRef</i>	IDREF	Link to the target resource.
<i>rSubRef</i> ? 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.
<i>Usage</i>	enumeration	Resource usage within this JDF node. Possible values are: <i>Input</i> – The resource is an input. <i>Output</i> – The resource is an output.
<i>AmountPool</i> ? 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 MUST NOT contain any of <i>Amount</i> , <i>ActualAmount</i> , <i>MaxAmount</i> or <i>MinAmount</i>
<i>Part</i> *	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,” on page 87. For details on partitioned resources, see Section 3.9.5, Description of Partitioned Resources.

- a. Note that in most cases this 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.

3.8.1.1 AmountPool and PartAmount

Whereas ResourceLink/Part identifies the 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/Part whereas tracking of amount related attributes are described by AmountPool/PartAmount. AmountPool/PartAmount/Part MUST refer to partitions that are implicitly or explicitly referred to by ResourceLink/Part.

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

Table 3-16: AmountPool element

Name	Data Type	Description
PartAmount * New in JDF 1.1	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 .

The following table lists the generic contents of a **PartAmount** element. Further parameters of the **PartAmount** are described in the respective sections below (Table 3-18, “Abstract ImplementationLink or //AmountPool/PartAmount element,” on page 70 and Table 3-19, “Abstract PhysicalLink or //AmountPool/PartAmount element,” on page 70). Note that **PartAmount** inherits values from its parent **ResourceLink**.

Table 3-17: PartAmount element

Name	Data Type	Description
DraftOK ? New in JDF 1.1 Deprecated in JDF 1.3	boolean	If <i>true</i> , the process can commence with a draft resource partition. Replaced with <i>MinLateStatus</i> and <i>MinStatus</i> in JDF 1.3 and beyond.
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. If not specified, defaults to the value of @MinStatus . Possible values are defined in Resource/@Status .
MinStatus ? New in JDF 1.3	enumeration	Minimum value of Resource/@Status for the execution of this node to commence. If not specified, a default of “ <i>Available</i> ” is assumed if @Usage=“Input” and a default of “ <i>Unavailable</i> ” is assumed if @Usage=“Output” . Possible values are defined in Resource/@Status .
PipeURL ? New in JDF 1.1	URL	Pipe request URL for this partition. Dynamic pipe requests from this end of a pipe SHOULD be made to this URL. ^a 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.
Part New in JDF 1.1	element	Specifies the selected part that the PartAmount is valid for. This MUST be a leaf partition of the resource.

- a. Note that in most cases this 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.

3.8.2 Links to Parameter Resources

Parameter resources are linked by an instance of a **ParameterLink** element. These elements contain no further attributes or elements besides those found in the abstract **ResourceLink** element.

3.8.3 Links to Implementation Resources

Implementation resources are linked by an instance of an **ImplementationLink** element. Since 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 MUST be *Available* for node execution to commence, a node can commence when implementation resources are still *InUse* by other processes if **Start** or **StartOffset** are specified. **ImplementationLink** elements always have a **Usage** of *Input*. The attributes in Table 3-18 can occur in either an abstract **ImplementationLink** or **//AmountPool/PartAmount** element that references an implementation resource.

Table 3-18: Abstract ImplementationLink or //AmountPool/PartAmount element

Name	Data Type	Description
<i>Duration</i> ?	duration	Estimated duration during which the resource will be used.
<i>Recommendation</i> ? Deprecated in JDF 1.2	boolean	If <i>true</i> and the request cannot be fulfilled, the change MAY be logged as a Modified Audit and the job can continue. If <i>false</i> , an error occurs if the request is not fulfilled. In JDF 1.2 and beyond use <i>SettingsPolicy</i> instead.
<i>Start</i> ?	dateTime	Time and date when the usage of the implementation resource starts.
<i>StartOffset</i> ?	duration	Offset time when the resource is needed after processing has begun. If both <i>Start</i> and <i>StartOffset</i> are specified, <i>Start</i> has precedence.

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

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

3.8.4 Links to Physical Resources

Just as a physical resource inherits the contents of the abstract resource element, a **PhysicalLink** inherits the contents of the abstract **ResourceLink** element. The Table 3-19 describes additional attributes for **PhysicalLink** elements. The attributes in Table 3-19 can occur in either an abstract **PhysicalLink** or **//AmountPool/PartAmount** element that references a physical resource.

It is important to note that the order of occurrence of links to physical resources MAY be significant—most specifically with **QuantityLink** elements. 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**.

Table 3-19: Abstract PhysicalLink or //AmountPool/PartAmount element (Section 1 of 2)

Name	Data Type	Description
<i>ActualAmount</i> ? New in JDF 1.2	double	Total amount of the resource that has been produced (in a ResourceLink with <i>Usage</i> = "Output") or consumed (in a ResourceLink with <i>Usage</i> = "Input") by this node in every execution. For details see Section 3.9.4, Resource Amount
<i>Amount</i> ?	double	For a link with a <i>Usage</i> 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 <i>Usage</i> 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 3.9.4, Resource Amount). If not specified, ResourceLink/@Amount defaults to Resource/@Amount .
<i>MaxAmount</i> ? New in JDF 1.3	double	Defines the planned <i>Amount</i> including the maximum overage. If not specified, defaults to a system specified value based on <i>Amount</i> .

Table 3-19: Abstract PhysicalLink or //AmountPool/PartAmount element (Section 2 of 2)

Name	Data Type	Description
MinAmount ? New in JDF 1.3	double	Defines the planned <i>Amount</i> including the maximum underage that the Customer is willing to accept. If not specified, defaults to a system specified value based on <i>Amount</i> .
Orientation ? New in JDF 1.1	Orientation	Named orientation describing the transformation of the orientation of a physical resource relative to the ideal process coordinate that uses this resource as input or output. If <i>Orientation</i> is specified for an output resource, the node that processes the physical resource is to manipulate the resource in such a way as to reflect the transformation. The coordinate system of the resource itself is <i>not</i> modified. At most one of <i>Orientation</i> or <i>Transformation</i> MUST be specified. For details on coordinate systems, see Section 2.5, Coordinate Systems in JDF.
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 <i>PipePause</i> , see Section 4.3.3, Overlapping Processing Using Pipes.
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 <i>PipeResume</i> , see Section 4.3.3, Overlapping Processing Using Pipes.
RemotePipeEndPause ?	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 <i>RemotePipeEndPause</i> , see Section 4.3.3, Overlapping Processing Using Pipes.
RemotePipeEndResume ?	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 <i>RemotePipeEndResume</i> , see Section 4.3.3, Overlapping Processing Using Pipes.
Transformation ? New in JDF 1.1	matrix	Matrix describing the transformation of the orientation of a physical resource relative to the ideal process coordinate using this resource as input or output. If <i>Transformation</i> is specified for an output resource, the node that processes the physical resource is to manipulate the resource in such a way as to reflect the transformation. The coordinate system of the resource itself is <i>not</i> modified. At most one of <i>Orientation</i> or <i>Transformation</i> MUST be specified. For details on coordinate systems, see Section 2.5, Coordinate Systems in JDF.
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 3.8.4.1, Identification of Physical Resources for details. For resources that are solely identified by <i>ProductID</i> , Lot element(s) NEED NOT be specified.

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>

```

3.8.4.1 Identification of Physical Resources

[New in JDF 1.3](#)

MIS systems frequently include functionality for managing inventory. Many physical resources 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 physical resources 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 MUST generate a unique identifier for each unique resource lot. This unique identifier MUST then be mapped back to the correct unique resource lot by the MIS.

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 within the consumable link, See [Table 3-19](#) on page 70.

Table 3-20: Lot element (Section 1 of 2)

Name	Data Type	Description
<i>ActualAmount</i> ?	double	Total amount of the resource that has been consumed from this resource lot. The sum of all values of <i>ActualAmount</i> for all Lot elements SHOULD equal the <i>ActualAmount</i> specified in the parent ResourceLink of the Lot elements.

Table 3-20: Lot element (Section 2 of 2)

Name	Data Type	Description
<i>Amount ?</i>	double	Total amount of the resource that is planned to be consumed from this resource lot. The sum of all values of <i>Amount</i> for all Lot elements SHOULD equal the <i>Amount</i> specified in the parent ResourceLink of the Lot elements.
<i>LotID</i>	string	Unique identifier related to this resource lot. The identifier MUST be unique within the scope of all resource lots for the related <i>ProductID</i> . An MIS that uses multiple identifiers to identify a resource lot MUST assign a single unique ID to each lot, and MUST map this single unique ID to the appropriate set of multiple identifiers.
<i>Consumption ?</i>	enumeration	Used for indicating level of consumption for the Lot. Possible values are: <i>Full</i> <i>Partial</i> This attribute MUST 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 resources 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.

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 3.10.1.4, ResourceAudit for the structure of the ResourceAudit element.

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" Version="1.3">
  <ResourcePool>
    <Media ID="RI007" Class="Consumable" ProductID="3002" Status="Unavailable"
Brand="Coated Roll Stock" Dimension="2520 8640000" MediaType="Paper" Thickness="36"/>
  </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>
  </ResourceLinkPool>
</JDF>
```

3.8.5 Links to Placeholder Resources

Placeholder resources are linked by a PlaceholderLink element. *Placeholder* links, used together with the **PlaceholderResource** resource, can be employed to predefine a skeleton of a processing network consisting of process group nodes without knowing the exact nature of the interchange resources. For instance, although the deadlines for the job might be known, it might not be known whether a press run will be defined for a digital press or a conventional press.

3.8.6 Links to Intent Resources

Intent resources are linked by an instance of a IntentLink element. They have no additional parameters.

3.9 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.

3.9.1 ResourceElement – Subelement of a Resource

A ResourceElement is always a subelement of a resource or subelement of a JMF message, and is defined in the following table: Table 3-21. A ResourceElement does not inherit from the abstract **Resource**. Examples of ResourceElement resources are **SeparationSpec** and **MISDetails**.

Table 3-21: Abstract ResourceElement

Name	Data Type	Description
<i>ID</i> ? 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 <i>ID</i> . This <i>ID</i> MUST NOT be referenced by ResourceRef/@ <i>rRef</i> or ResourceLink/@ <i>rRef</i> because a ResourceRef or ResourceLink element is able to reference a resource only if it is a direct child of a ResourcePool.

3.9.2 ResourceRef – Element for Inter-Resource Linking and refElement

In some cases, it is necessary to reference resource elements directly from other elements in order to reuse information. These references are abstract ResourceRef elements. The ResourceRef's name is generated by appending the string "Ref" to the element name. Candidate elements for inter-resource linking have a data type of **refelement** in the content description tables of this chapter and Section 7, Resources. A data type of **refElement** allows one of ResourceRef or ResourceElement. The following table defines the attributes of the abstract ResourceRef element (see also Figure 3-5 and ResourceElement in Table 3-9, "Abstract Resource element," on page 54).

Table 3-22: Abstract ResourceRef element

Name	Data Type	Description
<i>rRef</i>	IDREF	Reference to the resource. The linked resource MUST be a direct child of a ResourcePool.
<i>rSubRef</i> ? 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.

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. Otherwise, the referenced structure would be a partitioned element and thus invalid when inlined. Thus the following example equivalence applies.

ResourceRef example with partition:

```
<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>
```

Valid inlined ResourceRef example with no inline partition:

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

Invalid inlined ResourceRef example with 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>
```

ResourceRef elements MAY also occur in the AncestorPool/Ancestor element of a JDF node. Resources that are referenced MUST reside in a ResourcePool. The restrictions on locations of resource elements described in Section 3.7.3, Position of Resources within JDF Nodes that apply to ResourceLink elements similarly apply to ResourceRef elements.

3.9.2.1 Status of Resources That Contain rRef References

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 3-9, “Abstract Resource element,” on page 54:

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.

3.9.2.2 Alignment of ResourceLink and ResourceRef

[New in JDF 1.1A](#)

ResourceRef elements MUST NOT contain any of the attributes and elements that are specified in the ResourceLink as defined in Section 3.8, 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. 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"
Type="DigitalPrinting" Version="1.3">
  <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>
    </ResourcePool>
  </JDF>
```

```

    </DigitalPrintingParams>
  </DigitalPrintingParams>
</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"/>
</ResourceLinkPool>
</JDF>

```

3.9.3 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**.

3.9.4 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 physical resource 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 **PhysicalLink** 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 **PhysicalLink** 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 3.9.5, *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.

3.9.4.1 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 physical resources with an amount, it MUST calculate the needed production amount in the following order: *Production Amount(Output)=*

- 1 *ComponentLink(Output)/AmountPool/PartAmount/@Amount* -
ComponentLink(Output)/AmountPool/PartAmount/@ActualAmount
- 2 *ComponentLink(Output)/@Amount* -
ComponentLink(Output)/@ActualAmount
- 3 **Component**(*Output*)/@Amount -
ComponentLink(Output)/@ActualAmount
- 4 *PhysicalLink(Input)/AmountPool/PartAmount/@Amount* -
PhysicalLink(Input)/AmountPool/PartAmount/@ActualAmount
- 5 *PhysicalLink(Input)/@Amount* -
PhysicalLink(Input)/@ActualAmount
- 6 **PhysicalResource**(*Input*)/@Amount -
PhysicalLink(Input)/@ActualAmount
- 7 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 MUST 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*) MUST be ignored when summing, since they redundantly link to the same resources without specifying an additional production amount.

3.9.4.2 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 MUST produce only the remaining *Amount*. The following figure shows the various processes, resources and *ResourceLink* elements and their corresponding entries in Table 3-23 on page 78 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:

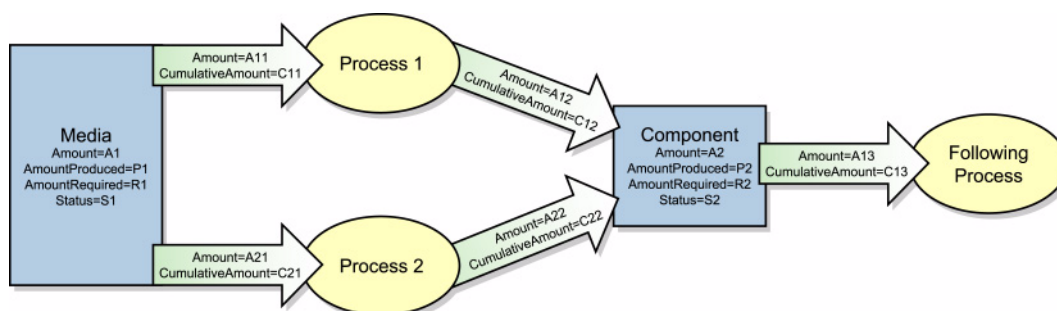


Figure 3-9: Amount handling

Table 3-23: Example of actual amount and amount handling (Section 1 of 2)

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	— —
Completed Overrun = 1234 Accumulated actual waste = 9323	390677 — 110000 Available	110000 109323 — —	100000 101234 — —	101234 101234 — Available	— —
Consumption of the output by a subsequent process					
A following process consumes 50,010 copies	390677 — 110000 Available	110000 109323 — —	100000 101234 — —	51224 101234 50000 Available	50000 50010
Additional Copy Request					

Table 3-23: Example of actual amount and amount handling (Section 2 of 2)

Process Step	A1 P1 R1 S1	A11 C11 A21 C21	A12 C12 A22 C22	A2 P2 R2 S2	A13 C13
A total of 120,000 copies are requested	390677 — 110000 Available	132000 109323 — —	120000 101234 — —	51224 101234 50000 Available	50000 50010
The 20,000 copies are produced(- underrun) Accumulated actual waste = 12123	367877 — 132000 Available	132000 132123 — —	120000 119999 — —	69989 119999 50000 Available	50000 50010
Parallel Production by a second device					
30,000 additional copies of the same resource are requested from a different device. 20% waste is assumed	367877 — 168000 Available	132000 132123 36000 0	120000 119999 30000 0	69989 119999 50000 Available	50000 50010
The 30,000 copies are produced	331856 — 168000 Available	132000 132123 36000 36021	120000 119999 30000 30100	100089 150099 50000 Available	50000 50010
Consumption by the following process					
The Consuming node is set up to consume all available Components	331856 — 168000 Available	132000 132123 36000 36021	120000 119999 30000 30100	100089 150099 50000 Available	150000 50010
All intermediate copies are consumed	331856 — 168000 Available	132000 132123 36000 36021	120000 119999 30000 30100	0 150099 150000 Unavailable	150000 150099

3.9.5 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 resource root. The partition-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.

3.9.5.1 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. For example, the following **SeparationSpec** is two color duo-tone (only *Black* and *SpotGreen*) in the part with *PageNumber* = "1".

```
<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 SeparationSpecs completely replace the CMYK in the root -->
    <SeparationSpec Name="Black"/>
    <SeparationSpec Name="SpotGreen"/>
  </LayoutElement>
</LayoutElement>
```

See also "Position of PlacedObject elements in Layout" on page 492 for additional examples and restrictions.

3.9.5.2 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.

The following example defines two sets of 12 plates for two sheets with three surfaces. Each has a common brand attribute called "Gooley". 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="Gooley" 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>
  </ExposedMedia>
  <ExposedMedia SheetName="S2" Side="Front">
    <ExposedMedia ProductID="S2FCPlateJ42" Separation="Cyan"/>
    <ExposedMedia ProductID="S2FMPlateJ42" Separation="Magenta"/>
    <ExposedMedia ProductID="S2FYPlateJ42" Separation="Yellow"/>
    <ExposedMedia ProductID="S2FKPlateJ42" Separation="Black"/>
  </ExposedMedia>
</ExposedMedia>

```

3.9.5.3 Relating PartIDKeys and Partitions

[New in JDF 1.2](#)

The *PartIDKeys* attribute (see Section 3.9.6, *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 **MUST** 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.

3.9.5.3.1 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 **MUST** also be omitted. The following example demonstrates a legal incomplete partition:

```

<Preview Class="Parameter" ID="P1" PartIDKeys="PreviewType Separation"
Status="Available">
  <Preview PreviewType="Separation">
    <Preview Separation="Cyan"/>
    <Preview Separation="Magenta"/>
  </Preview>
  <Preview PreviewType="ThumbNail"/>
</Preview>

```

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>

```

3.9.5.3.2 Number of Partition Keys per Partitioned Leaf or Node

[New in JDF 1.2](#)

Exactly one partition key **MUST** be specified per leaf or node, excluding the root node. This allows XPath-type searches on partitioned leaves. The following example demonstrates a legal partition:

```

<Preview Class="Parameter" ID="P3" PartIDKeys="PreviewType Separation"
Status="Available">
  <Preview PreviewType="Separation">
    <Preview Separation="Cyan"/>
  </Preview>
</Preview>

```

The following example demonstrates an illegal incomplete partition since more than one partition key is specified in the leaf:

```

<Preview Class="Parameter" ID="P4" PartIDKeys="PreviewType Separation"
Status="Available">
  <Preview PreviewType="Separation" Separation="Cyan"/>
</Preview>

```

3.9.5.3 Degenerate Partitions

[New in JDF 1.2](#)

A partitionable resource **MUST NOT** contain partition keys in the root. Mapping partitioned parameters to non-partitioned resources is achieved by partitioning the Resource with exactly one leaf. The following example specifies that only "c1" be folded:

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

The **Components** in the following example are **NOT** valid:

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

3.9.5.4 Partitioning of Resource sub-Elements

[New in JDF 1.2](#)

Only resources can be partitioned. If a resource contains subelements, the subelements **MUST NOT** be partitioned. Subelements **MUST** 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. 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>
```

In this incomplete example #2, the **Media** in the leaves is not complete because it does not contain the *MediaType* attribute. *MediaType* is **not** be 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>
```

In this invalid example #3, **Media** is a subelement that **MUST 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>
```

Partitioning MAY be combined with inter-resource links, (i.e., **ResourceRef** elements.) In the following valid example #4, 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 #1.)

```
<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>
```

In this invalid example #5, **MediaRef** does not reference the leaves of **Media** but, rather, to the root of **Media**. It is equivalent to the invalid example #3.

```
<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>
```

3.9.5.5 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.

Table 3-24: 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.

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 "Abstract Resource element" on page 54.) specified or inherited in the logical partition MUST be ignored and replaced by the attributes and subelements of the master partition.

Example: 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">
```

```

<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>

```

3.9.5.5.1 Restrictions when using Identical elements

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

The logical partition **MUST** 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** **MUST** be either a partition leaf or at the same partition level of the logical partition. In this way, the logical partition obeys the rules described in Section 3.9.5.3, *Relating PartIDKeys and Partitions*.

Example 1: 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 rRef="L1">
  <Part SheetName="S2" Side="Back" Separation="Black"/>
</ExposedMediaLink>

```

8 Example 2 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>

```

3.9.6 PartIDKeys Attribute and Partition Keys

[New in JDF 1.2](#)

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 3-25: Partitionable resource element

Name	Data Type	Description																																																									
PartIDKeys ? Modified in JDF 1.3	enumerations	<p>List of attribute names that are used to separate the individual parts. <i>PartIDKeys</i> also defines the sequence from root to leaf in which the <i>PartIDKeys</i> MUST occur in the partitioned resource. Each entry in the <i>PartIDKeys</i> list MUST occur only once. <i>PartIDKeys</i> MUST NOT be specified below the root of a partitioned resource. Possible values are:</p> <table border="1"> <tbody> <tr> <td><i>Bindery-</i></td> <td><i>DocSheetIndex</i></td> <td><i>RunPage</i></td> </tr> <tr> <td><i>SignatureName</i></td> <td><i>DocTags</i></td> <td><i>RunTags</i></td> </tr> <tr> <td><i>BlockName</i></td> <td><i>Edition</i></td> <td><i>RunSet</i></td> </tr> <tr> <td><i>BundleItemIndex</i></td> <td><i>EditionVersion</i></td> <td><i>SectionIndex</i></td> </tr> <tr> <td><i>CellIndex</i></td> <td><i>FountainNumber</i></td> <td><i>Separation</i></td> </tr> <tr> <td><i>Condition</i></td> <td><i>ItemNames</i></td> <td><i>SetDocIndex</i></td> </tr> <tr> <td><i>DeliveryUnit0</i></td> <td><i>LayerIDs</i></td> <td><i>SetIndex</i></td> </tr> <tr> <td><i>DeliveryUnit1</i></td> <td><i>Location</i></td> <td><i>SetRunIndex</i></td> </tr> <tr> <td><i>DeliveryUnit2</i></td> <td><i>Option</i></td> <td><i>SetSheetIndex</i></td> </tr> <tr> <td><i>DeliveryUnit3</i></td> <td><i>PageNumber</i></td> <td><i>SetTags</i></td> </tr> <tr> <td><i>DeliveryUnit4</i></td> <td><i>PageTags</i></td> <td><i>SheetIndex</i></td> </tr> <tr> <td><i>DeliveryUnit5</i></td> <td><i>PlateLayout</i></td> <td><i>SheetName</i></td> </tr> <tr> <td><i>DeliveryUnit6</i></td> <td><i>PartVersion</i></td> <td><i>Side</i></td> </tr> <tr> <td><i>DeliveryUnit7</i></td> <td><i>PreflightRule</i></td> <td><i>SignatureName</i></td> </tr> <tr> <td><i>DeliveryUnit8</i></td> <td><i>PreviewType</i></td> <td><i>SubRun</i></td> </tr> <tr> <td><i>DeliveryUnit9</i></td> <td><i>ProductionRun</i></td> <td><i>TileID</i></td> </tr> <tr> <td><i>DocCopies</i></td> <td><i>RibbonName</i></td> <td><i>WebName</i></td> </tr> <tr> <td><i>DocIndex</i></td> <td><i>Run</i></td> <td><i>WebProduct</i></td> </tr> <tr> <td><i>DocRunIndex</i></td> <td><i>RunIndex</i></td> <td></td> </tr> </tbody> </table> <p>For details, see Table 3-26, “Part element,” on page 87. Note that <i>Part/@Sorting</i> and <i>Part/@SortAmount</i> are not valid entries in <i>PartIDKeys</i>, although they are valid <i>Part</i> attributes.</p>	<i>Bindery-</i>	<i>DocSheetIndex</i>	<i>RunPage</i>	<i>SignatureName</i>	<i>DocTags</i>	<i>RunTags</i>	<i>BlockName</i>	<i>Edition</i>	<i>RunSet</i>	<i>BundleItemIndex</i>	<i>EditionVersion</i>	<i>SectionIndex</i>	<i>CellIndex</i>	<i>FountainNumber</i>	<i>Separation</i>	<i>Condition</i>	<i>ItemNames</i>	<i>SetDocIndex</i>	<i>DeliveryUnit0</i>	<i>LayerIDs</i>	<i>SetIndex</i>	<i>DeliveryUnit1</i>	<i>Location</i>	<i>SetRunIndex</i>	<i>DeliveryUnit2</i>	<i>Option</i>	<i>SetSheetIndex</i>	<i>DeliveryUnit3</i>	<i>PageNumber</i>	<i>SetTags</i>	<i>DeliveryUnit4</i>	<i>PageTags</i>	<i>SheetIndex</i>	<i>DeliveryUnit5</i>	<i>PlateLayout</i>	<i>SheetName</i>	<i>DeliveryUnit6</i>	<i>PartVersion</i>	<i>Side</i>	<i>DeliveryUnit7</i>	<i>PreflightRule</i>	<i>SignatureName</i>	<i>DeliveryUnit8</i>	<i>PreviewType</i>	<i>SubRun</i>	<i>DeliveryUnit9</i>	<i>ProductionRun</i>	<i>TileID</i>	<i>DocCopies</i>	<i>RibbonName</i>	<i>WebName</i>	<i>DocIndex</i>	<i>Run</i>	<i>WebProduct</i>	<i>DocRunIndex</i>	<i>RunIndex</i>	
<i>Bindery-</i>	<i>DocSheetIndex</i>	<i>RunPage</i>																																																									
<i>SignatureName</i>	<i>DocTags</i>	<i>RunTags</i>																																																									
<i>BlockName</i>	<i>Edition</i>	<i>RunSet</i>																																																									
<i>BundleItemIndex</i>	<i>EditionVersion</i>	<i>SectionIndex</i>																																																									
<i>CellIndex</i>	<i>FountainNumber</i>	<i>Separation</i>																																																									
<i>Condition</i>	<i>ItemNames</i>	<i>SetDocIndex</i>																																																									
<i>DeliveryUnit0</i>	<i>LayerIDs</i>	<i>SetIndex</i>																																																									
<i>DeliveryUnit1</i>	<i>Location</i>	<i>SetRunIndex</i>																																																									
<i>DeliveryUnit2</i>	<i>Option</i>	<i>SetSheetIndex</i>																																																									
<i>DeliveryUnit3</i>	<i>PageNumber</i>	<i>SetTags</i>																																																									
<i>DeliveryUnit4</i>	<i>PageTags</i>	<i>SheetIndex</i>																																																									
<i>DeliveryUnit5</i>	<i>PlateLayout</i>	<i>SheetName</i>																																																									
<i>DeliveryUnit6</i>	<i>PartVersion</i>	<i>Side</i>																																																									
<i>DeliveryUnit7</i>	<i>PreflightRule</i>	<i>SignatureName</i>																																																									
<i>DeliveryUnit8</i>	<i>PreviewType</i>	<i>SubRun</i>																																																									
<i>DeliveryUnit9</i>	<i>ProductionRun</i>	<i>TileID</i>																																																									
<i>DocCopies</i>	<i>RibbonName</i>	<i>WebName</i>																																																									
<i>DocIndex</i>	<i>Run</i>	<i>WebProduct</i>																																																									
<i>DocRunIndex</i>	<i>RunIndex</i>																																																										
PipePartIDKeys ? New in JDF 1.2	enumerations	<p>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 = “<i>SheetName Side Separation</i>”) and if the ResourceLink/@PipePartIDKeys = “<i>SheetName Side</i>”, then pipe requests would be issued only once per surface. The contents of <i>PipePartIDKeys</i> MUST be a subset of the <i>PartIDKeys</i> attribute of the resource that is linked by this <i>ResourceLink</i>. If <i>PipePartIDKeys</i> is not specified, it defaults to <i>PartIDKeys</i>, (i.e., maximum granularity.) <i>PipePartIDKeys</i> MUST NOT be specified below the root of a partitioned resource. For details on partitioned resources, see "Description of Partitioned Resources" on page 79.</p>																																																									
Identical ?	element	Cross reference to a logical partition. For details on logical partitions and the <i>Identical</i> element, see Section 3.9.5.5, Logical partitions and the <i>Identical</i> element.																																																									
Resource *	element	Nested resource elements that contain the appropriate partition keys as specified in <i>PartIDKeys</i> . These elements MUST be of the same name and type as the root <i>Resource</i> element. They represent the individual parts or groups of parts.																																																									

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 partitionable 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 3.9.7.2, 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.

Table 3-26: Part element (Section 1 of 7)

Name	Data Type	Description
<i>BinderySignatureName</i> ? New in JDF 1.2	NMTOKEN	Name of the BinderySignature used in a StrippingParams description.
<i>BlockName</i> ? New in JDF 1.1	NMTOKEN	Identifies a CutBlock from a Cutting process. The value of this attribute MUST match the value of the <i>BlockName</i> attribute of a CutBlock .
<i>BundleItemIndex</i> ? New in JDF 1.2	IntegerRangeList	The BundleItemIndex attribute selects a set of BundleItem elements from a Component resource.
<i>CellIndex</i> ? 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.
<i>Condition</i> ? New in JDF 1.2	NMTOKEN	The <i>Condition</i> 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. See Table 3-27 for values of <i>Condition</i> .
<i>DeliveryUnit0</i> ? ... <i>DeliveryUnit9</i> ? New in JDF 1.3	NMTOKEN	Specifies a hierarchical manifest of delivery packages where <i>DeliveryUnit0</i> specifies the most granular bundle. <i>DeliveryUnit<N+1></i> specifies the next most granular bundle in packing after <i>DeliveryUnit<N></i> . Bundles can be packaged with varying numbers of products. <i>DeliveryUnit<N+1></i> MUST occur before <i>DeliveryUnit<N></i> in PartIDKeys . Note that <i>N</i> is a placeholder for the values 0 through 9.
<i>DocCopies</i> ?	IntegerRangeList	Identifies a set of document copies to which the partition applies. <i>DocCopies</i> is a logical reference that is independent of the RunList structure and MUST NOT be used as an explicit partition key for RunList resources.
<i>DocIndex</i> ?	IntegerRangeList	The <i>DocIndex</i> attribute selects a set of logical instance documents from a RunList resource. <i>DocIndex</i> is a logical reference that is independent of the RunList structure and MUST NOT be used as an explicit partition key for RunList resources.

Table 3-26: Part element (Section 2 of 7)

Name	Data Type	Description
<i>DocRunIndex</i> ?	IntegerRangeList	The <i>DocRunIndex</i> attribute selects a set of logical pages from instance documents of a RunList resource. For example, <i>DocRunIndex</i> = "0 -1" specifies the first and last page of every copy of every selected instance document (assuming that additional partitioning using <i>DocCopies</i> and/or <i>DocIndex</i> is not also specified). <i>DocRunIndex</i> is a logical reference that is independent of the RunList structure and MUST NOT be used as an explicit partition key for RunList resources. The index always refers to entries of the entire RunList and MUST NOT be modified if only a part of the RunList is spawned. Specifying <i>DocRunIndex</i> does not modify the index of a RunList entry and therefore does not reposition pages on a Layout .
<i>DocSheetIndex</i> ?	IntegerRangeList	The <i>DocSheetIndex</i> attribute selects a set of logical sheets from individual instance documents. For example <i>DocSheetIndex</i> = "0 -1" specifies the first and last sheet of every selected copy of every instance document (assuming that additional partitioning using <i>DocCopies</i> and/or <i>DocIndex</i> is not also specified). <i>DocSheetIndex</i> is a logical reference that is independent of the RunList structure and MUST NOT be used as an explicit partition key for RunList resources. The index always refers to entries of the entire RunList and MUST NOT be modified if only a part of the RunList is spawned. Specifying <i>DocSheetIndex</i> does not modify the index of a RunList entry and therefore does not reposition pages on a Layout .
<i>DocTags</i> ? New in JDF 1.3	NMTOKENS	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, i.e., from the PDL, value of the document in the RunList matches any of the entries in <i>DocTags</i> Note that being a multi-set RunList implies being a multi-document RunList as well.
<i>Edition</i> ? New in JDF 1.3	string	An <i>Edition</i> 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.
<i>EditionVersion</i> ? New in JDF 1.3	string	An edition version is an OPTIONAL subset of a single edition. In order to ship inserts, editions might be subdivided into edition versions.
<i>FountainNumber</i> ?	integer	Zero-based position index of the fountain. Used to partition fountains along the axis of a roller; can be used for web printing.
<i>ItemNames</i> ? New in JDF 1.2	NMTOKENS	List of items to select from a Bundle . If not specified, all BundleItem elements are processed.
<i>LayerIDs</i> ? New in JDF 1.1	IntegerRangeList	The <i>LayerIDs</i> attribute selects a set layers that are defined by LayerID. If not specified, all layers are processed.

Table 3-26: Part element (Section 3 of 7)

Name	Data Type	Description
<i>Location ?</i>	string	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 3.9.6.2, Locations of Physical Resources for details on specifying locations.
<i>Option ?</i>	string	Option of an RFQ. Used mainly in Intent resources.
<i>PageNumber ?</i>	IntegerRangeList	Page number in a Component or document, (e.g., FileSpec that is not described as a RunList). References an index in a PageList .
<i>PageTags ?</i> New in JDF 1.3	NMTOKENS	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, i.e., from the PDL, value of the page in the RunList matches any of the entries in <i>PageTags</i> .
<i>PartVersion ?</i>	string	Version identifier, (e.g., the language version of a catalog).
<i>PlateLayout ?</i> 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)
<i>PreflightRule ?</i> New in JDF 1.2	string	Definition of the specific parts of a PreflightReportRulePool/PRRule used in preflight applications.
<i>PreviewType ?</i> New in JDF 1.1 Modified in JDF 1.2	enumeration	Type of the preview. Possible values are: <i>SeparatedThumbNail</i> – Very low resolution separated preview. <i>Separation</i> – Separated preview in medium resolution. <i>SeparationRaw</i> – Separated preview in medium resolution with no compensation. New in JDF 1.2 <i>ThumbNail</i> – Very low resolution RGB preview. <i>Viewable</i> – RGB preview in medium resolution. If both <i>PreviewType</i> and Preview/@PreviewUsage or PreviewGenerationParams/@PreviewUsage are specified, they MUST match.
<i>ProductionRun ?</i> New in JDF 1.3	string	Defines one run of a Device which will produce one or more components in parallel.
<i>RibbonName ?</i>	string	A string that uniquely identifies each ribbon. Multiple ribbons are created out of one web after dividing in case of web printing.
<i>Run ?</i>	string	The <i>Run</i> attribute selects an individual RunList partition from a RunList resource.

Table 3-26: Part element (Section 4 of 7)

Name	Data Type	Description
<i>RunIndex</i> ?	IntegerRangeList	The <i>RunIndex</i> 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 (~) and no whitespace. For example, <i>RunIndex</i> = "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 <i>RunIndex</i> = "0 ~ -1" refers to a complete range of pages, from first to last. <i>RunIndex</i> is a logical reference that is independent of the RunList structure and MUST NOT be used as an explicit partition key for RunList resources. The index always refers to entries of the entire RunList and MUST NOT be modified if only a part of the RunList is spawned. Specifying <i>RunIndex</i> does not modify the index of a RunList entry and therefore does not reposition pages on a Layout .
<i>RunPage</i> ? 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 : <pre><RunList URL="doc.pdf" (...)/></pre> is split into: <pre><RunList PartIDKeys="RunPage" (...)> <RunList URL="doc_page0.pdf" RunPage="0" (...)/> <RunList URL="doc_page1.pdf" RunPage="1" (...)/> </RunList></pre>
<i>RunSet</i> ? New in JDF 1.3	string	Generic group of elements in a RunList . If partitioning a RunList by <i>RunSet</i> and <i>Run</i> , then <i>RunSet</i> SHOULD be specified closer to the root.
<i>RunTags</i> ? New in JDF 1.1	NMTOKENS	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 <i>RunTag</i> of the RunList matches any of the entries in <i>RunTags</i> . <i>Note</i> the difference between <i>RunTags</i> and <i>PageTags</i> , <i>DocTags</i> or <i>SetTags</i> . <i>PageTags</i> is used to identify classes of individual pages having differing JDF parameterization. Similarly, <i>DocTags</i> is used to identify classes of individual documents and <i>SetTags</i> is used to identify classes of individual sets each having differing JDF parameterization. <i>RunTags</i> 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, <i>RunTags</i> MAY be explicitly set for an entire RunList by use of the <i>RunTag</i> attribute. The <i>SetTags</i> , <i>DocTags</i> and <i>PageTags</i> partition keys are always set implicitly and always refer to the granularity within a RunList implied by their names.

Table 3-26: Part element (Section 5 of 7)

Name	Data Type	Description
SectionIndex ? New in JDF 1.2	IntegerRangeList	List of sections in a StrippingParams .
Separation ?	string	<p>Identifies the separation name. Possible values include:</p> <p><i>Composite</i> – Non-separated resource.</p> <p><i>Separated</i> – The resource is separated, but the separation definition is handled internally by the resource, such as a PDF file that contains SeparationInfo dictionaries.</p> <p><i>Cyan</i> – Process color.</p> <p><i>Magenta</i> – Process color.</p> <p><i>Yellow</i> – Process color.</p> <p><i>Black</i> – Process color.</p> <p><i>Red</i> – Additional process color.</p> <p><i>Green</i> – Additional process color.</p> <p><i>Blue</i> – Additional process color.</p> <p><i>Orange</i> – Additional process color.</p> <p><i>Spot</i> – Generic spot color. Used when the exact nature of the spot color is unknown.</p> <p><i>Varnish</i> – Varnish.</p> <p>Other values include any separation name defined in the <i>Name</i> attribute of a Color element in the ColorPool.</p> <p>When <i>Separation</i> 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 "ColorantControl" on page 359.</p>
SetDocIndex ? New in JDF 1.2	IntegerRangeList	<p>The <i>SetDocIndex</i> attribute selects a set of logical instance documents from instance document sets of a RunList resource. For example, <i>SetDocIndex</i> = "0 -1" specifies the first and last page of every copy of every selected instance document set. <i>SetDocIndex</i> is a logical reference that is independent of the RunList structure and MUST NOT be used as an explicit partition key for RunList resources. The index always refers to entries of the entire RunList and MUST NOT be modified if only a part of the RunList is spawned. Specifying <i>SetDocIndex</i> does not modify the index of a RunList entry and therefore does not reposition pages on a Layout.</p>
SetIndex ? New in JDF 1.1	IntegerRangeList	<p>The <i>SetIndex</i> attribute selects a set of logical instance document sets from a RunList resource. <i>SetIndex</i> is a logical reference that is independent of the RunList structure and MUST NOT be used as an explicit partition key for RunList resources. The index always refers to entries of the entire RunList and MUST NOT be modified if only a part of the RunList is spawned. Specifying <i>SetIndex</i> does not modify the index of a RunList entry and therefore does not reposition pages on a Layout.</p>

Table 3-26: Part element (Section 6 of 7)

Name	Data Type	Description
<i>SetRunIndex</i> ? New in JDF 1.2	IntegerRangeList	The <i>SetRunIndex</i> attribute selects a set of logical pages from in-stance document sets of a RunList resource. For example, <i>SetRunIndex</i> = "0 -1" specifies the first and last page of every copy of every selected instance document set. <i>SetRunIndex</i> is a logical reference that is independent of the RunList structure and MUST NOT be used as an explicit partition key for RunList resources. The index always refers to entries of the entire RunList and MUST NOT be modified if only a part of the RunList is spawned. Specifying <i>SetRunIndex</i> does not modify the index of a RunList entry and therefore does not reposition pages on a Layout .
<i>SetSheetIndex</i> ? New in JDF 1.2	IntegerRangeList	The <i>SetSheetIndex</i> attribute selects a set of logical sheets from individual sets of instance documents. For example <i>SetSheetIndex</i> = "0 -1" specifies the first and last sheet of every selected copy of every set. <i>SetSheetIndex</i> is a logical reference that is independent of the RunList structure and MUST NOT be used as an explicit partition key for RunList resources. The index always refers to entries of the entire RunList and MUST NOT be modified if only a part of the RunList is spawned. Specifying <i>SetSheetIndex</i> does not modify the index of a RunList entry and therefore does not reposition pages on a Layout .
<i>SetTags</i> ? New in JDF 1.3	NMTOKENS	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, i.e., from the PDL, value of the set in the RunList matches any of the entries in <i>SetTags</i> .
<i>SheetIndex</i> ?	IntegerRangeList	The <i>SheetIndex</i> attribute selects a set of logical sheets from a RunList resource. In 1-up simplex printing, it is identical to <i>RunIndex</i> . <i>SheetIndex</i> is a logical reference that is independent of the RunList structure and MUST NOT be used as an explicit partition key for RunList resources.
<i>SheetName</i> ?	string	A string that uniquely identifies each sheet.
<i>Side</i> ?	enumeration	Denotes the side of the sheet. Possible values are: <i>Front</i> <i>Back</i> If <i>Side</i> 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, <i>Front</i> is a synonym for the upper side and <i>Back</i> for the down side of the web.
<i>SignatureName</i> ?	string	A string that uniquely identifies the signature within the partitionable resource.
<i>Sorting</i> ?	IntegerRangeList	Mapping from the implied partitionable resource order to a process order. The indices refer to the elements of the complete partitionable resource, not to the index in the selection of parts defined by the Part element. ^a If not specified the part order is the same as the sorting order. <i>Sorting</i> MUST NOT be used as a partition key.

Table 3-26: Part element (Section 7 of 7)

Name	Data Type	Description
<i>SortAmount ?</i>	boolean	If a sorted resource has an <i>Amount</i> attribute and <i>SortAmount</i> = <i>true</i> , each resource MUST be processed completely. If <i>SortAmount</i> = <i>false</i> (the default), each Part element MUST be processed the number of times specified in the <i>Amount</i> attribute before starting the next Part . <i>SortAmount</i> MUST NOT be used as a partition key.
<i>SubRun?</i> 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.
<i>TileID ?</i> 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 7.2.178, Tile. In JDF 1.3 and beyond, <i>TileID</i> SHOULD NOT be used to specify multiple plates per cylinder. Instead the new Resource CylinderLayout SHOULD be used.
<i>WebName ?</i>	string	A string that uniquely identifies each web.
<i>WebProduct ?</i> New in JDF 1.3	NMTOKEN	Name of a product that will be produced on a web-press.

- a. Note that *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.

— Attribute: Condition

Table 3-27: Condition attribute – possible values (Section 1 of 2)

Value	Description
<i>Good</i>	All correct components.
<i>Waste</i>	General waste.
<i>Overrun</i>	Excess Component resource(s) that were produced by running the device after the specified amount has been produces.
<i>xxxGood</i>	Like <i>Good</i> above, but where "xxx" can be the name of any JDF process, (e.g., "FeedingGood", "TrimmingGood", etc.). In the case of a combined process or process group the name of the last JDF process in the process chain is used.
<i>xxxWaste</i>	Like <i>Waste</i> above, but where "xxx" can be the name of any JDF process, (e.g., "FeedingWaste", "TrimmingWaste", etc.). In the case of a combined process or process group the name of the last JDF process in the process chain is used.
<i>BindingQualityTestFailed</i>	Failed binding quality test. The Component resource(s) with this <i>Condition</i> belong to the batch of Component resource(s) that did not pass the test.
<i>BindingQualityTestPassed</i>	Passed binding quality test. The Component resource(s) with this <i>Condition</i> belong to the batch of Component resource(s) that passed the test but were not destroyed in the process.

Table 3-27: Condition attribute – possible values (Section 2 of 2)

Value	Description
<i>BindingQualityTestWaste</i>	Passed binding quality test. The Component element(s) with this <i>Condition</i> belong to the batch of Component element(s) that passed the test but were destroyed in the process.
<i>CaliperWaste</i>	Waste by caliper on gathering / collecting.
<i>DoubleFeedWaste</i>	Waste by double feeds on feeders.
<i>IncorrectComponentWaste</i>	Waste by the attempted use of an incorrect components, (for example on a feeder.)
<i>BadFeedWaste</i>	Waste caused by a bad feed
<i>ObliqueSheetWaste</i>	Waste by oblique sheets on gathering / collecting chains.
<i>PaperJamWaste</i>	Waste by paper or other media jam.
<i>WhitePaperWaste</i>	White paper waste.

3.9.6.1 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.

3.9.6.2 Locations of Physical Resources

Unlike other kinds of resources, physical resources 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. The following example describes a set of plates that are distributed over two locations. (Note: See "Input Tray and Output Bin Names" on page 714 for additional detail on locating physical resources.

```
<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>
```

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>

```

3.9.7 Linking to Subsets of Resources

An agent can link to a subset of a resource by including a set of **Part** elements in a **ResourceLink** element in order to define a specific subset of a resource. For details of the **Part** element, please refer to Table 3-26, “Part element,” on page 87.

Partitionable hierarchies define an implied ordering of the individual parts. In the example in Section 3.9.5, Description of Partitioned Resources, the first element has a *ProductID* = S1FCPlateJ42 and the last has a *ProductID* = S2FKPlateJ42. If process ordering of a partitionable resource is important, the **Part** element of the **ResourceLink** MUST specify a *Sorting* attribute. If *Sorting* is not specified, process ordering is arbitrary. If *Sorting* is specified multiple times, the resolution of the sorting MUST be unambiguous.

The *Sorting* attribute maps the implied part ordering to a specified process ordering in a 0-based list. The first entry in *Sorting* defines the first entry to be processed. The following example, using a **ResourceLink** element, describes how the plates described in the previous example could be ordered by separation for the first sheet followed by the complete second sheet, in reverse order (back to front). Each set of two plates, as specified in the *Amount* attribute of the resource, would be processed together.

```

<ExposedMediaLink Usage="Input" rRef="E1">
  <Part SortAmount="false" Sorting="0 4 1 5 2 6 3 7 -1 ~ 8"/>
</ExposedMediaLink>

```

A partitionable resource MAY also be split into individual resources by an agent. In this case, one resource MUST be created for each individual part or set of parts. For example, a resource that describes a set of films that are also separated MAY be split into a set of resources that each describe all separations of a sheet.

3.9.7.1 Handling Amount in a ResourceLink to a Partitioned Resource

The *Amount* specified in a **ResourceLink** to a physical resource specifies the sum of individual resource partitions. Individual amounts are specified in the **PartAmount** elements of the **AmountPool**. The following example shows the **ResourceLink** that refers to the previous example 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>

```

3.9.7.2 Implicit, Sparse and Explicit PartUsage in Partitioned Resources

[New in JDF 1.2](#)

The *PartUsage* attribute defines how over-specialized ResourceLink elements are 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 sibling 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.

```

<ResourceLinkPool>
  <ExposedMediaLink Usage="Input" rRef="XM_ID">
    <Part Separation="a" SheetName="b" Side="c" PartVersion="d"/>
  </ExposedMediaLink>
</ResourceLinkPool>
<ResourcePool>
  <ExposedMedia Brand="Goey" 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>
  <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 SheetName="S1" Side="Back"/>
    </ExposedMedia>
  </ExposedMedia>
</ResourcePool>

```

The following table 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

3.9.7.3 Referencing Partitioned Resources from Nodes That Allow Multiple ResourceLink elements

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. The following examples of gathering three input sheets are equivalent.

Explicit reference of ordered partitioned resources

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Link0037" Status="Waiting"
Type="Gathering" Version="1.3">
  <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>
  </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>
  </ResourceLinkPool>
</JDF>
```

Implicit reference of ordered partitioned resources

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Link0037" Status="Waiting"
Type="Gathering" Version="1.3">
  <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>
  </ResourcePool>
  <ResourceLinkPool>
    <GatheringParamsLink Usage="Input" rRef="Gath01"/>
    <!--the ComponentLink implicitly references all three parts -->
    <ComponentLink Usage="Input" rRef="Sheets01"/>
  </ResourceLinkPool>
</JDF>
```

3.9.8 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 3-10 and Figure 3-11 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 3-10, the resource amount is split between subsequent processes. In Figure 3-11, 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 3-12 (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 3-13 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.

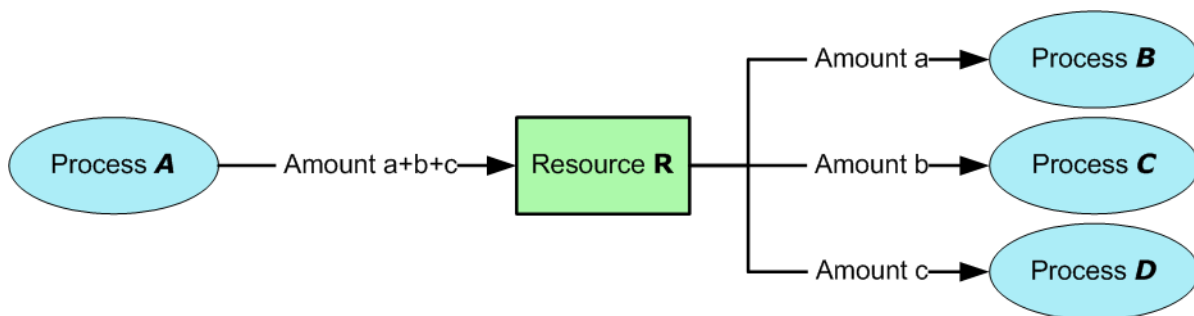


Figure 3-10: Workflow for splitting shared input resources

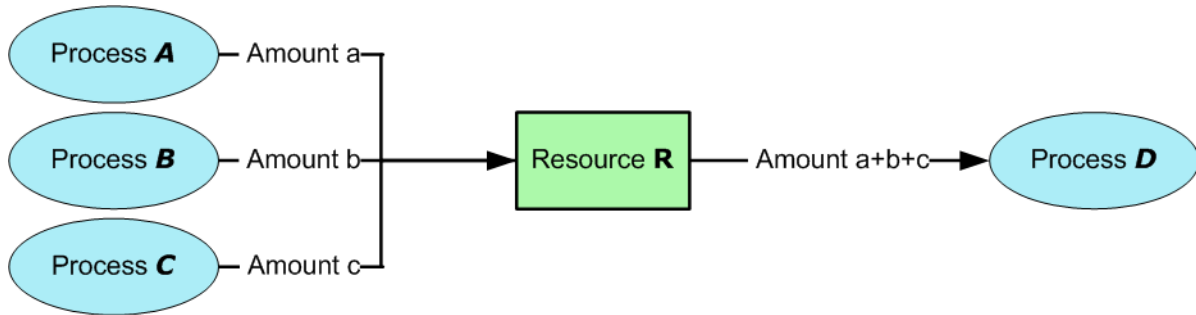


Figure 3-11: Workflow for combining shared output resources

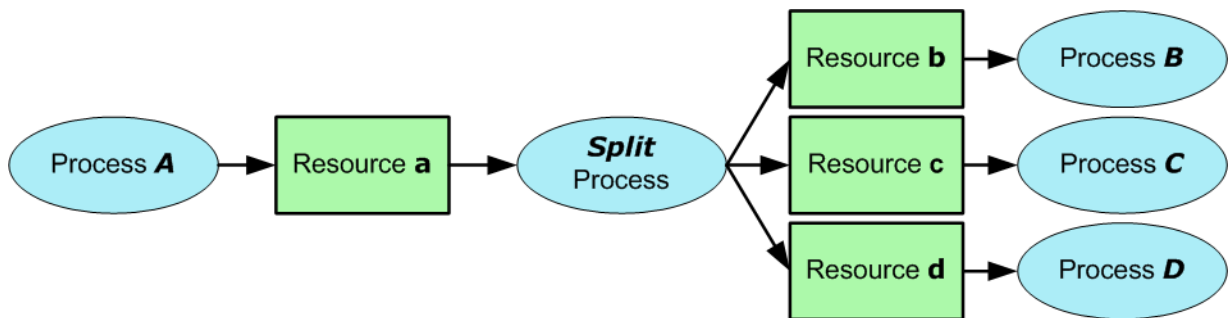


Figure 3-12: Workflow for splitting independent input resources

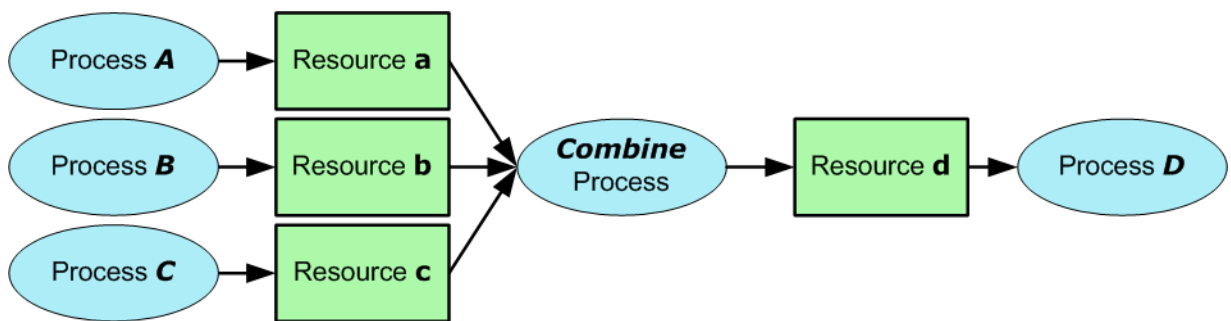


Figure 3-13: Workflow for combining independent output resources

3.10 AuditPool

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 MUST 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 element.
- Spawning and merging, including resource copying by spawned and merged elements.



AuditPool Elements

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

- Errors such as unnecessary **ResourceLink** elements, wrongly linked resources, missing resources or missing links, which might be detected by agents during a test run or by a **Notification** element.
- Actual data about the production and resource consumption by a **ResourceAudit** element.
- 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 implementation resource usage, such as a change of operators by a **PhaseTime** element, would also constitute an example of a phase time.
- Actual process scheduling data. For example, the process start and end times, as well as the final process state, as determined by a **ProcessRun** element.
- Any modification of a JDF node not covered by the preceding items, as recorded by a **Modified** or **Deleted** element.

Audit information might be used by MIS for operations such as evaluation or invoicing. The Figure 3-14 depicts the structure of the **AuditPool** and concrete elements, such as **Modified**, derived from the abstract **Audit** element. **Audit** entries are ordered chronologically, with the last entry in the **AuditPool** representing the newest. A **ProcessRun** element containing the scheduling data finalizes each process run. All subsequent entries belong to the next run. 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.

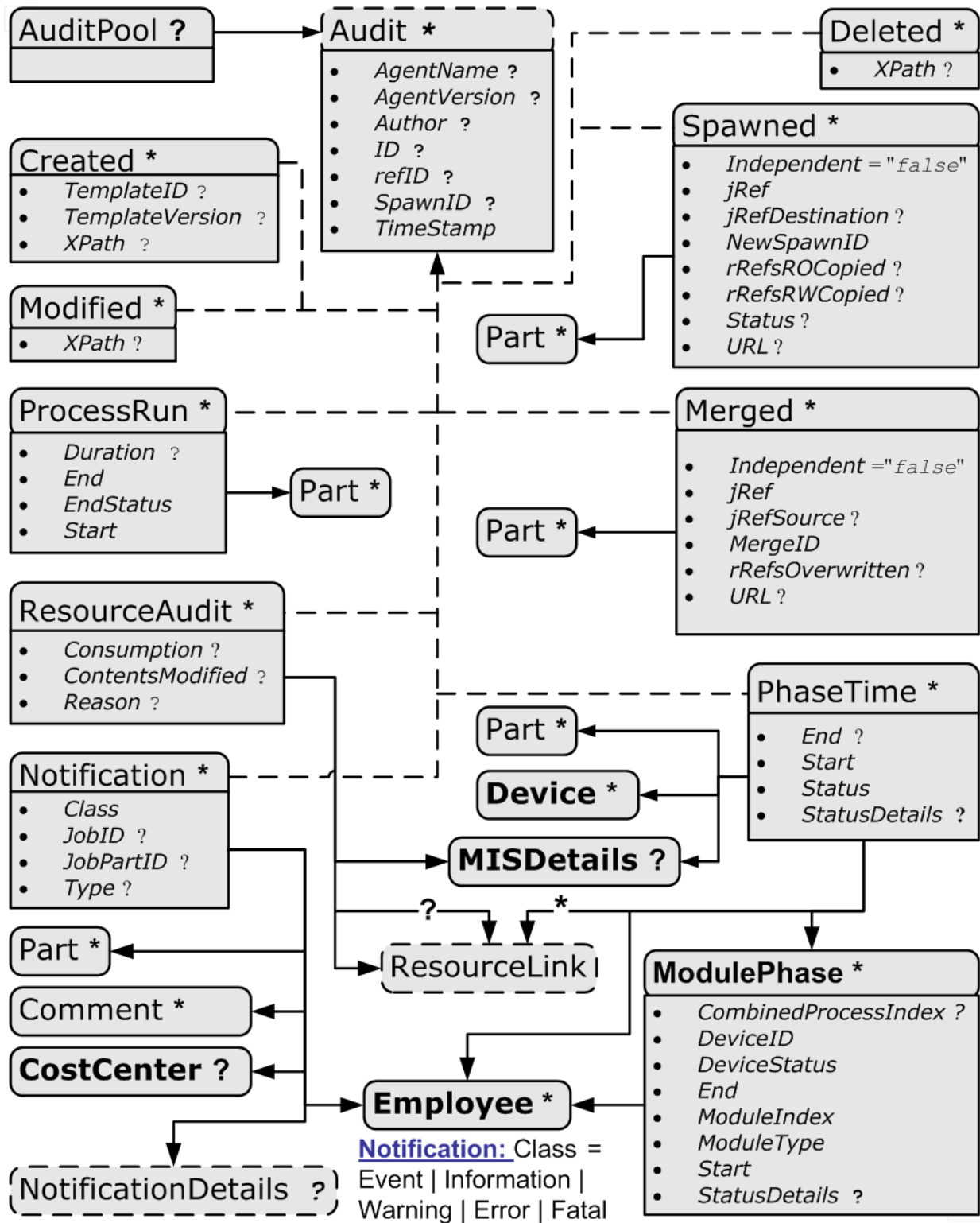


Figure 3-14: Abstract Audit element – a diagram of its structure

3.10.1 Audit Elements

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

Table 3-30: Abstract Audit element

Name	Data Type	Description
<i>AgentName</i> ? 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.
<i>AgentVersion</i> ? 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.
<i>Author</i> ? Modified in JDF 1.2	string	Text that identifies the person who made the entry. Prior to JDF 1.2, <i>Author</i> also contained information that is now encoded in <i>AgentName</i> and <i>AgentVersion</i> .
<i>ID</i> ? New in JDF 1.2	ID	<i>ID</i> of the Audit. <i>ID</i> MUST be specified if there is support to subsequently create correction Audit elements.
<i>refID</i> ? New in JDF 1.2	IDREF	Reference to a previous Audit that this Audit corrects. The referenced Audit MUST reside in the same AuditPool.
<i>SpawnID</i> ? New in JDF 1.1	NMTOKEN	Text that identifies the spawned processing step when the entry was generated. This is a copy of the <i>SpawnID</i> attribute of the root JDF node of the process that generates the Audit at the time the Audit is generated.
<i>TimeStamp</i>	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.

Listed in the following sections are the elements derived from the abstract Audit element. Following the description of each element is a table outlining the attributes associated with that element.

3.10.1.1 ProcessRun

This element serves two related functions; the second function is described after Table 3-31. 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 3-31.

Table 3-31: ProcessRun audit element (Section 1 of 2)

Name	Data Type	Description
<i>Duration</i> ?	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 <i>Status</i> is <i>InProgress</i> , <i>Setup</i> or <i>Cleanup</i> .
<i>End</i>	dateTime	Date and time at which the process ended.

Table 3-31: ProcessRun audit element (Section 2 of 2)

Name	Data Type	Description
<i>EndStatus</i>	enumeration	The <i>Status</i> of the process at the end of the run. For a description of process states, see Table 3-4, “JDF node,” on page 41. Possible values are: <i>Aborted</i> <i>Completed</i> <i>FailedTestRun</i> <i>Ready</i> <i>Stopped</i> – Deprecated in JDF 1.3 The execution of the node is stopped and might commence at a later time. In JDF 1.3 and beyond, <i>Stopped</i> is not an end state.
<i>Start</i>	dateTime	Date and time at which the process started.
<i>Part</i> * New in JDF 1.1	element	Describes which parts of a process this <i>ProcessRun</i> belongs to. If <i>Part</i> is not specified for a <i>ProcessRun</i> , 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 <i>Part</i> elements define, unambiguously, the processing of the sheet to which the <i>ProcessRun</i> refers.

The second function of a *ProcessRun* element 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 above. 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 MUST 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.

3.10.1.2 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 3-32: Notification audit element (Section 1 of 2)

Name	Data Type	Description
<i>Class</i>	enumeration	Class of the notification. Possible values, in order of severity from lowest to highest, are: <i>Event</i> – Indicates that a pure event due to any activity has occurred, (e.g., machine events, operator activities, etc.). This class is used for the transfer of conventional event messages. In case of <i>Class</i> = “ <i>Event</i> ”, further event information is to be provided by the <i>Type</i> attribute and <i>NotificationDetails</i> element. <i>Information</i> – Any information about a process which cannot be expressed by the other classes. No user interaction is needed. <i>Warning</i> – Indicates that a minor error has occurred, and an automatic fix was applied. Execution continues. <i>Error</i> – Indicates that an error has occurred that requires user interaction. Execution cannot continue. <i>Fatal</i> – Indicates that a fatal error led to abortion of the process.
<i>JobID</i> ? New in JDF 1.3	string	<i>JobID</i> that this <i>Notification</i> applies to. <i>JobID</i> MUST NOT be specified when <i>Notification</i> resides in an <i>Audit</i> element. <i>Notification/@JobID</i> MAY be specified within a JMF message.
<i>JobPartID</i> ? New in JDF 1.3	string	<i>JobPartID</i> that this <i>Notification</i> applies to. <i>JobPartID</i> MUST NOT be specified when <i>Notification</i> resides in an <i>Audit</i> element. <i>Notification/@JobPartID</i> MAY be specified within a JMF message.

Table 3-32: Notification audit element (Section 2 of 2)

Name	Data Type	Description
<i>Type</i> ?	NMTOKEN	Identifies the type of notification. Also defines the name of the abstract NotificationDetails element. ^a A list of predefined Notification types is compiled in "NotificationDetails" on page 709.
Comment *	telem	A Comment element contains a verbose, human-readable description of the event. If the value of the <i>Class</i> attribute is one of <i>Information</i> , <i>Warning</i> , <i>Error</i> or <i>Fatal</i> , at least one Comment element SHOULD be specified. Otherwise (including for <i>Class</i> = "Event"), Comment elements are OPTIONAL.
CostCenter ?	element	The cost center to which this event is to be charged.
Employee *	refelement	The employee associated with this event.
NotificationDetails ?	element	Abstract element which is a placeholder for additional structured information. It provides additional information beyond the <i>Class</i> and <i>Type</i> attribute and beyond the Comment element. For a list of supported NotificationDetails elements, see "NotificationDetails" on page 709.
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.

a. Type allows parsers that do not have access to the schema to find the instance of **NotificationDetails**.

3.10.1.2.1 NotificationDetails

The abstract **NotificationDetails** element is a placeholder only with no additional attributes. For a list of supported **NotificationDetails** elements, see "NotificationDetails" on page 709.

3.10.1.3 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 MAY also be used to log the actual time spans when implementation resources 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 MUST NOT overlap in time. **PhaseTime** elements that apply to different partitions MAY overlap in time in order to indicate parallel processing.

Table 3-33: PhaseTime audit element (Section 1 of 2)

Name	Data Type	Description
<i>End</i> ? 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 5.8.9, Status for details.,
<i>Start</i>	dateTime	Date and time of the beginning of the phase.

Table 3-33: PhaseTime audit element (Section 2 of 2)

Name	Data Type	Description
Status Modified in JDF 1.3	enumeration	Status of the phase. Possible values of JDF node states are: <i>TestRunInProgress</i> <i>Setup</i> <i>InProgress</i> <i>Cleanup</i> <i>Spawned</i> – Deprecated in JDF 1.3 <i>Suspended</i> New in JDF 1.3 <i>Stopped</i> The states listed above are a subset of the possible states of a JDF node. For all possible states of a JDF node see Table 3-4, “JDF node,” on page 41. The remaining set of states, i.e., <i>Ready</i> , <i>FailedTestRun</i> , <i>Aborted</i> and <i>Completed</i> , are end states and are specified in <i>ProcessRun/@EndStatus</i> .
StatusDetails ?	string	Description of the status phase that provides details beyond the enumerative values given by the Status attribute. For a list of supported values, see "StatusDetails Supported Strings" on page 703.
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 reelement 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 resource(s) was active during this phase, this reelement SHOULD link to that/those Employee resource(s).
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 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.

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 MAY overlap in time with one another. **ModulePhase** elements are defined in the following table.

Table 3-34: ModulePhase audit element

Name	Data Type	Description
<i>CombinedProcessIndex</i> ? New in JDF 1.3	IntegerList	<i>CombinedProcessIndex</i> attribute specifies the indices of individual processes in the <i>Types</i> attribute to which a ModulePhase in a <i>Combined</i> or <i>ProcessGroup</i> node belongs. Multiple entries in <i>CombinedProcessIndex</i> specify that the Module specified by ModulePhase is executing the respective multiple processes in the <i>Combined</i> node.
<i>DeviceID</i>	string	ID of the device that the module described by this ModulePhase belongs to. This MUST be the <i>DeviceID</i> attribute of one of the <i>Device</i> elements specified in the <i>PhaseTime</i> .
<i>DeviceStatus</i>	enumeration	Status of the device module. Possible values are: <i>Unknown</i> . – The module status is unknown. <i>Idle</i> – The module is not used, (e.g., a color print module that is inactive during a black-and-white print). <i>Down</i> – The module cannot be used. It might be broken, switched off etc. <i>Setup</i> – The module is currently being set up. <i>Running</i> – The module is currently executing. <i>Cleanup</i> – The module is currently being cleaned. <i>Stopped</i> – 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. These states are analog to the device states of Table 5-58, “ModuleStatus element,” on page 187.
<i>End</i> ? Modified in JDF 1.3	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.
<i>ModuleIndex</i> Modified in JDF 1.2	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 MUST be unique in the scope of the device.
<i>ModuleType</i>	NMTOKEN	Module description. The allowed values depend on the type of device that is described. The predefined values are listed in “ModuleType Supported Strings” on page 707.
<i>Start</i>	dateTime	Date and time of the beginning of the module phase.
<i>StatusDetails</i> ?	string	Description of the module status phase that provides details beyond the enumerative values given by the <i>DeviceStatus</i> attribute. For a list of supported values, see “StatusDetails Supported Strings” on page 703.
Employee *	refelement	References to Employee resources that are working during this module phase on this module. (The module is specified by the attributes <i>ModuleIndex</i> and <i>ModuleType</i>).

3.10.1.4 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 3-35: ResourceAudit element

Name	Data Type	Description
<i>ContentsModified</i> ?	boolean	Specifies that a modification has occurred but that the original resource has been deleted.
<i>Reason</i> ? New in JDF 1.1	enumeration	Reason for the modification. One of: <i>OperatorInput</i> – Human update that corrects inconsistencies from automated data collection. <i>PlanChange</i> – The resource was modified due to a change of plan before actual processing. <i>ProcessResult</i> – The actual consumption.
MISDetails ? New in JDF 1.3	element	Specifies how the costs associated with this ResourceAudit are to be charged.
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 <i>@Locked="true"</i> .
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 3.8, **ResourceLinkPool** and **ResourceLink**. The partitioning of resources using **Part** elements is defined in Section 3.9.5, **Description of Partitioned Resources**.

3.10.1.4.1 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.

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" Version="1.2">
  <ResourceLinkPool>
    <MediaLink Amount="400" Usage="Input" rRef="RLink"/>
  </ResourceLinkPool>
  <ResourcePool>
    <Media Amount="400" ID="RLink" Weight="80"/>
  </ResourcePool>
</JDF>
```

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" Version="1.2">
  <ResourceLinkPool>
    <!-- Note that ActualAmount has been added to the ResourceLink -->
    <MediaLink ActualAmount="421" Amount="400" Usage="Input" rRef="RLink"/>
  </ResourceLinkPool>
  <ResourcePool>
    <Media Amount="400" ID="RPrev" Weight="80"/>
    <!--Copy of the original resource-->
    <Media Amount="421" ID="RLink" Weight="90"/>
    <!--modified resource-->
  </ResourcePool>
  <AuditPool>
    <ResourceAudit>
      <MediaLink ActualAmount="421" Amount="400" Usage="Input" rRef="RLink"/>
      <MediaLink Amount="400" Usage="Input" rRef="RPrev"/>
    </ResourceAudit>
  </AuditPool>
</JDF>

```

3.10.1.4.2 Logging Changes in Product Descriptions by Using the ResourceAudit

ResourceAudit elements MAY also be used to store the original intent resources of a product specification in a change order or request for requote. The mechanism is the same as above. 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" Version="1.2">
  <ResourceLinkPool>
    <MediaIntentLink Usage="Input" rRef="id">
      <Part Option="Option2"/>
    </MediaIntentLink>
  </ResourceLinkPool>
  <ResourcePool>
    <MediaIntent PartIDKeys="Option">
      <!-- the common MediaIntent resource details -->
      <MediaIntent Option="Option1">
        <Weight Preferred="80"/>
      </MediaIntent>
      <MediaIntent Option="Option2">
        <Weight Preferred="90"/>
      </MediaIntent>
    </MediaIntent>
  </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>

```

3.10.1.5 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 **MUST** be located in the node where the resource resides so that the spawning and merging mechanism can work effectively.

Table 3-36: Created audit element

Name	Data Type	Description
<i>ref?</i> Deprecated in JDF 1.2	IDREF	Represents the ID of the created element. Defaults to the ID of the local JDF node. Replaced with <i>XPath</i> in JDF 1.2 and beyond.
<i>TemplateID?</i> New in JDF 1.2	string	Defines the template JDF that was used as the template to create the node.
<i>TemplateVersion?</i> New in JDF 1.2	string	Defines the version of template JDF that was used as the template to create the node.
<i>XPath?</i> New in JDF 1.2	XPath	Location of the created elements or attributes relative to the parent JDF node of the Created element.

3.10.1.6 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 3-37: Deleted audit element

Name	Data Type	Description
<i>XPath?</i>	XPath	Location of the deleted elements or attributes relative to the parent JDF node of the Deleted element.

3.10.1.7 Modified

This element allows any modifications affecting a JDF node (e.g., changes made to the **NodeInfo** element or **CustomerInfo** element) to be logged. Changes that can be logged by a more specialized **Audit** element (e.g., **ResourceAudit** for resource changes) **MUST 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 **MUST** reside in the same **AuditPool** as the corresponding **Created** element.

Table 3-38: Modified audit element

Name	Data Type	Description
<i>jRef?</i> 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 <i>XPath</i> in JDF 1.2 and beyond.
<i>XPath?</i> New in JDF 1.2	XPath	Location of the modified elements or attributes relative to the parent JDF node of the Modified element.

3.10.1.8 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 3-39: Spawned audit element

Name	Data Type	Description
<i>Independent</i> = "false"?	boolean	Declares that independent jobs that have previously been merged into a big job are spawned. If it is set to <i>true</i> , the attributes <i>jRefDestination</i> , <i>rRefsROCopied</i> and <i>rRefsRWCopied</i> have no meaning and SHOULD be omitted.
<i>jRef</i>	IDREF	ID of the JDF node that has been spawned.
<i>jRefDestination</i> ?	NMTOKEN	ID of the JDF node to which the job has been spawned. ^a This attribute MUST be specified in the parent of the original node if independent jobs are spawned.
<i>NewSpawnID</i> New in JDF 1.1	NMTOKEN	Copy of the <i>SpawnID</i> of the newly spawned node. Note that a Spawned Audit MAY also contain a <i>SpawnID</i> attribute, which is the <i>SpawnID</i> of the node that this Audit is being placed into prior to spawning.
<i>rRefsROCopied</i> ?	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.
<i>rRefsRWCopied</i> ?	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 MUST 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.
<i>Status</i> ? New in JDF 1.1	enumeration	<i>Status</i> of the spawned node at the time of spawning. Allowed values are defined in <i>Status</i> in Table 3-4, "JDF node," on page 41.
<i>URL</i> ? 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 3.9.5, Description of Partitioned Resources.

a. The data type is NMTOKEN and not IDREF because the attribute refers to an external ID.

3.10.1.9 Merged

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

Table 3-40: Merged audit element (Section 1 of 2)

Name	Data Type	Description
<i>Independent</i> = "false"	boolean	Declares that independent jobs are merged into a big job for common production. If it is set to <i>true</i> , the attributes <i>jRefSource</i> and <i>rRefsOverwritten</i> have no meaning and SHOULD be omitted.
<i>jRef</i>	IDREF	ID of the JDF node that has been returned or merged.
<i>jRefSource</i> ?	NMTOKEN	ID of the JDF root node of the big job from which the spawned structure has been returned. ^a
<i>MergeID</i> New in JDF 1.1	NMTOKEN	Copy of the <i>SpawnID</i> of the merged node. Note that a Merged element MAY also contain a <i>SpawnID</i> attribute, which is the <i>SpawnID</i> of the node that this Audit is being placed into prior to merging.
<i>rRefsOverwritten</i> ?	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.

Table 3-40: Merged audit element (Section 2 of 2)

Name	Data Type	Description
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 3.9.5, Description of Partitioned Resources .

- a. The data type is NMTOKEN and not IDREF because the attribute refers to an external ID.

3.11 JDF Extensibility

JDF is meant to be flexible and therefore useful to any vendor, as each vendor will have specific data to include in the JDF files. JDF is able to provide this kind of versatility by using the XML namespaces. This section describes how JDF uses the XML extension mechanisms.

3.11.1 Namespaces in XML

JDF Extensibility is implemented using XML Namespaces [XMLNS]. XML namespaces are defined by *xmlns* attributes. A general example is provided below. 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" ... >
. . .
<SomeJDFDefinedResource name="abc"
foo:specialname="cba">
. . .
    <foo:PrivateStuff type=""/>
. . .
</SomeJDFDefinedResource>
. . .
</JDF>
```



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.

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 (see [Section A.3.3, Defined JDF enumeration Data Types](#)). All namespace prefixes MUST be declared using the standard *xmlns:prefix* attribute declarations.

3.11.1.1 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.

3.11.1.2 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.

3.11.2 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 3.2.3, 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 MUST be separated by a single colon (:) as shown in the following example.

```
<JDF Type="myCompaniesNS:MyVeryImportantProcess" xmlns=
```

```
"http://www.CIP4.org/JDFSchema_1_1" xmlns:myCompaniesNS="my companies namespace URI" ...
>
. . .
</JDF>
```

The use of namespace prefixes in the *Type* attribute is for extensions only. Standard JDF process types MUST be specified without a prefix in the *Type* attribute or the *Types* attribute of a combined 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.

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.

3.11.3 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 MUST NOT be modified. However, the mechanism for creating new resources in a separate namespace is provided in the next section.

However, duplicate functionality MUST NOT be added to these resource types. JDF-defined attributes and elements MUST 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 resolution with a `foo:Resolution` or `foo:Res` attribute that overrides the JDF defined resolution parameter (see attribute *Resolution* of resource **RenderingParams** in Section 7.2.152, `RenderingParams`).

3.11.4 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*. For instance, if a JDF instruction contains the following text.

```
<TrappingParams TrapEndStyle="HDM:FooBar" (...)/>
```

Based on the definition of **TrappingParams**, the best assumption is to use *TrapEndStyle = Miter*.

Table 3-41: Example from TrappingParams

Name	Data Type	Description
<i>TrapEndStyle</i> = "Miter"	NMTOKEN	Instructs the trap engine how to form the end of a trap that touches another object. Possible values include: <i>Miter</i> <i>Overlap</i> Other values might be added later as a result of customer requests.

3.11.5 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 **MUST** be clearly specified and use its own namespace. These resource types **MUST** only be linked to custom-type JDF process nodes.

3.11.6 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.

3.11.7 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 **MUST** 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.

3.11.8 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 **MUST NOT** continue and a Notification audit element with *Class* = "Error" **SHOULD** be logged.

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

3.11.9 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.

3.12 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 MUST be specified with different versions in one document.

3.12.1 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 MUST 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 MUST remain valid when a new version of the JDF specification and schema is published.

3.12.2 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.

3.12.3 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 to JDF 1.2 or JDF 1.3, as well as future versions of JDF, but are not applicable to JDF 1.0.

3.12.3.1 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 MUST be optional.
 - New elements in existing elements MUST 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 MUST be extensions of existing data types. In other words the data type of an extended attribute MUST 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 MUST NOT be altered.
 - New attributes or elements MUST 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 MUST be clearly marked in the specification.
- The default values of attributes and elements MUST never be altered.
 - The default behavior that is specified when an attribute or element is missing MUST NOT be altered.

3.12.3.2 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 MUST always be backwards compatible.
 - JDF 1.x documents MUST validate against JDF 1.(x+n) schemas.
- Only one JDF schema namespace MUST be defined for all versions of JDF 1.x.
 - The namespace is http://www.CIP4.org/JDFSchema_1_1.
- The *xs:version* attribute MUST 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 or 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 MUST be available on the CIP4 website.
- The schema default values of deprecated attributes MUST be removed from the schema. Deprecated attributes MUST still be valid but MUST NOT be explicitly defaulted in the schema.

3.12.3.3 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.

3.12.3.3.1 JDF Agent Version Policies

JDF agents MUST ensure that the JDF that they generate is consistently versioned.

- An agent MUST update the *JDF/@Version* attribute when inserting new attributes or elements.
 - If an Agent is not aware of versions, it MUST 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 3.12.2, JDF Version Definition.
- The *JDF/@Version* of a spawned JDF node is identical to the *JDF/@Version* of that node in a complete JDF.

3.12.3.3.2 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 MUST 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.

Chapter 4 Life Cycle of JDF

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. Ancillary aspects of the life cycle, such as test running and error handling, are also discussed.

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 JDF node than the original, creating agent knew. For example, one agent might create a product node that specifies the product intent of a series of pages. This product 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 parameter resource.

Regardless of where in the life cycle they are written, nodes and their resources **MUST** 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 intent resources. In contrast to process resources that define precise values, intent resources 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 Section 7, Resources.

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 of processes. The various kinds of process nodes are described in Section 3.2.1, Section 3.2.2, and Section 3.2.3. Any job that specifies product intent **MUST** include nodes whose *Type* = *Product*. This representation is described in the following section.



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.

4.1.1.1 Representation of Product Intent

The product description of a job is a hierarchy of *Product* 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* nodes begin specifying the processes needed to produce the products.

Product 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 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 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 node will use the output of a product 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 product intent resources, such as **BindingIntent** or **InsertingIntent**, define how to combine multiple products. To accomplish this, the respective **Component** resources MUST be labeled according to their usage. For example, the *Cover* and *Insert* attributes use the *ProcessUsage* attribute of the respective ResourceLink elements. For more information about product intent, see Section 3.2.1, Product Intent Nodes.

4.1.2 Defining Business Objects Using Intent Resources

Business objects like requests-for-quote, quote, invoice, etc. need to reference processes at a level that is well represented by product intent nodes. It is assumed that business object metadata such as financial information, business document type, customer information, etc. is defined by an XML envelope that contains JDF as a job description. If this is not the case, the business related metadata MAY be placed into the root **NodeInfo/BusinessInfo** resource, and the customer related data MAY be placed into the root **CustomerInfo** resource.



PrintTalk Implementation

A PrintTalk implementation guide can be found at <http://www.printtalk.org>

This section sketches the usage of JDF in an E-commerce environment using the business object model that was defined by the PrintTalk [PrintTalk] consortium and is being maintained by CIP4.

The following table describes the individual business objects and their relationships. “Object type” defines the name of the XML element that defines the metadata. All object types are inherited from the abstract PrintTalk **Request** element. “References” defines the business objects that are responded to when generating the business object, and the “buyer-provider” arrow defines the direction of the transaction.

Table 4-1: Business objects as defined by PrintTalk (Section 1 of 2)

Object Type	Description	References	Direction
RFQ (Request for Quote)	Initiated by a buyer to a print supplier. It might instigate a new product process or it might supersede an existing RFQ. The Change Order and Request for Requote variations are included within Request for Quote.	None, Quote, Confirmation	B→P
Quote	Normally sent in response to a RFQ. The Requote and Change Order Quote variations are included within Quote. A Quote might supersede an existing Quote before the Print Buyer has answered with a RFQ or an Order.	RFQ, PO, Confirmation	B←P

Table 4-1: Business objects as defined by PrintTalk (Section 2 of 2)

Object Type	Description	References	Direction
Purchase Order	Typically sent as a response to a quote but might be the initial document in a well defined buyer / print supplier relationship or when ordering finished goods items. The Change Order variation is included within Purchase Order. An order might supersede an existing Order prior to the Print Provider having confirmed it.	None, Quote, Confirmation	B→P
(Order) Confirmation	Sent by the print supplier to the buyer acknowledging receipt of the purchase order. It might contain information about expected due dates and final pricing that were undetermined at the time of the quote.	PO	B←P
Cancellation	Cancels a complete job. To cancel parts of a job, one sends a new RFQ, Quote or PO for the non-cancelled parts of the job. For canceling parts of a confirmed order, the Change Order variations of these Business Objects are sent.	RFQ, Quote, PO, Confirmation	B↔P
Refusal	Used to explicitly decline a Business Object sent by the counter party. Alternatively, the non-accepted Business Object expires.	RFQ, Quote, PO	B↔P
Order Status Request	Generated anytime one party requests status from another party.	Confirmation	B↔P
Order Status Response	An Order Status Response can be sent as a response to an Order Status Request or it can be sent automatically.	Confirmation, Order Status Request	B↔P
Proof Approval Request	Provides a transport for proofing from supplier to buyer. This MAY contain MIME data or a URL where the proof is located.	Confirmation	B←P
Proof Approval Response	Contains buyer's approval or denial of a proof.	Proof Approval Request	B→P
Invoice	Typically sent once the job is shipped, but can also be sent several times when certain milestones during production are reached. can include additional charges or discounts.	Confirmation, Cancellation	B←P

In the following figure the workflow of these business objects is partly illustrated in a simplified manner. See the PrintTalk specification [PrintTalk] for a complete picture.

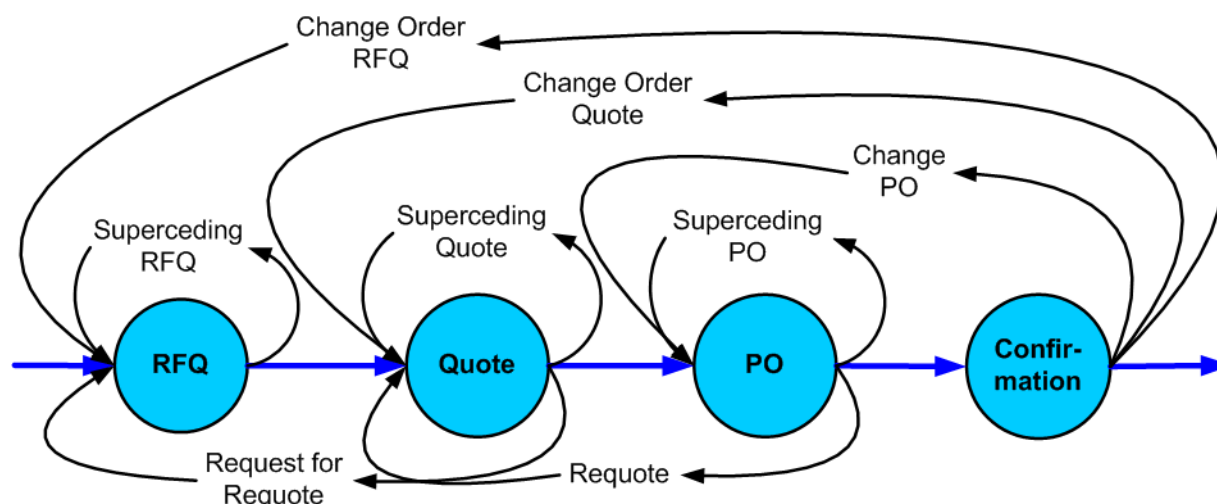


Figure 4-1: Simplified PrintTalk workflow (negotiation phase)

The node that defines an RFQ MUST contain one or more **DeliveryIntent** resources that define the amounts and methods of delivery. The *Usage* of the ResourceLink elements is *Input*, its *Type* is *Product* and the Business object is an RFQ.

The examples quoted in this section use an object model as defined by PrintTalk with the business objects defined in BusinessInfo. This does not preclude the use of other E-commerce systems. The following examples show equivalent PrintTalk and pure JDF document text. The **highlights** show the respective position of an RFQ.

PrintTalk example

```
<PrintTalk xmlns="http://www.printtalk.org/schema">
  <Header>
    Standard CXML header
  </Header>
  <Request>
    <RFQ AgentDisplayName="Lara Garcia-Daniels" AgentID="Lara" BusinessID="RFQ_ID"
    Currency="EUR" Estimate="false" Expires="2002-04-15T1700- 0800" RequestDate="2002-04-
    05T1700-0800">
      <jdf:JDF xmlns:jdf="http://www.CIP4.org/JDFSchema_1_1" ID="ScreenTest"
      JobID="ScreenJob" Status="Waiting" Type="Product" Version="1.2">
        <jdf:NodeInfo LastEnd="2000-12-24T06:02:42+01:00"/>
        </jdf:JDF>
      </RFQ>
    </Request>
  </PrintTalk>
```

Equivalent pure JDF Example

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="ScreenTest" JobID="ScreenJob"
Status="Waiting" Type="Product" Version="1.2">
  <NodeInfo LastEnd="2000-12-24T06:02:42+01:00">
    <BusinessInfo>
      <pt:RFQ xmlns:pt="http://www.printtalk.org/schema" AgentDisplayName="Lara Garcia-
      Daniels" AgentID="Lara" BusinessID="RFQ_ID" Currency="EUR" Estimate="false"
      Expires="2002-04-15T1700- 0800" RequestDate="2002-04-05T1700-0800"/>
    </BusinessInfo>
  </NodeInfo>
</JDF>
```

4.1.3 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 JDF [*@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 **Drop** elements. Each **Drop** element has a common delivery address and time, and a set of **DropItem** 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.

4.1.4 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**. The green highlighted section of the following example 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="1.3">
  <ResourcePool>
    <Component Amount="10000" Class="Quantity"
      DescriptiveName="Complete 16-page Brochure" ID="Link0003" Status="Unavailable"/>
    <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="NameSpan" Preferred="None"/>
      <MediaIntent Option="1">
        <FrontCoatings DataType="NameSpan" Preferred="Glossy"/>
      </MediaIntent>
      <BackCoatings DataType="NameSpan" 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** element 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 MUST 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 is either *Waiting* or *Ready*. If a **Device** resource is specified as input to the node, the resource MUST 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` element, the specified start and/or end time MUST 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 MUST 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.

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 4-2.

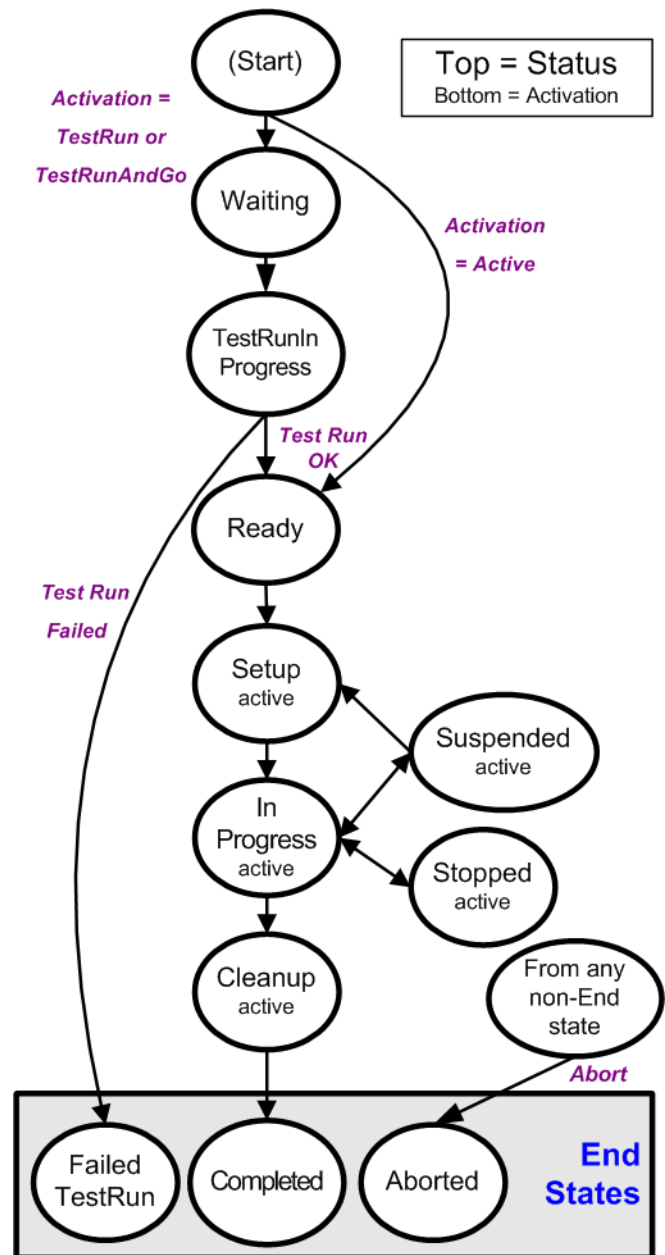


Figure 4-2: Life Cycle of a JDF node

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 **MUST** be able to identify process nodes it is capable of executing. Furthermore, each device **MUST** 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* element 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 **MUST** 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* element of a node, JDF **MAY** contain implicit routing information in the form of **Device** implementation resources.

JMF defines the *KnownControllers* query to find controllers and the *KnownDevices* query to find devices that are controlled by a controller. The information provided by these queries can be used by a controller to infer the appropriate routing for a node. In a system that does not support messaging, this information **MUST** 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 act both 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 *PhysicalLink* element to determine whether or not the input resource can be considered available. If no amount is specified in the *PhysicalLink*, the process is assumed to consume the entire physical resource.

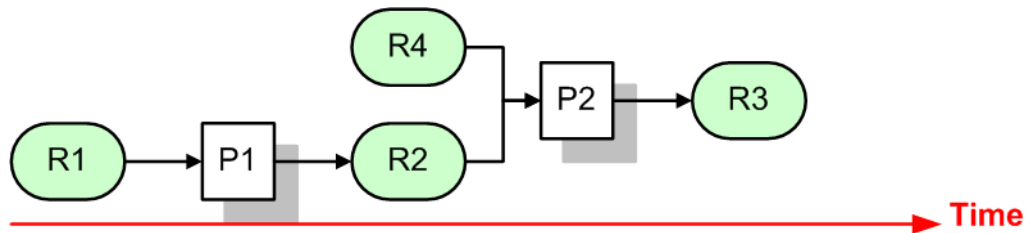


Figure 4-3: Example of a simple process chain linked by resources

Figure 4-3 depicts a simple process chain that produces and consumes *Quantity* resources and uses an implementation resource. The resources R1, R2 and R3 represent *Quantity* resources. Process P1 consumes resource R1 and produces resource R2. R2 is then completely consumed by P2, which also requires the implementation resource R4 for processing. Process P2 uses these two resources and produces resource R3. All of this is accomplished along a linear time axis.

Table 4-2 shows the value of the *Status* attribute of each of the resources and processes used in Figure 4-3. The time axis runs from left to right both in Figure 4-3 and in Table 4-2. 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 3-9, “Abstract Resource element,” on page 54. For more information about the values of the *Status* attribute of processes, see Table 3-4, “JDF node,” on page 41.

Table 4-2: 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	<i>Available</i>	<i>InUse</i>	<i>Unavailable</i>	<i>Unavailable</i>	<i>Unavailable</i>
resource R2	<i>Unavailable</i>	<i>Unavailable</i>	<i>Available</i>	<i>InUse</i>	<i>Unavailable</i>
resource R3	<i>Unavailable</i>	<i>Unavailable</i>	<i>Unavailable</i>	<i>Unavailable</i>	<i>Available</i>
resource R4	<i>Available</i>	<i>Available</i>	<i>Available</i>	<i>InUse</i>	<i>Available</i>
process P1	<i>Waiting or Ready</i>	<i>InProgress</i>	<i>Completed</i>	<i>Completed</i>	<i>Completed</i>
process P2	<i>Waiting or Ready</i>	<i>Waiting or Ready</i>	<i>Waiting or Ready</i>	<i>InProgress</i>	<i>Completed</i>

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 attribute *Amount* is used in connection with the quantifiable resources R1, R2 or R3 and their links, then the controller MUST 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* resources. 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 *PhysicalLink* elements MUST 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 “Resource Amount” on page 76. 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 MUST 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 MUST 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 MUST explicitly calculate the result. The following examples in Table 4-3 illustrate the restriction algorithms:

Table 4-3: Examples of partitioning across multiple resources (Section 1 of 2)

Input Partition 1	Input Partition 2	Output Partition	Node Partition	Description
<i>SheetName</i> = "S1"	—	—	<i>SheetName</i> = "S1"	If only the input is partitioned, the node partition is defined by the input.
<i>SheetName</i> = "S1" <i>Separation</i> = "Cyan"	—	—	<i>SheetName</i> = "S1" <i>Separation</i> = "Cyan"	If only the input is partitioned, the node partition is defined by the input.
<i>SheetName</i> = "S1" <i>Separation</i> = "Cyan"	<i>Separation</i> = "Cyan" + <i>Separation</i> = "Black" (<i>PartUsage</i> = "Implicit")	—	<i>SheetName</i> = "S1" <i>Separation</i> = "Cyan" + <i>SheetName</i> = "S1" <i>Separation</i> = "Black"	The first input is partitioned by <i>SheetName</i> and <i>Separation</i> which defines the partition key granularity. The second input is partitioned by <i>Separation</i> only but has an implied <i>SheetName</i> and has a larger but overlapping set of separation values. The separation value set is therefore defined by the second key.
<i>SheetName</i> = "S1"	—	<i>SheetName</i> = "S1" <i>Separation</i> = "Cyan"	<i>SheetName</i> = "S1" <i>Separation</i> = "Cyan"	The input and output base partitions are identical. The output further restricts the partition.
<i>SheetName</i> = "S1"	—	<i>SheetName</i> = "S2" <i>Separation</i> = "Cyan"	Error	Input and output are not overlapping. This specifies the null set.

Table 4-3: Examples of partitioning across multiple resources (Section 2 of 2)

Input Partition 1	Input Partition 2	Output Partition	Node Partition	Description
<i>SheetName</i> = "s1" <i>Separation</i> = "Magenta"	<i>Separation</i> = "Cyan" + <i>Separation</i> = "Black"	—	Error	This is an error and defines the null set. The first input is partitioned by <i>SheetName</i> and <i>Separation</i> which defines the partition key granularity. The second input is partitioned by <i>Separation</i> only and has a larger but non-overlapping set of separation values. The separation value set is therefore the null set.
<i>SheetName</i> = "s1" <i>Separation</i> = "Cyan"	<i>Separation</i> = "Cyan" + <i>Separation</i> = "Black" (<i>PartUsage</i> = "Explicit")	—	Error	The first input is partitioned by <i>SheetName</i> and <i>Separation</i> which defines the partition key granularity. The second input is partitioned by <i>Separation</i> only but has no implied <i>SheetName</i> and therefore has a non-overlapping set of partition keys. The separation value set is therefore defined by the second key.
<i>RunIndex</i> = "0 ~ 7"	—	<i>SheetName</i> = "s2"	Special	This specifies sheet s2, with all <i>PlacedObject</i> elements with an <i>Ord</i> 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, pipe dynamics are declared in the *ResourceLink* elements that reference the pipe. This allows multiple nodes to access one pipe, each of them with its own pipe buffering parameters.

In some situations, resource linking is a continuous process rather than a chronological one. 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 of quantifiable resources 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). The level is controlled by the *PhysicalLink* attributes *PipeResume*, *PipePause*, *RemotePipeEndPause* and *RemotePipeEndResume* (see Table 3-19, "Abstract *PhysicalLink* or //AmountPool/PartAmount element," on page 70). If none of them are specified, any pro-



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 every-day 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.

duced *Quantity* can be immediately consumed by the consuming end of the pipe. The unit of the buffers is defined by the *Unit* attribute of the resource.

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:

PipePartIDKeys

PipePause

PipeProtocol

PipeResume

RemotePipeEndPause

RemotePipeEndResume

The latter two—*RemotePipeEndPause* and *RemotePipeEndResume*—are used to control the level in context with pipe command messages which will be described in Section 4.3.3.2, *Dynamic Pipes*. The specified value of each of these attributes in any given node dictates the levels at which a pipe SHOULD resume or pause execution. Figure 4-5 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 consistent 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 3-9, “Abstract Resource element,” on page 54).

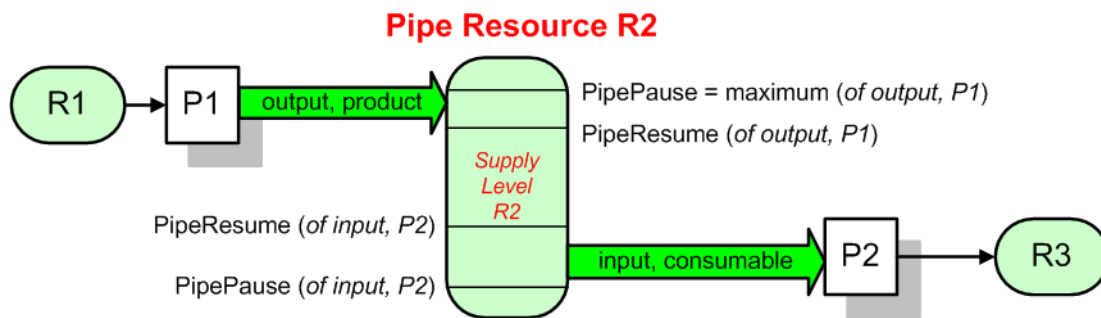


Figure 4-4: Example of a Pipe resource linking two processes

Figure 4-4 is a view on the structure and Figure 4-5 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.

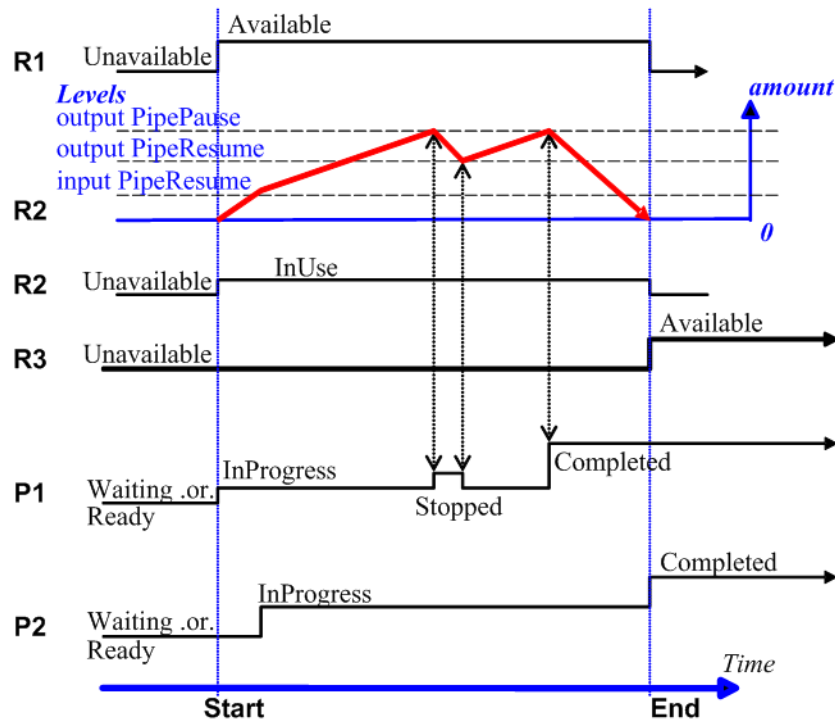


Figure 4-5: Example of status transitions in case of overlapping processing

Resource linking through pipes is controlled through the specification of the *PipePause* and *PipeResume* attributes. The intended amount of a resource MUST 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 output *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.

Conversely, if the level representing the actual amount exceeds the *PipeResume* level of the input *ResourceLink*, P2 can start or resume execution. If it falls below the *PipePause* level, P2 is halted (*Status = stopped*) unless the intended amount of the pipe resource R2 has already been produced. Then the *PipePause* level is ignored and the pipe resource is completely consumed.

In the case of output *ResourceLink* elements, the *PipeResume* value MUST be smaller than the *PipePause* value, whereas in the case of input *ResourceLink* elements, the *PipeResume* value MUST be greater than the *PipePause* value. If *PipePause* is specified for an input or an output *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 5.9, 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 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*".

4.3.3.2 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.

Dynamic pipes provide a *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 5.9, 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 **ResourceLink/@Usage** attribute, the following actions are possible.

- *Input*: The consumer sends a **PipePull** message to its *PipeURL* in order to request additional resources or a **PipePause** to halt production by the creator. The consumer sends a **PipeClose** message to the producer if the consumer does not require any further resources.
- *Output*: The creator sends a **PipePush** message to its *PipeURL* in order to deliver additional resources or a **PipePause** to halt consumption by the consumer.

When dynamic pipes are used, (i.e., when the *PipeURL* attribute is specified), the pipe buffering parameters *RemotePipeEndResume* and *RemotePipeEndPause* define the buffering parameters of the remote (controlled) end. *PipeResume* and *PipePause*, meanwhile, define the buffering parameters of the local node as described in Section 4.3.3, Overlapping Processing Using Pipes. The buffering parameters of a non-dynamic pipe might control the process that contains the **ResourceLink**, whereas the buffering parameters of a dynamic pipe control the process at the other end of the pipe. The pipe control messages described later in Section 5.9, 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 4-4: Actions generated when a dynamic-pipe buffer passes various levels

Controlling Pipe End	Situation	Message	Description
Output (creator)	$L > RemotePipeEndResume$	PipePush	Sufficient resources have been produced by the creator and are ready for delivery to the consumer.
Output (creator)	$L < RemotePipeEndPause$	PipePause	The consumer has consumed to the low water mark and MUST pause until a sufficient amount of resources have been produced.
Input (consumer)	$L < RemotePipeEndResume$	PipePull	More resources are requested from the creator and processing MAY continue by the consumer.
Input (consumer)	$L > RemotePipeEndPause$	PipePause	The creator has produced to the high water mark and MUST wait until a sufficient amount of resources have been consumed.

Dynamic pipes are initially dormant and MUST be activated by an explicit request. Dynamic pipe requests MAY be initiated by both ends of the pipe. For example, a print process might notify an off-line finishing process when a cer-

tain amount is ready by sending a `PipePush` message, or the printing process might request a new plate by sending a `PipePull` message.

4.3.3.3 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 **MUST** 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.

When using pipe resources, it is **RECOMMENDED** that scheduling data for the process be specified only in the `NodeInfo` element of the parent node of the processes linked by pipe resources in order to avoid scheduling deadlocks. In [Figure 4-5](#), for instance, the actual start and end time of the corresponding parent of P1 and P2 are marked on the time axis.

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. Implementation resources, for example, cannot be shared and are therefore always independent, and *Consumable* and *Quantity* resources 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 resources 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 **MUST 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 **MUST NOT** be *Available*.

The `ResourceLink/@MinStatus` of a `ResourceLink` that links to a draft input resource MUST 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 `DraftOK` attribute, 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 MUST 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 **MUST NOT** be re-spawned. If a node is to be spawned a second time, the previously submitted version **MUST** first be deleted, and the spawning procedure **MUST** 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 **MUST** 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 **MUST** be logged, or the appropriate **Spawned** audit element **MUST** be removed from the **AuditPool** element.

Figure 4-6 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.

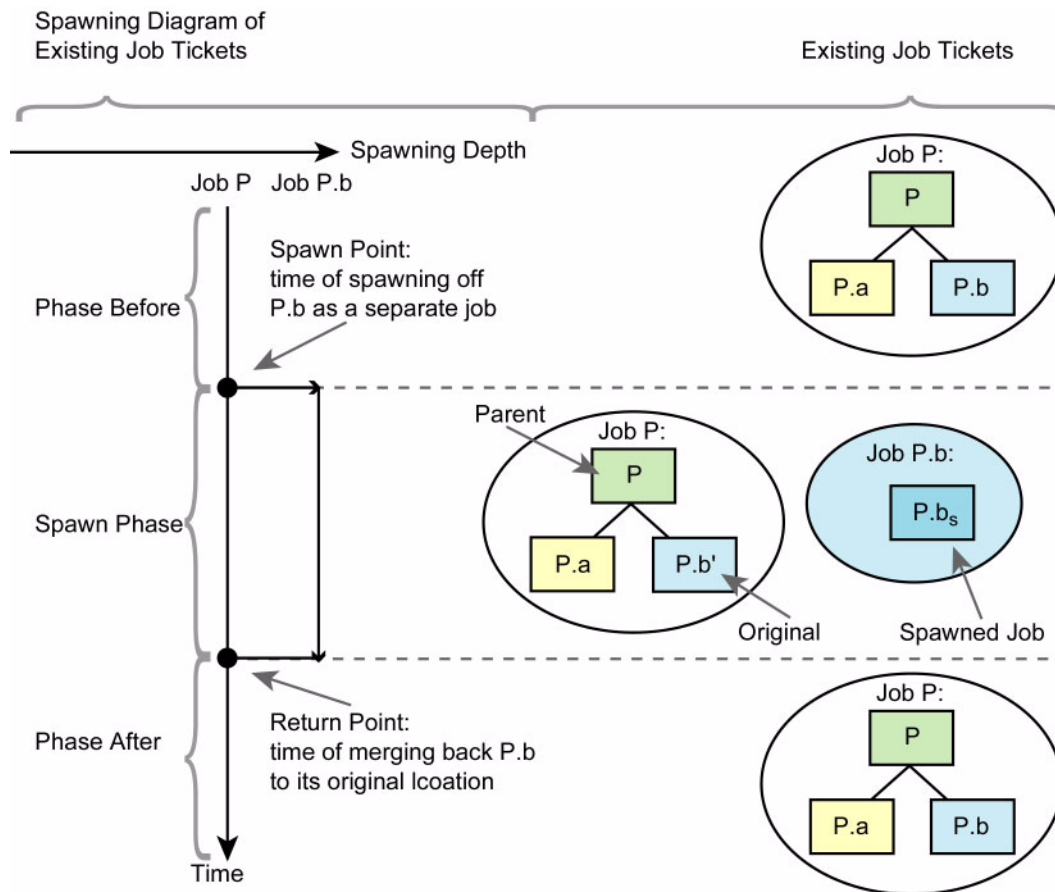


Figure 4-6: The spawning and merging mechanism and its phases

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 4-6, 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 **MUST** 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 **MUST** be set to the value *Spawned* (see Table 3-4, “JDF node,” on page 41). 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 **MUST** 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 **MAY** be copied into the respective *Ancestor* elements. 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>
```

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 **MUST** 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 **MUST** be appended to the *AuditPool* container of the original parent node. If no *AuditPool* container exists in the parent node, one **MUST** be created for the purpose.

Merging

After processing, the spawned JDF node **MUST** be merged back to its original location. Before this can occur, however, duplicate information contained in any elements (such as *CustomerInfo* or *NodeInfo*) **MUST** 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 **MUST** add a *Merged* audit element to the *AuditPool* (see Section 3.10, *AuditPool*). 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 **MUST**

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.

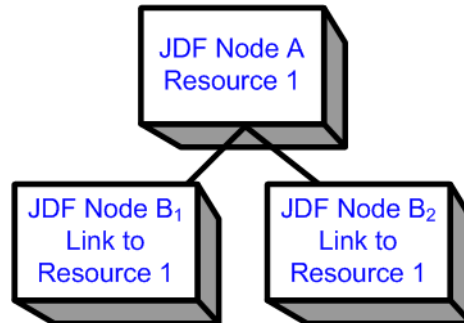


Figure 4-7: JDF node structure that requires resource copying during spawning and merging

When node B_1 is spawned, its resources MUST also be duplicated. To accomplish this, the affected resources MUST 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**, **CustomerInfo** or **NodeInfo** element of the node.

A spawning or merging agent MUST 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 MUST 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 MUST 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 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 MUST also be set *true*. Furthermore, the value of

1. Usually resources become locked (*Locked* = *true*) if they are referenced by **Audit** elements (see also Section 3.10, **AuditPool**).

the *ID* attribute of each copied resource **MUST** 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 **MUST** 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 **MUST** 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 **MUST** 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 **MUST** 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., physical resources where only 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 “*Pool*” and a **StatusPool** is generated with the appropriate information. The **PartStatus** 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 **MUST** 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 **PartStatus** in the **StatusPool** 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** **MUST NOT** modify any elements except for:

- Resources that were spawned with read-write permission, and
- Adding **Audit** elements.

Synchronizing multiple **NodeInfo**, **CustomerInfo** elements, or newly inserted JDF subnodes 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 "Spawning and Merging" on page 774.

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 4-8 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.

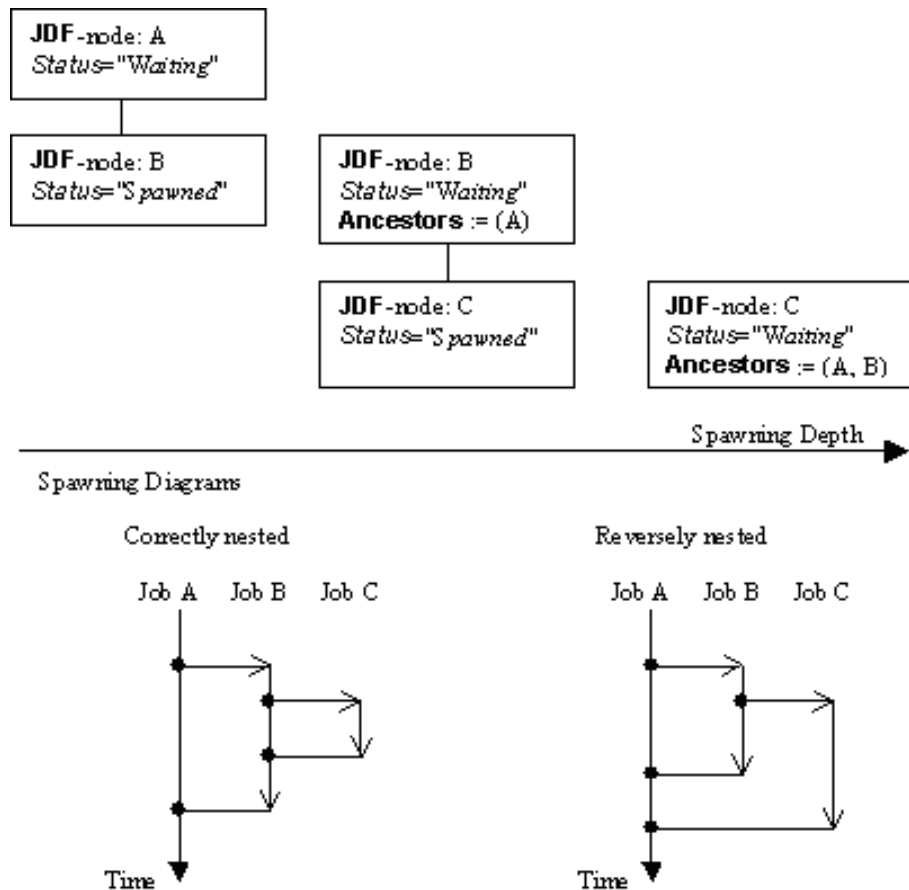


Figure 4-8: Example for a JDF node structure with nested spawning

4.4.5 Case 5: Spawning and Merging of Independent Jobs

Compatibility Warning. Note that Spawning and Merging of Independent Jobs is under development and subject to major changes in a future release of this specification.

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 implementation resources to be performed more efficiently.

Note that production devices will generally require their resources to unambiguously define the production details. Thus a JDF Agent MUST 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 4-9, 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.

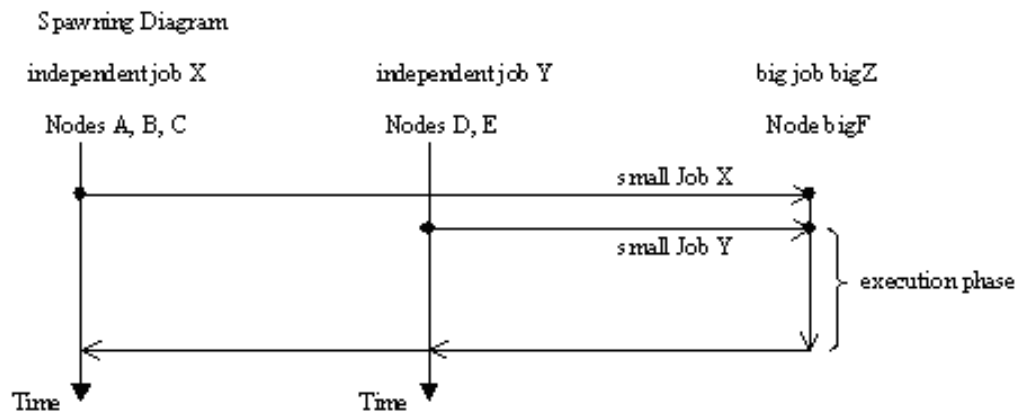
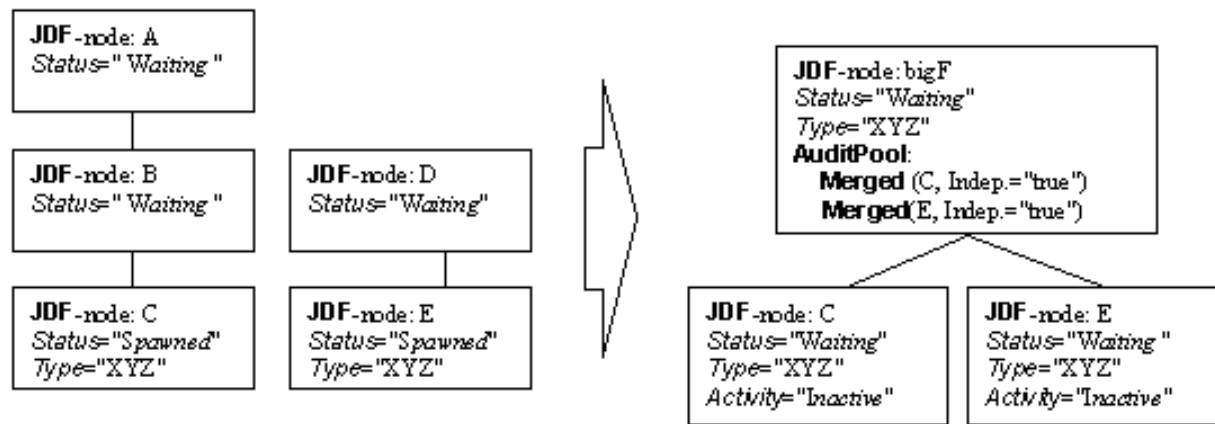


Figure 4-9: Example of the spawning and merging of independent jobs

Spawning

Spawning begins as it did in Case 1 or Case 2. Then, the process to be spawned (job C in Figure 4-9) is copied into a newly created or already existing big job (big job bigZ in Figure 4-9). The process type of the root node of the big job MUST 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 attribute *jRefDestination* specified is added to the parent of the original job. The attribute *jRefDestination* 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 attribute *jRefDestination* in the *Spawned* element.

Where necessary, resource instances MUST 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 MUST 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 attribute *Activation* is restored by setting it to the activation of the big-job node after completion.

Eventually, copied resources **MUST** be purged and handled just as they were in Case 2. Then, the merging **MUST** be logged by appending the **Merged** element to the **AuditPool** container of the parent of the original node. In independent spawning and merging, the attribute *jRefSource* **MUST** be specified in the appropriate **Merged** element.

If the big job is retained, a **Spawned** element with the attribute *Independent* = *true* **MUST** 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.

4.4.6 Case 6: Simultaneous Spawning and Merging of Multiple Nodes

It is not possible to explicitly spawn multiple nodes simultaneously. The nodes **MUST** be grouped into a single *ProcessGroup* 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 **MUST** 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 **MUST** 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 **MUST** 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 pureID, delimited by a period. For example:

- ID of the big job with node *ID* = "A"
- Receives small job A₁ with some IDs: *ID* = "A" *ID* = "A.A" *ID* = "A.B" where the first is the ID of the root node.
- Receives small job A₂ 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 A₁ become: *ID* = "A1.A" *ID* = "A1.A.A" *ID* = "A1.A.B", and the IDs of the small job A₂ 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 A₁ with ID: *ID* = "A1.A.C". Here the agent **MUST** 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 **MUST** 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 Signal messages. The content of the `Notification` element is described in Table 3-32, “Notification audit element,” on page 103. For a list of predefined error codes, see “Supported Error Codes in JMF and Notification elements” on page 717. 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 attribute `Class`. Every workflow implementation MUST associate a class with all events on an event-by-event basis. The following table shows the possible values for `Class`:

Table 4-5: Class attribute values for the Notification audit element

Value of Class	Description
<i>Event</i>	Indicates a pure event which occurred due to a certain operation-related action, (e.g., machine events, operator activities, etc.). This class is used for messaging.
<i>Information</i>	Indicates not an error, but rather any information about a process that cannot be expressed by the other classes, (e.g., the beginning of execution).
<i>Warning</i>	Indicates that a minor error has occurred, and an automatic fix was applied. Execution continues. The node’s <i>Status</i> 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
<i>Error</i>	Indicates that an error has occurred that requires user interaction. Execution cannot continue until the problem has been fixed. The node’s <i>Status</i> is <i>Stopped</i> . This value appears in situations such as when resources are missing, when major incompatibilities are detected, or when the toner is empty.
<i>Fatal</i>	Execution has been aborted. The node’s <i>Status</i> is <i>Aborted</i> . This value is seen with most protocol errors or when major device malfunction has occurred.

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 3.10, `AuditPool`.

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` element 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 Signal message 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 5.3.3, `Persistent Channels` and Section 5.7.1, `Events`. Note that this is different from the `NotificationFilter` elements of the `NodeInfo` element, 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 attribute *Activation* to *TestRun* (see Table 3-4, “JDF node,” on page 41), which is inherited by all descendent nodes that are not inactive (*Activation = Inactive*). If a device or controller¹ 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 MUST 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 3.10, **AuditPool**.

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 Testrun

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*.

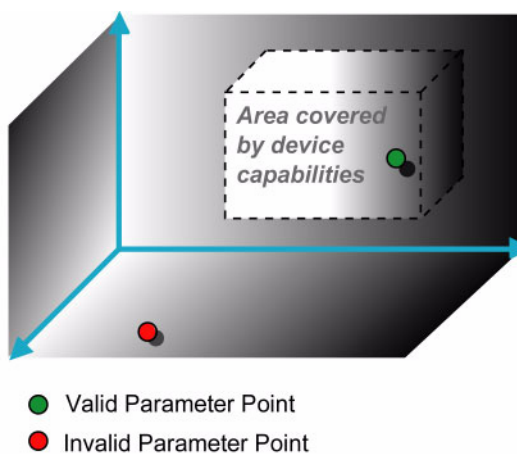


Figure 4-10: Parameter space in device capabilities^a

- a. Note that the restriction to three dimensions is for graphical demonstration purposes only.

1. Note that only devices and controllers with agent capabilities can write in a JDF document.

4.8 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 JDF 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).

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 "Device Capability Definitions" on page 613.

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).

Chapter 5 JDF Messaging with the Job 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

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 MUST support MIME for messages.

There are two types of JMF messaging: bidirectional and unidirectional. Bidirectional JMF messaging uses a Bidirectional protocol — currently HTTP and HTTPS. Unidirectional JMF messaging uses JMF files, placed into a “hot folder” using either a network shared folder or FTP folder to move the file between client and server.

There is a special case of unidirectional JMF messaging: a JDF file to be placed in the input folder of a JDF controller or device. Placing a JDF file rather than a JMF file implies the `SubmitQueueEntry` message and is analogous to placing a JMF file containing the `SubmitQueueEntry` message with a reference to the JDF file.

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 “JDF Packaging” on page 680.

Certain attributes in various JDF and JMF elements exist only to facilitate unidirectional JMF Messaging. To reduce confusion such attributes are marked as *Unidirectional* in the tables through which they are defined. Others exist only for bidirectional JMF Messaging and are marked as *Bidirectional* in the tables through which they are defined.

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. `Message` elements are abstract, as is indicated by the dashed line surrounding the `Message` element in Figure 5-1



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.

Table 5-1: JMF element

Name	Data Type	Description
<i>DeviceID</i> ?	string	Identifies the recipient device or controller. If <i>DeviceID</i> 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 <i>DeviceID</i> that does not match the controller's <i>ControllerID</i> , 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 <i>Response/@ReturnCode</i> = 121, " <i>Unknown DeviceID</i> ". If a Device receives a message with a <i>DeviceID</i> that does not match its own, it SHOULD also respond to the message with <i>Response/@ReturnCode</i> = 121.
<i>ICSVersions</i> ? 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 . For details, see Table 3-4, "JDF node," on page 41
<i>MaxVersion</i> ? 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 3.12, JDF Versioning for a discussion of versioning in JDF.
<i>ResponseURL</i> ? New in JDF 1.2 <i>Unidirectional</i>	URL	URL of the direct response to this JMF. <i>ResponseURL</i> is REQUIRED when using an unidirectional protocol that does not automatically provide a response channel, (e.g., the file protocol). If <i>ResponseURL</i> is specified, a <i>Response</i> MUST be generated and written to <i>ResponseURL</i> , even if no <i>ResponseTypeObj</i> is REQUIRED for the <i>Message</i> . The <i>Response</i> MAY be empty. It MUST NOT be present when a bidirectional protocol is used, (e.g., in HTTP). The URL MUST be an explicit locator. It is up to the sending agent to generate a unique locator for the response. Example: <code>"file://master/JMFResponseFolder/Rip1/r12345.jmf"</code>
<i>SenderID</i>	string	String that identifies the sender device, controller or agent. For a sender Device, the sender's <i>DeviceID</i> . For a sender controller, the sender's <i>ControllerID</i> . <i>SenderID</i> MUST NOT be modified when a JMF is passed through a proxy.
<i>TimeStamp</i>	dateTime	Time stamp that identifies when the message was created.
<i>Version</i> 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" and "1.3". 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 "JDF Versioning" on page 114. Note that <i>Version</i> 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 <i>Version</i> MUST default to "1.1".
<i>xmlns</i> ? New in JDF 1.1	URI	JDF supports use of XML namespaces. The JDF namespace MUST be declared. For details on using namespaces in XML, see [XMLNS].
<i>Message</i> +	element	Abstract message element(s). Note that while a JMF instance MAY include multiple messages, the order of execution of the <i>Message</i> elements within a JMF is not deterministic.

— Element: Message

The following table describes the contents of the abstract `Message` element. All messages contain an `ID` and a `Type` attribute.

Table 5-2: Abstract message element

Name	Data Type	Description
<code>ID</code>	ID	Identifies the message.
<code>Time ?</code>	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 <code>TimeStamp</code> attribute of the JMF element.
<code>Type</code>	NMTOKEN	Name that identifies the message type. Message types are described in Section 5.10 and Section 5.15.
<code>xsi:type ?</code> New in JDF 1.2	NMTOKEN	Informs schema aware validators of the JMF message type definition that the message is to 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 “JDF Nodes” on page 37.

The following figure depicts the basic JMF messaging structure and the message families. Dashed boxes show abstract objects.

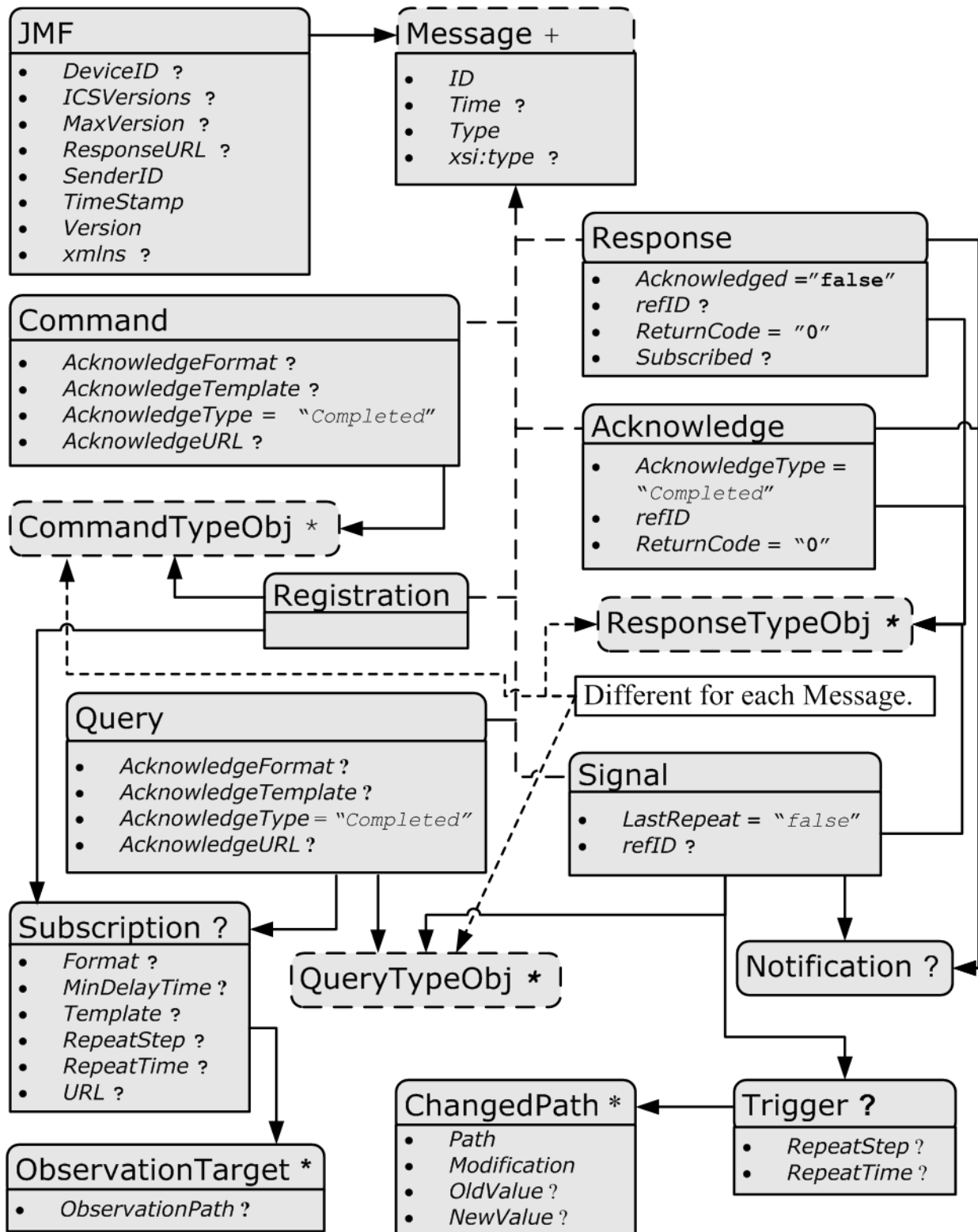


Figure 5-1: JMF root element – a diagram of its structure

5.2 JMF Message Families

A message contains one or more of the following five high level elements, referred to as **message families**, in the root node. These families are Query, Command, Response, Acknowledge and Signal. 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 is sent, a Response is returned. If the Query included a Subscription, Signals are sent to the designated URL until a StopPersistentChannel Command is sent.

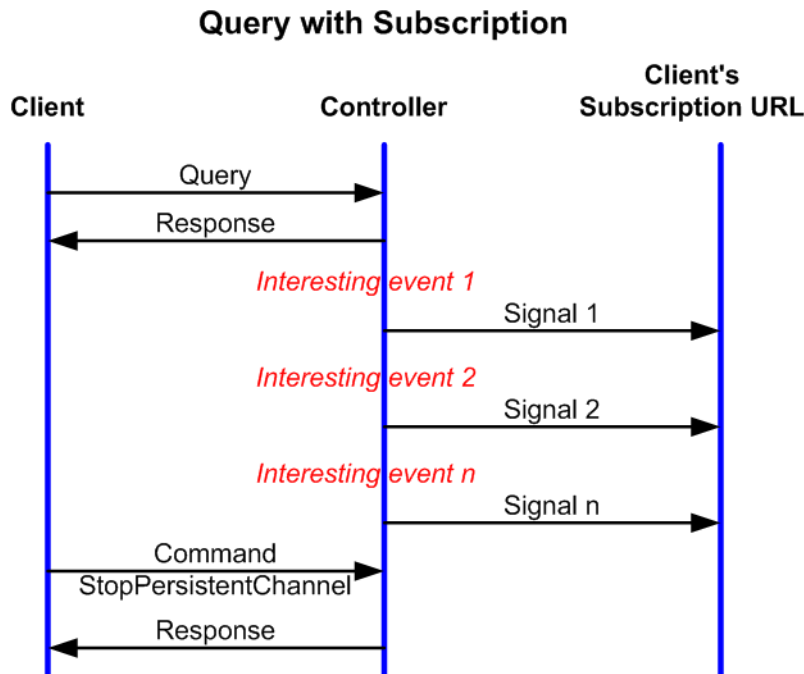


Figure 5-2: Interaction of Messages with a subscription

The Query contains an *ID* attribute and a *Type* attribute, which it inherits from the abstract message type described in Table 5-2, “Abstract message element,” on page 145. JMF supports a number of well defined query types, and each query type can contain additional descriptive elements, which are described in Section 5.10 and Section 5.15. The following table shows the content of a Query message element:

Table 5-3: Query message element

Name	Data Type	Description
<i>AcknowledgeFormat</i> ? New in JDF 1.3 <i>Unidirectional</i>	string	A formatting string used with the <i>AcknowledgeTemplate</i> attribute to define a sequence of generated URLs. See “Generating strings with Format and Template” on page 743. If <i>AcknowledgeFormat</i> is specified, then <i>AcknowledgeTemplate</i> MUST also be specified and <i>AcknowledgeURL</i> MUST NOT be specified.
<i>AcknowledgeTemplate</i> ? New in JDF 1.3 <i>Unidirectional</i>	string	A template, used with <i>AcknowledgeFormat</i> , to define a sequence of generated URLs. See “Generating strings with Format and Template” on page 743. The resulting set of URLs MUST be qualified URLs and not a folder. If <i>AcknowledgeTemplate</i> is specified, then <i>AcknowledgeFormat</i> MUST also be specified and <i>AcknowledgeURL</i> MUST NOT be specified.
<i>AcknowledgeURL</i> ? New in JDF 1.3 <i>Bidirectional</i>	URL	URL of the recipient of any <i>Acknowledge</i> . If specified, the command requests for a <i>Acknowledge</i> message depending on the value of <i>AcknowledgeType</i> . The protocol of the acknowledgement is specified either by the scheme of <i>AcknowledgeURL</i> for bidirectional JMF messaging or through the use of <i>AcknowledgeFormat</i> for unidirectional JMF messaging. If <i>AcknowledgeURL</i> is specified, then both <i>AcknowledgeFormat</i> and <i>AcknowledgeTemplate</i> MUST NOT be specified
<i>AcknowledgeType</i> = “ <i>Completed</i> ” 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. <i>Received</i> – The <i>Query</i> has been received and understood, (e.g., by an operator). <i>Applied</i> – The <i>Query</i> has been applied to the machine, (e.g., by an operator). <i>Completed</i> – The <i>Query</i> has been completely responded to.
<i>QueryTypeObj</i> *	element	Abstract element that is a placeholder for any descriptive elements that provide details for the query. The element type of <i>QueryTypeObj</i> is defined by the <i>Type</i> attribute of the abstract <i>Message</i> element.
<i>Subscription</i> ?	element	If specified creates a persistent channel. For the structure of a <i>Subscription</i> element, see Section 5.3.3, Persistent Channels.

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" Version="1.3">
  <Query ID="M007" Type="KnownDevices"/>
</JMF>
```

5.2.2 Response

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* is returned from a controller to the controller that submitted the *Query* or *Command*; however, *Response*(s) are not acknowledged themselves.

A command response indicates that the command has been received and interpreted. The response of commands with short latency also includes the information about the execution. Commands with long latency MAY additionally generate a separate *Acknowledge* message (see Section 5.2.5, *Acknowledge*) to broadcast the execution of the command. Command responses SHOULD contain a *Notification* element that describes the return status in text if *ReturnCode* is greater than 0. Responses contain an attribute called *refID*, which identifies the initiating query or command. The following table shows the content of a *Response* message.

Table 5-4: Response message element

Name	Data Type	Description
<i>Acknowledged</i> = "false"	boolean	Used only in responses to command messages. Indicates whether the command will be acknowledged separately. If "true", an Acknowledge message will be supplied after command execution. If "false", no Acknowledge message will be supplied.
<i>refID</i> ? Modified in JDF 1.2	NMTOKEN	Copy of the <i>ID</i> attribute of the initiating query or command message to which the response refers. If not specified, the response refers to the entire JMF message, (e.g., if the JMF was not parseable). <i>Response/@Type</i> is set to "Notification" if the <i>Type</i> of the incoming Message is corrupted or unknown.
<i>ReturnCode</i> = "0"	integer	Describes the result. "0" indicates success. For all other possible codes see "Supported Error Codes in JMF and Notification elements" on page 717.
<i>Subscribed</i> ?	boolean	If a Subscription element has been supplied by the corresponding query, this attribute indicates whether the subscription has been refused or accepted. If <i>true</i> , the requested subscription is accepted. If <i>false</i> , the subscription is refused because the controller does not support persistent channels. For details, see Section 5.3.3, Persistent Channels.
<i>Notification</i> ?	element	Additional information including textual description of the return code. The Notification element SHOULD be provided if the <i>ReturnCode</i> is greater than 0, which indicates that an error has occurred. See Section 3.10.1.2, Notification.
<i>ResponseTypeObj</i> *	element	Abstract element that is a placeholder for any descriptive elements that provide details queried for or details about command execution.

An example of a response to a command is provided in the Section 5.2.4, Command. The encoding example for the query, shown above, might generate the following response:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="RIP-1"
  TimeStamp="2000-07-25T11:38:25+02:00" Version="1.3">
  <Response ID="M107" Type="KnownDevices" refID="M007">
    <DeviceList>
      <DeviceInfo DeviceStatus="Unknown">
        <Device DeviceID="Rip1"/>
      </DeviceInfo>
      <DeviceInfo DeviceStatus="Unknown">
        <Device DeviceID="Rip2"/>
      </DeviceInfo>
    </DeviceList>
  </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 transmitted via a message channel that includes a **Subscription** element. The second way is to subscribe for them with an initiating query defined in the **NodeInfo** element of a JDF node that also includes a **Subscription** element (see JMF elements in See "NodeInfo" on page 524.). The first query 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 controllers about their capabilities. Hard-wired signals, however, MUST NOT have a *refID* attribute. If no *refID* is specified, the corresponding query parameters MUST be specified instead.

Table 5-5: Signal message element

Name	Data Type	Description
<i>LastRepeat</i> = "false"	boolean	If <i>true</i> , the persistent channel is being closed by the Device and no further messages will be generated that fulfill the persistent channel criteria. If <i>false</i> , further signals will be sent. For further details, see Section 5.3.3, Persistent Channels.
<i>refID</i> ?	NMTOKEN	Identifies the initiating query message that subscribed this signal message. Hard-wired signals MUST NOT contain a <i>refID</i> attribute.
Notification ?	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 <i>warning</i> , or if pure events have been subscribed. See Section 3.10.1.2, Notification. For details about subscribing pure events see Section 5.7.1, Events.
<u>QueryTypeObj</u> * <u>Modified in JDF 1.2</u>	element	If no <i>refID</i> is specified, the corresponding query parameters MUST be specified instead by providing this 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. The element type of QueryTypeObj is defined by the <i>Type</i> 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 5.3.3, Persistent Channels.

— Element: Trigger

The following table describes the structure of the **Trigger** element.

Table 5-6: Trigger element (Section 1 of 2)

Name	Data Type	Description
<i>RepeatStep</i> ?	integer	Recalls the <i>RepeatStep</i> attribute specified during subscription of the signal. For details see Table 5-11.
<i>RepeatTime</i> ?	double	Recalls the <i>RepeatTime</i> attribute specified during subscription of the signal. For details see Table 5-11.
Added ? <u>Deprecated in JDF 1.2</u>	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 "Signal" on page 816 for details.
ChangedAttribute * <u>Deprecated in JDF 1.2</u>	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 "Signal" on page 816 for details.

Table 5-6: Trigger element (Section 2 of 2)

Name	Data Type	Description
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.
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 "Signal" on page 816 for details.

— Element: 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
<i>Path</i>	XPath	XPath of the element or attribute that was modified.
<i>Modification</i>	enumeration	Specifies the modification that occurred with the object specified in <i>Path</i> . Allowed values are: <i>Create</i> – The object was created. <i>Delete</i> – The object was deleted. <i>Modify</i> – The object was modified.
<i>OldValue ?</i>	string	Old value of the attribute if <i>Path</i> specifies an attribute and <i>Modification</i> <i>!= Create</i> . The string MUST be cast to the appropriate data type that depends on the attribute's data type.
<i>NewValue ?</i>	string	New value of the attribute if <i>Path</i> specifies an attribute and <i>Modification</i> <i>!= Delete</i> . The string MUST be cast to the appropriate data type that depends on the attribute's data type.

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" Version="1.3">
  <Signal ID="s123" Type="Status">
    <StatusQuParams JobID="42" JobPartID="66"/>
    <DeviceInfo DeviceStatus="Setup"/>
  </Signal>
</JMF>
```

5.2.4 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](#) 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](#) with [Acknowledge](#) = "true", and send an [Acknowledge](#) to the [AcknowledgeURL](#) when the [Command](#) completes.

Table 5-8: Command message element

Name	Data Type	Description
<i>AcknowledgeFormat</i> ? New in JDF 1.2 <i>Unidirectional</i>	string	A formatting string used with the <i>AcknowledgeTemplate</i> attribute to define a sequence of generated URLs. See “Generating strings with Format and Template” on page 743. If <i>AcknowledgeFormat</i> is specified, then <i>AcknowledgeTemplate</i> MUST also be specified and <i>AcknowledgeURL</i> MUST NOT be specified.
<i>AcknowledgeTemplate</i> ? New in JDF 1.2 <i>Unidirectional</i>	string	A template, used with <i>AcknowledgeFormat</i> , to define a sequence of generated URLs. See “Generating strings with Format and Template” on page 743. The resulting set of URLs MUST be qualified URLs and not a folder. If <i>AcknowledgeTemplate</i> is specified, then <i>AcknowledgeFormat</i> MUST also be specified and <i>AcknowledgeURL</i> MUST NOT be specified.
<i>AcknowledgeURL</i> ? Modified in JDF 1.2 <i>Bidirectional</i>	URL	URL of the recipient of any <i>Acknowledge</i> . If specified, the command requests for a <i>Acknowledge</i> message depending on the value of <i>AcknowledgeType</i> . The protocol of the acknowledgement is specified either by the scheme of <i>AcknowledgeURL</i> for bidirectional JMF messaging or through the use of <i>AcknowledgeFormat</i> for unidirectional JMF messaging. If <i>AcknowledgeURL</i> is specified, then both <i>AcknowledgeFormat</i> and <i>AcknowledgeTemplate</i> MUST NOT be specified
<i>AcknowledgeType</i> = “ <i>Completed</i> ” 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. <i>Received</i> – The Command has been received and understood, (e.g., by an operator). <i>Applied</i> – The Command has been applied to the machine, (e.g., by an operator). <i>Completed</i> – The Command has been executed.
<i>CommandTypeObj</i> *	element	Abstract element that is a placeholder for any descriptive elements that provide details of the command.

The following example demonstrates how a *ResumeQueueEntry* command 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" Version="1.3">
  <Command ID="M009" Type="ResumeQueueEntry">
    <QueueEntryDef QueueEntryID="job-0032"/>
  </Command>
</JMF>
```

The following example shows a possible response to the command example above:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
TimeStamp="2000-07-25T12:32:48+02:00" Version="1.3">
  <Response ID="M109" Type="ResumeQueueEntry" refID="M009">
    <Queue DeviceID="A3 Printer" Status="Full">
      <QueueEntry JobID="job-0032" QueueEntryID="job-0032" Status="Running"/>
    </Queue>
  </Response>
</JMF>
```

5.2.5 Acknowledge

An *Acknowledge* element is a message that is an asynchronous answer to a *Command* issued by a controller. Each *Acknowledge* message is unidirectional and similar to a *Response*, 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 sender about the results. Acknowledge messages are only generated if the initiating command has specified the attribute *AcknowledgeURL*.

Command with Acknowledge

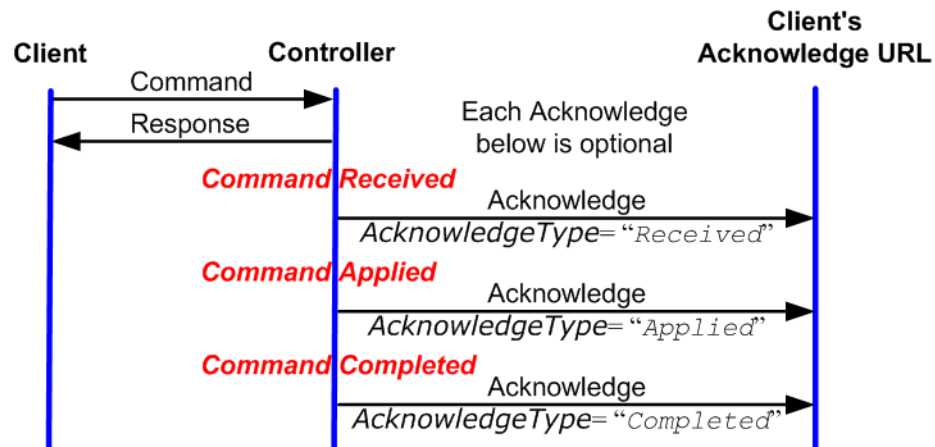


Figure 5-3: Interaction of Command and Acknowledge Messages

They are announced in the Response message to the command by the setting the attribute *Acknowledged* = *true*.

Table 5-9: Acknowledge message element

Name	Data Type	Description
<i>AcknowledgeType</i> = "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. <i>Received</i> – The initiating Command has been received and understood, (e.g., by an operator). <i>Applied</i> – The initiating Command has been applied to the machine, (e.g., by an operator). <i>Completed</i> – The initiating Command has been executed. No further acknowledgement will be sent after an acknowledgement with <i>AcknowledgeType</i> = "Completed" has been sent.
<i>refID</i>	NMTOKEN	Identifies the initiating command message that the Acknowledge refers to.
<i>ReturnCode</i> = "0"	integer	Describes the result. "0" indicates success. For all other possible codes see "Supported Error Codes in JMF and Notification elements" on page 717.
<i>Notification</i> ? Modified in JDF 1.1A	element	Textual description of the command execution. See Section 3.10.1.2, Notification.
<i>ResponseTypeObj</i> *	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.

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" Version="1.3">
  <Acknowledge ID="M109" Type="PipePush" refID="M010">
    <JobPhase JobID="J1" JobPartID="1" Status="InProgress"/>
  </Acknowledge>
</JMF>

```

5.2.6 Registration

[New in JDF 1.3](#)

A Registration is a request to the recipient of the JMF to send Commands to a Command recipient who is specified in Subscription. See Section 5.3.3.2, 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.
Subscription	element	Creates a persistent channel for a Command. For the structure of a Subscription element, see Section 5.3.3, 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 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

JMF Signal and Acknowledge messages are "fire and forget." In case of success, no Response 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 as defined in "Error and Event Messages" on page 156.

5.3.3 Persistent Channels

Queries and Commands are subscribed for using Subscription elements.

5.3.3.1 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). The responding controller SHOULD initially send a Response 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.

5.3.3.2 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 receiver to subsequently send Commands 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 Commands to a press controller whenever a plate or preview has been produced.

5.3.4 Subscription Element

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

Name	Data Type	Description
Format ? New in JDF 1.2 <i>Unidirectional</i>	string	A formatting string used with the <i>Template</i> attribute to define a sequence of generated URLs. See “Generating strings with Format and Template” on page 743. If <i>Format</i> is specified, then <i>Template</i> MUST also be specified and <i>URL</i> MUST NOT be specified
MinDelayTime ? New in JDF 1.3	duration	Minimum delay between two subsequent <i>Signals</i> that are triggered by this <i>Subscription</i> . If not specified a <i>Signal</i> SHOULD be fired when any of the conditions described in <i>Subscription</i> is met. Note that <i>Signals</i> that would be fired before <i>MinDelayTime</i> are lost. <i>MinDelayTime</i> SHOULD NOT be applied to <i>Signals</i> that affect costing.
Template ? New in JDF 1.2 <i>Unidirectional</i>	string	A template, used with <i>Format</i> , to define a sequence of generated URLs. See “Generating strings with Format and Template” on page 743. If <i>Template</i> is specified, then <i>Format</i> MUST also be specified and <i>URL</i> MUST NOT be specified.
RepeatStep ?	integer	Requests an update signal whenever the <i>Amount</i> associated with the query is an integer multiple of <i>RepeatStep</i> . If not specified, it is up to the sending controller to generate signals.
RepeatTime ?	double	Requests an update signal every <i>RepeatTime</i> seconds. If defined, the signal is generated periodically independent of any other trigger conditions.
URL ? Modified in JDF 1.2 <i>Bidirectional</i>	URL	URL of the persistent channel receiving end. The protocol of the subscription is specified by the scheme of <i>URL</i> for bidirectional JMF messaging or <i>Format</i> for unidirectional JMF messaging. If <i>URL</i> is specified, then both <i>Format</i> and <i>Template</i> MUST NOT be specified
ObservationTarget *	element	Requests an updating <i>Signal</i> message whenever the value of one of the attributes specified in <i>ObservationTarget</i> changes.

— Element: 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 <i>Attributes</i> is modified. A value of "*" denotes a message request for any attribute change which is the default. Replaced with <i>ObservationPath</i> 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 <i>ResponseTypeObj</i> of the message. Replaced with <i>ObservationPath</i> 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. Replaced with <i>ObservationPath</i> in JDF 1.2 and above.
ObservationPath ? New in JDF 1.2	XPath	XPath of the elements or attributes that are observed. If not specified, a <i>Signal</i> is emitted on any change in the abstract <i>ResponseType</i> of the message.

If a persistent signal channel has been set up and the device knows that this is the last time that the condition for signaling will be *true*, it SHOULD set the *LastRepeat* flag of the corresponding *Signal* message to *true*. In general, this will happen for a *Status* query, as when the job that has been tracked is completed. It can also happen when

a device is shut down and will, therefore, not send any further updates. If a controller that does not support persistent channels is queried to set up a persistent channel, it **MUST** answer the query with a **Response**, 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.

5.3.5 Creating Persistent Channels in a JDF Node

The **NodeInfo** element of a JDF node **MAY** contain JMF elements that contains a set of **queries** (not **commands**) that define persistent channels. Parsing a JDF instance that contains JMF with a **Subscription** element is equivalent to receiving the messages that are specified in the JMF node. If the parsing controller cannot handle the request, it generates a **Response** with *ReturnCode* = “111” and *Subscribed* = “*false*”, accompanied by a **Notification** element describing the rejection. It is **OPTIONAL** to emit the **Response**, (e.g., if the agent parses a **Resource** request but has no access to the device information).

5.3.6 Deleting Persistent Channels

A persistent channel is to be deleted by sending a **StopPersistentChannel** command, as described in Section 5.7.7, **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 can be defined in a JDF node or hard-wired. In order to set up notification messages via a **NodeInfo** element, the controller **MUST** be able to read JMF query elements from a JDF document.
- **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. Queries require bidirectional communication capabilities.
- **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 **MUST** support MIME Multipart/Related documents. For more details on submission, see Section 5.12.8, **SubmissionMethods**.

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 a command or a query message is not successfully handled, a processor **MUST** reply with a standardized response that **MAY** contain a **Notification** element. **Notification** elements, described in detail in Section 3.10.1.2,



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.

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 `Response` and `Acknowledge` 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 "Supported Error Codes in JMF and Notification elements" on page 717. The responding application SHOULD fill additional `Notification/Error` elements that describe the details of the error.

The following example uses a `Notification` element to describe an error:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
  Timestamp="2005-03-25T12:32:48+02:00" Version="1.3">
  <Response ID="M109" ReturnCode="5" Type="ResumeQueueEntry" 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="false">
        <ErrorData Path="/JMF/Command" ErrorType="Unsupported"/>
      </Error>
    </Notification>
  </Response>
</JMF>
```

5.5.1 Pure Event 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 pure events can be subscribed to by the `Events` message described in Section 5.7.1, `Events`. These `Signals` always have a `Type = "Notification"`:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
  Timestamp="2005-07-25T12:32:48+02:00" Version="1.3">
  <Signal ID="S1" Type="Notification">
    <Notification Class="Event" Timestamp="2000-07-25T12:32:48+02:00" Type="Barcode">
      <Comment>Palette completed</Comment>
      <Barcode Code="99923AAA123"/>
    </Notification>
  </Signal>
</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 or query described in this list. It is, however, possible to discover which messages are supported in a workflow. A controller responds to the `KnownMessages` query by publishing a list of all the messages it supports (see [Section 5.7.3, KnownDevices](#), 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.)

- Q: Query
- C: Command and Registration
- 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** and **Command** messages cause a **Response** message to be returned. If the message can occur as a **Signal**, 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 or command 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 or Command , 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 or Acknowledge to a Query or Command followed by cardinality symbol.	Short description of the subelement(s) if applicable.

The name of the abstract subelement of a **Query** element is `QueryTypeObj`, the name of the abstract subelement of a **Command** element is `CommandTypeObj`, and the name of the abstract subelement of a **Response** as well as an **Acknowledge** element is `ResponseTypeObj`.

5.7 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 5-14: Messages for events and capabilities

Message type	Family	Description
Events	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	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.
RepeatMessages	QR	Returns a set of previously sent messages that have been stored by the controller.
StopPersistentChannel	CR	Closes a persistent channel.

5.7.1 Events

The **Events** message type is intended to be used to query for supported **Signal** messages and to subscribe for asynchronous, randomly occurring **Signals** 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 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 "Pure Event Messages" on page 157, which are supported by a device or controller. If all signals are requested, a **NotificationFilter** with *SignalTypes* = "all" MUST be included in the query.

Table 5-15: Events message

Object Type	Element Name	Description
QueryTypeObj	NotificationFilter ?	Refines the list of events queried.
ResponseTypeObj	NotificationDef *	List of Notification types that match NotificationFilter.

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 Signals, 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 3.10.1.2, Notification. Possible NotificationDetails elements are defined in "NotificationDetails" on page 709. Example of a subscription of all Events and the response, including the feature of subscribing for all messages by setting NotificationFilter/@SignalTypes = "All":

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
  TimeStamp="2005-07-25T12:32:48+02:00" Version="1.3">
  <Query ID="M170" Type="Events">
    <Subscription URL="http://www.anycompany.com/MIS/JMF/JobTracker"/>
    <NotificationFilter Classes="Event Warning Error Fatal" SignalTypes="All"/>
  </Query>
</JMF>
```

The Response to the previous Query:

```
<JMF SenderID="A3 Printer" TimeStamp="2005-07-25T12:32:48+02:00"
  xmlns="http://www.CIP4.org/JDFSchema_1_1" Version="1.3">
  <Response ID="M1001" refID="M170" Type="Events"
    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>
</JMF>
```

— Element: NotificationFilter

Table 5-16: NotificationFilter element (Section 1 of 2)

Name	Data Type	Description
<i>Classes</i> ?	enumerations	<p>Defines the set of Notification/@Class to be queried/subscribed for. Possible values are:</p> <p><i>Event</i></p> <p><i>Information</i></p> <p><i>Warning</i></p> <p><i>Error</i></p> <p><i>Fatal</i></p> <p>If not specified, all notification classes are subscribed to. If the values both <i>Classes</i> and <i>Types</i> are lists of values, the NotificationFilter defines an OR of all combinations.</p>

Table 5-16: NotificationFilter element (Section 2 of 2)

Name	Data Type	Description
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. Use JMF/@ <i>DeviceID</i> in JDF 1.3 and beyond.
JobID ?	string	JobID of the job whose messages are queried/subscribed.
JobPartID ?	string	JobPartID of the job whose messages are queried/subscribed.
QueueEntryID ? New in JDF 1.2	string	<i>QueueEntryID</i> of the job whose messages are queried/subscribed. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> and <i>Part</i> are ignored. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> or <i>QueueEntryID</i> are specified, <i>NotificationFilter</i> applies to all jobs.
SignalTypes = "Notification" New in JDF 1.2	NMTOKENS	Possible <i>Signal/@Type</i> values of the subscribed messages. The special token "all" specifies that all <i>Signals</i> , regardless of <i>Type</i> are queried/subscribed.
Types ?	NMTOKENS	Possible <i>Notification/@Type</i> names are defined in "NotificationDetails" on page 709. Matching notification types are returned/subscribed. Defaults to all supported notification types.
Part * New in JDF 1.2	element	<i>Part</i> elements that describe the partition of the job whose messages are queried/subscribed. For details on job partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 126.

— Element: NotificationDef**Table 5-17: NotificationDef element**

Name	Data Type	Description
Classes ? Modified in JDF 1.2	enumerations	<i>Notification/@Class</i> of the <i>Notification</i> in a <i>Signal</i> . Possible values are: <i>Event</i> <i>Information</i> <i>Warning</i> <i>Error</i> <i>Fatal</i> <i>Classes</i> MUST NOT be specified unless <i>SignalType</i> = "Notification". For details, see Section 4.6.1, Classification of Notifications.
SignalType = "Notification" New in JDF 1.2	NMTOKEN	<i>Signal/@Type</i> value of the subscribed message.
Type ? Modified in JDF 1.2	NMTOKEN	Notification type, that is the name of the element derived from the abstract <i>NotificationDetails</i> element. <i>Type</i> MUST NOT be specified unless <i>SignalType</i> = "Notification". For a list of predefined names see "NotificationDetails" on page 709.

5.7.2 KnownControllers

Table 5-18: KnownControllers message

Object Type	Element Name	Description
QueryTypeObj	-	-
ResponseTypeObj	JDFController *	Known controllers.

The KnownControllers query requests information about the Controllers that are known to the Controller that is queried and can be directly accessed by JMF messaging. KnownControllers is designed to define a registration server. A processor that needs information about its system environment can query a registration server for a list of known controllers. A single controller 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 MUST be defined using a method outside of JDF.

— Element: JDFController

Table 5-19: JDFController element

Name	Data Type	Description
<u>ControllerID ?</u> <u>New in JDF 1.2</u>	string	String that identifies the Controller or Agent. The <i>ControllerID</i> is used as the <i>SenderID</i> of JMF messages that are produced by this Controller. A JMF message that is intended for a specific controller SHOULD specify the controller's <i>ControllerID</i> value in JMF/@ <i>DeviceID</i> .
<i>URL</i>	URL	URL of the controller. If the URL scheme is " <i>file:</i> ", <i>URL</i> MUST specify the directory where the JMF messages are to be deposited.

The following is an example of a response to a KnownControllers query:

```
<Response ID="M1" Type="KnownControllers" refID="Q1">
  <JDFController DescriptiveName="Printer Controller" URL="http://www.anycompany.com/
controller"/>
</Response>
```

5.7.3 KnownDevices

Table 5-20: 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 is to return a list of all known devices.
ResponseTypeObj <u>Modified in JDF 1.1A</u>	DeviceList ?	The list of known devices. ^a

- a. This was Device* prior to version 1.1 a. It was changed due to inconsistencies of the inheritance model in the JDF schema.

The KnownDevices query 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 list of DeviceInfo elements controlled by the controller that receives the query, as demonstrated in the following example:

```
<Response ID="M1" refID="Q1" Type="KnownDevices">
  <DeviceList>
    <DeviceInfo DeviceStatus="Unknown">
      <Device DeviceID="Joe SpeedMaster" DeviceType="Heidelberg SM102/6 rev. 47"/>
    </DeviceInfo>
  </DeviceList>
</Response>
```

— Element: 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 5-21: DeviceFilter element

Name	Data Type	Description
<i>DeviceDetails</i> = "None" New in JDF 1.1	enumeration	Refines the level of provided information about the device. Possible values are: <i>None</i> <i>Brief</i> – Provide all available device information except for Device elements. <i>Modules</i> – ModuleStatus elements are to be provided without module specific status details and without module specific employee information. <i>Details</i> – Provide maximum available device information excluding device capability descriptions. Includes Device elements which represent details of the device. <i>NamedFeature</i> – 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. <i>Capability</i> – Provide Device/DeviceCap subelements which represent details of the capabilities of the device. <i>Full</i> – Provide maximum available device information including device capability descriptions. Includes Device elements which represent details of the device.
<i>Localization</i> ? New in JDF 1.2	languages or "all"	If present, <i>Localization</i> 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 Devices are returned. As this is a filter, only information that can be used to identify a device MUST be specified. This precludes use of DeviceCap and IconList in this Device . The data type of Device is ResourceElement . See "ResourceElement – Subelement of a Resource" on page 74.

— Element: DeviceList

The **DeviceList** element contains a list of information about devices that are returned.

[New in JDF 1.1A](#)

Table 5-22: 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 5-56, "DeviceInfo element," on page 184 in the message description Section 5.8.9, Status.

5.7.4 KnownJDFServices

[Deprecated in JDF 1.2](#)

In JDF 1.2 and beyond, KnownJDFServices has been replaced with KnownDevices and *DeviceDetails* = "*Capabilities*". See "KnownJDFServices" on page 820 for the details of this deprecated message.

5.7.5 KnownMessages

Table 5-23: 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.

The KnownMessages query returns a list of all message types that are supported by the controller.

— Element: KnownMsgQuParams

The flags of the KnownMsgQuParams element specify the message families to include in the response list. Multiple flags are allowed.

Table 5-24: KnownMsgQuParams element

Name	Data Type	Description
<i>Exact</i> = "false" New in JDF 1.1	boolean	Requests an exact description of the known messages. If <i>true</i> , the response also contains the requested <i>DevCaps</i> of the messages.
<i>ListCommands</i> = "true"	boolean	Lists all supported Command types.
<i>ListQueries</i> = "true"	boolean	Lists all supported Query types.
<i>ListRegistrations</i> = "true" New in JDF 1.3	boolean	Lists all supported Registration types.
<i>ListSignals</i> = "true"	boolean	Lists all supported Signal types.
<i>Persistent</i> = "false"	boolean	If <i>true</i> , only lists messages that can use persistent channels. If <i>false</i> , ignores the ability to use persistent channels.

— Element: MessageService

The response is a list of MessageService elements, one for each supported message type. The flags of the MessageService response 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*. Multiple flags are allowed.

Table 5-25: MessageService element (Section 1 of 2)

Name	Data Type	Description
<i>Acknowledge</i> = "false" New in JDF 1.1	boolean	If <i>true</i> , the device supports asynchronous Acknowledge answers to this message.
<i>Command</i> = "false"	boolean	If <i>true</i> the message is supported as a Command.

Table 5-25: MessageService element (Section 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 <i>State</i> elements (see the following Section 7.3.7, <i>State</i>) overwrite the description in <i>GenericAttributes</i> , which MUST NOT be specified if <i>KnownMsgQuParams/@Exact = "false"</i>
JMFRole ? New in JDF 1.3	enumeration	The role of the Device that responds with the <i>MessageService</i> . One of: <i>Receiver</i> – The Device that responds to <i>KnownMessages</i> responds to the message specified in <i>Type</i> . This <i>MessageService</i> specifies <i>Query</i> , <i>Signal</i> , <i>Command</i> and <i>Registration</i> messages that the Device understands. <i>Sender</i> – The Device that responds to <i>KnownMessages</i> is the originator of the message specified in <i>Type</i> . This <i>MessageService</i> specifies <i>Responses</i> and <i>Acknowledges</i> that the Device understands as a <i>Response</i> to these messages. – I understand these responses to messages that I send.
<i>Persistent = "false"</i>	boolean	If <i>true</i> the message is supported as a persistent channel.
<i>Query = "false"</i>	boolean	If <i>true</i> the message is supported as a <i>Query</i> .
<i>Registration = "false"</i> New in JDF 1.3	boolean	If <i>true</i> the message is supported as a <i>Registration</i> .
<i>Signal = "false"</i>	boolean	If <i>true</i> the message is supported as a <i>Signal</i> .
<i>Type</i>	NMTOKEN	Type of the supported message. Extension types are specified by stating the namespace prefix in <i>Type</i> .
URLSchemes ? New in JDF 1.3	NMTOKENS	List of schemes supported for the Message defined by this <i>MessageService</i> . Values include: <i>file</i> – The file scheme according to [RFC1738] and [RFC3986]. <i>http</i> – HTTP (Hypertext Transport Protocol) <i>https</i> – HTTPS (Hypertext Transport Protocol – Secure)
ActionPool ? New in JDF 1.3	element	Container for zero or more <i>Action</i> elements for use as constraints. For details on <i>Action</i> elements, see "ActionPool" on page 617. <i>ActionPool</i> MUST NOT be specified if <i>KnownMsgQuParams/@Exact = "false"</i>
DevCapPool ? New in JDF 1.3	element	Pool of <i>DevCap</i> elements that can be referenced from multiple elements within the <i>DeviceCap</i> structure. <i>DevCapPool</i> MUST NOT be specified if <i>KnownMsgQuParams/@Exact = "false"</i>
DevCaps * New in JDF 1.1	element	Specifies the restrictions of the parameter space of the supported messages. For details on using <i>DevCaps</i> , see Section 7.3.5, <i>DevCaps</i> . <i>DevCaps</i> MUST NOT be specified if <i>KnownMsgQuParams/@Exact = "false"</i> .
ModulePool ? New in JDF 1.3	element	Pool of <i>ModuleCap</i> elements that specify the availability of a given <i>Module</i> . See Table 7-368, "ModuleCap element," on page 618 for details of <i>ModuleCap</i> . <i>ModulePool</i> MUST NOT be specified if <i>KnownMsgQuParams/@Exact = "false"</i>
TestPool ? New in JDF 1.3	element	Container for zero or more <i>Test</i> elements that are referenced from <i>ActionPool/Action</i> elements. <i>TestPool</i> MUST NOT be specified if <i>KnownMsgQuParams/@Exact = "false"</i>

The following is an example of a response to a *KnownMessages* query:

```
<Response ID="M1" Type="KnownMessages" refID="Q1">
```

```

<MessageService JMFRole="Receiver" Query="true" Type="KnownMessages"/>
<MessageService JMFRole="Receiver" Persistent="true" Query="true" Signal="true"
Type="Status"/>
</Response>

```

5.7.6 RepeatMessages

Table 5-26: RepeatMessages message

Object Type	Element Name	Description
QueryTypeObj	MsgFilter ?	A filter for the messages to be repeated. For details, see Section 5.7.1, Events.
ResponseTypeObj	Message *	The recent messages queried.

The RepeatMessages query 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.

— Element: MsgFilter

Table 5-27: MsgFilter element (Section 1 of 2)

Name	Data Type	Description
<i>After</i> ?	dateTime	Messages sent only after a certain time.
<i>Before</i> ?	dateTime	Messages sent only before a certain time.
<i>Count</i> ?	integer	Maximum number of messages, most recent first.
<i>DeviceID</i> ?	string	ID of the Device whose messages are requested.
<i>Family</i> ? Modified in JDF 1.3	enumeration	Filter for Message family. Possible values are: <i>Acknowledge</i> <i>Command</i> – Repeat Commands that are triggered by a Registration. New in JDF 1.3 <i>Response</i> <i>Signal</i> <i>All</i> – <i>Response</i> , <i>Signal</i> and <i>Acknowledge</i> messages are queried. Deprecated in JDF 1.2.
<i>JobID</i> ? New in JDF 1.2	string	<i>JobID</i> of the job whose messages are queried/subscribed.
<i>JobPartID</i> ? New in JDF 1.2	string	<i>JobPartID</i> of the job whose messages are queried/subscribed.
<i>MessageRefID</i> ?	NMTOKEN	The <i>refID</i> attribute MUST match the value of <i>MessageRefID</i> .
<i>MessageID</i> ?	NMTOKEN	The <i>ID</i> attribute MUST match the value of <i>MessageID</i> .
<i>MessageType</i> ?	NMTOKEN	<i>Type</i> attribute of the requested messages.
<i>QueueEntryID</i> ? New in JDF 1.2	string	<i>QueueEntryID</i> of the job whose messages are queried/subscribed. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> and <i>Part</i> are ignored. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> or <i>QueueEntryID</i> are specified, MsgFilter applies to all jobs that will be processed by the receiver.
<i>ReceiverURL</i> ?	URL	URL for which the messages are intended.

Table 5-27: MsgFilter element (Section 2 of 2)

Name	Data Type	Description
Part * New in JDF 1.2	element	Part of the job whose messages are queried/subscribed. For details on Node partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 126.

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"`].

The following is an example of a response to a `RepeatMessages` query. Note the nesting of `Response` messages, where the first layer is the response to the `RepeatMessages` query and its contents are the repeated messages.

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
  Timestamp="2005-07-25T12:32:48+02:00" Version="1.3">
  <Response ID="RepMsg" Type="RepeatMessages">
    <Response ID="R1" Time="2000-06-14T11:00:01+02:00" Type="Status"/>
    <Response ID="R2" Time="2000-06-14T10:50:22+02:00" Type="Occupation"/>
    <Signal ID="R3" Time="2000-06-14T08:20:23+02:00" Type="Resource"/>
    <Signal ID="R4" Time="2000-06-14T03:01:22+02:00" Type="Notification"/>
  </Response>
</JMF>
```

5.7.7 StopPersistentChannel

Table 5-28: StopPersistentChannel message

Object Type	Element Name	Description
CommandTypeObj	StopPersChParams	Specifies the persistent channel and the message types to be unsubscribed.
ResponseTypeObj	—	—

The `StopPersistentChannel` command unregisters a listening controller from a persistent channel. No more messages are sent to the controller once the command has been issued. A certain subset of signals can be addressed for unsubscription by specifying a `StopPersChParams` element.

— Element: StopPersChParams

If the OPTIONAL attributes are not specified, those attributes default to match anything. Therefore, it is possible to cancel the persistent channel for messages belonging to a certain type of message or to a certain job.

Table 5-29: StopPersChParams element (Section 1 of 2)

Name	Data Type	Description
<i>ChannelID</i> ?	NMTOKEN	<i>ChannelID</i> of the persistent channel to be deleted. If the channel has been created with a <code>Query</code> message, the <i>ChannelID</i> specifies the <i>ID</i> of the <code>Query</code> message (identical to the <i>refID</i> of the <code>Response</code> message).
<i>MessageType</i> ?	NMTOKEN	Only messages with a matching message type are suppressed. Message types are specified in the <i>Type</i> attribute of each <code>Message</code> element. Defaults to all message types.
<i>DeviceID</i> ?	string	Only messages from devices or controllers with a matching <i>DeviceID</i> attribute are suppressed.
<i>JobID</i> ?	string	Only messages with a matching <i>JobID</i> attribute are suppressed.
<i>JobPartID</i> ?	string	Only messages with a matching <i>JobPartID</i> attribute are suppressed.

Table 5-29: StopPersChParams element (Section 2 of 2)

Name	Data Type	Description
QueueEntryID ? New in JDF 1.2	string	<i>QueueEntryID</i> of the job whose messages are queried/subscribed. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> and <i>Part</i> are ignored. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> or <i>QueueEntryID</i> are specified, <i>StopPersChParams</i> applies to all jobs that will be processed by the receiver.
<i>URL</i>	URL	URL of the receiving controller. This MUST be identical to the URL that was used to create the persistent channel. If no <i>ChannelID</i> is specified, all persistent channels to this URL are deleted.
Part * New in JDF 1.2	element	<i>Part</i> elements that describe the partition of the job whose messages are suppressed. For details on Node partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 126.

5.8 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 7.2.58, *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 5-30: Messages to query/affect a job, device or controller

Message type	Family	Description
FlushResources New in JDF 1.2	CRS	Remove temporary resource from a Device.
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).
<i>Occupation</i>	QRS	Queries the occupation of an employee.
<i>Resource</i>	CQRS	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	CR	Creates a new <i>QueueEntry</i> from an already existing <i>QueueEntry</i> and submits it to the queue in order to be executed.
ShutDown New in JDF 1.2	CRS	Shuts down a device.
<i>Status</i>	QRS	Queries the general status of a device, controller or job.
<i>Track</i>	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	CRS	Wakes up a device that is in standby mode.

5.8.1 FlushResources

[New in JDF 1.2](#)

Table 5-31: FlushResources message

Object Type	Element Name	Description
CommandTypeObj	QueueFilter ?	Defines a filter for the returned <code>Queue</code> element in the <code>FlushResources</code> message.
	FlushResourceParams?	Defines the resources to be removed.
ResponseTypeObj	FlushedResources ?	This element is a placeholder for future use.
For the definition of the <code>Queue</code> element, see "Elements for Queues" on page 211.		

`FlushResources` is used to remove temporary resources from a Device. `FlushResourceParams` allows the specification of which resources to remove. The `QueueFilter` in the `FlushQueue` message is applied to the `Queue` returned after the command is executed. The `QueueFilter` contained within the `FlushResourceParams` is used to specify `QueueEntry` elements to which the resources to be removed belong.

— Element: FlushResourceParams

Table 5-32: FlushResourceParams element

Name	Data Type	Description
QueueFilter ?	element	Defines a <code>QueueFilter</code> that specifies the <code>QueueEntry</code> elements to which the resources to be removed belong. If not specified, all temporary re-sources on the device are completely flushed.
<i>FlushPolicy</i> = "QueueEntry"	enumeration	Policy that defines how much of the <code>QueueEntry</code> resources is requested to be flushed. One of: <i>Complete</i> – Remove the entire temporary resources belonging to the <code>QueueEntry</code> . <i>QueueEntry</i> – The resources belonging to <code>QueueEntry</code> are completely re-moved and no longer available — the default. <i>Intermediate</i> – Remove any intermediate resources that belong to the <code>QueueEntry</code> (e.g., intermediate raster files in a combined RIP and Image-Setting process) and retain the original input resources. A <code>ResourcePull</code> message is possible.

— Element: FlushedResources

Table 5-33: FlushedResources element

Name	Data Type	Description

5.8.2 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.

Table 5-34: ModifyNode message

Object Type	Element Name	Description
CommandTypeObj	ModifyNodeCmdParams ?	Defines the details of the <code>ModifyNode</code> message.
ResponseTypeObj	-	-

— Element: ModifyNodeCmdParams

The ModifyNodeCmdParams specifies the **JDF** node to be modified.

Table 5-35: ModifyNodeCmdParams element

Name	Data Type	Description
<i>Activation</i> ?	enumeration	The new value for <i>Activation</i> . For a list of allowed values, see Table 3-4, “JDF node,” on page 41.
<i>JobID</i>	string	<i>JobID</i> of the node to be modified. In case of adding a Comment to a Resource or Audit , this <i>JobID</i> MUST be an attribute of the node where the AuditPool or ResourcePool resides.
<i>JobPartID</i>	string	<i>JobPartID</i> of the node to be modified. In the case of adding a Comment to a Resource or Audit , this <i>JobPartID</i> MUST be an attribute of the node where the AuditPool or ResourcePool resides.
<i>CommentURL</i> ?	URL	The new value for <i>CommentURL</i> . Note that <i>CommentURL</i> is specified in Table 3-1, “Any element (generic content),” on page 37 and that the semantics are overridden by the definition in this table.
<i>NewComment</i> *	element	Details of modifications of Comment elements.

— Element: NewComment

Table 5-36: NewComment element

Name	Data Type	Description
<i>Action</i>	enumeration	<i>Add</i> – A new Comment is added. If <i>refID</i> is specified, the Comment is stored in the Resource or Audit with <i>@ID = refID</i> . <i>Concat</i> – Comment is concatenated to the Comment with <i>Comment/@ID = CommentID</i> . <i>Replace</i> – Comment replaces the Comment with <i>Comment/@ID = CommentID</i> . <i>Remove</i> – The Comment with <i>Comment/@ID = CommentID</i> is removed.
<i>CommentID</i> ?	NMTOKEN	<i>ID</i> of the existing Comment . MUST be specified if <i>Action = Concat, Replace</i> or <i>Remove</i> .
<i>refID</i> ?	NMTOKEN	<i>ID</i> of the Resource or Audit where the Comment MUST be added. The <i>refID</i> MUST NOT be set unless <i>Action = "Add"</i> .
<i>Comment</i> ?	element	The Comment to <i>Add, Concat</i> or <i>Replace</i> . Comment MUST NOT be specified if <i>Action = "Remove"</i> . Note that Comment * is specified in Table 3-1, “Any element (generic content),” on page 37 and that the cardinality and semantics are overridden by the definition in this table.

5.8.3 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 or signal informs a Controller or MIS system that a JDF node has been created. A command initiates a modification.

5.8.3.1 NewJDF Query

Table 5-37: 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.

The NewJDF query 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 ProcessGroup node with new subnodes. NewJDF is made selective by specifying a NewJDFQuParams element.

The query response returns a list of IDInfo elements that contains the queried information concerning the newly created Nodes.

— Element: NewJDFQuParams

Table 5-38: NewJDFQuParams element

Name	Data Type	Description
<i>JobID</i> ?	string	Job ID of the JDF node that is being queried.
<i>JobPartID</i> ?	string	Job part ID of the JDF node that is being queried.
<i>QueueEntryID</i> ?	string	<i>QueueEntryID</i> of the job that is currently being executed. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> and <i>Part</i> are ignored.

The NewJDF command 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 does not imply any job submission or request for job submission. Job queue submission MUST still be requested with a RequestQueueEntry message, and the MIS MUST still subsequently submit the job to the requesting Controller or Device.

5.8.3.2 NewJDF Command

Table 5-39: 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.

— Element: NewJDFCmdParams

Table 5-40: NewJDFCmdParams element

Name	Data Type	Description
<i>JDFDetails</i> = " <i>Brief</i> "	string	Level of detail requested for the returned IDInfo elements. Values are: <i>None</i> : Do not return any IDInfo elements. <i>Brief</i> : Return IDInfo elements without embedded JDF or Device. <i>Full</i> : Return IDInfo elements with embedded JDF and Device.
IDInfo	element	Details of the new JDF node that is to be created.

— Element: IDInfo

Table 5-41: IDInfo element

Name	Data Type	Description
<i>Category</i> ?	NMTOKEN	JDF/@Category of the JDF node.
<i>JobID</i> ?	string	Job ID of the JDF node.
<i>JobPartID</i> ?	string	Job part ID of the JDF node.
<i>ParentJobID</i> ?	string	<i>JobID</i> of the parent node of the JDF node. If not specified, it defaults to the value of <i>JobID</i> .
<i>ParentJobPartID</i> ?	string	Job part ID of the parent node of the JDF node.
<i>Type</i> ?	NMTOKEN	JDF/@Type of the JDF node.
<i>Types</i> ?	NMTOKENS	JDF/@Types of the JDF node.
Device ?	element	Description of the Device that the JDF is targeted for. The data type of Device is ResourceElement. See “ResourceElement – Subelement of a Resource” on page 74.
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.

5.8.4 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 "NodeInfo" on page 817.

5.8.5 Occupation

Table 5-42: Occupation message

Object Type	Element Name	Description
QueryTypeObj	EmployeeDef *	Defines the employees queried.
ResponseTypeObj	Occupation *	The occupation status of the employees.

Occupation queries the occupation status of an employee. No job context is needed to issue an Occupation message.

— Element: EmployeeDef

The Occupation query 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 5-43: EmployeeDef element

Name	Data Type	Description
<i>PersonalID</i> ?	string	<i>PersonalID</i> of the employee being tracked.

— Element: 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 5-44: Occupation element

Name	Data Type	Description
<i>Busy</i> = "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 <i>Busy</i> values of one employee SHOULD NOT exceed 100.
<i>JobID</i> ?	string	<i>JobID</i> of the JDF node that the employee is assigned to. If no <i>JobID</i> is specified but devices are, the employee is performing tasks not related to a job.
<i>JobPartID</i> ?	string	Job part ID of the JDF node that is currently being executed.
<i>QueueEntryID</i> ? New in JDF 1.2	string	<i>QueueEntryID</i> of the job that is currently being executed. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> and <i>Part</i> are ignored. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> or <i>QueueEntryID</i> are specified, <i>Occupation</i> applies to all jobs.
Device *	element	Devices that the employee is currently assigned to. The data type of Device is <i>ResourceElement</i> . See "ResourceElement – Subelement of a Resource" on page 74.
Employee	element	Description of the employee being tracked. The data type of Employee is <i>ResourceElement</i> . See "ResourceElement – Subelement of a Resource" on page 74.
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 "Partial Processing of Nodes with Partitioned Resources" on page 126

The following is an example of response to an *Occupation* query:

```
<Response ID="M1" Type="Occupation" 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>
```

5.8.6 Resource

The **Resource** message can be used as a command or a query 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 simply retrieves information about the resources without modifying them, while the command modifies those settings within the resource that are specified. Settings that are not specified remain unchanged.

5.8.6.1 Resource Query

Table 5-45: 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.

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 Response 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"].

— Element: ResourceQuParams

Table 5-46: ResourceQuParams element (Section 1 of 2)

Name	Data Type	Description
<i>Classes</i> ?	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 <i>Classes</i> = "Consumable" in the query without a JobID attribute. If <i>Classes</i> is not used or empty, then all classes known shall be queried. For possible resource class names, see the <i>Class</i> attribute in Table 3-9, "Abstract Resource element," on page 54.
<i>Exact</i> = "false"	boolean	Requests an exact description of the JDF resource. If <i>true</i> , the response will also return the requested JDF resource.
<i>JobID</i> ?	string	Job ID of the JDF node that is being queried. If no JobID is specified, global device settings are queried.
<i>JobPartID</i> ?	string	Job part ID of the JDF node that is being queried.
<i>Location</i> ?	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 <i>Location</i> of distributed resources (see also Section 3.9.6.2, Locations of Physical Resources). If not specified, the location will be selected by the device.
<i>ProcessUsage</i> ?	string	Selects a resource in which the value of the <i>ProcessUsage</i> attribute of the ResourceLink (see Table 3-15, "Abstract ResourceLink element," on page 66) 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 Component input ExposedMedia of a ConventionalPrinting process can be distinguished by specifying <i>ProcessUsage</i> = "Plate" and <i>ProcessUsage</i> = "Proof", respectively. The <i>ResourceName</i> , <i>Usage</i> and <i>ProcessUsage</i> attributes are combined by a logical AND conjunction to select the resource to be queried.
<i>ProductID</i> ? New in JDF 1.2	string	<i>ProductID</i> of the resource that is queried.

Table 5-46: ResourceQuParams element (Section 2 of 2)

Name	Data Type	Description
QueueEntryID ? New in JDF 1.2	string	<i>QueueEntryID</i> of the job that is currently being executed. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> and <i>Part</i> are ignored. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> or <i>QueueEntryID</i> are specified, <i>ResourceQuParams</i> applies to all jobs.
ResourceID ? New in JDF 1.3	NMTOKEN	<i>Resource/@ID</i> of the Resource that is queried. Note: The data type is NMTOKEN and not IDREF because the referenced <i>ID</i> need not be present in the JMF.
<i>ResourceName</i> ?	NMTOKEN	Name of the resource being queried. For possible resource names, see titles in Section 7, Resources.
<i>Usage</i> ?	enumeration	<i>Input</i> – The resource is an input. <i>Output</i> – The resource is an output. Selects a resource in which the value of the <i>ResourceLink/@Usage</i> attribute (see Table 3-15, “Abstract ResourceLink element,” on page 66) matches the token specified here in this attribute. Only necessary if a resource name is used both as input and output by one node.
Part * New in JDF 1.2	element	<i>Part</i> elements that describe the resource whose messages are queried.

The following is an example of a **Resource Signal** used to report the inventory identification of the resources that were used:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="ProductionSystem"
  TimeStamp="2005-09-23T17:47:18-05:00" Version="1.3">
  <Signal ID="P172" Type="Resource">
    <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>
</JMF>
```


5.8.6.2 Resource Command

Table 5-47: 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.

The **Resource** command is used to modify or create either global device settings or 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 **MUST** be created as an Input resource to the JDF node.

The **Response** 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 was successful, the value of the *ReturnCode* attribute 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.

— Element: ResourceCmdParams

Table 5-48: ResourceCmdParams element (Section 1 of 3)

Name	Data Type	Description
<i>Activation</i> = "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. Possible values, in order of involvement from least to most active, are: <i>Held</i> – Used for uploading a resource that requires operator intervention before being applied. <i>TestRun</i> – 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. <i>TestRunAndGo</i> – Similar to <i>TestRun</i> , but requests a subsequent automatic update of the resource if the test run has been completed successfully. <i>Active</i> – Used for applying the update immediately. Note that <i>Activation</i> uses an identical syntax to JDF/@Activation , but that it does not explicitly set JDF/@Activation in the JDF on the Device. The <i>Inactive</i> value defined in JDF/@Activation is not a valid value in this list.
<i>Exact</i> = "false"	boolean	Requests an exact description of the JDF resource. If "true", the response will also return the requested JDF resource.
<i>JobID</i> ?	string	<i>JobID</i> of the JDF node that the resource being modified is linked to. If no <i>JobID</i> is specified, global resource settings are modified.
<i>JobPartID</i> ?	string	<i>JobPartID</i> of the JDF node that the resource being modified is linked to.

Table 5-48: ResourceCmdParams element (Section 2 of 3)

Name	Data Type	Description
<i>ProcessUsage</i> ?	NMTOKEN	Selects a resource in which the value of the ResourceLink/ <i>@ProcessUsage</i> attribute (see Table 3-15, “Abstract ResourceLink element,” on page 66) 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 <i>ProcessUsage = Plate</i> and <i>ProcessUsage = Proof</i> , respec- tively. The <i>ResourceName</i> and <i>ProcessUsage</i> attributes are combined by a logical AND conjunction to select the resource to be modified.
<i>ProductID</i> ? New in JDF 1.2	string	<i>ProductID</i> of the resource that is updated.
<i>ProductionAmount</i> ?	double	New amount of resource production. This value replaces the ResourceLink/ <i>@Amount</i> of the resource specified by <i>ResourceName</i> .
<i>QueueEntryID</i> ? New in JDF 1.2	string	<i>QueueEntryID</i> of the job that is currently being executed. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> and <i>Part</i> are ignored. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> or <i>QueueEntryID</i> are specified, ResourceCmdParams applies to all jobs.
<i>ResourceName</i> ?	NMTOKEN	Name of the resource whose production amount will be modified. For possible resource names see titles in Section 7, Resources.
<i>ResourceID</i> ? New in JDF 1.3	NMTOKEN	Resource/ <i>@ID</i> of the Resource that is modified. Note: The data type is NMTOKEN and not IDREF because the referenced <i>ID</i> need not be present in the JMF.
<i>Status</i> ? New in JDF 1.2	enumeration	Updated <i>Status</i> of the selected resource. The list of possible values is defined in Table 3-9, “Abstract Resource element,” on page 54.
<i>UpdateIDs</i> ? New in JDF 1.1 Deprecated in JDF 1.3	NMTOKENS	The <i>UpdateID</i> attributes of one or more ResourceUpdate that are defined in resources known to the recipient. The data type is NMTO- KENS and not IDREFS because no matching IDs exist within this mes- sage. The order of tokens in defines the order in which the updates are applied.
<i>UpdateMethod</i> = “Complete” New in JDF 1.3	enumeration	Method how the resource is updated. Values are: <i>Complete</i> – The resource partitions defined by Part are completely overwritten by Resource in this message. <i>Incremental</i> – The resource partitions defined by Part are incre- mentally updated by the values that are explicitly set in Resource in this message.
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 1.2	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.

Table 5-48: ResourceCmdParams element (Section 3 of 3)

Name	Data Type	Description
Resource *	element	Resources to be uploaded to the Device. They completely replace the original resources with the same ID. The resources to be modified are identified by their <i>ID</i> values, which means that the <i>ID</i> attributes MUST be known to the controller that issued the Resource command. The data type and <i>Class</i> of Resource is derived from the abstract resource. See “Abstract Resource” on page 54.

The following is an example for specifying that a the Cyan, Front plate of *Sheet2*, Signature 1 has become available

```
<Command ID="C1" Type="Resource">
  <ResourceCmdParams JobID="MakeBrochure 012" ResourceID="ExposedMediaID">
    <Part SignatureName="Sig1" SheetName="Sheet2" Side="Front" Separation="Cyan"/>
  </ResourceCmdParams>
</Command>
```

The following is an example for specifying that a 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">
  <ResourceCmdParams JobID="MakeBrochure 012" ResourceID="ExposedMediaID">
    <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>
```

— Element: ResourceInfo

Table 5-49: ResourceInfo element (Section 1 of 3)

Name	Data Type	Description
<i>ActualAmount</i> ? New in JDF 1.2	double	Reflects the current accumulated amount of the resource that has been consumed (input) or produced (output) by the process. This corresponds to the value of the <i>ActualAmount</i> attribute in the corresponding ResourceLink of the resource where it to be written now.
<i>Amount</i> ?	double	Reflects the intended accumulated amount of the resource that will be consumed (input) or produced (output) by the process. This corresponds to the value of the <i>Amount</i> attribute in the corresponding ResourceLink of the resource where it to be written now.
<i>AvailableAmount</i> ?	double	Device-specific amount of the <i>Consumable</i> resource that is available in the device.
<i>Level</i> = "OK"	enumeration	This attribute is device dependent. A Device MAY specify a level status that describes a low or empty consumable level. Possible values are: <i>Empty</i> – Specification is left to the device manufacturer. <i>Low</i> – Specification is left to the device manufacturer. <i>OK</i> – Specification is left to the device manufacturer.

Table 5-49: ResourceInfo element (Section 2 of 3)

Name	Data Type	Description
<i>Location</i> ?	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 <i>Location</i> of distributed resources (see also Section 3.9.6.2, Locations of Physical Resources). If not specified, the location will be defined by the device.
<i>ModuleID</i> ? New in JDF 1.3	string	<i>ModuleID</i> of the Module that the Resource is consumed or produced by. If neither of <i>ModuleID</i> or <i>ModuleIndex</i> are specified, defaults to the entire device specified by JMF/@ <i>SenderID</i> .
<i>ModuleIndex</i> ? 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 <i>ModuleID</i> or <i>ModuleIndex</i> are specified, defaults to the entire device specified by JMF/@ <i>SenderID</i> .
<i>ResourceName</i> ?	NMTOKEN	Name of the resource if <i>Exact</i> = <i>false</i> in the query. Exactly one of <i>Resource</i> or <i>ResourceName</i> MUST be specified.
<i>ProcessUsage</i> ?	NMTOKEN	Selects a resource in which the value of the ResourceLink/@ <i>ProcessUsage</i> attribute (see Table 3-15, “Abstract ResourceLink element,” on page 66) 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 <i>ProcessUsage</i> = <i>Proof</i> and <i>ProcessUsage</i> = <i>Plate</i> , respectively. The <i>ResourceName</i> and <i>ProcessUsage</i> attributes are combined by a logical AND conjunction to select the resource to be queried.
<i>ProductID</i> ? New in JDF 1.2	string	<i>ProductID</i> of the resource.
<i>Status</i> ? New in JDF 1.2	enumeration	Updated <i>Status</i> of the selected resource. The list of possible values is defined in Table 3-9, “Abstract Resource element,” on page 54.
<i>Unit</i> ?	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).
<i>AmountPool</i> ? New in JDF 1.3	element	Definition of partial amounts and pipe parameters for this Resource. The contents of the <i>AmountPool</i> are described for the various types of ResourceLink elements in Table 3-16, “AmountPool element,” on page 69. If <i>AmountPool</i> is specified, the ResourceInfo MUST NOT contain any of the amount related attributes defined in <i>AmountPool/PartAmount</i> .
<i>CostCenter</i> ?	element	Cost center to which the resource consumption is allocated.
<i>MISDetails</i> ? New in JDF 1.2	element	Definition how the costs for the production of the Resource are to be charged.

Table 5-49: ResourceInfo element (Section 3 of 3)

Name	Data Type	Description
Part * New in JDF 1.2 Deprecated in JDF 1.3	element	Part elements that describe the resource. In JDF 1.3 and beyond, only AmountPool/PartAmount SHOULD be specified. Part is no longer specified in ResourceInfo because it is unambiguously selected by ResourceCmdParams/Part or ResourceQuParams/Part.
Resource ?	element	JDF description of the resource. If the query or command leading to this response element contains Part elements, the resource MUST contain only the appropriate matching partitions. The data type and Class of Resource is derived from the abstract resource. See "Abstract Resource" on page 54.

The following is an example for retrieving settings:

```
<Query ID="Q1" Type="Resource">
  <ResourceQuParams Classes="Consumable" Exact="true"/>
</Query>
```

The following is a possible response to the query above:

```
<Response ID="M1" Type="Resource" refID="Q1">
  <ResourceInfo AvailableAmount="2120" Location="Paper Tray 1">
    <Media>
      <!-- Media resource defined in JDF -->
    </Media>
  </ResourceInfo>
  <ResourceInfo AvailableAmount="0" Level="Empty" Location="Ink1" Unit="1">
    <Ink>
      <!-- Ink description resource defined in JDF -->
    </Ink>
  </ResourceInfo>
</Response>
```

The following is an example for modifying the production amount of a specific job to produce brochures

```
<Command ID="C1" Type="Resource">
  <ResourceCmdParams JobID="MakeBrochure 012" ProductionAmount="7500"
  ResourceName="Component"/>
</Command>
```

The following is a possible response to the resource command above:

```
<Response ID="M2" Type="Resource" refID="C1">
  <ResourceInfo Amount="7500" ResourceName="Component"/>
</Response>
```

5.8.7 ResourcePull

[New in JDF 1.2](#)

Table 5-50: ResourcePull message

Object Type	Element	Description
CommandTypeObj	ResourcePullParams	Defines the parameters of the repeated job.
	QueueFilter	Defines a filter for the returned Queue element in the ResourcePull message.
ResponseTypeObj	QueueEntry	Provides the queue entry of the repeated job.
	Queue	Describes the state of the queue after the command has been executed.
Definition of the QueueEntry and Queue elements, see "Elements for Queues" on page 211.		

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*.

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 MUST 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.

— Element: 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.

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">
  <ResourcePullParams QueueEntryID="AllPlates" Priority="100" ResourceID="R42">
    <Part SheetName="Sheet1" Side="Front" Separation="Yellow"/>
  </ResourcePullParams>
</Command>
```

Table 5-51: ResourcePullParams element (Section 1 of 2)

Name	Data Type	Description
<i>Amount</i> ?	double	The <i>Amount</i> attribute identifies the amount of the output resource to be created by the JDF node that is executed by the cloned <code>QueueEntry</code> . This <i>Amount</i> is the amount to be produced by the process that is executed due to the <code>ResourcePull</code> . Thus if 200 copies had been created previously and 100 copies are requested by the <code>ResourcePull</code> , <i>Amount</i> = "100" and not "300".
<i>Hold</i> = "false"	boolean	If "true", the entry is submitted as held.
<i>NextQueueEntryID</i> ?	string	ID of the queue entry that is to be positioned directly behind the entry.
<i>PrevQueueEntryID</i> ?	string	ID of the queue entry that is to be positioned directly in front of the entry.
<i>JobID</i> ?	string	Job ID of the JDF node that creates the requested resource. If <i>QueueEntryID</i> is specified, <i>JobID</i> is ignored. Exactly one of <i>JobID</i> or <i>QueueEntryID</i> MUST be specified.
<i>Priority</i> = "1"	integer	Number from 0 to 100, where 0 is the lowest priority and 100 is the maximum priority.
<i>QueueEntryID</i> ?	string	<i>QueueEntryID</i> of the JDF node that creates the requested resource. If <i>QueueEntryID</i> is specified, <i>JobID</i> is ignored. Exactly one of <i>JobID</i> or <i>QueueEntryID</i> MUST be specified.

Table 5-51: ResourcePullParams element (Section 2 of 2)

Name	Data Type	Description
<i>RepeatPolicy</i> ?	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). One of: <i>Complete</i> – 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. <i>CompleteOnly</i> – Restart from the original input resources. The process MUST NOT run if any original resources are not available. <i>Fast</i> – Reuse as many intermediate resources as possible, (e.g., restart ImageSetting from stored intermediate raster files and do not reRIP if possible).
<i>ResourceID</i>	string	ID attribute of the Resource requested.
<i>ReturnURL</i> ?	URL	URL where the JDF file is to be written when the job is completed or aborted. If not specified, the JDF is to be placed in the default output hot folder of the queue controller. The <i>ReturnURL</i> takes precedence when <i>NodeInfo/@TargetRoute</i> is specified in the previously submitted JDF.
<i>WatchURL</i> ?	URL	URL of the controller that is to be notified when the status of the <i>QueueEntry</i> or the underlying job changes. Specifying <i>WatchURL</i> is equivalent to sending a subscription for an <i>Events</i> message with <i>SignalTypes</i> = "All".
Part *	element	The <i>Part</i> elements identify the parts of a partitioned output resource to be created by the JDF node. The structure of the <i>Part</i> element is defined in Table 3-26, "Part element," on page 87. For details on partitioned resources, see Section 3.9.5, Description of Partitioned Resources. For details on Node partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 126.
Disposition ?	element	Specifies how long the <i>QueueEntry</i> SHOULD be retained in the queue. If not specified, the <i>QueueEntry</i> MAY be removed from the queue immediately after process completion of the <i>QueueEntry</i> .
MISDetails ?	element	Definition how the costs for the production of the <i>Resource</i> are to be charged.

5.8.8 ShutDown

[New in JDF 1.2](#)

The *ShutDown* message shuts down a controller or device. Note that a Device MUST use the *Status* message if it signals its own shutdown.

Table 5-52: ShutDown message (Section 1 of 2)

Object Type	Element	Description
CommandTypeObj	<i>ShutDownCmdParams</i>	Defines the details of a shutdown.
	<i>QueueFilter</i>	Defines a filter for the returned <i>Queue</i> element in the <i>ShutDown</i> message.

Table 5-52: ShutDown message (Section 2 of 2)

Object Type	Element	Description
ResponseTypeObj	DeviceInfo	Describes the device status as anticipated after the shut-down.
	Queue	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 "Elements for Queues" on page 211s.

— Element: ShutDownCmdParams

Table 5-53: ShutDownCmdParams element

Name	Data Type	Description
<i>ShutDownType</i> = "StandBy"	enumeration	Defines the device shutdown method. Possible values are: <i>StandBy</i> – The device is set to standby mode. It can be restarted with a <i>WakeUp</i> JMF message. <i>Full</i> – Completely shut down the device. It is no longer accessible via JMF after the shutdown.
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.

5.8.9 Status

Table 5-54: Status message

Object Type	Element Name	Description
QueryTypeObj	StatusQuParams	Refines the query to include various aspects of the device and job states.
ResponseTypeObj Modified in JDF 1.2	DeviceInfo +	Describes the actual device status. If the controller handles multiple devices, one <i>DeviceInfo</i> MUST be specified for each device.
	Queue ?	Provides information about the queue and all its entries. This element will only be provided if the device has queue capabilities. The <i>Queue</i> element is described in Section 5.13, Elements for Queues.

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. The response MAY also provide a *Queue* element.

— Element: StatusQuParams

The various aspects of the device, queue and job states are refined by the *StatusQuParams* element. This element contains three groups of parameters. The first group serves to refine the device-specific status information queried. The parameters *EmployeeInfo* and *DeviceDetails* belong to this group. The second group serves to refine the job specific status information. These are *JobDetails*, *JobID* and *JobPartID*. And the third determines simply whether a queue element is requested to be appended. This is specified by the attribute *QueueInfo*.

In order to focus on the status of a certain job, the job MUST be uniquely identified using the *JobID* attribute. It might be necessary to define a process or a part of a job as the query target under certain circumstances, such as when a job is processed in parallel. This is accomplished using the *JobPartID* attribute of the *StatusQuParams* element. A value of *JobDetails* = "Full" requests a complete JDF description of a snapshot of the specified job or job part.

If the specified job or job part is unknown, the value of the *ReturnCode* attribute is 103 or 104 (for error codes, see "Supported Error Codes in JMF and Notification elements" on page 717).

Table 5-55: StatusQuParams element

Name	Data Type	Description
<i>DeviceDetails</i> = "None"	enumeration	Refines the provided status information about the device. Possible values are: <i>None</i> <i>Brief</i> – Provide all available device information except for Device elements. <i>Modules</i> – Provide ModuleStatus elements without module specific status details and without module specific employee information. <i>Details</i> – Provide maximum available device information excluding device capability descriptions. Includes Device elements which represent details of the device. <i>Capability</i> – Provide Device elements with DeviceCap subelements which represent details of the capabilities of the device. <i>Full</i> – Provide maximum available device information including device capability descriptions. Includes Device elements which represent details of the device.
<i>EmployeeInfo</i> = "false"	boolean	If <i>true</i> , Employee elements are to be provided in the response. Those elements describe the employees which are associated to the device independent on any job.
<i>JobDetails</i> = "None" Modified in JDF 1.2	enumeration	Refines the provided status information about the jobs associated with the device. Each higher entry includes the values specified in the lower entries. Possible values are: <i>None</i> – Specify only <i>JobID</i> , <i>JobPartID</i> and <i>Amount and/or PercentCompleted</i> . <i>MIS</i> – Provide business with the relevant information contained in the CostCenter element and the <i>DeadLine</i> , <i>DeviceStatus</i> , <i>Status</i> , <i>StatusDetails</i> and the various <i>Counter</i> attributes. In JDF 1.2 and beyond, this value is identical to <i>Brief</i> . Deprecated in JDF 1.2 <i>Brief</i> – Provide all available status information including JobPhase elements except for JDF. <i>Full</i> – Provide maximum available status information. Includes an actual JDF which represents a snapshot of the current job state.
<i>JobID</i> ?	string	Job ID of the JDF node whose status is being queried. If not specified, list all known jobs.
<i>JobPartID</i> ?	string	JobPart ID of the JDF node whose status is being queried.
<i>QueueEntryID</i> ? New in JDF 1.2	string	<i>QueueEntryID</i> of the job that is being queried. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> and <i>Part</i> are ignored. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> or <i>QueueEntryID</i> are specified, StatusQuParams applies to all jobs.
<i>QueueInfo</i> = "false"	boolean	If <i>true</i> , a Queue element is requested to be provided. This is analogous to a QueueStatus query (see Section 5.12.6, QueueStatus).
<i>Part</i> * 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 "Partial Processing of Nodes with Partitioned Resources" on page 126.

— Element: DeviceInfo

The response returns a **DeviceInfo** element for the queried device.

Table 5-56: DeviceInfo element (Section 1 of 2)

Name	Data Type	Description
<i>CounterUnit</i> ?	string	The unit of the <i>ProductionCounter</i> , the <i>TotalProductionCounter</i> and numerator unit of <i>Speed</i> . 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 printer, it is the number of sheets; in case of a web printer, it is the length of printed web in meters.
<i>DeviceCondition</i> ? New in JDF 1.2	enumeration	The general condition of a device. <i>OK</i> – The device is in working condition. <i>NeedsAttention</i> – The device is still in working condition but requires attention. <i>Failure</i> – The device is not in working condition. <i>OffLine</i> – The device is off line and its condition is unknown.
<i>DeviceID</i> ? New in JDF 1.3	string	<i>DeviceID</i> of the Device that this DeviceInfo describes. <i>DeviceID</i> MUST match Device/@DeviceID if Device is specified in this DeviceInfo.
<i>DeviceOperationMode</i> ? New in JDF 1.2	enumeration	<i>DeviceOperationMode</i> 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. <i>Productive</i> – The device is used to produce good product. Any times recorded in this mode are to be allocated against the job. <i>NonProductive</i> – The device is used without the intention to produce good product. Any times recorded in this mode are not be allocated against the job. <i>Maintenance</i> – The device is used without the intention to produce good product, e.g., to perform (preventative) maintenance.
<i>DeviceStatus</i>	enumeration	The status of a device. Possible values are: <i>Unknown</i> – No device is known or the device cannot provide a <i>DeviceStatus</i> . <i>Idle</i> – No job is being processed and the device is accepting new jobs. <i>Down</i> – No job is being processed and the device currently cannot execute a job. The device might be broken, switched off, etc. <i>Setup</i> – The device is currently being set up. This state is allowed to occur also during the execution of a job. <i>Running</i> – The device is currently executing a job. <i>Cleanup</i> – The device is currently being cleaned. This state is allowed to occur also during the execution of a job. <i>Stopped</i> – 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.
<i>HourCounter</i> ?	duration	The total integrated time (life time) of device operation in hours.
<i>PowerOnTime</i> ?	dateTime	Date and time when the device was switched on.

Table 5-56: DeviceInfo element (Section 2 of 2)

Name	Data Type	Description
<i>ProductionCounter</i> ?	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 an Events message of <i>Type</i> = <i>CounterReset</i> (see "NotificationDetails" on page 709).
<i>Speed</i> ?	double	The current machine speed. <i>Speed</i> is defined in the same units as <i>ProductionCounter</i> / hour.
<i>StatusDetails</i> ?	string	String that defines the device state more specifically. For a list of supported values, see "StatusDetails Supported Strings" on page 703.
<i>TotalProductionCounter</i> ?	double	The current total machine production counter since the machine was produced.
Device ?	element	A Device resource that describes details of the device. The data type of Device is ResourceElement. See "ResourceElement – Subelement of a Resource" on page 74.
Employee *	element	Employee resources that describe which employees are currently working at the device. The data type of Employee is ResourceElement. See "ResourceElement – Subelement of a Resource" on page 74.
JobPhase *	element	Describes the actual status of jobs in the device. All jobs that are active on the device MUST be specified. Supplying no JobPhase specifies that no job is currently active on the device. Active jobs have JDF/@Activation = <i>Active</i> , <i>TestRun</i> or <i>TestRunAndGo</i> and JDF/@Status or JDF/StatusPool/PartStatus/@Status = <i>TestRunInProgress</i> , <i>Setup</i> , <i>InProgress</i> , <i>Cleanup</i> or <i>Stopped</i> . For details on using JobPhase elements, see Table 5-57 on page 185.
ModuleStatus *	element	Status of individual modules. ModuleStatus MUST NOT be specified for modules that are specified in JobPhase/ModuleStatus . For details on using ModuleStatus elements, see Table 5-58 on page 187.

— Element: JobPhase

A **Status** response 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 3.10.1.3, **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 embed 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 as those which apply for the **Resource** response.

Table 5-57: JobPhase element (Section 1 of 3)

Name	Data Type	Description
<i>Activation</i> ? New in JDF 1.1	enumeration	The activation of the JDF node. Possible values are the same as the possible values of a JDF node's <i>Activation</i> attribute. For details, see Table 3-4, "JDF node," on page 41.

Table 5-57: JobPhase element (Section 2 of 3)

Name	Data Type	Description
<i>Amount</i> ?	double	Total <i>Amount</i> that the node defined in this <i>JobPhase</i> produced since <i>StartTime</i> . If <i>Waste</i> is also specified, the value is without waste. The unit is specified in the <i>CounterUnit</i> attribute of the parent element <i>DeviceInfo</i> .
<i>DeadLine</i> ?	enumeration	Scheduling state of the job. Possible values are: <i>InTime</i> – The job or job part will probably not miss the deadline. <i>Warning</i> – The job or job part could miss the deadline. <i>Late</i> – The job or job part will miss the deadline. For more details on scheduling, see NodeInfo .
<i>JobID</i> ?	string	<i>JobID</i> of the JDF node that is executing.
<i>JobPartID</i> ?	string	<i>JobPartID</i> of the JDF node that is executing.
<i>PercentCompleted</i> ?	double	Node processing progress in percent (%) completed.
<i>PhaseAmount</i> ? New in JDF 1.2	double	Amount that the node defined in this <i>JobPhase</i> produced during this <i>JobPhase</i> . If <i>PhaseWaste</i> is also specified, the value is without waste. The unit is specified in the <i>CounterUnit</i> attribute of the parent element <i>DeviceInfo</i> .
<i>PhaseStartTime</i> ? New in JDF 1.2	dateTime	Time that this <i>JobPhase</i> started.
<i>PhaseWaste</i> ? New in JDF 1.2	double	Amount of waste that the node defined in this <i>JobPhase</i> produced during this <i>JobPhase</i> . The unit is specified in the <i>CounterUnit</i> attribute of the parent element <i>DeviceInfo</i> .
<i>QueueEntryID</i> ?	string	If the job was submitted to a <i>Queue</i> and the <i>QueueEntryID</i> is known, this attribute SHOULD be provided.
<i>RestTime</i> ? New in JDF 1.1	duration	Estimated duration of time to finishing processing this node.
<i>Speed</i> ?	double	The current job speed. <i>Speed</i> is defined in the same units as <i>ProductionCounter</i> / hour. Defaults to the speed specified in the <i>DeviceInfo</i> element.
<i>StartTime</i> ? New in JDF 1.1	dateTime	Time when execution of the node that is described by this <i>JobPhase</i> has been started, defined by the transition of JDF/@Status from <i>Waiting</i> or <i>Ready</i> to any active value.
<i>Status</i>	enumeration	The status of the JDF node. Possible values are the same as the possible values of a JDF node's <i>Status</i> attribute. For details, see Table 3-4, "JDF node," on page 41.
<i>StatusDetails</i> ?	string	String that defines the job state more specifically. For a list of supported values, see "StatusDetails Supported Strings" on page 703.
<i>TotalAmount</i> ? New in JDF 1.1	double	Amount that will be produced when this job phase is 100% completed. The unit is specified in the <i>CounterUnit</i> attribute of the parent element <i>DeviceInfo</i> .
<i>Waste</i> ? New in JDF 1.1	double	Total <i>Amount</i> of waste that the node defined in this <i>JobPhase</i> produced since <i>StartTime</i> . The unit is specified in the <i>CounterUnit</i> attribute of the parent element <i>DeviceInfo</i> .
CostCenter ?	element	The cost center that the job is currently being charged to. Defaults to the cost center specified in the <i>DeviceInfo</i> element.

Table 5-57: JobPhase element (Section 3 of 3)

Name	Data Type	Description
JDF ?	element	Complete JDF node that represents a snapshot of the job that is currently being processed. This element is for reference only and MUST NOT be merged with the main JDF of the job using spawning and merging methods. JDF/@Activation MUST be set to " <i>Informative</i> " in this JDF element.
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 MUST NOT be specified for modules that are specified in DeviceInfo/ModuleStatus. For details on using ModuleStatus elements, see Table 5-58 on page 187.
Part * Modified in JDF 1.1	element	Describes which parts of a job are currently being processed. For details on Node partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 126.

— Element: ModuleStatus

The ModuleStatus element is identical to the ModulePhase element of the PhaseTime audit element (see Table 3-34, "ModulePhase audit element," on page 106), except that the attributes *Start* and *End* are missing. These attributes specify the time interval in the Audit pendant ModulePhase and the *DeviceID* attribute, which is unnecessary here. The ModuleStatus element is described in the following table.

Table 5-58: ModuleStatus element (Section 1 of 2)

Name	Data Type	Description
<i>CombinedProcessIndex</i> ? New in JDF 1.3	IntegerList	<i>CombinedProcessIndex</i> attribute specifies the indices of individual processes in the <i>Types</i> attribute to which a ModuleStatus that describes a <i>Combined</i> or <i>ProcessGroup</i> node belongs. Multiple entries in <i>CombinedProcessIndex</i> specify that the Module specified by ModuleStatus is executing the respective multiple processes in the <i>Combined</i> node.
<i>DeviceStatus</i>	enumeration	Status of the module. Possible values are: <i>Unknown</i> – The module status is unknown. <i>Idle</i> – The module is not used. An example is a color print module that is inactive during a black-and-white print. <i>Down</i> – The module cannot be used. It might be broken, switched off etc. <i>Setup</i> – The module is currently being set up. <i>Running</i> – The module is currently executing. <i>Cleanup</i> – The module is currently being cleaned. <i>Stopped</i> – 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.
<i>ModuleID</i> ? New in JDF 1.3	string	<i>ModuleID</i> of the Module that ModuleStatus refers to. If not specified, the module is specified in <i>ModuleIndex</i> . At least one of <i>ModuleID</i> or <i>ModuleIndex</i> MUST be specified.

Table 5-58: ModuleStatus element (Section 2 of 2)

Name	Data Type	Description
<i>ModuleIndex</i> ? 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 MUST also be unique.
<i>ModuleType</i>	NMTOKEN	Module description. The allowed values depend on the type of device that is described. The predefined values are listed in "ModuleType Supported Strings" on page 707.
<i>StatusDetails</i> ?	string	Description of the module status phase that provides details beyond the enumerative values given by the <i>DeviceStatus</i> attribute. For a list of supported values, see "StatusDetails Supported Strings" on page 703.
Employee *	element	Links to Employee resources that are working at this module (the module is specified by the attributes <i>ModuleIndex</i> and <i>ModuleType</i>). The data type of Employee is ResourceElement. See "ResourceElement – Subelement of a Resource" on page 74.

The following is an example of a response to a **Status** query. 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:

```
<JMF SenderID="A3 Printer" TimeStamp="2005-07-25T12:32:48+02:00"
  xmlns="http://www.CIP4.org/JDFSchema_1_1" Version="1.3">
  <Response ID="M1" refID="Q1" Type="Status">
    <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>
</JMF>
```

5.8.10 Track

Table 5-59: Track message

Object Type	Element Name	Description
QueryTypeObj	TrackFilter ?	Refines the Track query.
ResponseTypeObj	TrackResult *	Details of the tracked jobs.

The Track query 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 is a list of TrackResult elements.

— Element: 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 5-60: TrackFilter element

Name	Data Type	Description
<i>JobID</i> ?	string	<i>JobID</i> of the JDF node that is being tracked. Defaults to list JobPhase elements of all known nodes.
<i>JobPartID</i> ?	string	<i>JobPartID</i> of the JDF node that is being tracked.
<i>ProjectID</i> ? New in JDF 1.2	string	<i>ProjectID</i> of the JDF node that is being tracked.
<i>QueueEntryID</i> ? New in JDF 1.2	string	<i>QueueEntryID</i> of the job that is currently being executed. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> and <i>Part</i> are ignored. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> , <i>ProjectID</i> or <i>QueueEntryID</i> are specified, <i>TrackFilter</i> applies to all jobs.
<i>Status</i> ?	enumerations	The JDF/@Status of the jobs being tracked. Possible values are a combination of any of the possible values of a JDF node's <i>Status</i> attribute. When not known or specified, all enumerations are applicable. Details and possible values are defined in Table 3-4, "JDF node," on page 41 and Table 3-5, "Status attribute – possible values," on page 45.
<i>Part</i> * New in JDF 1.2	element	<i>Part</i> elements that describe the partition of the job that is being tracked. For details on Node partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 126.

— Element: TrackResult

One *TrackResult* is returned for each known JDF or spawned JDF part. *TrackResult* elements contain information about the location of distributed jobs.

Table 5-61: TrackResult element

Name	Data Type	Description
<i>JobID</i>	string	<i>JobID</i> of the JDF node that is being tracked.
<i>JobPartID</i> ?	string	<i>JobPartID</i> of the highest level node of the JDF node that is being tracked.
<i>ProjectID</i> ? New in JDF 1.2	string	<i>ProjectID</i> of the highest level node of the JDF node that is being tracked.
<i>QueueEntryID</i> ? New in JDF 1.2	string	<i>QueueEntryID</i> of the job that is currently being tracked.
<i>URL</i>	URL	URL of the controller that owns this job.
<i>IsDevice</i>	boolean	If <i>true</i> , the controller that emitted this message is the device that has access to the job and can be queried for details of the job.
<i>Part</i> * New in JDF 1.2	element	<i>Part</i> elements that describe the partition of the job that is being tracked. For details on Node partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 126.

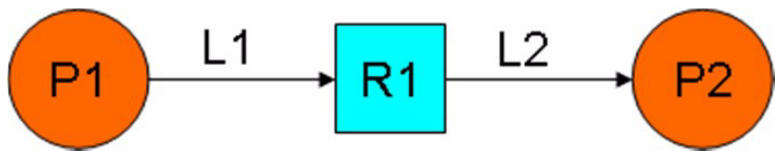
The following is an example of a response on a *Track* message:

```
<Response ID="M1" Type="Track" refID="Q1">
  <TrackResult IsDevice="true" JobID="1" JobPartID="42"
    URL="http://www.anycompany.com/controller"/>
</Response>
```

5.8.11 UpdateJDF

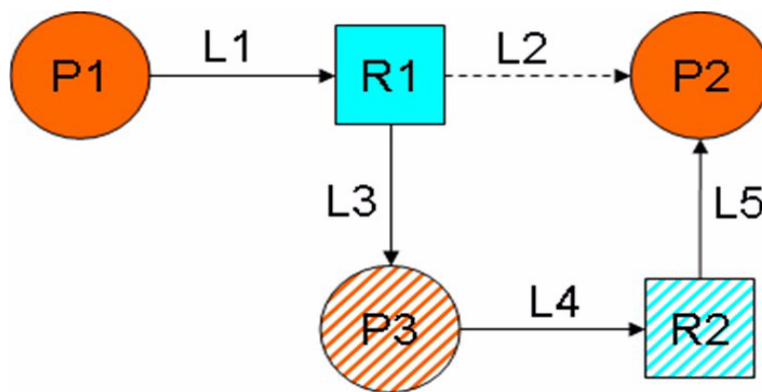
[New in JDF 1.3](#)

This JMF is used to update a JDF node that has been submitted to a Controller or a Device. Typically this JMF will be sent from the MIS to a Controller which received the original job. The changes **MUST** 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.



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 5-4: Without 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 **MUST** be removed from the JDF.

Figure 5-5: With UpdateJDF Message

Table 5-62: UpdateJDF message

Object Type	Element Name	Description
CommandTypeObj	UpdateJDFCmdParams ?	Defines the details of the UpdateJDF message.
ResponseTypeObj	-	-

— Element: UpdateJDFCmdParams

The UpdateJDFCmdParams specifies a JDF Node, new **Resources** and new ResourceLink elements to add to existing nodes.

Table 5-63: UpdateJDFCmdParams element (Section 1 of 2)

Name	Data Type	Description
ParentJobID	string	JobID of the node in which the new node is to be inserted.
ParentJobPartID	string	JobPartID of the node in which the new node is to be inserted.
CreateLink *	element	New ResourceLink elements to be added to the previously submitted JDF nodes.

Table 5-63: UpdateJDFCmdParams element (Section 2 of 2)

Name	Data Type	Description
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.
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.
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.

— Element: CreateLink**Table 5-64: 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 MUST be added to the existing ResourceLink.

— Element: CreateResource**Table 5-65: CreateResource element**

Name	Data Type	Description
JobID	string	JobID of the node in which the new Resources are to be inserted.
JobPartID	string	JobPartID of the node in which the new Resources are to be inserted.
Resource +	element	The new Resource . 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 “Abstract Resource” on page 54.

— Element: MoveResource**Table 5-66: MoveResource element**

Name	Data Type	Description
JobID	string	JobID of the node to which the new Resource is to be moved.
JobPartID	string	JobPartID of the node in which the new Resource is to 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.

— Element: RemoveLink**Table 5-67: RemoveLink element**

Name	Data Type	Description
<i>JobID</i>	string	<i>JobID</i> of the node from which the ResourceLink elements are to be removed.
<i>JobPartID</i>	string	<i>JobPartID</i> 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 Parts specified in this ResourceLink are to be removed.

Sample:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" TimeStamp="2005-07-25T12:32:48+02:00"
  DeviceID="AnyDevice" SenderID="JDFMaster" Version="1.3">
  <Command ID="ID1" Type="UpdateJDF">
    <UpdateJDFCmdParams ParentJobID="100" ParentJobPartID="112">
      <CreateLink JobID="100" JobPartID="111">
        <MediaLink Usage="Input" rRef="link001111"/>
      </CreateLink>
      <CreateResource JobID="100" JobPartID="110">
        <Component rRef="link001111"/>
      </CreateResource>
      <RemoveLink JobID="100" JobPartID="111">
        <MediaLink Usage="Input" rRef="link001111"/>
      </RemoveLink>
      <MoveResource JobID="100" JobPartID="101" ResourceID="link000001"/>
      <JDF JobPartID="200" Type="Cutting">
        <AuditPool>
          <Created Author="MIS" TimeStamp="2005-06-02T09:01:45+01:00"
            AgentName="MIS" AgentVersion="1.0"/>
        </AuditPool>
        <ResourceLinkPool>
          <ComponentLink Usage="Output" rRef="link000001"/>
        </ResourceLinkPool>
      </JDF>
    </UpdateJDFCmdParams>
  </Command>
</JMF>
```

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.

5.8.12 WakeUp[New in JDF 1.2](#)

All queues that belong to the device are held upon its receiving a WakeUp and MUST be resumed with an explicit ResumeQueue message. All jobs that were running on the device at shutdown are also in a held state and MUST be explicitly resumed with a ResumeQueueEntry message.

Table 5-68: 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.

— Element: WakeUpCmdParams

WakeUpCmdParams is a placeholder for future use and for extensions to the WakeUp message.

Table 5-69: WakeUpCmdParams element

Name	Data Type	Description
—	—	—

5.9 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 5-70: 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.
PipePull	CR	Requests a new resource from a pipe.
PipePush	CR	Notifies that a new resource is available in a pipe.
PipePause	CR	Pauses a process if no further resources can be consumed or produced.

5.9.1 PipeClose

Table 5-71: PipeClose message

Object Type	Element Name	Description
CommandTypeObj	PipeParams	Describes the pipe resource. The PipeParams element is described in Section 5.9.2, PipePull.
ResponseTypeObj	JobPhase	The status of the responding process. The JobPhase element is defined in Table 5-57 on page 185.

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 response is equivalent to the PipePull and PipePush command responses described below.

5.9.2 PipePull

Table 5-72: PipePull message

Object Type	Element Name	Description
CommandTypeObj	PipeParams	Describes the requested pipe resource.
ResponseTypeObj	JobPhase	The status of the responding process. The JobPhase element is defined in Table 5-57 on page 185.

The PipePull message requests resources that are described in a JDF dynamic pipe (see Section 3.7.4, 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 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 Supported Error Codes in JMF and Notification elements. The Response MAY contain a Notification element. The JobPhase element (see Section 5.8.9, Status) returned SHOULD provide only the *Status* attribute that describes the job status of the responding process after receiving the command.

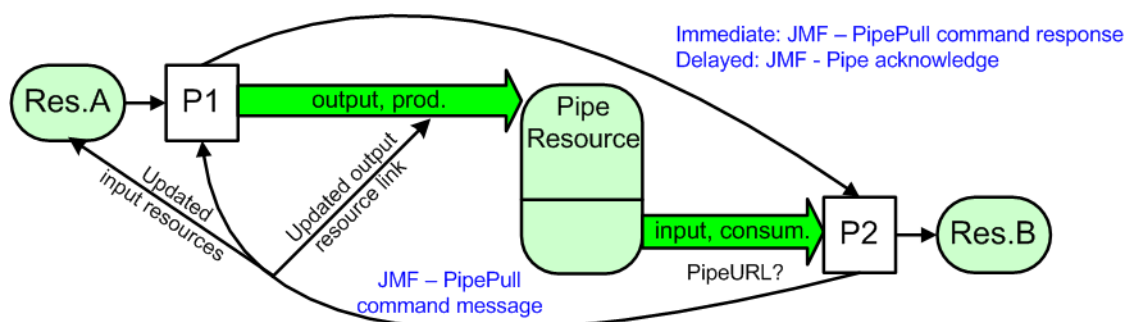


Figure 5-6: Mechanism of a PipePull message

— Element: PipeParams

The PipeParams element is also used by the messages PipeClose, PipePush and PipePause. The URL where an OPTIONAL Acknowledge will be sent when the pipe command has been executed is defined in the initiating command message by the attribute *AcknowledgeURL*. The Acknowledge is sent for the following commands:

- for PipeClose: when the process has been finished,
- for PipePull: when the resource is available,
- for PipePush: when the resource has been accepted and
- for PipePause: when the process has been stopped.

Table 5-73: PipeParams element (Section 1 of 2)

Name	Data Type	Description
<i>JobID</i> ? New in JDF 1.2	string	Specifies the <i>JobID</i> of the node at the receiving end of the message that links to the resource specified in <i>PipeID</i> .
<i>JobPartID</i> ? New in JDF 1.2	string	Specifies the <i>JobPartID</i> of the node at the receiving end of the message that links to the resource specified in <i>PipeID</i> .
<i>PipeID</i>	string	Pipe ID of the JDF resource that defines the dynamic pipe.
<i>Status</i> = "InProgress"	enumeration	Process status after the request. Possible values are defined in Table 3-4, "JDF node," on page 41.

Table 5-73: PipeParams element (Section 2 of 2)

Name	Data Type	Description
<i>UpdatedStatus</i> ?	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. For possible values of the resource <i>Status</i> attribute see Table 3-9, “Abstract Resource element,” on page 54.
Resource *	element	Updated input resources to be used by the process that receives the pipe command: <i>PipePull</i> (the receiver creates the pipe resource), <i>PipePush</i> (the receiver consumes the pipe resource) and <i>PipePause</i> (the receiver only updates the inputs). The resource to be updated is identified by the <i>ID</i> , that means the <i>ID</i> attribute MUST be known to the controller that issued the pipe command. Possible commands are: <i>PipePull</i> , <i>PipePush</i> or <i>PipePause</i> . In case of the <i>PipeClose</i> command, the resources are ignored. The data type and <i>Class</i> of Resource is derived from the abstract resource. See “Abstract Resource” on page 54.
ResourceLink ?	element	Updated ResourceLink to the pipe resource: <i>PipePull</i> (it is an output link), <i>PipePush</i> (it is an input link) and <i>PipePause</i> (depends on the pipe end). This ResourceLink MAY be used by the process that links to the pipe resource. The attributes <i>rRef</i> and <i>Usage</i> of a ResourceLink MUST NOT be modified by the agent that sends the Pipe message because these attributes are used by the JMF receiver to identify the ResourceLink that is to be modified. For details see Section 3.7.5, ResourceUpdate Elements. In the context of dynamic pipes these two attributes have no meaning. In case of the <i>PipeClose</i> command, the ResourceLink is ignored.

5.9.3 PipePush

Table 5-74: PipePush message

Object Type	Element Name	Description
CommandTypeObj	PipeParams	Describes the produced pipe resource. The PipeParams element is described in Section 5.9.2, PipePull.
ResponseTypeObj	JobPhase	The status of the responding process. The JobPhase element is defined in Table 5-57 on page 185.

The *PipePush* message notifies the availability of pipe resources that are described in a JDF dynamic pipe (see Section 3.7.4, Pipe Resources and Section 4.3.3, Overlapping Processing Using Pipes). *PipePush* messages are the JMF equivalent of a dynamic output *ResourceLink*. The Figure 5-7 depicts the mode of operation of a *PipePush* message. The *PipePush* command response is equivalent to the *PipePull* command response described above.

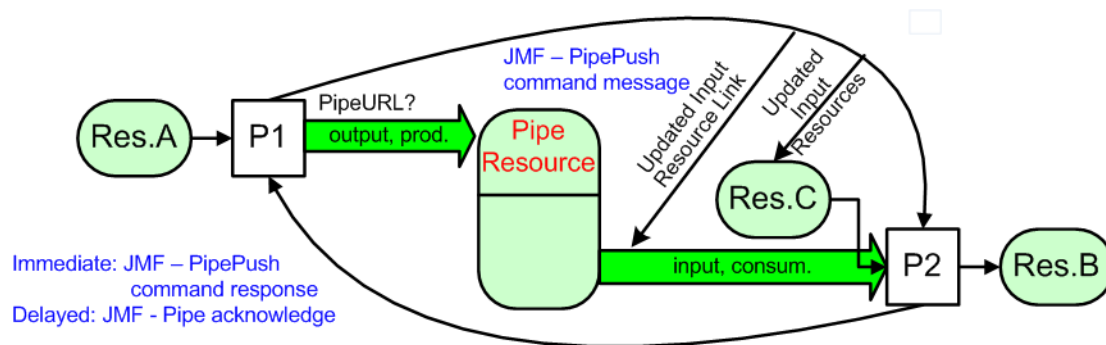


Figure 5-7: Mechanism of a PipePush message

5.9.4 PipePause

Table 5-75: PipePause message

Object Type	Element Name	Description
CommandTypeObj	PipeParams	Describes the pipe resource. The PipeParams element is described in Section 5.9.2, PipePull.
ResponseTypeObj	JobPhase	The status of the responding process. The JobPhase element is defined in Table 5-57 on page 185.

The PipePause message pauses execution of a process that is at the other end of a dynamic pipe. The PipePause command response is equivalent to the PipePull command response described above.

5.10 Queue Support

In JMF, a Controller or Device is assumed to have one input queue that accepts submitted jobs. Controllers which receive submitted jobs MUST 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 MUST be represented by multiple logical devices in JDF. In other words, a Device MUST 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 MUST 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).

5.10.1 Queue Entry ID Generation

Queue entries are accessed using a *QueueEntryID* attribute, which the queue's controller generates when it receives the submitted job, and which is returned in the *SubmitQueueEntry* response. *QueueEntryID* MUST uniquely identify an entry within the scope of one queue. An implementation is free to choose the algorithm that generates *QueueEntryID* values.

5.10.2 Use of QueueFilter in Queue Entry Handling commands

[New in JDF 1.2](#)

Each of the Queue Handling Commands contains an OPTIONAL *QueueFilter* element (See “*QueueFilter* element” on page 214.) which selects the queue entries to be returned and the contents of each. If multiple filter attributes are supplied in the *QueueFilter*, the individual filters are all applied, resulting in a *Queue* element that contains only *QueueEntry* elements that fulfill all conditions defined by *QueueFilter*. If *QueueFilter* is not supplied, the entire *Queue* is returned.

5.11 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.

Table 5-76: Messages for queue entry handling

Message type	Family	Description
AbortQueueEntry Modified in JDF 1.2	CR	The <i>QueueEntry</i> is aborted and remains in the <i>Queue</i> with <i>QueueEntry/@Status = "Aborted"</i> .
<i>HoldQueueEntry</i>	CR	The entry remains in queue but is not executed until a <i>ResumeQueueEntry</i> command is received.
<i>RemoveQueueEntry</i>	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 resources available.
<i>ResubmitQueueEntry</i>	CR	Replaces a queue entry without affecting the entry's parameters. The command is used, for example, for late changes to a submitted JDF.
<i>ResumeQueueEntry</i>	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 <i>SubmitQueueEntry</i> to the queue that represents the controller that originally submitted the job.
<i>SetQueueEntryPosition</i>	CR	Queues a job behind a given position <i>n</i> , where <i>n</i> represents a numerical value. "0" = pole position. Priority is set to the priority of the job at position <i>n</i> .
<i>SetQueueEntryPriority</i>	CR	Sets the priority of a queued job to a new value. This does not apply to jobs that are already running.
<i>SubmitQueueEntry</i>	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 <i>ResumeQueueEntry</i> command 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 "Supported Error Codes in JMF and Notification elements" on page 717.

Table 5-77: Status transitions for QueueEntry handling messages (Section 1 of 2)

Previous Status Message type	Non-existent	Waiting	Held	Running	Suspended	Completed	Aborted
<i>AbortQueueEntry</i>	error(105)	Aborted	Aborted	Aborted	Aborted	error(114)	error(113)

Table 5-77: Status transitions for QueueEntry handling messages (Section 2 of 2)

Previous Status Message type	Non-existent	Waiting	Held	Running	Suspended	Completed	Aborted
HoldQueueEntry	error(105)	Held	error(113)	error(106)	error(106)	error(114)	error(114)
RemoveQueueEntry	error(105)	Removed	Removed	error(106)	error(106)	Removed	Removed
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	error(105)	Waiting	Held	error(107)	error(107)	error(114)	error(114)
ResumeQueueEntry	error(105)	error(113)	Waiting	error(113)	Running	error(114)	error(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.						
SetQueueEntryPosition	error(105)	Waiting	Held	error(107)	error(107)	error(114)	error(114)
SetQueueEntryPriority	error(105)	Waiting	Held	error(107)	error(107)	error(114)	error(114)
SubmitQueueEntry	Waiting, Held, Running	A new QueueEntryID is generated by the queue owner on submission. Therefore these states are not applicable.					
SuspendQueueEntry	error(105)	error(115)	error(115)	Suspended	error(113)	error(114)	error(114)

The following *Status* transition diagram depicts the life cycle of a queue entry.

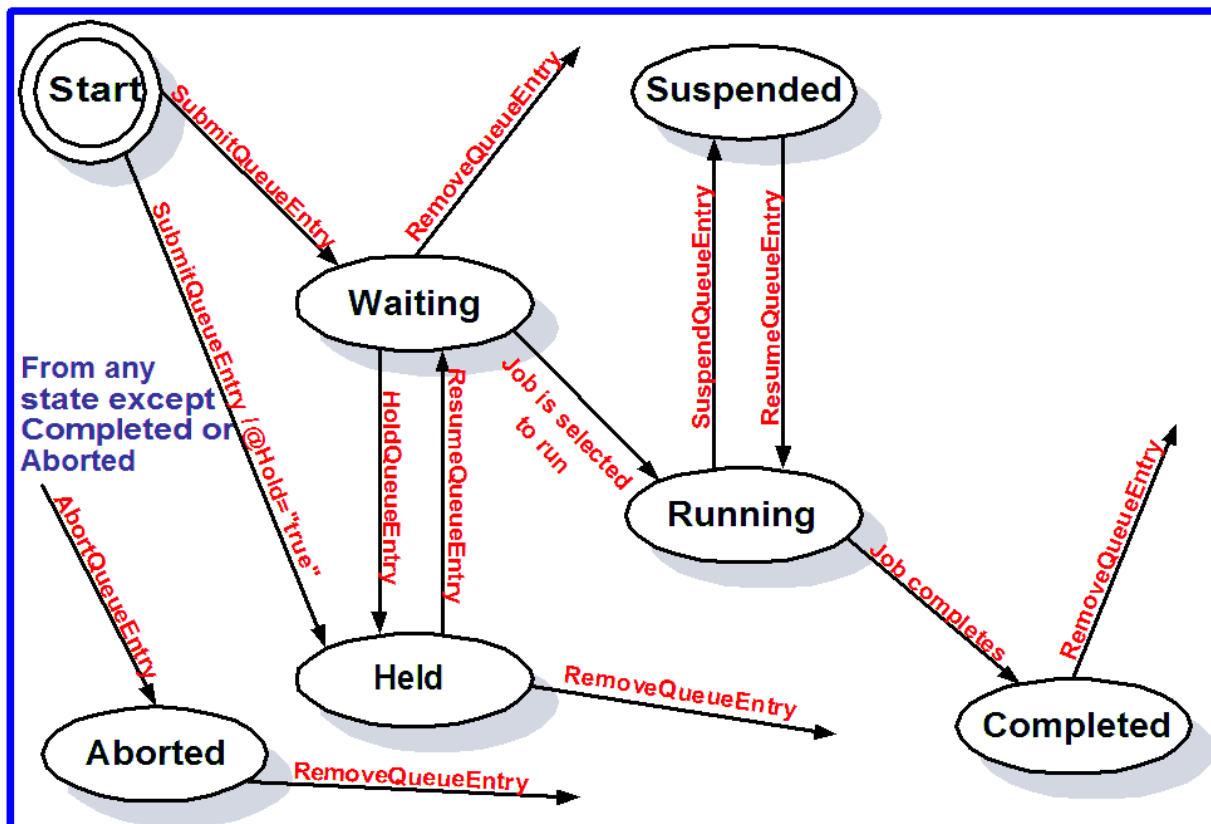


Figure 5-8: JMF QueueEntry Status transition diagram

5.11.1 AbortQueueEntry

Once this command is issued, the entry specified by `QueueEntryDef` is aborted and remains in the `Queue` with `QueueEntry/@Status = "Aborted"`. The `Audits` and `JDF/@Status` of the processing JDF node are to be appropriately set to `"Aborted"` and the JDF node is to be delivered to the URL as specified by `SubmitQueueEntry/@ReturnURL`, `SubmitQueueEntry/@ReturnJMF` or `NodeInfo/@TargetRoute`.

Table 5-78: AbortQueueEntry message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	QueueEntryDef	Defines the queue entry.
	QueueFilter ? New in JDF 1.2	Defines a filter for the returned <code>Queue</code> element in the <code>AbortQueueEntry</code> message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.
For the definition of the elements listed above, see Section 5.13, Elements for Queues.		

The following example demonstrates how an `AbortQueueEntry` command 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`:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="A3 Printer"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00" Version="1.3">
  <Command ID="M009" Type="AbortQueueEntry">
    <QueueEntryDef QueueEntryID="job-0032"/>
    <QueueFilter>
      <QueueEntryDef QueueEntryID="job-0032"/>
    </QueueFilter>
  </Command>
</JMF>
```

The following example shows a possible response to the command example above:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
  TimeStamp="2005-07-25T12:32:48+02:00" Version="1.3">
  <Response ID="M109" Type="AbortQueueEntry" refID="M009">
    <Queue DeviceID="A3 Printer" Status="Running">
      <QueueEntry JobID="job-0032" QueueEntryID="job-0032" Status="Aborted"/>
    </Queue>
  </Response>
</JMF>
```

5.11.2 HoldQueueEntry

The entry specified by `QueueEntryDef` remains in the queue but is never executed. If its `Status` is `"Waiting"`, its `Status` is set to `"Held"`. The `HoldQueueEntry` command has no effect on jobs with a `Status` other than `"Waiting"`. If `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 "Status transitions for `QueueEntry` handling messages" on page 197.

Table 5-79: HoldQueueEntry message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	QueueEntryDef	Defines the queue entry.
	QueueFilter ? New in JDF 1.2	Defines a filter for the returned <code>Queue</code> element in the <code>HoldQueueEntry</code> message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.
For the definition of the elements listed above, see Section 5.13, Elements for Queues.		

5.11.3 RemoveQueueEntry

This command causes the entry specified by `QueueEntryDef` 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 "Status transitions for `QueueEntry` handling messages" on page 197.

Table 5-80: RemoveQueueEntry message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	QueueEntryDef	Defines the queue entry.
	QueueFilter ? New in JDF 1.2	Defines a filter for the returned <code>Queue</code> element in the <code>RemoveQueueEntry</code> message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.
For the definition of the elements listed above see, Section 5.13, Elements for Queues.		

5.11.4 RequestQueueEntry

[New in JDF 1.2](#)

Table 5-81: RequestQueueEntry message

Object Type	Element Name	Description
CommandTypeObj	RequestQueueEntryParams	Defines the specifics for the requested job.
ResponseTypeObj	—	The controller does not send any immediate response. Any job submission is handled using hot folders or the standard <code>SubmitQueueEntry</code> message.
For the definition of the elements listed above see, Section 5.13, Elements for Queues.		

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 dispatcher and not to the queue, as is the case with most other queue handling commands.

— Element: RequestQueueEntryParams

Table 5-82: RequestQueueEntryParams element (Section 1 of 2)

Name	Data Type	Description
<i>JobID</i> ?	string	<i>JobID</i> of the requested <code>QueueEntry</code> .
<i>JobPartID</i> ?	string	<i>JobPartID</i> of the requested <code>QueueEntry</code> .
<i>QueueURL</i>	URL	URL of the <code>Queue</code> controller that is requesting the <code>QueueEntry</code> and will accept <code>Queue</code> manipulation messages.

Table 5-82: RequestQueueEntryParams element (Section 2 of 2)

Name	Data Type	Description
SubmitPolicy ? New in JDF 1.3	enumeration	Defines the requested policy for submitting the node: <i>Standard</i> : All linked resources MUST have a <code>Resource/@Status</code> as defined by <code>ResourceLink/@MinStatus</code> . <i>Late</i> : All linked resources MUST have a <code>Resource/@Status</code> as defined by <code>ResourceLink/@MinLateStatus</code> . <i>Force</i> : The node MUST be submitted regardless of the values of linked <code>Resource/@Status</code> . If not specified, the submission policy is dependent on the Controller implementation. <i>SubmitPolicy</i> allows a device to request a node that would otherwise not be submitted by the controller due to missing resources.
Part *	element	Partition parts of the requested <code>QueueEntry</code> .
Queue ?	element	Representation of the current status of the device's <code>Queue</code> .

5.11.5 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 specified in `ResubmissionParams/@URL`. The `QueueEntry/@Status` MUST be `"Waiting"` or `"Held"`. Job resubmission does not affect other queue parameters as specified. For example, resubmission does not affect queue ordering. For details, see "Status transitions for `QueueEntry` handling messages" on page 197.

Table 5-83: ResubmitQueueEntry message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	<code>ResubmissionParams</code>	Defines the job resubmission.
	QueueFilter ? New in JDF 1.2	Defines a filter for the returned <code>Queue</code> element in the <code>ResubmitQueueEntry</code> message.
ResponseTypeObj	<code>Queue</code>	Describes the state of the queue after the command has been executed.
For the definition of the <code>Queue</code> element, see Section Section 5.13, Elements for Queues.		

— Element: ResubmissionParams

Table 5-84: ResubmissionParams element

Name	Data Type	Description
<code>QueueEntryID</code>	string	ID of the queue entry to be replaced.
<code>URL</code>	URL	Location of the JDF to be submitted. It MAY be a URL with a "cid" scheme in the case of MIME Multipart/Related.

5.11.6 ResumeQueueEntry

The hold status of the queue entry specified by `QueueEntryDef` is removed. A `QueueEntry` with `Status = "Held"` gets a `Status` of `"Waiting"`. A `QueueEntry` with `Status = "Suspended"` gets a `Status` of `"Running"`. If `GangPolicy` is other than `"NoGang"`, a resumed `QueueEntry` joins its respective gang. For details, see "Status transitions for `QueueEntry` handling messages" on page 197.

Table 5-85: ResumeQueueEntry message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	QueueEntryDef	Defines the queue entry.
	QueueFilter ? New in JDF 1.2	Defines a filter for the returned Queue element in the ResumeQueueEntry message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.
For the definition of the elements listed above, see Section 5.13, Elements for Queues.		

5.11.7 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 5-86: 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	-	
For the definition of the elements listed above, see Section 5.13, Elements for Queues.		

— Element: 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 5-87: ReturnQueueEntryParams element

Name	Data Type	Description
<i>Aborted</i> ?	IDREFS	ID of the JDF nodes that have been executed and aborted or failed test running. If <i>Aborted</i> and <i>Completed</i> are empty, no executable node was found.
<i>Completed</i> ?	IDREFS	ID of the JDF nodes that have been executed and completed or succeeded in test run.
<i>Priority</i> ?	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.
<i>URL</i>	URL	Location of the JDF to be returned. Note that the <i>URL</i> SHOULD be queried with a SubmissionMethods query to determine whether MIME Multipart/Related is supported

5.11.8 SetQueueEntryPosition

The position of the queue entry is modified. The QueueEntryPosParams element provides the parameters. The position of a queue entry MUST NOT be modified unless *Status* = "Waiting" or *Status* = "Held". For details, see "Status transitions for QueueEntry handling messages" on page 197.

Table 5-88: SetQueueEntry message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	QueueEntryPosParams	Defines the queue entry.
	QueueFilter ? New in JDF 1.2	Defines a filter for the returned Queue element in the SetQueueEntryPosition message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.
For the definition of the Queue element, see Section 5.13, Elements for Queues.		

— Element: 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 5-89: QueueEntryPosParams element

Name	Data Type	Description
<i>NextQueueEntryID</i> ?	string	ID of the queue entry that is to be positioned directly behind the entry. Exactly one of <i>NextQueueEntryID</i> , <i>PrevQueueEntryID</i> or <i>Position</i> MUST be specified.
<i>QueueEntryID</i>	string	ID of a queue entry.
<i>PrevQueueEntryID</i> ?	string	ID of the queue entry that is to be positioned directly in front of the entry. Exactly one of <i>NextQueueEntryID</i> , <i>PrevQueueEntryID</i> or <i>Position</i> MUST be specified.
<i>Position</i> ?	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 <i>Position</i> . Exactly one of <i>NextQueueEntryID</i> , <i>PrevQueueEntryID</i> or <i>Position</i> MUST be specified.

5.11.9 SetQueueEntryPriority

The priority of the queue entry is modified. The QueueEntryPriParams element provides the parameters. For details, see "Status transitions for QueueEntry handling messages" on page 197.

Table 5-90: SetQueueEntryPriority message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	QueueEntryPriParams	Defines the queue entry.
	QueueFilter ? New in JDF 1.2	Defines a filter for the returned Queue element in the SetQueueEntryPriority message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.
For the definition of the Queue element, see Section 5.13, Elements for Queues.		

— Element: QueueEntryPriParams

QueueEntryID, described in the table below, specifies the queue entry that has its priority modified.

Table 5-91: QueueEntryPriParams element

Name	Data Type	Description
<i>Priority</i>	integer	Number from 0 to 100, where "0" = lowest priority and "100" = maximum priority.
<i>QueueEntryID</i>	string	ID of a queue entry.

5.11.10 SubmitQueueEntry

SubmitQueueEntry submits a job to a queue of a Device or Controller. QueueSubmissionParams provides the parameters of the submission.

Table 5-92: SubmitQueueEntry message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	QueueSubmissionParams	Defines the job submission.
	QueueFilter ? New in JDF 1.2	Defines a filter for the returned Queue element in the SubmitQueueEntry message.
ResponseTypeObj	QueueEntry ? Modified in JDF 1.2	Provides the queue entry of the submitted job. QueueEntry MUST be specified if the submission was successful and MUST be omitted in case the submission was rejected.
	Queue	Describes the state of the queue after the command has been executed.

Definition of the QueueEntry and Queue elements, see Section 5.13, Elements for Queues.

— Element: 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 is to be sent after the job is completed or aborted.

Table 5-93: QueueSubmissionParams element (Section 1 of 2)

Name	Data Type	Description
<i>GangName</i> ? New in JDF 1.3	NMTOKEN	Name of the Gang for the job. If <i>GangName</i> is specified, the QueueEntry SHOULD be executed together with other QueueEntry elements that share a common value of <i>GangName</i> . If <i>GangName</i> is not known, the receiving Device MAY either return an error 131 or create the gang with <i>GangName</i> on the fly.
<i>GangPolicy</i> ? New in JDF 1.3	enumeration	Ganging policy for the QueueEntry. Allowed values are: <i>Gang</i> - The Job MUST be ganged in the gang that is specified by <i>GangName</i> or MUST be calculated from other properties of the submitted job. A gang job that MAY contain this submitted QueueEntry MAY be queued. <i>GangAndForce</i> - The Job MUST be ganged in the gang that is specified by <i>GangName</i> or MUST be calculated from other properties of the submitted job. A gang job that MUST contain this submitted QueueEntry MUST be queued. <i>NoGang</i> - The Job MUST NOT be ganged and <i>GangName</i> MUST be ignored. The job MUST be queued individually.

Table 5-93: QueueSubmissionParams element (Section 2 of 2)

Name	Data Type	Description
<i>Hold</i> = "false"	boolean	If <i>true</i> , the entry is submitted as with <i>QueueEntry/@Status</i> ="Held". If a JDF node is not executable due to resources being unavailable, it MUST be submitted with <i>Hold</i> = "true". If a <i>QueueEntry</i> is submitted with <i>Hold</i> = "true" and <i>GangPolicy</i> is other than "NoGang", the <i>QueueEntry</i> retains its respective gang data but does not influence execution of other jobs that are in the gang.
<i>NextQueueEntryID</i> ?	string	ID of the queue entry that is to be positioned directly behind the entry. At most one of <i>NextQueueEntryID</i> , <i>PrevQueueEntryID</i> or <i>Priority</i> MUST be specified.
<i>PrevQueueEntryID</i> ?	string	ID of the queue entry that is to be positioned directly in front of the entry. At most one of <i>NextQueueEntryID</i> , <i>PrevQueueEntryID</i> or <i>Priority</i> MUST be specified.
<i>Priority</i> = "1"	integer	Number from 0 to 100, where "0" = lowest priority and "100" = maximum priority. Exactly one of <i>NextQueueEntryID</i> , <i>PrevQueueEntryID</i> or <i>Priority</i> MUST be specified. Note that <i>QueueSubmissionParams/@Priority</i> is not the same as <i>JDF/NodeInfo/@Priority</i> . <i>QueueSubmissionParams/@Priority</i> specifies the priority in the context of the device queue whereas <i>JDF/NodeInfo/@Priority</i> specifies the priority of the task in general. <i>QueueSubmissionParams/@Priority</i> MAY be modified due to additional scheduling information, (e.g., <i>JDF/NodeInfo/@FirstStart</i>).
<i>refID</i> ? New in JDF 1.2	NMTOKEN	Copy of the <i>ID</i> attribute of the initiating <i>RequestQueueEntry</i> message.
<i>ReturnJMF</i> ? New in JDF 1.2	URL	Address of a JMF queue where a <i>ReturnQueueEntry</i> message is to be sent when the <i>QueueEntry</i> is completed or aborted. Note that the <i>ReturnJMF</i> queue SHOULD be queried with a <i>SubmissionMethods</i> query to determine whether MIME Multipart/Related is supported by the return queue. <i>ReturnJMF</i> MUST NOT be specified if <i>ReturnURL</i> is present.
<i>ReturnURL</i> ? Modified in JDF 1.2	URL	URL where the JDF file is to be written when the <i>QueueEntry</i> is completed or aborted. A Controller MUST write only a JDF file to the URL and MUST NOT write a MIME Multipart package to the URL. <i>ReturnURL</i> MUST take precedence when <i>NodeInfo/@TargetRoute</i> is specified in the submitted JDF. Note: A Controller MUST NOT return a JDF file or MIME Multipart/Related file by performing a <i>SubmitQueueEntry</i> or <i>ReturnQueueEntry</i> to the <i>ReturnURL</i> URL. The controller specified by <i>ReturnURL</i> MUST NOT accept JMF messages. See instead <i>ReturnJMF</i> . <i>ReturnURL</i> MUST NOT be specified if <i>ReturnJMF</i> is present.
<i>URL</i> 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.
<i>WatchURL</i> ? Modified in JDF 1.2	URL	URL of the controller that is to be notified when the status of the <i>QueueEntry</i> or the underlying job changes. Specifying <i>WatchURL</i> is equivalent to sending a subscription for an <i>Events</i> message with <i>SignalTypes</i> = "ALL".
<i>Disposition</i> ? New in JDF 1.2	element	Definition how long the <i>QueueEntry</i> SHOULD be retained in the queue. If not specified, the <i>QueueEntry</i> MAY be removed from the queue immediately after process completion of the <i>QueueEntry</i> .

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:

```
<Command ID="M1" Type="SubmitQueueEntry">
  <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.

```
<Command ID="M2" Type="SubmitQueueEntry">
  <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 "JDF Packaging" on page 680 for details of MIME Multipart/Related packaging.

5.11.11 SuspendQueueEntry

[New in JDF 1.2](#)

Table 5-94: SuspendQueueEntry message

Object Type	Element Name	Description
CommandTypeObj	QueueEntryDef	Defines the queue entry.
	QueueFilter ?	Defines a filter for the returned <i>Queue</i> element in the <i>SuspendQueueEntry</i> message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed. See “Elements for Queues” on page 211. for the definition of the elements listed above. The entry specified by <i>QueueEntryDef</i> remains in the queue but moved into the <i>Suspended</i> state.
For the definition of the elements listed above, see Section 5.13, Elements for Queues.		

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 *SuspendQueueEntry* command has no effect on jobs with a *Status* other than “*Running*”. For details, see “Status transitions for *QueueEntry* handling messages” on page 197.

5.12 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 MUST 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 applied to all queues that are controlled by the controller that received the message. The following individual messages are defined:

Table 5-95: Messages for global handling of queues (Section 1 of 2)

Message type	Family	Description
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.

Table 5-95: Messages for global handling of queues (Section 2 of 2)

Message type	Family	Description
OpenQueue	CR	The queue is opened. Jobs are to be accepted.
QueueEntryStatus <i>Deprecated in JDF 1.2</i>	QRS	Returns a QueueEntry element.
QueueStatus	QRS	Returns the Queue elements that describe a queue or set of queues.
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 5-96: Definition of the Queue Status Attribute values

Closed	Held	Queue Full	Processor Full	Status
Yes	Yes	Any	Any	<i>Blocked</i>
Yes	No	Any	Any	<i>Closed</i>
No	Yes	Any	Any	<i>Held</i>
No	No	Any	No	<i>Waiting</i>
No	No	No	Yes	<i>Running</i>
No	No	Yes	Yes	<i>Full</i>

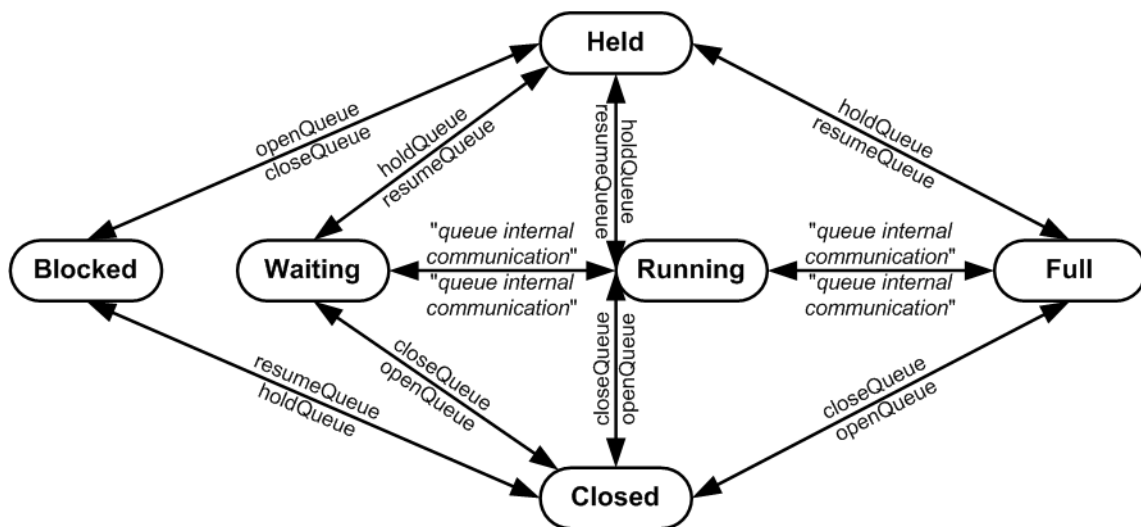


Figure 5-9: Effects of the global queue messages on the queue Status

5.12.1 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 prior entries can be executed.

Table 5-97: CloseQueue message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	QueueFilter ? New in JDF 1.2	Defines a filter for the returned Queue element in the CloseQueue message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.
For the definition of the Queue element, see Section 5.13, Elements for Queues.		

5.12.2 FlushQueue

5.12.2.1 FlushQueue Command

Table 5-98: FlushQueue command message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	QueueFilter ? New in JDF 1.2	Defines a filter for the returned Queue element in the FlushQueue message.
	FlushQueueParams ? New in JDF 1.2	Defines the QueueEntry elements to be removed. If not specified then only pending (<i>Status</i> = "Waiting" and <i>Status</i> = "Held" queue entries are removed.)
ResponseTypeObj Modified in JDF 1.2	Queue	Describes the state of the queue after the command has been executed.
	FlushQueueInfo ? New in JDF 1.2	Defines the QueueEntry elements that were removed.
For the definition of the Queue element, see Section 5.13, Elements for Queues.		

FlushQueue 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.

— Element: FlushQueueParams

[New in JDF 1.2](#)

Table 5-99: 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.

5.12.2.2 FlushQueue Query

Table 5-100: FlushQueue query message

Object Type	Element Name	Description
QueryTypeObj	QueueFilter ?	Defines a filter for the returned Queue element in the FlushQueue message.
ResponseTypeObj	Queue	Describes the state of the queue after the elements have been flushed.
	FlushQueueInfo ? New in JDF 1.2	Defines the QueueEntry elements that were removed.
For the definition of the Queue element, see Section 5.13, Elements for Queues.		

When used as a Signal or Query, FlushQueue 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.

— Element: FlushQueueInfo

[New in JDF 1.2](#)

Table 5-101: FlushQueueInfo element

Name	Data Type	Description
QueueFilter	element	Defines a QueueFilter that specifies the QueueEntry elements that were removed.

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.

5.12.3 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 MUST be suspended individually using the SuspendQueueEntry command.

Table 5-102: HoldQueue message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	QueueFilter ? New in JDF 1.2	Defines a filter for the returned Queue element in the HoldQueue message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.
For the definition of the Queue element, see Section 5.13, Elements for Queues.		

5.12.4 OpenQueue

The queue is opened and new queue entries can be accepted by the queue. A held queue remains held. The OpenQueue command is the opposite of a CloseQueue command.

Table 5-103: OpenQueue message (Section 1 of 2)

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	QueueFilter ? New in JDF 1.2	Defines a filter for the returned Queue element in the OpenQueue message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.

Table 5-103: OpenQueue message (Section 2 of 2)

Object Type	Element Name	Description
For the definition of the <code>Queue</code> element, see Section 5.13, Elements for Queues.		

5.12.5 QueueEntryStatus

[Deprecated in JDF 1.2](#)

In JDF 1.2 and beyond, use `QueueStatus` with an appropriate `QueueFilter` instead of `QueueEntryStatus`. See "QueueEntryStatus" on page 821 for details of this deprecated JMF element.

5.12.6 QueueStatus

Returns a queue description.

Table 5-104: QueueStatus message

Object Type	Element Name	Description
<code>QueryTypeObj</code> Modified in JDF 1.2	<code>QueueFilter ?</code> New in JDF 1.2	Defines a filter for the <code>QueueStatus</code> message.
<code>ResponseTypeObj</code>	<code>Queue</code>	Describes the status of the queue.
For the definition of the <code>Queue</code> element, see Section 5.13, Elements for Queues.		

5.12.7 ResumeQueue

The queue is activated and queue entries can be executed. The `ResumeQueue` command is the opposite of a `HoldQueue` command.

Table 5-105: ResumeQueue message

Object Type	Element Name	Description
<code>CommandTypeObj</code> Modified in JDF 1.2	<code>QueueFilter ?</code> New in JDF 1.2	Defines a filter for the <code>ResumeQueue</code> message.
<code>ResponseTypeObj</code>	<code>Queue</code>	Describes the state of the queue after the command has been executed.
For the definition of the <code>Queue</code> element, see Section 5.13, Elements for Queues.		

5.12.8 SubmissionMethods

Table 5-106: SubmissionMethods message

Object Type	Element Name	Description
<code>QueryTypeObj</code>	—	—
<code>ResponseTypeObj</code>	<code>SubmissionMethods ?</code>	Describes the submission methods supported by the queue.

The `SubmissionMethods` message returns the submission methods that are supported by a queue controller.

— Element: SubmissionMethods

The response element MAY contain multiple attributes, as defined below. If an attribute is not specified, the corresponding submission method is not supported.

Table 5-107: SubmissionMethods element (Section 1 of 2)

Name	Data Type	Description
<code>File ?</code> Deprecated in JDF 1.2	boolean	Can retrieve a JDF from a File specified in the URL In JDF 1.2 and beyond, include " <code>file</code> " in <code>URLSchemes</code> .
<code>HotFolder ?</code>	URL	URL specification of a hot folder location.

Table 5-107: SubmissionMethods element (Section 2 of 2)

Name	Data Type	Description
HttpGet ? Deprecated in JDF 1.2	boolean	Can retrieve a JDF via HTTP <code>get</code> commands. In JDF 1.2 and beyond, include <code>"http"</code> in <code>URLSchemes</code> .
Packaging ? New in JDF 1.2	enumerations	List of packaging methods supported. If not specified, the controller does not support receiving packaged messages and MUST retrieve JDF files using a URL with a scheme other than <code>"cid"</code> . Allowed values are: <i>MIME</i> – Accepts MIME Multipart/Related packaging of JMF, JDF and digital assets. This packaging is intended for bidirectional JMF messaging. When used for unidirectional JMF messaging, the package MUST be saved as a single file with a <code>".mjm"</code> extension.
MIME ? Deprecated in JDF 1.2	boolean	Accepts MIME Multipart/Related submission messages via a message post. In JDF 1.2 and beyond, use <code>Packaging = "MIME"</code> .
URLSchemes ? New in JDF 1.2	NMTO-KENS	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: <i>file</i> – The file scheme according to [RFC1738] and [RFC3986]. <i>ftp</i> – FTP (File Transfer Protocol) <i>http</i> – HTTP (Hypertext Transport Protocol) <i>https</i> – HTTPS (Hypertext Transport Protocol — Secure)

The following is an example of a response to a `SubmissionMethods` query:

```
<Response ID="M1" Type="SubmissionMethods" refID="Q1">
  <SubmissionMethods HotFolder="file://MyDevice/HotFolder" Packaging="MIME"
    URLSchemes="http file ftp"/>
</Response>
```

5.13 Elements for Queues

In this section elements used by queue-handling commands are defined.

5.13.1 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 5-108: Queue element (Section 1 of 2)

Name	Data Type	Description
<code>DeviceID</code>	string	Identifies the device that is represented by the queue.
QueueSize ? New in JDF 1.2	integer	The maximum number of <code>QueueEntry</code> elements that can be in the Queue. Note: <code>QueueEntry</code> [<code>@Status = "Completed"</code> or <code>@Status = "Aborted"</code>] elements MUST NOT count towards determining <code>Queue/@Status</code> based on the number of <code>QueueEntry</code> elements versus the <code>QueueSize</code> .

Table 5-108: Queue element (Section 2 of 2)

Name	Data Type	Description
<i>Status</i>	enumeration	Status of the queue. Possible values are: <i>Blocked</i> – Queue is completely inactive. Entries MUST NOT be added and no entries are executed. The queue is closed and held. The queue requires an interaction like <i>OpenQueue</i> or <i>ResumeQueue</i> to reactivate it. <i>Closed</i> – Queue entries that are in the queue are executed, but new entries MUST NOT be submitted. The lock MUST be removed explicitly by the <i>OpenQueue</i> command. <i>Full</i> – Queue entries that are in the queue are executed but new entries MUST NOT be submitted. The lock is removed by the queue controller as soon as it is able to do so. <i>Running</i> – A process is executing. Entries can be submitted and will be executed when they reach their turn in the queue. <i>Waiting</i> – Queue accepts new entries and has free resources to immediately commence processing. <i>Held</i> – Entries can be submitted but will not be executed until the queue is resumed by the <i>ResumeQueue</i> command.
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	element	QueueEntry elements (see Table 5-109, “QueueEntry element,” on page 213, below). The entries are 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 5-109, “QueueEntry element,” on page 213. A QueueEntry is not automatically deleted when executed or aborted, but rather it remains in the Queue and its <i>Status</i> is changed to <i>Completed</i> or <i>Aborted</i> accordingly. QueueEntry [<i>@Status</i> = “ <i>Completed</i> ” or <i>@Status</i> = “ <i>Aborted</i> ”] elements MUST NOT count towards determining Queue/@Status based on the number of QueueEntry elements versus the <i>QueueSize</i> .

Example of a **Queue** message element:

```
<Queue DeviceID="Q12345" Status="Running">
  <QueueEntry JobID="111" JobPartID="0" Priority="1" QueueEntryID="111-0"
    Status="Completed"/>
  <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"/>
</Queue>
```

5.13.2 QueueEntry

Table 5-109: QueueEntry element (Section 1 of 2)

Name	Data Type	Description
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. <i>GangName</i> MUST be specified, if the QueueEntry is member of a gang job.
GangPolicy ? New in JDF 1.3	enumeration	Ganging policy for the QueueEntry. Allowed values are defined in: <i>SubmitQueueEntry/@GangPolicy</i>
JobID ? Modified in JDF 1.1	string	The Job ID of the JDF process.
JobPartID ?	string	The JobPartID of the JDF process.
<i>Priority = "1"</i>	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 is generated by the queue owner.
StartTime ? New in JDF 1.1	dateTime	Time when the job has been started.
Status Modified in JDF 1.3	enumeration	Status of the individual entry. Possible values are: <i>Running</i> – The queue entry is running on the device. A QueueEntry is <i>Running</i> when <i>JDF/@Status</i> is one of <i>Setup</i> , <i>InProgress</i> or <i>Cleanup</i> . <i>Waiting</i> – The queue entry is waiting and will be executed when resources are available. <i>Held</i> – The queue entry is held and MUST NOT execute until resumed. A held QueueEntry with <i>GangPolicy</i> other than "NoGang" does not interact with its respective gang. <i>Removed</i> – 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. <i>Suspended</i> – The queue entry was running and has been held. It will not continue to execute until resumed. A QueueEntry is <i>Suspended</i> when <i>JDF/@Status</i> is one of <i>Stopped</i> or <i>Suspended</i> . New in JDF 1.2 <i>PendingReturn</i> : 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 <i>Completed</i> – Indicates that the queue entry has been executed correctly, and is finished. New in JDF 1.2 <i>Aborted</i> – Indicates that the process executing the node has been aborted, which means that execution will not be resumed again. New in JDF 1.2
SubmissionTime ?	dateTime	Time when the entry was submitted to the queue.

Table 5-109: QueueEntry element (Section 2 of 2)

Name	Data Type	Description
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 s MAY be associated with a QueueEntry and used for display purposes. Preview/@PreviewUsage SHOULD be <i>ThumbNail</i> or <i>Viewable</i> .

5.13.3 QueueEntryDef

The element specifies a queue entry and is used to refer to a certain queue entry.

Table 5-110: QueueEntryDef element

Name	Data Type	Description
QueueEntryID	string	ID of the queue entry. The ID is generated by the queue owner.

5.13.4 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 5-111: QueueFilter element (Section 1 of 2)

Name	Data Type	Description
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 .
MaxEntries ?	integer	Maximum number of QueueEntry elements to provide in the Queue element. If not specified, fill in all matching QueueEntry elements.
OlderThan ?	dateTime	Only QueueEntry elements with a <i>SubmissionTime</i> older than or equal to this dateTime are provided in the Queue element or removed by the FlushQueue message. If not specified, there is no dateTime lower bound on candidates.
NewerThan ?	dateTime	Only QueueEntry elements with a <i>SubmissionTime</i> newer than or equal to this dateTime are provided in the Queue element or removed by the FlushQueue message. If not specified, there is no dateTime upper bound on candidates.

Table 5-111: QueueFilter element (Section 2 of 2)

Name	Data Type	Description
<i>QueueEntryDetails</i> = "Brief"	enumeration	Refines the level of provided information about the Queue. Possible values are: <i>None</i> – Do not fill in the <i>QueueEntry</i> elements into the Queue. <i>Brief</i> – Provide all available <i>QueueEntry</i> information except for the associated <i>JobPhase</i> element. <i>JobPhase</i> – Provide all available <i>QueueEntry</i> information including the associated <i>JobPhase</i> elements. <i>JDF</i> – Provide all available <i>QueueEntry</i> information including the associated <i>JobPhase</i> element and the associated JDF element in the <i>JobPhase</i> element.
<i>StatusList</i> ?	enumerations	Only <i>QueueEntry</i> elements with a <i>Status</i> matching one of the entries in <i>StatusList</i> are considered. For a list of allowed values, see Table 5-109, "QueueEntry element," on page 213.
<i>QueueEntryDef</i> *	element	Defines an explicit list of queue entries. If not specified, all entries in the Queue are considered.
<i>Device</i> *	element	Devices that returned queue entries are targeted to. <i>QueueEntry/@DeviceID</i> MUST match <i>QueueFilter/Device/@DeviceID</i> for the <i>QueueEntry</i> to be returned in the queue. If not specified, all entries in the Queue are considered.

5.14 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*.

Table 5-112: Messages for Gang Jobs

Message type	Family	Description
<i>ForceGang</i> New in JDF 1.3	CR	A gang is forced to execute.
<i>GangStatus</i> New in JDF 1.3	CR	The status of a gang is queried.

5.14.1 ForceGang

[New in JDF 1.3](#)

The *ForceGang* message forces all *QueueEntry* [*@Status* = "waiting"] elements that belong to a gang to be executed, even though the device dependent queue entry collecting algorithm might not be completed.

Table 5-113: Contents of the ForceGang Command message

Object Type	Element Name	Description
<i>CommandTypeObj</i>	<i>GangCmdFilter</i>	Defines the gang(s) to be force executed.

— Element: *GangCmdFilter*

Table 5-114: *GangCmdFilter* element

Object Type	Element Name	Description
<i>GangNames</i> ?	NMTOKENS	<i>GangName</i> of the gang(s) being queried.

5.14.2 GangStatus

[New in JDF 1.3](#)

GangStatus returns a description of the gang(s). Details are specified in GangInfo element.

Table 5-115: 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).

— Element: GangQuFilter

Table 5-116: GangQuFilter element

Name	Data Type	Description
<i>GangNames</i> ?	NMTOKENS	<i>GangName</i> of the gang(s) being queried.

— Element: 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 5-117: GangInfo element

Name	Data Type	Description
<i>GangName</i>	NMTOKEN	Name of the gang.

5.15 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 3.11, JDF Extensibility. Note, the generic content of Section 3.1.1, Generic Contents of JDF 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 3.11, JDF Extensibility. An example is provided:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Circus"
  Timestamp="2005-07-25T12:32:48+02:00" Version="1.3" xmlns:Circus="Circus Schema URI">
  <Query ID="Q1" Type="Circus:IsClownHappy">
    <Circus:ClownParams Gender="male"/>
  </Query>
</JMF>
```

The response 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" Version="1.3" xmlns:Circus="Circus Schema URI">
  <Response ID="M1" Type="Circus:IsClownHappy" refID="Q1">
    <Circus:Clown happy="true" name="Joe"/>
    <Circus:Clown happy="false" name="John"/>
  </Response>
```

5.15.1 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:

```
<JMF xmlns="http://
www.CIP4.org/JDFSchema_1_1" SenderID="IFRA" TimeStamp="2003-07-25T12:32:48+02:00"
Version="1.3" xmlns:IFRA="IfraTrack URI">
  <Query ID="Q1" Type="IFRA:IMF">
    <imf:IMF xmlns:imf="IfraTrack URI">
      Whatever you want (might be multiple top level elements)
    </imf:IMF>
  </Query>
</JMF>
```

The legal response would be:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="IFRA"
TimeStamp="2003-07-25T12:32:48+02:00" Version="1.3" xmlns:IFRA="IfraTrack URI">
  <Response ID="M1" Type="IFRA:IMF" refID="Q1">
    <imf:IMF xmlns:imf="IfraTrack URI">
      The appropriate IFRA response(s)
    </imf:IMF>
  </Response>
</JMF>
```

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 examples above the default namespace associated with the JMF query and response elements has been overwritten by the Ifra namespace.



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/\(StructuredSearchAll\)?OpenAgent&IFRATRACK](http://www.ifra.com/WebSite/news.nsf/(StructuredSearchAll)?OpenAgent&IFRATRACK)

Chapter 6 Processes


The following chapter describes the processes that are defined in detail for JDF.

6.1 Process Template

Processes are defined by their input and output resources, therefore, 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 the following table for all processes defined in this chapter.

Note: for the Input Resource Template and Output Resource Template tables below:

- 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 1.3.4, 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



The JDF Cookbook

Chapter 6 and Chapter 7 are "the list of ingredients" in the JDF "cookbook." The following processes and resources are fairly exhaustive. You can choose to use only what fits your workflow.

Table 6-1: Template for input resources (Section 1 of 2)

Name	Description
<i>Resource-Name</i> <i>Cardinality</i>	<i>Information about the input resource.</i> 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.
<i>Resource-Name (someValue)</i> <i>Cardinality</i>	<i>Information about the input resource</i> Note: <i>ProcessUsage</i> attribute of the specified resource MUST match the “ <i>someValue</i> ” value specified in the parentheses. When a process potentially contains multiple input resources of the same type, the value of <i>ProcessUsage</i> 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.
Implementation *	Abstract resource that is a placeholder for any implementation resource (examples are Employee or Device) that is associated with processing this node.
MiscConsumable * New in JDF 1.3	Miscellaneous consumable resources.
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 (Section 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 “Preflight” on page 235.
Preview * New in JDF 1.1A	Any number of previews MAY be associated with a process and used for display purposes. Preview/@PreviewUsage SHOULD be <i>ThumbNail</i> or <i>Viewable</i> .
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
<i>Resource-Name</i> <i>Cardinality</i>	<i>Information about the output resource.</i>
<i>Resource-Name (someValue)</i> <i>Cardinality</i>	<i>Information about the output resource</i> Note: <i>ProcessUsage</i> attribute of the specified resource MUST match the “ <i>someValue</i> ” value specified in the parentheses. When a process potentially contains multiple output resources of the same type, the value of <i>ProcessUsage</i> 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 **Approval** 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 proofed. The input will most often be a resource of class <i>Handling</i> or <i>Quantity</i> . 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 (Section 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 <i>Status</i> of <i>Draft</i> to <i>Available</i> (see also Section 4.3.5.2, Formal Iterative Processing).

Table 6-4: Approval – Output resources (Section 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 <i>Rejected</i> .

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 MUST 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 physical resources to be buffered. These MAY be any resource whose class is <i>Consumable</i> , <i>Handling</i> or <i>Quantity</i> .

Table 6-6: Buffer – Output resources

Name	Description
Resource	The same resource after buffering. The resource MUST have a <i>Class</i> of <i>Consumable</i> , <i>Handling</i> or <i>Quantity</i> .

6.2.3 Combine

The **Combine** process is used to combine multiple physical resources 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 MUST be equal. The ordering of the input **ResourceLink** elements MUST 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 physical resource 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 physical resource or a Parameter resource that is delivered electronically. 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	Any resources delivered from a location. These MUST be physical resources.

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
Resource *	Resources that are REQUIRED to create the output Resource.
ManualLaborParams	Details on the ManualLabor process.

Table 6-12: ManualLabor – Output resources

Name	Description
Resource ? Modified in JDF 1.3	The resource that was created by manual work. In general these will be Component resources, but handling resources MAY also be processed manually. If no output Resource is specified, the ManualLabor process describes incidental work.

6.2.6 Ordering

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 6-13: 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 6-14: Ordering – Output resources

Name	Description
Resource + Modified in JDF 1.1	All kinds of physical resources can be ordered.

6.2.7 Packing

[Deprecated in JDF 1.1](#)

See "Packing" on page 821 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 6-15: QualityControl – Input resources

Name	Description
Resource	The resource to be quality controlled. In general this will be a Component resource.
QualityControlParams	Detailed definition of the QualityControl process.

Table 6-16: QualityControl – Output resources

Name	Description
QualityControlResult	Results of the process, (e.g., measurement statistics).
Resource	The resource after QualityControl is applied. Note that this resource will generally be partitioned by <i>Condition</i> 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 6-17: 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 6-18: 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 MUST be equal.

Table 6-19: Split – Input resources

Name	Description
Resource	The resource to be split.

Table 6-20: 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 6-21: Verification – Input resources

Name	Description
DBSchema ?	Schema description of the cross-reference database.
DBSelection ?	Database link that defines the database that contains cross-reference data.
IdentificationField *	Identifies the position and type of data for an automated, OCR-based verification process.
Resource ? New in JDF 1.2	The resource to be verified. The input will most often be a resource with <i>Class</i> = "Quantity", e.g., Component or <i>Class</i> = "Parameter", e.g. RunList .
VerificationParams	Controls the verification requirements.

Table 6-22: Verification – Output resources

Name	Description
ApprovalSuccess ?	Signature file that defines verification success.
DBSelection ?	Database link where the verification data is to be recorded.
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 Product Intent Descriptions

Product intent is also described as a JDF node. The following table defines the list of JDF intent resources used to describe product intent.

Table 6-23: Product Intent – Input resources (Section 1 of 2)

Name	Description
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.
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.
DeliveryIntent ?	Summarizes the options that describe pickup or delivery time and location of the physical resources 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.

Table 6-23: Product Intent – Input resources (Section 2 of 2)

Name	Description
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.
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 6-24: Product Intent – Output resources

Name	Description
Component +	Resource representation of the output this Product Intent Node. Multiple Component resources MUST be specified in a root node that contains a DeliveryIntent that references multiple Component resources as delivery end products.

6.4 Prepress Processes

6.4.1 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 6-25: AssetListCreation – Input resources

Name	Description
RunList	List of assets used to create a listing of dependent assets.
AssetListCreationParams	Parameters of the AssetListCreation process

Table 6-26: 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 .

6.4.2 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 ImageSetting* and **Bending** process.

Table 6-27: 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 <i>MediaType</i> = "Plate" serves as an input resource.

Table 6-28: Bending – Output resources

Name	Description
ExposedMedia	The bent/punched ExposedMedia Resource.

6.4.3 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 colorspace might be.

The **ColorCorrection** process MAY be combined 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 6-29: 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 6-30: ColorCorrection – Output resources

Name	Description
RunList	List of color-corrected pages.

6.4.4 ColorSpaceConversion

ColorSpaceConversion, as the name implies, is the process of converting all colors used in the job to a known colorspace. 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 content.

The parameters of this resource provide the ability to selectively control the conversion or tagging of raster data or graphical objects based on object class and/or incoming color space.

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 **MUST** be paired with the identified final target device profile to create the transformation.

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** select a DeviceLink¹ profile as the assumed color space characterization. In these instances, the final target profile is ignored. Since there is no algorithmic way to determine that the output characterization in a device link profile is equivalent to another profile, some of the responsibility to select a sensible combination falls on the agent or end user.

Table 6-31: ColorSpaceConversion – Input resources

Name	Description
ColorantControl ? Modified in JDF 1.1A	Identifies the assumed color model for the job.
ColorSpaceConversionParams	Parameters that define how colorspace will be converted in the file.
RunList	List of pages, sheets or byte maps on which to perform the selected operation.

Table 6-32: ColorSpaceConversion – Output resources

Name	Description
ColorantControl ?	Identifies the assumed color model for the job. The ColorantControl resource can be modified by a ColorSpaceConversion Process.
RunList	List of pages, sheets or byte maps on which the selected operation has been performed.

6.4.5 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.

Table 6-33: ContactCopying – Output resources

Name	Description
ExposedMedia	The resulting exposed contact copy.

6.4.6 ContoneCalibration

This process specifies the process of contone calibration. It consumes contone raster data such as that output from an interpreting and rendering process. It produces contone raster data which has been calibrated to a press using a well defined screening process.

1. DeviceLink profiles are ICC profiles that map directly from one device color space to another device color space. Therefore, it represents a one-way link or connection between devices. Examples for Device-Link profiles are CMYK to CMYK print process conversions or RGB to CMYK color separations.

Table 6-34: ContoneCalibration – Input resources

Name	Description
RunList	Ordered list of rasterized byte maps representing pages or surfaces.
ScreeningParams ? Modified in JDF 1.1	Parameters specifying which halftoning mechanism is to be applied and with what specific controls.
TransferFunctionControl ? Modified in JDF 1.1	Specifies which calibration to apply.

Table 6-35: ContoneCalibration – Output resources

Name	Description
RunList	Ordered list of rasterized byte maps representing pages or surfaces.

6.4.7 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 6-36: 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 6-37: 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.

6.4.8 DBDocTemplateLayout

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 **LayoutElementProduction** process except that the output is a set of document templates. Document template are represented in JDF as **LayoutElement** resources with *Template* = "true".

Table 6-38: 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 6-39: 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 <i>Template</i> attribute MUST be <i>true</i> .

6.4.9 DBTemplateMerging

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 6-40: 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 6-41: 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.

6.4.10 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*"¹ 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 "Examples" on page 771.

Table 6-42: 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 6-43: DigitalDelivery – Output resources

Name	Description
RunList + Modified in JDF 1.3	The list of digital files which were actually delivered to the destination.

1. When **ArtDeliveryIntent/ArtDelivery/@ArtDeliveryType** = "*DigitalFile*", the process MAY also be **Delivery**, in case the file is delivered on digital media.

6.4.11 FilmToPlateCopying

[Deprecated in JDF 1.1](#)

FilmToPlateCopying has been replaced by the more generic **ContactCopying**. See "FilmToPlateCopying" on page 822 for details of this deprecated process.

6.4.12 FormatConversion

[New in JDF 1.1](#)

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**.

Table 6-44: 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 6-45: 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 MUST match the input FormatConversionParams/@MimeType . If FileSpec/@MimeType is not specified, then FormatConversionParams/@MimeType is used to update the output resource.

6.4.13 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 6-46: 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 6-47: ImageReplacement – Output resources

Name	Description
RunList	List of page contents with images that have been manipulated as indicated by the ImageReplacementParams resource.

6.4.14 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 "Approval" on page 220.)

Table 6-48: 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 . Exactly one of Media or ExposedMedia MUST be specified.
ImageSetterParams ? Modified in JDF 1.1	Controls the device specific features of the imagesetter.
Media ?	The unexposed media. Exactly one of Media or ExposedMedia MUST 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 6-49: ImageSetting – Output resources

Name	Description
ExposedMedia	The exposed media resource.

6.4.15 Imposition

The **Imposition** process is responsible for combining several pages of input graphical content on to a single surface whose dimensions are reflective of the physical output media. 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., **Layout**) is defined either by **LayoutPreparation**, **Stripping** or by the generic **ResourceDefinition** process.

There are two mechanisms provided for controlling the flow of page images onto **Media**. The default mechanism, which provides the functionality of **Layout** in PJTF, explicitly identifies all page content for each sheet imaged and references these pages by means of the **Documents** and/or **MarkDocuments** array. Setting the **Automated** attribute of the **Layout** resource to *true* activates a template approach to printing and relies upon the full **Documents** hierarchy to specify the page content to image. Automated impositioning is equivalent to the Print-Layout functionality in PJTF.

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, the **Layout** resources include an array of signatures. Each signature in turn specifies an array of sheets, and each sheet can have up to two surfaces (*Front* and *Back*), on which the page images and any marks are to be placed using **PlacedObject** elements. A sheet that specifies no surface content will be blank. Pages that are to be printed MUST be placed onto surfaces using **ContentObject** subelements which explicitly identify the page (via the *Ord* attribute which specifies an index into the document **RunList**). Thus, the **Layout** hierarchy specifies explicitly which pages will be imaged.

When describing automated imposition, **Layout** resources specify a single signature of sheet(s) where page contents are imaged. The (virtual) sequence of pages which is to be imaged via automated layout is defined by the Document **RunList**. Pages are drawn in order from this sequence to satisfy the **ContentObject** elements in the surfaces for the signature in the **Layout**, and the signature is repeated until all pages of the sequence are consumed. Each time the signature is repeated, pages are consumed in “chunks”. The total of number of pages per iteration of the

Layout is determined by the value of *MaxOrd* + 1 (if present in the **Layout**), or by the largest *Ord* value or calculated *OrdExpression* value for any **ContentObject** in the signature (if *MaxOrd* is absent).

Attributes of the **Media** are given for each sheet used in printing. Because the same signature is repeated until all pages are consumed, the **Layout** hierarchy can provide hints or preferences about special needs for sets of page content via **InsertSheet** elements. Inserting media is a way to separate sections of the document content. Thus alternate content is printed only as necessary to fill areas which would normally have page content because new media has been added or to designate where a document section will begin as specified by the odd or even position of the signature.

In a JDF model, the **Imposition** process is defined separately from other processes, which can precede or follow it. A *Combined* node MAY combine **Imposition** with other processes (e.g., **Separation** or **Interpreting**) to describe a device that happens to perform both in a single execution module.

Table 6-50: Imposition – Input resources

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.
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 6-51: Imposition – Output resources

Name	Description
RunList	Structured list of imposed surfaces. The <i>Type</i> of the LayoutElement resources MUST all be <i>Surface</i> . Typically the output RunList will be partitioned by <i>PartIDKeys</i> = " <i>SheetName Side Separation</i> ". If the Imposition process is executed before RIPing, this will generally be consumed by an Interpreting process. In the case of post-RIP Imposition , it will be consumed by DigitalPrinting or ImageSetting .

6.4.16 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 6-52: 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 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 6-53: InkZoneCalculation – Output resources

Name	Description
InkZoneProfile	Contains information about ink coverage along and perpendicular to the ink zones for a specific press geometry.

6.4.17 Interpreting

The interpreting device consumes page descriptions and instructions for controlling the marking device, (e.g., image-setter, digital printers, CTP, combined digital printing 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 MUST 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 6-54: 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 6-55: Interpreting – Output resources

Name	Description
RunList ? New in JDF 1.2	Pipe of streamed data which represents the results of <i>Interpreting</i> the pages in the RunList . The data is specified in InterpretedPDLData sub-elements. The format and detail of these is implementation specific. In general, it is assumed that the <i>Interpreting</i> and <i>Rendering</i> processes are tightly coupled and that there is no value in attempting to develop a general specification for the format of this data.
InterpretedPDLData ? Deprecated in JDF 1.2	Pipe of streamed data which represents the results of <i>Interpreting</i> the pages in the RunList . In JDF 1.2 and beyond, a RunList with InterpretedPDLData subelements describes the output content data for <i>Interpreting</i> .

6.4.18 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 InDesign®, 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 6-56: LayoutElementProduction – Input resources

Name	Description
LayoutElement *	Metadata about the PDL or application file, bitmap image file, text file, vector graphics file, etc.
LayoutElementProductionParams ? New in JDF 1.3	The parameters for the LayoutElementProduction process.

Table 6-57: 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 MUST be specified.
RunList ?	A RunList of LayoutElement resources of <i>ElementType Page</i> or <i>Document</i> is produced if this LayoutElementProduction task is the last process of type LayoutElementProduction . Exactly one of LayoutElement or RunList MUST be specified.

6.4.19 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 6-58: 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 6-59: LayoutPreparation – Output resources

Name	Description
Layout	The layout of the document to be imposed.
RunList (Marks) ?	List of marks that is to be used as input of the following Imposition process.
TransferCurvePool ?	Definition of the transfer curves and coordinate systems of the devices.

6.4.20 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* node with a **PSToPDFConversion** process.

It is RECOMMENDED to replace **PDFToPSConversion** with the combination of **Interpreting** and **PDLCreation** processes.

Table 6-60: 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 6-61: 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.

6.4.21 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 6-62: 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 MUST specify either an InterpretedPDLData or ByteMap element, but not both.

Table 6-63: 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 MUST match PDLCreationParams/@MimeType . If not specified, then PDLCreationParams/@MimeType is inserted.

6.4.22 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 finished 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 6-64: Preflight – Input resources

Name	Description
PreflightParams	A specified list of tests against which documents and/or pages are to be tested.
PreflightReportRulePool	A list of rules used to build the PreflightReport . Those rules are attached to actions in the ActionPool .
RunList	The list of documents and/or pages to be preflighted.

Table 6-65: Preflight – Output resources

Name	Description
PreflightReport	PreflightReport is a container for logging information that is generated by the Preflight process.

6.4.23 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) MUST fulfill the following requirements:

$$\text{MediaBox-length} / 72 * \text{x-resolution} = \text{width} \pm 1$$

$$\text{MediaBox-height} / 72 * \text{y-resolution} = \text{height} \pm 1$$

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).

Rules for the Generation of the Preview Image

To be useful for the ink consumption calculation, the preview data MUST 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%.

Spatial Resolution

Since some pixel of the preview image might fall on the border between two zones, their tonal values MUST 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:

$$\text{error}[\%] = \frac{100}{4 * \text{resolution}[\text{L} / \text{mm}] * \text{zone_width}[\text{mm}]}$$

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.

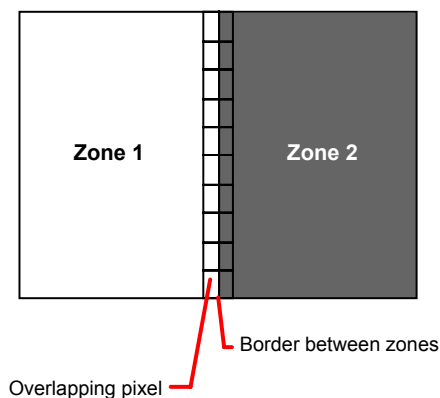


Figure 6-1: Worst case scenario for area coverage calculation

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.

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.

Conclusion

There are quite a few different ways to meet the requirements listed above. The following list includes several examples:

- The job can be RIPPed 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 RIPPed 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).

Recommendations for Implementation

The following three guidelines are strongly RECOMMENDED:

- The resolution of RIPPed 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 6-66: PreviewGeneration – Input resources (Section 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 MUST be specified in any PreviewGeneration process.
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 MUST 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 MUST be specified in any PreviewGeneration process.

Table 6-66: PreviewGeneration – Input resources (Section 2 of 2)

Name	Description
TransferCurvePool ? New in JDF 1.1	Area coverage correction and coordinate transformations of the device.

Table 6-67: 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 as thumbnail images.

6.4.24 Proofing

[Deprecated in JDF 1.2](#)

The **Proofing** process is deprecated in JDF/1.2. Instead, use a combined process to produce 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 "Proofing" on page 826 for details of this deprecated process. In JDF 1.2 and beyond, proofing is a combined process.

6.4.25 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 6-68: 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 is to be found.

Table 6-69: PStoPDFConversion – Output resources

Name	Description
RunList	This resource identifies the location of the resulting PDF pages.

6.4.26 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**. See also **FormatConversion**.

Table 6-70: RasterReading – Input resources (Section 1 of 2)

Name	Description
RasterReadingParams ?	Additional parameters for reading raster files.

Table 6-70: RasterReading – Input resources (Section 2 of 2)

Name	Description
RunList	This resource identifies a set of raster pages or surfaces that will be inserted into the display list. This resource MUST reference ByteMap images.

Table 6-71: 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 MUST specify an InterpretedPDLData element that describes the output content data for RasterReading .

6.4.27 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 6-72: Rendering – Input resources

Name	Description
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 .
RunList ? New in JDF 1.2	Pipe of streamed data that represents the results of Interpreting the pages in the RunList . The data is specified in InterpretedPDLData sub-elements. 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.
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 sub-element describes the input content data for Rendering .
RenderingParams ?	This resource describes the format of the byte maps to be created and other specifics of the Rendering process.

Table 6-73: Rendering – Output resources

Name	Description
RunList Modified 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 sub-elements. 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.

6.4.28 RIPing

RIPing is a gray box (See “Use of the Types attribute in ProcessGroup nodes – Gray Boxes” on page 47.) 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:

```
<JDF Type="ProcessGroup" Types="RIPing" Category="RIPing" ... />
```

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 combined 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:

RIP→Screening→ImageSetting→ContactCopying→ConventionalPrinting

RIP→(Screening)→DigitalPrinting

Since RIPing is not a predefined JDF process, see the processes that contribute to the RIP for input and output resources.

6.4.29 Scanning

The **Scanning** process creates bitmaps from analog images using a scanner.

Table 6-74: 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 6-75: Scanning – Output resources

Name	Description
RunList	List of ByteMap resources or LayoutElement resources of <i>Type = "Image"</i> .

6.4.30 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 post-RIP halftoning 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**.

Table 6-76: Screening – Input resources

Name	Description
RunList	Ordered list of rasterized ByteMap or InterpretedPDLData representing pages or surfaces.
ScreeningParams	Parameters specifying which halftone mechanism is to be applied and with what specific controls.

Table 6-77: 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.

6.4.31 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 combined with an **ImageReplacement** process in order to do image substitution for omitted or proxy images.

Table 6-78: 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 6-79: Separation – Output resources

Name	Description
RunList	List of separated pages or separated raster bytemaps.

6.4.32 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 "SoftProofing" on page 827 for details of this deprecated process. In JDF 1.2 and beyond, soft proofing is a combined process.

6.4.33 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 6-80: Stripping – Input resources

Name	Description
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 <i>JobID</i> and <i>AssemblyIDs</i> attributes.
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 [<i>@ElementType = "Reservation"</i>]) 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 6-81: 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.

Examples

The first example specifies three sheets based on folding catalog example F16-6. More examples can be found in Section N.6, Stripping.

```
<StrippingParams ID="FoldCatalogSample" Class="Parameter" Status="Available"
WorkStyle="WorkAndBack" PartIDKeys="SheetName">
  <BinderySignature FoldCatalog="F16-6"/>
  <StrippingParams SheetName="Sheet1"/>
  <StrippingParams SheetName="Sheet2"/>
  <StrippingParams SheetName="Sheet3"/>
</StrippingParams>
```

The following example specifies three sheets: *Sheet1* and *Sheet2* are based on a *B2x4* **BinderySignature** using the *WorkAndBack* workstyle, while *Sheet3* is based on **BinderySignature** *B2x2* using the *WorkAndTurn* workstyle.

WorkAndBack B2x4			
8	7	4	11
15	0	3	12

WorkAndTurn B2x2			
4	3	2	5
7	0	1	6

```
<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>

```

6.4.34 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** MUST always follow **Imposition**, but it can operate on imposed PDL page contents or on contone or halftone data. **Tiling** will generally be combined with other processes. For example, **Tiling** might be combined with **ImageSetting**. In that case, the input would be a **RunList** that contains **ByteMap** resources for each surface.

Table 6-82: Tiling – Input resources

Name	Description
RunList (Surface)	Structured list of imposed page contents or ByteMaps that are to be decomposed to produce the images for each tile. The <i>Type</i> value of LayoutElement resources MUST all be <i>Surface</i> .
RunList (Marks) ?	Structured list of incoming marks. These are typically printer's marks that provide the information needed to combine the tiles.
Tile	A partitioned Tile resource that describes how the surface contents are to be decomposed.

Table 6-83: Tiling – Output resources

Name	Description
RunList	Structured list of portions of the decomposed surfaces. The value of the <i>Type</i> attribute of the LayoutElement resources MUST be <i>Tile</i> .

6.4.35 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 combined with **RIPing** or specified as a stand-alone process.

Table 6-84: 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 6-85: Trapping – Output resources

Name	Description
RunList	Structured list of the modified page contents after Trapping has been executed.

6.5 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 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.

6.5.1 ConventionalPrinting

This process covers several conventional printing tasks, including sheetfed 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**, **Cutting** and **Numbering**) 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 the **DeviceColorantOrder** attribute of the appropriate **ColorantControl** resource.

In the context of web printing, the **ConventionalPrinting** process MUST be combined with the **WebInlineFinishing** process. The following drawing gives an overview about web printing in general.

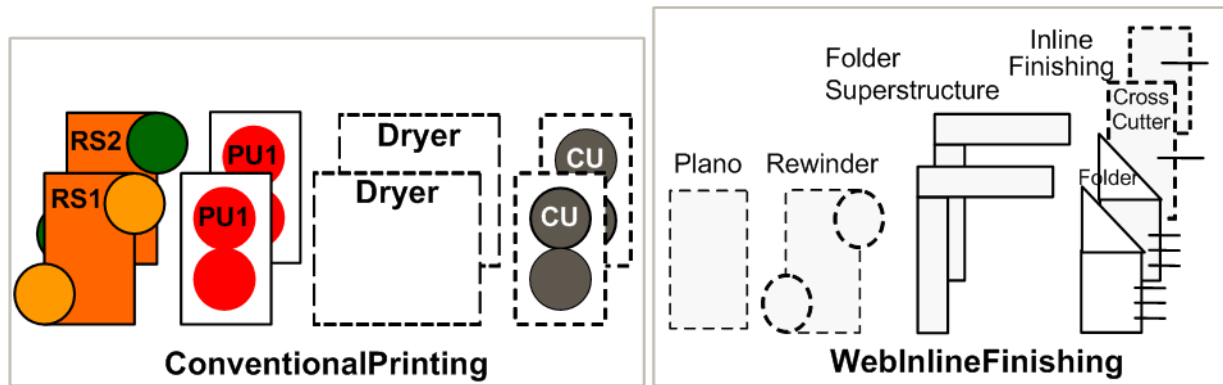


Figure 6-2: Overview of Web Printing

Table 6-86: ConventionalPrinting – Input resources (Section 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 (Input) ?	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.
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) MUST 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) MUST 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 Component (Proof) or ExposedMedia (Proof) SHOULD be specified.
Ink ? Modified in JDF 1.1	Information about the ink (e.g. brand, color) is useful to set up the press.

Table 6-86: ConventionalPrinting – Input resources (Section 2 of 2)

Name	Description
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.
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 MUST be present if no preprinted Component (Input) resource is used.
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 6-87: 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 <i>Amount</i> attribute of the ResourceLink to this resource indicates the number of copies of the entire job which will be produced. Prior to JDF 1.2 this Component was marked with a <i>ProcessUsage</i> = "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 <i>Condition</i> in order to distinguish waste Component resources from good Component resources.

6.5.2 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 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 6-88: DigitalPrinting – Input resources (Section 1 of 2)

Name	Description
ColorantControl ?	The ColorantControl resources that define the ordering and usage of inks in print modules.
Component (Input) *	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 .
Component (Proof) ?	A Proof component is used if a proof was produced during an earlier print run, (see description in Section 6.5.1, 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 pre-printed Component (Input) resource is present. Unprinted Media used for covers are also defined as Media . Note that 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 ByteMaps that will be printed on the digital press are needed for DigitalPrinting . The RunList contains only ByteMaps .
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.

Table 6-88: DigitalPrinting – Input resources (Section 2 of 2)

Name	Description
TransferCurvePool ? New in JDF 1.1	Area coverage correction and coordinate transformations of the device.

Table 6-89: DigitalPrinting – Output resources

Name	Description
Component Modified in JDF 1.2	Components are produced for other printing processes or postpress processes. Note that the <i>Amount</i> attribute of the ResourceLink to this resource indicates the number of copies of the entire job which will be produced. Prior to JDF 1.2 this Component was marked with a <i>ProcessUsage</i> = "Good", which is OPTIONAL, but supported in JDF 1.2 and beyond.
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 <i>Condition</i> in order to distinguish waste Component resources from good Component resources.

6.5.3 IDPrinting

[Deprecated in JDF 1.1](#)

The IDPrinting process was deprecated in JDF/1.1. Instead, implementations SHOULD use the **DigitalPrinting** process combined with 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. See "IDPrinting" on page 828 for details of this deprecated process.

6.6 Postpress Processes

In this specification, the postpress processes are presented in two parts: an alphabetical list of processes that is then followed by a Postpress Processes Structure section that divides these processes into subchapters for structuring purposes. This structuring is useful to find specific processes. Please note that processes, in some cases can be used to describe operations that go beyond the scope of a specific chapter. Therefore, it is a good idea not only to look at certain processes within a subchapter but also to find out what functionality other processes offer if a specific task needs to be addressed.

6.6.1 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.

6.6.2 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 6-90: 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 6-91: BlockPreparation – Output resources

Name	Description
Component	One Component is produced: the prepared book block. Its <i>ProductType</i> = “BookBlock”

6.6.3 BoxFolding

[New in JDF 1.3](#)

BoxFolding defines the process of folding and gluing blanks into folded flat boxes for packaging.

Table 6-92: BoxFolding – Input resources

Name	Description
Component	The BoxFolding process consumes one Component , the folding blank. Its <i>ProductType</i> = “BlankBox”
Component (Application) *	This process MAY consume additional Component resources, such as windows, handles or inlets. These Component resources MUST additionally be referenced from BoxFoldingParams /Application elements.
BoxFoldingParams	Specific parameters to set up the folder gluer.

Table 6-93: BoxFolding – Output resources

Name	Description
Component	One Component is produced: the folded flat box. Its <i>ProductType</i> = “FlatBox”

6.6.4 BoxPacking

[New in JDF 1.1](#)

A pile, stack or bundle of products can be packed into a box or carton.

Table 6-94: BoxPacking – Input resources

Name	Description
BoxPackingParams	Specific parameters to set up the machinery.
Component	The BoxPacking process puts a set of Component resources into the box Component .
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 6-95: BoxPacking – Output resources

Name	Description
Component	One Component is produced: the boxed Component .

6.6.5 Bundling

[New in JDF 1.2](#)

JDF-Spec 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 6-3: Bundle Creation



Figure 6-4: Bundle Transport

Table 6-96: Bundling – Input resources

Name	Description
Component	Component to be bundled
BundlingParams	Bundling parameters.
Media ?	End boards to protect the bundle. For each bundle a pair of end boards is needed.

Table 6-97: Bundling – Output resources

Name	Description
Component	The completed bundle.

Parameters like manufacturer and device type are defined in the **Device** element.

6.6.6 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 6-98: CaseMaking – Input resources (Section 1 of 2)

Name	Description
Component (CoverMaterial) ?	The cover material is either a preprinted or processed sheet of paper. Exactly one of Media (CoverMaterial) or Component (CoverMaterial) MUST be specified.
CaseMakingParams	Specific parameters to set up the machinery.
Media (CoverMaterial) ?	The CaseMaking process MAY also consume unprocessed Media as cover material. Exactly one of Media (CoverMaterial) or Component (CoverMaterial) MUST be specified.

Table 6-98: CaseMaking – Input resources (Section 2 of 2)

Name	Description
Media (CoverBoard) Modified in JDF 1.1A	The cardboard Media used for the cover board.
Media (SpineBoard) ?	The cardboard Media used for the spine board. If not specified, the Media (CoverBoard) MUST be used for the spine board.

Table 6-99: CaseMaking – Output resources

Name	Description
Component	One Component is produced: the produced book case. Its <i>ProductType</i> = “ <i>BookCase</i> ”.

6.6.7 CasingIn

[New in JDF 1.1](#)

The hard cover book case and the book block are joined in the **CasingIn** process.

Table 6-100: CasingIn – Input resources

Name	Description
Component	The prepared book block.
Component (Case)	The hard cover book case.
CasingInParams	Specific parameters to set up the machinery.

Table 6-101: CasingIn – Output resources

Name	Description
Component	One ^a Component is produced: the completed hard cover book.

a.Note that JDF 1.1 defined two output **Component** resources. This was an editing error in the specification.

6.6.8 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 6-102: ChannelBinding – Input resources

Name	Description
Component (BookBlock)	The operation requires one component: the block of sheets to be bound.
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.
ChannelBindingParams	Specific parameters to set up the machinery.

Table 6-103: ChannelBinding – Output resources

Name	Description
Component	One Component is produced: the channel-bound component forming an item such as a brochure.

6.6.9 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 6-104: CoilBinding – Input resources

Name	Description
Component	The operation requires one component: the pile of prepunched sheets often including a top and button cover.
CoilBindingParams	Specific parameters to set up the machinery.

Table 6-105: CoilBinding – Output resources

Name	Description
Component	One Component is produced: the coil-bound component forming an item such as a calendar.

6.6.10 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 6-106: Collecting – Input resources

Name	Description
Assembly ? New in JDF 1.3	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 *	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 ?	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 6-107: Collecting – Output resources

Name	Description
Component	A block of collected sheets is produced. This Component can be joined in further postpress processes.

6.6.11 CoverApplication

[New in JDF 1.1](#)

CoverApplication describes the process of applying a soft cover to a book block.

Table 6-108: CoverApplication – Input resources

Name	Description
CoverApplicationParams	Specific parameters to set up the machinery.
Component	The book block on which the cover is applied
Component (Cover)	The soft cover that is applied.

Table 6-109: CoverApplication – Output resources

Name	Description
Component	The book block with the applied soft cover.

6.6.12 Creasing

[New in JDF 1.1](#)

Sheets are creased or grooved to enable folding or to create even, finished page delimiters.

Table 6-110: 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 6-111: Creasing – Output resources

Name	Description
Component	One creased Component is produced.

6.6.13 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 MUST determine the sequence.

Media might also be cut in a precutting step. In this case, **Cutting** SHOULD deliver **Media** as the output resource.

Table 6-112: Cutting – Input resources (Section 1 of 2)

Name	Description
Component ?	This process consumes one Component : the printed sheets. Exactly one of Component or Media MUST 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 MUST 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.

Table 6-112: Cutting – Input resources (Section 2 of 2)

Name	Description
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 MUST be specified as input.

Table 6-113: 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 MUST be a set of Component resources. Either Component or Media MUST 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 MUST be specified as output, but not both.

6.6.14 Dividing

[Deprecated in JDF 1.1](#)

Dividing has been replaced by **Cutting**. See "Dividing" on page 829 for details of this deprecated process.

6.6.15 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 6-114: Embossing – Input resources

Name	Description
Component	This process consumes one Component :
EmbossingParams	Parameters to setup the machinery.
Media ?	If foil stamping or foil embossing, the stamping foil material is REQUIRED.
Tool ?	The embossing stamp or calendar.

Table 6-115: Embossing – Output resources

Name	Description
Component	One Component is created.

6.6.16 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 6-116: EndSheetGluing – Input resources (Section 1 of 2)

Name	Description
Component (BackEndSheet)	A back-end sheet to be mounted on the book block.
Component (BookBlock)	A back-end sheet and a front-end sheet are glued onto the book block.

Table 6-116: EndSheetGluing – Input resources (Section 2 of 2)

Name	Description
Component (FrontEndSheet)	A front-end sheet to be mounted on the book block.
EndSheetGluingParams	Specific parameters to set up the machinery.

Table 6-117: EndSheetGluing – Output resources

Name	Description
Component	A book block is produced that includes the end sheets.

6.6.17 Feeding

[New in JDF 1.2](#)

The **Feeding** process separates sheets or signatures from a stack or stream and feeds single **Component**(s) to processes such as **Folding**, **Gathering**, **Collecting**, **ConventionalPrinting**, etc. In general, the **Feeding** process will be combined with the processes that consume the feed of **Component**(s) 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).

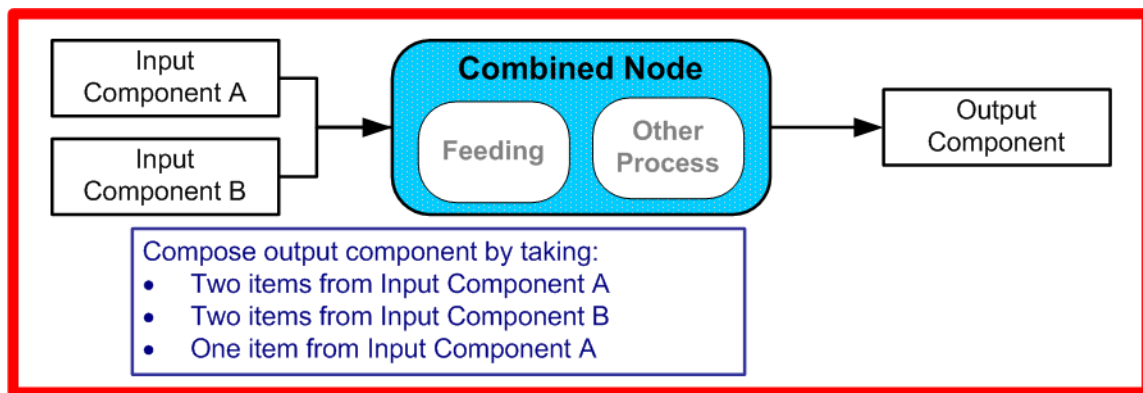


Figure 6-5: Combined Process with Feeding Process

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 6-6). Figure 6-7 shows the output for the case of **Gathering**.

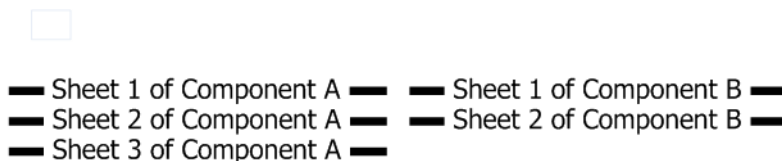


Figure 6-6: Input Components

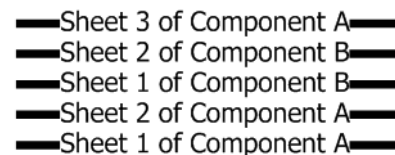


Figure 6-7: Output Component

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 6-118: Feeding – Input resources

Name	Description
Component *	Sheets or signatures to be fed to the machinery. The <i>ProcessUsage</i> of the Component MAY be specified as any valid <i>ProcessUsage</i> of the a feed consuming process.
FeedingParams	Specific parameters to set up the Feeding Process
Media *	Media to be fed to the feeder machinery.

Table 6-119: Feeding – Output resources

Name	Description
Component *	Component (s) fed to the consuming process.
Media *	Media fed to the consuming process.

6.6.18 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 sheet fed printed materials – and in-line equipment of web printing 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 6-120: 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 6-121: Folding – Output resources

Name	Description
Component Modified in JDF 1.1	The process produces a Component , which in most cases is a folded sheet.

6.6.19 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. The figure on the right shows typical gathered piles.

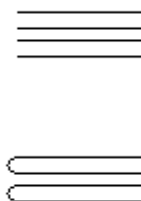


Table 6-122: Gathering – Input resources

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.
GatheringParams	Specific parameters to set up the machinery.
DBRules *	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 ?	Database input that describes which sheets are to be gathered 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 6-123: Gathering – Output resources

Name	Description
Component	Components gathered together, (e.g., a pile of folded sheets).

6.6.20 Gluing

[New in JDF 1.1](#)

Gluing describes arbitrary methods of applying glue to a **Component**.

Table 6-124: Gluing – Input resources

Name	Description
Component	This process consumes one Component : the printed sheets.
GluingParams	Details of the Gluing process.

Table 6-125: Gluing – Output resources

Name	Description
Component	One Component is produced, the input Component with glue applied to it.

6.6.21 HeadBandApplication

[New in JDF 1.1](#)

Head bands are applied to the hard cover book block.

Table 6-126: HeadBandApplication – Input resources

Name	Description
Component	The prepared book block.
HeadBandApplicationParams	Specific parameters to set up the machinery.

Table 6-127: HeadBandApplication – Output resources

Name	Description
Component	One Component is produced: the hard cover block with head bands.

6.6.22 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 6-128: HoleMaking – Input resources

Name	Description
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 6-129: 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.

6.6.23 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.

Table 6-130: Inserting – Input resources

Name	Description
Component (Mother)	Designates where to insert the child Component .
Component (Child)	The Component to be inserted in the mother Component .
InsertingParams	Specific parameters (e.g., placement) to set up the machinery.
DBRules ?	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 ?	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.

Table 6-131: Inserting – Output resources

Name	Description
Component	A mother Component is produced containing the inserted child Component .

6.6.24 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 creates the jacket, this can be described with a **Combined** process of **Jacketing** and **Creasing**.

Table 6-132: Jacketing – Input resources

Name	Description
JacketingParams	Specific parameters to set up the machinery.
Component (Book)	The book that the jacket is wrapped around.
Component (Jacket)	The description of the jacket.

Table 6-133: Jacketing – Output resources

Name	Description
Component	The jacketed book.

6.6.25 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 6-134: 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 6-135: Labeling – Output resources

Name	Description
Component	One Component is produced: the labeled Component .

6.6.26 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 6-136: 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 6-137: Laminating – Output resources

Name	Description
Component	One Component is produced: the laminated component.

6.6.27 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 node with **ConventionalPrinting**. See "LongitudinalRibbonOperations" on page 830 for details of this deprecated process.

6.6.28 Numbering

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 6-138: Numbering – Input resources

Name	Description
Component	One Component (e.g., a printed sheet or a pile of sheets) are modified in the Numbering process.
NumberingParams	Specific parameters to set up the machinery.

Table 6-139: Numbering – Output resources

Name	Description
Component	One Component is produced: the numbered sheet.

6.6.29 Palletizing

[New in JDF 1.1](#)

Bundles, stacks, piles or boxes can be loaded onto a palette.

Table 6-140: Palletizing – Input resources

Name	Description
Component	The Palletizing process describes placing the bundle that is represented by the Component onto a palette.
PalletizingParams	Specific parameters to set up the machinery.
Pallet	The palette.

Table 6-141: Palletizing – Output resources

Name	Description
Component	One Component is produced: the loaded palette.

6.6.30 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 6-142: Perforating – Input resources

Name	Description
Component	This process consumes one Component : the printed sheets.
PerforatingParams	Details of the Perforating process.

Table 6-143: Perforating – Output resources

Name	Description
Component	One Component is produced.

6.6.31 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 6-144: 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 6-145: PlasticCombBinding – Output resources

Name	Description
Component	One Component is produced: the plastic-comb-bound component forming an item such as a calendar.

6.6.32 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*.



Table 6-146: PrintRolling – Input resources

Name	Description
Component	Component to be rolled.
PrintRollingParams ?	Print rolling parameters.
RollStand ?	Rollstand to store the component(s) as rolls.

Table 6-147: PrintRolling – Output resources

Name	Description
Component	The print roll.

6.6.33 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 **MUST** be used.

Table 6-148: RingBinding – Input resources (Section 1 of 2)

Name	Description
Component (BookBlock)	The operation requires one component: the pile of prepunched sheets to be inserted into the ring binder.
Component (RingBinder) ?	The empty ring binder that might have been printed, for example, before it is used during the <i>RingBinding</i> process.

Table 6-148: RingBinding – Input resources (Section 2 of 2)

Name	Description
RingBindingParams	Specific parameters to set up the process/machinery.

Table 6-149: RingBinding – Output resources

Name	Description
Component	One Component is produced: the ring-bound component forming an item such as a calendar.

6.6.34 SaddleStitching

[Deprecated in JDF 1.1](#)

SaddleStitching has been replaced by **Stitching** in JDF 1.1. See "SaddleStitching" on page 830 for details of this deprecated process.

6.6.35 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 6-150: 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 6-151: ShapeCutting – Output resources

Name	Description
Component + Modified in JDF 1.3	One or more Component resources are produced by the ShapeCutting process.

6.6.36 Shrinking

[New in JDF 1.1](#)

Shrink-wrap foil MUST be treated in order to shrink.

Table 6-152: Shrinking – Input resources

Name	Description
Component	The Shrinking process shrinks the shrink-wrap that is wrapped around a bundle. The bundle including the shrink-wrap media is represented by this Component .
ShrinkingParams	Specific parameters to set up the machinery.

Table 6-153: Shrinking – Output resources

Name	Description
Component	One Component is produced: the bundle including bundle including the shrunk shrink-wrap media.

6.6.37 SideSewing

[Deprecated in JDF 1.1](#)

Replaced by *ThreadSewing*. See "SideSewing" on page 830 for details of this deprecated process.

6.6.38 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 6-154: SpinePreparation – Input resources

Name	Description
Component	The raw book block.
SpinePreparationParams	Specific parameters to set up the machinery.

Table 6-155: SpinePreparation – Output resources

Name	Description
Component	The book block with a processed spine.

6.6.39 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 6-156: SpineTaping – Input resources

Name	Description
Component	The book block that the spine is taped to.
SpineTapingParams	Specific parameters to set up the machinery.

Table 6-157: SpineTaping – Output resources

Name	Description
Component	The book block with the spine.

6.6.40 Stacking

[New in JDF 1.1](#)

The stacking process collects physical resources (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 6-158: Stacking – Input resources

Name	Description
Component	The <i>Stacking</i> process consumes one Component and stacks it onto a stack.
StackingParams	Specific parameters to set up the machinery.

Table 6-159: Stacking – Output resources

Name	Description
Component	One Component is produced: the stack of input Components.

6.6.41 Stitching

Gathered or collected sheets or signatures are stitched together with a cover.

Table 6-160: Stitching – Input resources

Name	Description
Component	The only REQUIRED Component is the pile of gathered sheets, including the cover.
StitchingParams	Specific parameters to set up the machinery.

Table 6-161: Stitching – Output resources

Name	Description
Component	One Component is produced: the gathered or collected sheets including the cover stitched together.

Components containing staples of different characteristics like shape, width, etc. are defined by a combined process. For example:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="CombinedStitch" JobID="Stitching
special" Type="Combined" Types="Stitching Stitching" Version="1.3">
  <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"/>
  </ResourcePool>
  <ResourceLinkPool>
    <StitchingParamsLink CombinedProcessIndex="0" Usage="Input" rRef="Stitch1"/>
    <StitchingParamsLink CombinedProcessIndex="1" Usage="Input" rRef="Stitch2"/>
  </ResourceLinkPool>
</JDF>
```

6.6.42 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 6-162: 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 6-163: Strapping – Output resources

Name	Description
Component	One Component is produced: the strapped Component .

6.6.43 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 **MUST** 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 6-164: 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 6-165: StripBinding – Output resources

Name	Description
Component	One Component is produced: the strip-bound component forming an item such as a book.

6.6.44 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 6-166: ThreadSealing – Input resources

Name	Description
Component	This process consumes one Component : the printed sheets.
ThreadSealingParams	Details of the ThreadSealing process.

Table 6-167: ThreadSealing – Output resources

Name	Description
Component	One Component is produced.

6.6.45 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 6-168: ThreadSewing – Input resources

Name	Description
Component	The operation requires one component: the gathered sheets.
ThreadSewingParams	Specific parameters to set up the machinery.

Table 6-169: ThreadSewing – Output resources

Name	Description
Component	One Component is produced: the thread-sewn components forming an item such as a raw book block.

6.6.46 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* operation 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.

Some trimming machines such as magazine production systems can produce N-ups. In every case, however, the additional trimming cuts that divide the N-ups result in separated book blocks. Sometimes a strip is trimmed out between the book blocks. It takes multiple **Trimming** processes to describe such operations in JDF.

Table 6-170: 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 6-171: Trimming – Output resources

Name	Description
Component	One Component is produced: the trimmed component.

6.6.47 WebInlineFinishing

[New in JDF 1.3](#)

The process **WebInlineFinishing** 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 6-172: 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 6-173: 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 <i>WebProduct</i>

6.6.48 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 6-174: 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 6-175: WireCombBinding – Output resources

Name	Description
Component	One Component is produced: the wire-comb bound component forming an item such as a calendar.

6.6.49 Wrapping

[New in JDF 1.1](#)

Single products, bundles or pallets can be wrapped by film or paper.

Table 6-176: Wrapping – Input resources

Name	Description
Component	The Wrapping process wraps a bundle that is represented by a Component .
WrappingParams	Specific parameters to set up the machinery.
Media ?	The wrapping material.

Table 6-177: Wrapping – Output resources

Name	Description
Component	One Component is produced: the wrapped Component .

6.7 Postpress Processes Structure

6.7.1 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**, **SaddleStitching**, **SideSewing**, **SpineTaping**, **Stitching** and **ThreadSewing**. The workflow using these processes eventually produces a **Component** that can be trimmed.

6.7.1.1 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**

6.7.1.2 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 6.7.1.2.1. 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:

- **AdhesiveBinding**
- **CoverApplication**
- **EndSheetGluing**
- **Gluing**
- **SpinePreparation**
- **SpineTaping**
- **Stitching**
- **ThreadSewing**

6.7.1.2.1 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.

6.7.1.2.2 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**

6.7.1.2.3 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**

6.7.2 HoleMaking

- **HoleMaking**

6.7.3 Laminating

- **Laminating**.

6.7.4 Numbering

- **Numbering**.

6.7.5 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:

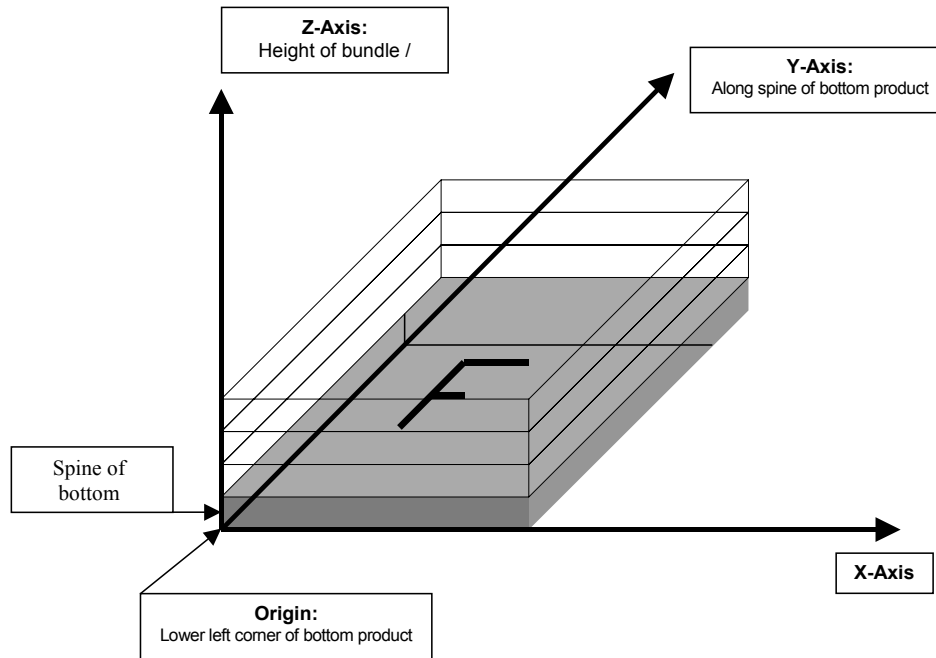


Figure 6-8: Packaging Process Coordinate System

6.7.6 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**

6.7.7 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 sheetfed 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**

6.7.8 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**

6.7.9 Trimming

- **Trimming.**

6.7.10 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.

Chapter 7 Resources

Introduction

Resources represent inputs and outputs, the “things” that are produced, modified, consumed, or in any way used by nodes. A more thorough description was provided in Section 3.7, *ResourcePool* and its *Resource Children*. The resources in this chapter are divided into two sections. The first section documents all of the resources of class *Intent*. The second section documents the rest of the resources that have been defined for JDF.

7.1 Intent Resources

As was described in Section 4.1.1, *Product Intent Constructs*, intent resources are designed to narrow down the available options when defining a JDF job. Many of the elements in intent resources 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 node has no *ResourceLink* to **NumberingIntent**, then no numbering is requested. The characteristics of the product that are not specified through the use of intent resources will be selected by the system that processes the intent resources. The system that processes the intent data in a JDF job ticket MAY insert the details of its selection into the JDF data for the job. See “Conformance Requirements for Support of Attributes and Attribute Values” on page 8 for more information on the handling and processing of systems-specified default values.

All intent resources 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 (Section 7.1.1, *Span Subelements of an Intent Resource*) describes these elements.

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 3.7.2, *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 also: 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 1.3.4, *Specification of Cardinality*

Resource Properties Template

Resource class:	<i>Defines the resource class.</i>
Resource referenced by:	<i>List of parent resources that MAY contain elements of this type.</i>
Process Pairing:	<i>List of process resources to 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.</i>
Example Partition:	<i>List of RECOMMENDED partitioning keys: For a complete list of partition keys, see the description of PartIDKeys in Table 3-25, “Partitionable resource element,” on page 86. Note that resources MAY be partitioned by keys that are not specified in this list.</i>
Input of processes:	<i>List of JDF node types that use the resource as an input resource.</i>
Output of processes:	<i>List of JDF node types that create the resource as an output resource.</i>

Table 7-1: Template for intent resources

Name	Data Type	Description
<i>Attribute-Name</i> <i>Cardinality</i>	<i>Attribute-data-type</i>	<i>Information about the attribute.</i>
<i>Element-Name</i> <i>Cardinality</i>	element	<i>Information about the element.</i> Note: the “element” data type means that the specified element MUST be an in-line sub-element within the resource.
<i>Element-Name</i> <i>Cardinality</i>	refelement	<i>Information about the element</i> Note: the “refelement” data type means that the specified element is based on other atomic resources or resource elements. The specified element MUST be either an in-line element or an instance of a ResourceRef element (see Section 3.9.2, ResourceRef – Element for Inter-Resource Linking and refElement). In case of a ResourceRef element, a “Ref” MUST be appended to the name specified in the table column entitled “Name”.

7.1.1 Span Subelements of an Intent Resource

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 abstract span element to be used is determined by the data type of the values to be recorded. Span elements are defined to facilitate negotiation between buyer and provider as defined in Section 4.1.2, Defining Business Objects Using Intent Resources.

Note: The tables used below to define the structure of intent resources can contain attributes with both JDF data types (as defined in the "Data Structures" on page 10 and "Defined JDF enumeration Data Types" on page 697) as well as the following span elements (used in the data type column of the intent resource's structure table). All possible span elements are listed in the following table. The formatting of *XXXSpan* elements was chosen to be the same as *attribute* formatting although the 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 further attributes or subelements. The contents shared by all span elements are listed in Section 7.1.1.1, Abstract Span Element, below, and the contents particular to each span element type are described in the sections that follow.

Table 7-2: List of span elements

Span Element Types	Data Type	Description
DurationSpan New in JDF 1.1	element	Describes a set of duration values.
EnumerationSpan	element	Describes a set of enumeration values.
IntegerSpan	element	Describes a numerical range of integer values.
NameSpan	element	Describes a set of NMTOKEN values.
NumberSpan	element	Describes a numerical range of values.
OptionSpan	element	Describes an intent in which the principal information is that a specific option is requested.
ShapeSpan New in JDF 1.1	element	Describes a set of shape values.
StringSpan	element	Describes a set of string values.
TimeSpan	element	Describes a set of dateTime values.
XYPairSpan	element	Describes a set of XYPair values.

7.1.1.1 Abstract Span Element

Abstract span elements of intent resources have a common set of attributes and elements that define the priority, data type and requested identity of the element. These attributes are described in the following table. 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 7-3: Abstract span element

Name	Data Type	Description								
<i>DataType</i>	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. Possible values are:								
		<table border="0"> <tr> <td><i>DurationSpan</i></td> <td><i>OptionSpan</i></td> </tr> <tr> <td><i>EnumerationSpan</i></td> <td><i>ShapeSpan</i></td> </tr> <tr> <td><i>IntegerSpan</i></td> <td><i>StringSpan</i></td> </tr> <tr> <td><i>NameSpan</i></td> <td><i>TimeSpan</i></td> </tr> <tr> <td><i>NumberSpan</i></td> <td><i>XYPairSpan</i></td> </tr> </table>	<i>DurationSpan</i>	<i>OptionSpan</i>	<i>EnumerationSpan</i>	<i>ShapeSpan</i>	<i>IntegerSpan</i>	<i>StringSpan</i>	<i>NameSpan</i>	<i>TimeSpan</i>
<i>DurationSpan</i>	<i>OptionSpan</i>									
<i>EnumerationSpan</i>	<i>ShapeSpan</i>									
<i>IntegerSpan</i>	<i>StringSpan</i>									
<i>NameSpan</i>	<i>TimeSpan</i>									
<i>NumberSpan</i>	<i>XYPairSpan</i>									
<i>Priority?</i> Deprecated in JDF 1.2	enumeration	<p>Indicates the importance of the specific intent. The following values have prescribed meanings:</p> <p><i>None</i></p> <p><i>Suggested</i> – The customer will accept a value of <i>Actual</i> that is different than the value of <i>Preferred</i> or outside of <i>Range</i>.</p> <p><i>Required</i> – The customer expects the <i>Actual</i> to be equal to <i>Preferred</i> or within <i>Range</i>. Note that the attribute <i>Preferred</i> is available in the data types which inherit from this abstract type.</p> <p>Note: Replaced by <i>SettingsPolicy</i> in JDF 1.2 and beyond.</p>								

7.1.1.2 DurationSpan Element

[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 7.1.1.1, Abstract Span Element.

Table 7-4: DurationSpan element

Name	Data Type	Description
<i>Actual?</i>	duration	The actual value selected for the quote.
<i>OfferRange?</i> New in JDF 1.3	DurationRange	Provides an offered range of time durations. If not specified, it defaults to the value of <i>Actual</i> .
<i>Preferred?</i>	duration	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of <i>Preferred</i> MUST fall within the range of values specified in <i>Range</i> .
<i>Range?</i>	DurationRange	Provides a valid range of time durations. If not specified, it defaults to the value of <i>Preferred</i> .

7.1.1.3 EnumerationSpan Element

This Span subelement is used to describe ranges of enumerative values. It inherits from the abstract Span element described in Section 7.1.1.1, Abstract Span Element. It is identical to the NameSpan element except for the fact that it describes a closed list of enumeration values.

Table 7-5: EnumerationSpan element

Name	Data Type	Description
<i>Actual</i> ?	enumeration	The actual value selected for the quote.
<i>OfferRange</i> ? New in JDF 1.3	enumerations	Provides an offered range of values. If not specified, it defaults to the value of <i>Actual</i> .
<i>Preferred</i> ?	enumeration	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of <i>Preferred</i> MUST fall within the range of values specified in <i>Range</i> .
<i>Range</i> ?	enumerations	Provides a set of discreet enumeration values. If not specified, it defaults to the value of <i>Preferred</i> .

7.1.1.4 IntegerSpan Element

This Span subelement is used to describe ranges of integer values. It inherits from the abstract Span element described in Section 7.1.1.1, Abstract Span Element.

Table 7-6: IntegerSpan element

Name	Data Type	Description
<i>Actual</i> ?	integer	The actual value selected for the quote.
<i>OfferRange</i> ? 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. If not specified, it defaults to the value of <i>Actual</i> .
<i>Preferred</i> ?	integer	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of <i>Preferred</i> MUST fall within the range of values specified in <i>Range</i> .
<i>Range</i> ?	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. If not specified, it defaults to the value of <i>Preferred</i> .

7.1.1.5 NameSpan Element

This Span subelement is used to describe name ranges. It inherits from the abstract Span element described in Section 7.1.1.1, Abstract Span Element. It is identical to the EnumerationSpan element except for the fact that it describes an extensible list of NMTOKEN values.

Table 7-7: NameSpan element

Name	Data Type	Description
<i>Actual</i> ?	NMTOKEN	The actual value selected for the quote.
<i>OfferRange</i> ? New in JDF 1.3	NMTOKENS	Provides a set of discreet values that comprise all offered values for the Span. If not specified, it defaults to the value of <i>Actual</i> .
<i>Preferred</i> ?	NMTOKEN	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of <i>Preferred</i> MUST fall within the range of values specified in <i>Range</i> .
<i>Range</i> ?	NMTOKENS	Provides a set of discreet values that comprise all allowed values for the Span. If not specified, it defaults to the value of <i>Preferred</i> .

7.1.1.5.1 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:

```
<HoleType DataType="NameSpan" Range="36Hole 42Hole">
  <Comment Name="36Hole">
    6 equidistant holes on each side of a hexagonal piece of paper
  </Comment>
  <Comment Name="42Hole">
    7 equidistant holes on each side of a hexagonal piece of paper
  </Comment>
</HoleType>
```

7.1.1.6 NumberSpan Element

This Span subelement is used to describe a numerical range of values. It inherits from the abstract Span element described in Section 7.1.1.1, Abstract Span Element.

Table 7-8: NumberSpan element

Name	Data Type	Description
<i>Actual</i> ?	double	The actual value selected for the quote.
<i>OfferRange</i> ? 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. If not specified, it defaults to the value of <i>Actual</i> .
<i>Preferred</i> ?	double	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of <i>Preferred</i> MUST fall within the range of values specified in <i>Range</i> .
<i>Range</i> ?	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. When not known, the value of <i>Range</i> shall default to the value of <i>Preferred</i> .

7.1.1.7 OptionSpan Element

This Span subelement is used to describe a range of options or Boolean values. It inherits from the abstract Span element described in Section 7.1.1.1, Abstract Span Element.

Table 7-9: OptionSpan element

Name	Data Type	Description
<i>Actual</i> ?	boolean	The actual value selected for the quote. If the option is included = "true".
<i>OfferRange</i> ? New in JDF 1.3	enumerations	Provides a set of the discreet Boolean values "true" and "false". If not specified, it defaults to the value of <i>Actual</i> .
<i>Detail</i> ? Deprecated in JDF 1.2	string	<i>Detail</i> provides information about the option. Replaced with <i>DescriptiveName</i> in JDF 1.2 and above.
<i>Preferred</i> ?	boolean	Provides a value specified by the person submitting the request, indicating what that person prefers.
<i>Range</i> ? New in JDF 1.2	enumerations	Provides a set of the discreet Boolean values "true" and "false".

7.1.1.8 ShapeSpan Element

[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 7.1.1.1, Abstract Span Element.

Table 7-10: ShapeSpan element

Name	Data Type	Description
<i>Actual</i> ?	shape	The actual value selected for the quote.
<i>OfferRange</i> ? New in JDF 1.3	ShapeRangeList	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. If not specified, it defaults to the value of <i>Actual</i> .
<i>Preferred</i> ?	shape	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of <i>Preferred</i> MUST fall within the range of values specified in <i>Range</i> .
<i>Range</i> ?	ShapeRangeList	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. If not specified, it defaults to the value of <i>Preferred</i> .

7.1.1.9 StringSpan Element

This Span subelement is used to describe string ranges. It inherits from the abstract Span element described in Section 7.1.1.1, Abstract Span Element.

Table 7-11: StringSpan element

Name	Data Type	Description
<i>Actual</i> ?	string	The actual value selected for the quote.
<i>Preferred</i> ?	string	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of <i>Preferred</i> MUST fall within the range of values specified in <i>Range</i> .
<i>OfferRange</i> * New in JDF 1.3	telem	Provides a set of discreet values that comprise all offered values for the Span. If not specified, it defaults to the value of <i>Actual</i> .
<i>Range</i> *	telem	Provides a set of discreet values that comprise all allowed values for the Span. If not specified, it defaults to the value of <i>Preferred</i> .

7.1.1.10 TimeSpan Element

This Span subelement is used to describe a selection of instances in time. It inherits from the abstract Span element described in Section 7.1.1.1, Abstract Span Element.

Table 7-12: TimeSpan element

Name	Data Type	Description
<i>Actual</i> ?	dateTime	The actual value selected for the quote.
<i>OfferRange</i> ? New in JDF 1.3	DateTimeRange	Provides a range of values that comprise all offered values for the Span. If not specified, it defaults to the value of <i>Actual</i> .
<i>Preferred</i> ?	dateTime	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of <i>Preferred</i> MUST fall within the range of values specified in <i>Range</i> .
<i>Range</i> ?	DateTimeRange	Provides a range of values that comprise all allowed values for the Span. If not specified, it defaults to the value of <i>Preferred</i> .

7.1.1.11 XYPairSpan Element

This Span subelement is used to describe ranges of numerical value pairs. It inherits from the abstract Span element described in Section 7.1.1.1, Abstract Span Element.

Table 7-13: XYPairSpan element

Name	Data Type	Description
<i>Actual</i> ?	XYPair	The actual value selected for the quote.
<i>OfferRange</i> ? New in JDF 1.3	XYPairRangeList	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. If not specified, it defaults to the value of <i>Allowed</i> .
<i>Preferred</i> ?	XYPair	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of <i>Preferred</i> MUST fall within the range of values specified in <i>Range</i> .
<i>Range</i> ?	XYPairRangeList	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. If not specified, it defaults to the value of <i>Preferred</i> .

7.1.2 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
Resource referenced by:	—
Process Resource Pairing:	DeliveryParams, DigitalDeliveryParams
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

Table 7-14: ArtDeliveryIntent resource (Section 1 of 4)

Name	Data Type	Description
<i>ArtDeliveryDate</i> ? New in JDF 1.1	TimeSpan	Specifies the latest time by which the transfer of the artwork will be made.
<i>ArtDeliveryDuration</i> ? 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 <i>ArtDeliveryDate</i> or <i>ArtDeliveryDuration</i> MUST be specified. Within a purchase order, only <i>ArtDeliveryDate</i> is allowed.

Table 7-14: ArtDeliveryIntent resource (Section 2 of 4)

Name	Data Type	Description
<p>ArtHandling ? New in JDF 1.1</p>	EnumerationSpan	<p>Describes what is to happen to the artwork after usage. The address for the <i>Return</i> and <i>Pickup</i> values MUST be specified by a Contact [contains (@<i>ContactTypes</i>, "ArtReturn")]/Address. Possible values are:</p> <p><i>ReturnWithProof</i> – The artwork is delivered back to the customer together with the proof if there is any.</p> <p><i>ReturnWithProduct</i> – The artwork is delivered back to the customer together with the final product.</p> <p><i>Return</i> – The artwork is delivered back independently directly after usage.</p> <p><i>Pickup</i> – The customer picks up the artwork.</p> <p><i>Destroy</i> – The printer destroys the artwork.</p> <p><i>PrinterOwns</i> – The artwork belongs to the printer.</p> <p><i>Store</i> – The printer has to store the artwork for future purposes.</p>
<p>DeliveryCharge ? New in JDF 1.1 Modified in JDF 1.3</p>	EnumerationSpan	<p>Specifies who pays for a delivery being made by a third party. Possible values are specified in Possible values are defined in DeliveryIntent/@DeliveryCharge.</p>
<p>Method ? Modified in JDF 1.3</p>	NameSpan	<p>Specifies the delivery method, which can be a generic method. Possible values are:</p> <p><i>EMail</i></p> <p><i>ExpressMail</i></p> <p><i>InterofficeMail</i></p> <p><i>OvernightService</i></p> <p><i>Courier</i></p> <p><i>CompanyTruck</i></p> <p><i>ISDNSoftware</i></p> <p><i>Local</i> – The files are already in place and a DigitalDelivery process is not needed. New in JDF 1.3</p> <p><i>NetworkCopy</i> – This includes LAN and VPN.</p> <p><i>WebServer</i> – Upload / download from HTTP / FTP server.</p> <p><i>InstantMessaging</i></p> <p>Can also be a delivery service brand, such as:</p> <p><i>UPS</i></p> <p><i>DHL</i></p> <p><i>FedEx</i></p> <p>Or any digital delivery service brand.</p>

Table 7-14: ArtDeliveryIntent resource (Section 3 of 4)

Name	Data Type	Description
<p><i>PreflightStatus</i> = "NotPerformed"</p> <p>New in JDF 1.1</p> <p>Modified in JDF 1.2</p>	enumeration	<p>Information about a preflight process probably applied to the artworks before being submitted. Possible values are:</p> <p><i>NotPerformed</i> – No preflighting was applied.</p> <p><i>WithErrors</i> – Preflighting resulted in error messages and possibly warning messages.</p> <p><i>WithWarnings</i> – Preflighting resulted in warning messages and no errors.</p> <p><i>WithoutErrors</i> – Preflighting was successful. No errors and no warnings occurred.</p>
<p><i>ReturnList</i> = "None"</p> <p>New in JDF 1.1</p>	NMTOKENS	<p>Type of printer created intermediate materials that are to be sent to the customer after usage. Possible values include:</p> <p><i>DigitalMedia</i> – Digital data on media, (e.g., a CD).</p> <p><i>DigitalNetwork</i> – Digital data via network.</p> <p><i>ExposedPlate</i> – Pre-exposed press plates, usually used for a rerun.</p> <p><i>ImposedFilm</i> – Film of the imposed surfaces.</p> <p><i>LooseFilm</i> – Film of individual pages or sections.</p> <p><i>OriginalPhysicalArt</i> – Analog artwork, (e.g., reflective or transparencies).</p> <p><i>Tool</i> – Tools needed for processing the job, (e.g., a die for die cutting or embossing stamp).</p> <p><i>None</i> – No intermediate materials are to be returned to the customer.</p>
<p><i>ReturnMethod</i> ?</p> <p>New in JDF 1.1</p>	NameSpan	<p>Specifies a delivery method for returning the artwork if <i>ArtHandling</i> = "Return" and for the printer created materials listed in <i>ReturnList</i>. The predefined values are the same as the list specified in <i>Method</i>.</p>
<p><i>ServiceLevel</i> ?</p> <p>New in JDF 1.2</p>	StringSpan	<p>The service level of the specific carrier. Contain values "Next Day", "2nd Day Air", "Ground", etc.</p>
<p><i>Transfer</i> ?</p> <p>New in JDF 1.1</p>	EnumerationSpan	<p>Describes the responsibility of the transfer. Possible values are:</p> <p><i>BuyerToPrinterDeliver</i> – The buyer delivers the artwork to the printer. The printer MAY specify in the quote a special Contact [contains (@<i>ContactTypes</i>, "Delivery")] to specify where the buyer is to send the artwork.</p> <p><i>BuyerToPrinterPickup</i> – The printer picks up the artwork. The Contact [contains (@<i>ContactTypes</i>, "Pickup")] specifies where the printer has to pick up the artwork.</p>
<p>ArtDelivery +</p> <p>Modified in JDF 1.1</p>	element	Individual delivery.
<p>Company ?</p> <p>Deprecated in JDF 1.1</p>	refelement	<p>Address and further information of the art delivery. Company MUST 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.</p>

Table 7-14: ArtDeliveryIntent resource (Section 4 of 4)

Name	Data Type	Description
Contact * New in JDF 1.1	refelement	Address and further information about the transfer of the artwork. The actual delivery address MUST be specified by Contact [contains (@ContactTypes, "Delivery")]/Address. <i>At most one such Contact MUST be specified.</i> The actual pickup address MUST be specified by Contact [contains (@ContactTypes, "Pickup")]/Address. <i>At most one such Contact MUST be specified.</i>

— Element: 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 7-15: ArtDelivery element (Section 1 of 3)

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 <i>ArtDeliveryDate</i> or <i>ArtDeliveryDuration</i> MUST be specified. Within a purchase order, only the <i>ArtDeliveryDate</i> is allowed.
ArtDeliveryType New in JDF 1.1 Modified in JDF 1.2	NMTOKEN	Type of artwork supplied. Possible values include: <i>DigitalFile</i> – Digital data irrespective of the delivery mechanism. <u>The union of <i>DigitalMedia</i> and <i>DigitalNetwork</i>.</u> New in JDF 1.2 <i>DigitalMedia</i> – Digital data on media, (e.g., a CD). <i>DigitalNetwork</i> – Digital data via network. <i>ExposedPlate</i> – Pre-exposed press plates, usually used for a rerun. <i>ImposedFilm</i> – Film of the imposed surfaces. <i>LooseFilm</i> – Film of individual pages or sections. <i>OriginalPhysicalArt</i> – Analog artwork, (e.g., reflective or transparencies). <i>Proof</i> – Physical proof delivered with digital scan or separated film asset. <i>Tool</i> – Tools needed for processing the job, (e.g., a die for die cutting or embossing stamp). <i>None</i> – No artwork exists, and it will be created later.

Table 7-15: ArtDelivery element (Section 2 of 3)

Name	Data Type	Description
ArtHandling ? New in JDF 1.1	Enumeration-Span	Describes what is to happen to the artwork after usage. The address for the <i>Return</i> and <i>Pickup</i> values MUST be specified by <i>Contact</i> [contains (<i>@ContactTypes</i> , "ArtReturn")]/ <i>Address</i> . Possible values are: <i>ReturnWithProof</i> – The artwork is delivered back to the customer together with the proof if there is any. <i>ReturnWithProduct</i> – The artwork is delivered back to the customer together with the final product. <i>Return</i> – The artwork is delivered back independently directly after usage. <i>Pickup</i> – The customer picks up the artwork. <i>Destroy</i> – The printer destroys the artwork. <i>PrinterOwns</i> – The artwork belongs to the printer. <i>Store</i> – The printer has to store the artwork for future purposes. Defaults to the value of ArtDeliveryIntent/@ArtHandling .
DeliveryCharge ? New in JDF 1.1 Modified in JDF 1.3	Enumeration-Span	Specifies who pays for a delivery being made by a third party. Possible values are specified in DeliveryIntent/@DeliveryCharge . Defaults to the value of ArtDeliveryIntent/@DeliveryCharge .
<i>HasBleeds</i> = "false"	boolean	If "true", the file has bleeds.
<i>IsTrapped</i> = "false"	boolean	If "true", the file has been trapped.
Method ? Modified in JDF 1.2	NameSpan	Specifies a delivery method. It MAY be a generic item from the list defined in <i>Method</i> in ArtDeliveryIntent . Defaults to the value of ArtDeliveryIntent/@Method .
PageList ?	IntegerRange-List	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 <i>PageList</i> = "3 ~ 5", page 0 of the <i>ArtDelivery</i> (e.g., <i>RunList</i>) is page 3 in the product, page 1 is page 4, etc. If not specified, the <i>PageList</i> MUST include all pages in reader order. The indices specified in <i>PageList</i> reference the <i>PageData</i> elements defined in PageList .
PreflightOutput ? New in JDF 1.1	URL	Pointer to the output information created by the preflight tool if <i>PreflightStatus</i> is either <i>WithoutErrors</i> or <i>WithErrors</i> .
PreflightStatus ? New in JDF 1.1	enumeration	Information about a preflight process. The values are identical to those of <i>PreflightStatus</i> in ArtDeliveryIntent . Defaults to the value of ArtDeliveryIntent/@PreflightStatus .
ReturnMethod ? New in JDF 1.1	NameSpan	Specifies a delivery method for returning the artwork if <i>ArtHandling</i> = "Return". Defaults to the value of ArtDeliveryIntent/@ReturnMethod .
ServiceLevel ? New in JDF 1.2	StringSpan	The service level of the specific carrier. Contain values "Next Day", "2nd Day Air", "Ground", etc. Defaults to the value of ArtDeliveryIntent/@ServiceLevel .
Transfer ? New in JDF 1.1	Enumeration-Span	Describes the responsibility of the transfer. The values are identical to those of <i>Transfer</i> in ArtDeliveryIntent . Defaults to the value of ArtDeliveryIntent/@Transfer .

Table 7-15: ArtDelivery element (Section 3 of 3)

Name	Data Type	Description
Company ? Deprecated in JDF 1.1	refelement	Address and further information about the art delivery. This MUST 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 <i>ArtDeliveryType</i> and <i>Amount</i> are known.
Contact * New in JDF 1.1	refelement	Address and further information about the art transfer. Defaults to the value of 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 <i>ArtDeliveryType</i> and <i>Amount</i> 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 <i>ArtDeliveryType</i> and <i>Amount</i> , 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 <i>ArtDeliveryType</i> and <i>Amount</i> 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 <i>ArtDeliveryType</i> = "Tool". If neither ExposedMedia , RunList , DigitalMedia , nor Tool are specified, no details of the ArtDelivery except the <i>ArtDeliveryType</i> and <i>Amount</i> are known.

7.1.3 BindingIntent

This resource specifies the binding intent for a JDF job using information that identifies the desired type of binding and which side is to be bound. The input components that are used as a cover MUST have a *ProcessUsage* of *Cover*. The input components that are used as a hard cover jacket MUST have a *ProcessUsage* of *Jacket*. The input components that are used as end sheets for hardcover or soft cover binding MUST have a *ProcessUsage* of *EndSheet*. All other input components are bound in the order of their appearance in the ResourceLinkPool of the JDF node that contains the **BindingIntent**.

Resource Properties

Resource class: Intent

Resource referenced by: —

Process Resource Pairing: **BlockPreparationParams, CaseMakingParams, CasingInParams, ChannelBindingParams, CoilBindingParams, CoverApplicationParams, EndSheetGluingParams, GlueApplication, GluingParams, GlueLine, InsertingParams, JacketingParams, PlasticCombBindingParams, RingBindingParams, SaddleStitchingParams, SpinePreparationParams, SpineTapingParams, StitchingParams, StripBindingParams, ThreadSealingParams, ThreadSewingParams, WireCombBindingParams**

Example Partition: *Option*

Input of processes: Any product node

Output of processes: —

Table 7-16: BindingIntent resource (Section 1 of 2)

Name	Data Type	Description
<i>BackCoverColor</i> ? New in JDF 1.1	EnumerationSpan	Defines the color of the back cover material of the binding. Allowed values are defined in Section A.3.3.2, NamedColor. If not specified, it defaults to the value of <i>CoverColor</i> .
<i>BindingColor</i> ?	EnumerationSpan	Defines the color of the spine material of the binding. Allowed values are defined in Section A.3.3.2, NamedColor.
<i>BindingLength</i> ?	EnumerationSpan	Indicates which side is to be bound when no content. Thus, no orientation is available, but a quote for binding is needed. Possible values are: <i>Long</i> <i>Short</i>
<i>BindingOrder</i> = "Gathering" New in JDF 1.1	enumeration	Specifies whether the child Component resources are to be collected or gathered if multiple child Component resources are combined. One of: <i>Collecting</i> – The child Component resources are collected on a spine and placed within one another. The first Component is on the outside. <i>Gathering</i> – The child Component resources are gathered on a pile and placed on top of one another. The first Component is on the top. <i>List</i> – More complex ordering of child Component resources is specified using the BindList in this intent resource for this product.
<i>BindingSide</i> ?	EnumerationSpan	Indicates which side are to be bound. Possible values are: <i>Top</i> <i>Bottom</i> <i>Right</i> <i>Left</i> Each of these values is intended to identify an edge of the job. These edges are defined relative to the orientation of the first page in the job with content on it. Default = <i>BindingLength</i> value, unless a non-empty BindList was specified. If both <i>BindingSide</i> and <i>BindingLength</i> are specified, <i>BindingSide</i> has precedence.
<i>BindingType</i> Modified in JDF 1.2	EnumerationSpan	Describes the desired binding for the job. See Table 7-17 for values of <i>BindingType</i> .
<i>CoverColor</i> ?	EnumerationSpan	Defines the color of the cover material of the binding. Allowed values are defined in Section A.3.3.2, NamedColor.
<i>AdhesiveBinding</i> ? Deprecated in JDF 1.1	element	Details of AdhesiveBinding. Replaced with SoftCoverBinding in JDF 1.1.
<i>BindList</i> ? New in JDF 1.1	element	Details of binding of individual child Component resources.
<i>BookCase</i> ? 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.
<i>ChannelBinding</i> ?	element	Details of ChannelBinding.

Table 7-16: BindingIntent resource (Section 2 of 2)

Name	Data Type	Description
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.
Tape ? New in JDF 1.1	element	Details of Tape binding.
Tabs ?	element	Details of Tabs.
ThreadSealing ?	element	Details of ThreadSealing.
ThreadSewing ?	element	Details of ThreadSewing.
StripBinding ? New in JDF 1.1	element	Details of StripBinding.
VeloBinding ? Removed in JDF 1.1	element	Details of VeloBinding. Renamed to StripBinding in JDF 1.1.
WireCombBinding ?	element	Details of WireCombBinding.

— Attribute: BindingType**Table 7-17: BindingType attribute – possible values (Section 1 of 2)**

Value	Description
<i>Adhesive</i>	This type of binding can be handled with the AdhesiveBinding process. It includes perfect binding. Deprecated in JDF 1.1 and replaced with <i>SoftCover</i> or <i>HardCover</i> .
<i>ChannelBinding</i>	This type of binding can be handled with the ChannelBinding process.
<i>CoilBinding</i>	This type of binding can be handled with the CoilBinding process.
<i>CornerStitch</i>	Stitch in the corner that is at the clockwise end binding edge. For example, to stitch in the top left corner, set <i>BindingSide</i> = "Left". This type of binding can be handled with the Stitching process. New in JDF 1.2
<i>EdgeGluing</i>	Gluing gathered sheets at one edge of the pile. This Type of Binding can be handled with the <i>Gluing</i> process.
<i>HardCover</i>	This type of binding defines a hard-cover bound book.
<i>LooseBinding</i>	This type of binding defines a stack of pages with no additional binding.
<i>PlasticComb</i>	This type of binding can be handled with the PlasticCombBinding process.
<i>Ring</i>	This type of binding can be handled with the RingBinding process.
<i>SaddleStitch</i>	This type of binding can be handled with the Stitching process.
<i>Sewn</i>	This type of binding can be handled with the ThreadSewing process.
<i>SideSewn</i>	This type of binding can be handled with the ThreadSewing process.
<i>SideStitch</i>	This type of binding can be handled with the Stitching process.

Table 7-17: BindingType attribute – possible values (Section 2 of 2)

Value	Description
<i>SoftCover</i>	This type of binding defines a soft cover bound book. It includes perfect binding.
<i>StripBind</i>	This type of binding can be handled with the StripBinding process.
<i>Tape</i>	This type of binding is an inexpensive version of the <i>SoftCover</i> .
<i>ThreadSealing</i>	This type of binding can be handled with the ThreadSealing process.
<i>WireComb</i>	This type of binding can be handled with the WireCombBinding process.

— Element: 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 7-18: BindList element

Name	Data Type	Description
BindItem *	element	Individual bind item description. Defaults to BindingIntent/BindingSide value if empty, (i.e., as if the BindList element weren't there).

— Element: 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 7-19: BindItem element (Section 1 of 2)

Name	Data Type	Description
<i>BindingType</i> ?	EnumerationSpan	Describes the desired binding for the individual BindItem . The list of possible values is defined in BindingIntent/@BindingType . Defaults to the value of BindingIntent/@BindingType .
<i>ChildFolio</i> ?	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 <i>ChildFolio</i> 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.
<i>ParentFolio</i>	XYPair	Definition of the fold between two pages in the Cover Component that receive the BindItem . The two numbers (as integers) in the <i>ParentFolio</i> 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.
<i>Transformation</i> ?	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
<i>WrapPages</i> ?	IntegerRangeList	List of pages of the Cover that wrap around a BindItem after all folds are correctly positioned. It is sufficient to specify the pages of the <i>Front</i> surface of the cover. Note that this key MUST NOT be specified unless the folding is ambiguous.

Table 7-19: BindItem element (Section 2 of 2)

Name	Data Type	Description
BookCase ? Deprecated in JDF 1.1	element	Details of the hard cover Book Case. Used in Combination with HardCoverBinding .
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 .
Tape ?	element	Details of Tape binding.
Tabs ?	element	Details of Tabs .
ThreadSealing ?	element	Details of ThreadSealing .
ThreadSewing ?	element	Details of ThreadSewing .
WireCombBinding ?	element	Details of WireCombBinding .

— Element: AdhesiveBinding[Deprecated in JDF 1.1](#)

The table defining the deprecated [AdhesiveBinding](#) subelement has been moved to "BindingIntent Deprecated Subelements" on page 831.

— Element: BookCase[Deprecated in JDF 1.1](#)

The table defining the deprecated [BookCase](#) subelement has been moved to "BindingIntent Deprecated Subelements" on page 831.

— Element: ChannelBinding**Table 7-20: ChannelBinding element**

Name	Data Type	Description
ChannelBrand ? New in JDF 1.3	StringSpan	Strings providing available brand names for the ChannelBinding .
Cover ?	OptionSpan	If " <i>true</i> ", 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.

— Element: **CoilBinding**Table 7-21: **CoilBinding** element

Name	Data Type	Description
CoilBrand ? New in JDF 1.3	StringSpan	Strings providing available brand names for the coil.
CoilMaterial ?	EnumerationSpan	The coil materials available for CoilBinding . Possible values are: <i>Steel</i> – Plain steel. <i>ColorCoatedSteel</i> – Coated steel. <i>Plastic</i> – any kind of plastic.
HoleList ? New in JDF 1.2	refelement	Details of the holes for coil binding.

— Element: **EdgeGluing**[New in JDF 1.1](#)Table 7-22: **EdgeGluing** element

Name	Data Type	Description
EdgeGlue ?	EnumerationSpan	Glue type used to glue the edge of the gathered sheets. Possible values are: <i>ColdGlue</i> <i>Hotmelt</i> <i>PUR</i> – Polyurethane rubber.

— Element: **HardCoverBinding**[New in JDF 1.1](#)Table 7-23: **HardCoverBinding** element (Section 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: <i>Simple</i> – Single layer cover board, see Figure 7-1 <i>Padded</i> – Padded cover board, see Figure 7-2
EndSheets ?	OptionSpan	Specified if end sheets are applied. Additional details of the EndSheets MAY be specified by supplying an input Component with <i>ProcessUsage</i> = "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 ?	EnumerationSpan	Defines the color of the headband. Allowed values are defined in Section A.3.3.2, NamedColor.
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 <i>ProcessUsage</i> = "Jacket". Possible values: <i>None</i> – No jacket is needed. <i>Loose</i> – The jacket is loosely wrapped. <i>Glue</i> – The jacket is glued to the spine

Table 7-23: HardCoverBinding element (Section 2 of 2)

Name	Data Type	Description
<i>JacketFoldingWidth</i> ? New in JDF 1.3	NumberSpan	Dimension of the jacket folds. See JacketingParams for details.
<i>JapanBind</i> ?	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.
<i>SpineBrushing</i> ?	OptionSpan	Brushing option for SpinePreparation .
<i>SpineFiberRoughing</i> ?	OptionSpan	Fiber roughing option for SpinePreparation .
<i>SpineGlue</i> ?	EnumerationSpan	Glue type used to glue the book block to the cover. Possible values are: <i>ColdGlue</i> <i>Hotmelt</i> <i>PUR</i> – Polyurethane rubber.
<i>SpineLevelling</i> ?	OptionSpan	Leveling option for SpinePreparation .
<i>SpineMilling</i> ?	OptionSpan	Milling option for SpinePreparation .
<i>SpineNotching</i> ?	OptionSpan	Notching option for SpinePreparation .
<i>SpineSanding</i> ?	OptionSpan	Sanding option for SpinePreparation .
<i>SpineShredding</i> ?	OptionSpan	Shredding option for SpinePreparation .
<i>StripMaterial</i> ?	EnumerationSpan	Spine taping strip material. Possible values are: <i>Calico</i> <i>Cardboard</i> <i>CrepePaper</i> <i>Gauze</i> <i>Paper</i> <i>PaperlinedMules</i> <i>Tape</i>
<i>Thickness</i> ?	NumberSpan	Specifies thickness of board which is wrapped as front and back covers of a case bound book, in points.
<i>TightBacking</i> ?	EnumerationSpan	Definition of the geometry of the back of the book block. This can be one of: <i>Flat</i> – A flat backing. <i>Round</i> – Rounding way. <i>FlatBacked</i> – Backing way. <i>RoundBacked</i> – Rounding way, backing way.
RegisterRibbon *	refelement	Number, materials, colors and details of register ribbons.

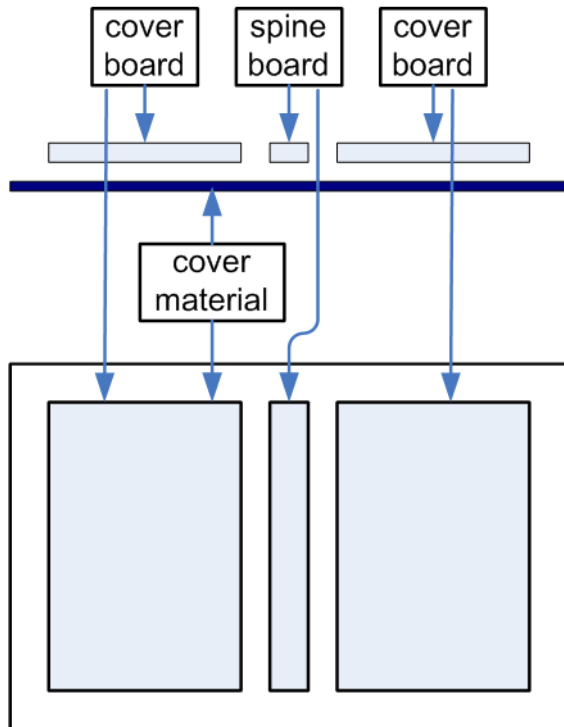


Figure 7-1: Structure of a normal hardcover book

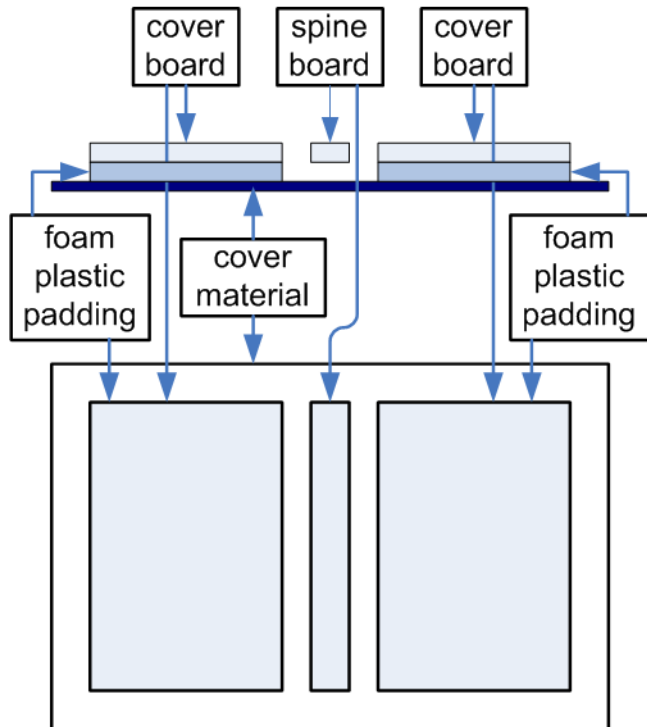


Figure 7-2: Structure of a padded hardcover book

— Element: PlasticCombBinding

Table 7-24: PlasticCombBinding element

Name	Data Type	Description
<u>CombBrand ?</u> <u>New in JDF 1.3</u>	StringSpan	Strings providing available brand names for the plastic comb.
<u>PlasticCombType ?</u> <u>Modified in JDF 1.1</u>	NameSpan	The distance between the “teeth” in PlasticCombBinding and the distance between the holes of the prepunched leaves MUST be the same. The following values from the hole type catalog in "JDF/CIP4 Hole Pattern Catalog" on page 761 exist: <i>P12m-rect-02</i> – Distance = 12 mm; Holes = 7 mm x 3 mm <i>P16_gi-rect-0t</i> – Distance = 14.28 mm; Holes = 8 mm x 3 mm The following values are <u>deprecated in JDF 1.1</u> . <i>Euro</i> – Distance = 12 mm; Holes = 7 mm x 3 mm <i>USA1</i> – Distance = 14.28 mm; Holes = 8 mm x 3 mm
<u>HoleList ?</u> <u>New in JDF 1.2</u>	element	Details of the holes for the plastic comb. Note that <i>Shape</i> is always rectangular by design of the plastic combs.

— Element: RingBinding

Table 7-25: RingBinding element

Name	Data Type	Description
<i>BinderBrand</i> ? New in JDF 1.3	StringSpan	Strings providing available brand names for RingBinding.
<i>BinderMaterial</i> ?	NameSpan	The following describe RingBinding binder materials used. Values include: <i>Cardboard</i> – Cardboard with no covering. <i>ClothCovered</i> – Cardboard with cloth covering. <i>Plastic</i> – Binder cover fabricated from solid plastic sheet material, (e.g., PVC sheet). <i>VinylCovered</i> – Cardboard with colored vinyl covering.
<i>HoleType</i> ? New in JDF 1.1	EnumerationSpan	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 and a list of allowed values, refer to "JDF/CIP4 Hole Pattern Catalog" on page 761.
<i>RingDiameter</i> ?	NumberSpan	Size of the rings in points. The value used in production MUST be suitable for specified values of <i>HoleType</i> . Note that in ring shapes other than round, this size is specified by industry-standard method.
<i>RingMechanic</i> ?	OptionSpan	The ring binder used includes a lever for opening and closing.
<i>RingShape</i> ?	NameSpan	The following RingBinding shapes are used: <i>Round</i> <i>Oval</i> <i>D-shape</i> <i>SlantD</i>
<i>RingSystem</i> ? Deprecated in JDF 1.1	NameSpan	<i>2HoleEuro</i> <i>3HoleUS</i> <i>4HoleEuro</i> <i>RingSystem</i> have been replaced by <i>HoleType</i> .
<i>RivetsExposed</i> ?	OptionSpan	The following RingBinding choice describes mounting of the ring mechanism in binder case. If " <i>true</i> ", the heads of the rivets are visible on the exterior of the binder. If " <i>false</i> ", the binder covering material covers the rivet heads.
<i>ViewBinder</i>	NameSpan	The following RingBinding clear vinyl outer wrap types are used on top of a colored base wrap: <i>Embedded</i> – Printed material is embedded by sealing between the colored and clear vinyl layers during binder manufacturing. <i>Pocket</i> – Binder is designed so that printed material can be inserted between the color and clear vinyl layers after binder manufacturing.

— Element: **SaddleStitching**

Table 7-26: SaddleStitching element

Name	Data Type	Description
<i>StitchNumber</i> ? New in JDF 1.1	IntegerSpan	Number of stitches used for saddle stitching.

— Element: **SideSewing**

This is a placeholder that might be filled with private or future data.

Table 7-27: SideSewing element

Name	Data Type	Description

— Element: **SideStitching**

Table 7-28: SideStitching element

Name	Data Type	Description
<i>StitchNumber</i> ? New in JDF 1.2	IntegerSpan	Number of stitches used for side stitching.

— Element: **SoftCoverBinding**

[New in JDF 1.1](#)

Table 7-29: SoftCoverBinding element (Section 1 of 2)

Name	Data Type	Description
<i>BlockThreadSewing</i> ?	OptionSpan	Specifies whether the block is also thread sewn.
<i>EndSheets</i> ? New in JDF 1.3	OptionSpan	Specified if end sheets are applied. Additional details of the <i>EndSheets</i> MAY be specified by supplying an input Component with <i>ProcessUsage</i> = "EndSheet".
<i>FoldingWidth</i> ? New in JDF 1.3	NumberSpan	Definition of the dimension of the folding width of the front cover fold. See JacketingParams for details.
<i>FoldingWidthBack</i> ? New in JDF 1.3	NumberSpan	Definition of the dimension of the folding width of the back cover fold. If not specified, <i>FoldingWidthBack</i> defaults to <i>FoldingWidth</i> .
<i>GlueProcedure</i> ?	EnumerationSpan	Glue procedure used to glue the book block to the cover. Possible values are: <i>Spine</i> <i>SideOnly</i> – Glued at the side or endsheets but not at the spine. <i>SideOnly</i> books are also referred to as “layflat” if <i>EndSheets</i> are also specified. See Figure "Structure of a book with GlueProcedure = “SideOnly” (Layflat)" on page 292 <i>SingleSide</i> – Swiss Brochure. <i>SideSpine</i> – Both side gluing and spine gluing.

Table 7-29: SoftCoverBinding element (Section 2 of 2)

Name	Data Type	Description
<i>Scoring ?</i>	EnumerationSpan	Scoring option for SoftCoverBinding . Possible values are: <i>TwiceScored</i> <i>QuadScored</i> <i>None</i> Values are based on viewing the cover in its flat, prebound state.
<i>SpineBrushing ?</i>	OptionSpan	Brushing option for SpinePreparation .
<i>SpineFiberRoughing ?</i>	OptionSpan	FiberRoughing option for SpinePreparation .
<i>SpineGlue ?</i>	EnumerationSpan	Glue type used to glue the book block to the cover. Possible values are: <i>ColdGlue</i> <i>Hotmelt</i> <i>PUR</i> – Polyurethane rubber.
<i>SpineLevelling ?</i>	OptionSpan	Leveling option for SpinePreparation .
<i>SpineMilling ?</i>	OptionSpan	Milling option for SpinePreparation .
<i>SpineNotching ?</i>	OptionSpan	Notching option for SpinePreparation .
<i>SpineSanding ?</i>	OptionSpan	Sanding option for SpinePreparation .
<i>SpineShredding ?</i>	OptionSpan	Shredding option for SpinePreparation .

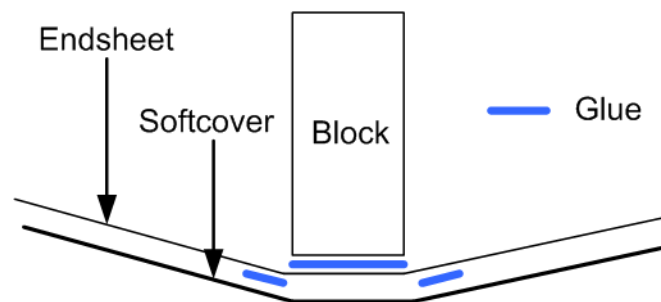


Figure 7-3: Structure of a book with GlueProcedure = "SideOnly" (Layflat)

— Element: StripBinding

[New in JDF 1.1](#)

Table 7-30: StripBinding element

Name	Data Type	Description
HoleList ? New in JDF 1.2	refelement	Note that <i>Shape</i> is always round by design of the strip poles.

— Element: Tape[New in JDF 1.1](#)**Table 7-31: Tape element**

Name	Data Type	Description
<i>TabColor</i> ?	EnumerationSpan	Defines the color of the tape material of the binding. Allowed values are defined in Section A.3.3.2, NamedColor.

— Element: Tabs

Specifies tabs in a bound document.

Table 7-32: Tabs element

Name	Data Type	Description
<i>TabBanks</i> = "1"	Integer	Number of rows of tabs on the face of the book.
<i>TabBrand</i> ? New in JDF 1.3	StringSpan	Strings providing available brand names for the Tabs.
<i>TabsPerBank</i> ?	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
<i>TabExtensionDistance</i> ?	NumberSpan	Distance tab extends beyond the body of the book block, in points.
<i>TabExtensionMylar</i> ?	OptionSpan	If "true", the tab extension will be mylar reinforced.
<i>TabBindMylar</i> ?	OptionSpan	If "true", the tab bind edge will be mylar reinforced.
<i>TabBodyCopy</i> ?	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.
<i>TabMylarColor</i> ?	EnumerationSpan	Specifies the color of the mylar used to reinforce the tab extension. This is conditional on <i>TabExtensionMylar</i> being "true". Allowed values are defined in Section A.3.3.2, NamedColor.

— Element: ThreadSealing

This is a placeholder that might be filled with private or future data.

Table 7-33: ThreadSealing element

Name	Data Type	Description

— Element: ThreadSewing.**Table 7-34: ThreadSewing element**

Name	Data Type	Description
<i>Sealing</i> ?	OptionSpan	If "true", thermo-sealing is needed in ThreadSewing .

— Element: **WireCombBinding**Table 7-35: **WireCombBinding** element

Name	Data Type	Description
<i>WireCombBrand</i> ? New in JDF 1.3	StringSpan	Strings providing available brand names for the WireCombBinding .
<i>WireCombMaterial</i> ?	EnumerationSpan	The material used for forming the WireCombBinding . Possible values are: <i>Steel-Silver</i> <i>ColorCoatedSteel</i>
<i>WireCombShape</i> ?	EnumerationSpan	The shape of the WireCombBinding . Possible values are: <i>Single</i> – Each “tooth” is made with one wire. <i>Twin</i> – The shape of each “tooth” is made with a double wire, (e.g., Wire-O).
HoleList ? New in JDF 1.2	refelement	Details of the holes for the wire comb.

7.1.4 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.

Resource Properties

Resource class: Intent

Resource referenced by: —

Process Resource Pairing: **Color, ColorantControl, ColorCorrectionParams, ColorPool, ColorSpaceConversionParams**

Example Partition: *Option, PageNumber, Side*

Input of processes: Any product node

Output of processes: —

Table 7-36: **ColorIntent** resource (Section 1 of 3)

Name	Data Type	Description
<i>Coatings</i> ? Modified in JDF 1.3	StringSpan	Material usually applied to a full surface on press as a protective or gloss-enhancing layer over ink. Possible values include: <i>Aqueous</i> <i>DullVarnish</i> <i>GlossVarnish</i> <i>Protective</i> <i>SatinVarnish</i> <i>Silicone</i> <i>UV</i> The individual strings within <i>Coatings</i> are of type NMTOKENS and MAY contain multiple entries from the above list. Note that spot coating is specified in ColorsUsed .

Table 7-36: ColorIntent resource (Section 2 of 3)

Name	Data Type	Description
ColorStandard? Modified in JDF 1.2	NameSpan	The color process (i.e., printing condition) requested for the job. <i>ColorStandard</i> does not imply values for <i>ColorsUsed</i> . For instance, if <i>ColorStandard</i> is <i>CMYK</i> , <i>ColorsUsed</i> MUST still contain the four process colors <i>Cyan</i> , <i>Magenta</i> , <i>Yellow</i> and <i>Black</i> . If both of <i>ColorICCStandard</i> and <i>ColorStandard</i> are specified, then <i>ColorICCStandard</i> defines the ICC specific details, whereas <i>ColorStandard</i> defines the generic color standard. See Table 7-36 for values of <i>ColorStandard</i> :
ColorICCStandard? New in JDF 1.2	StringSpan	<i>ColorICCStandard</i> 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 OutputIntent 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: <i>FOGRA11</i> – 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. <i>FOGRA15</i> – 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. <i>CGATS TR001</i> – 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 <i>ColorICCStandard</i> or <i>ColorsUsed</i> are specified, the union of the two is specified. If both of <i>ColorICCStandard</i> and <i>ColorStandard</i> are specified, then <i>ColorICCStandard</i> defines the ICC specific details, whereas <i>ColorStandard</i> defines the generic color standard.
Coverage?	NumberSpan	Cumulative colorant coverage percentage. For example, a full sheet of 100% deep black in CMYK has <i>Coverage</i> = "400". Typical coverages based on one color plane are: <i>Light</i> = 1-9% <i>Medium</i> = 10-35% <i>Heavy</i> = 36+%
InkManufacturer? Deprecated in JDF 1.2	NameSpan	Name of the manufacturer of the ink requested, (e.g., "ACMEInk", "CIP4_Ink_Company" etc.).

Table 7-36: ColorIntent resource (Section 3 of 3)

Name	Data Type	Description
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 <i>ColorStandard</i> process colors.
ColorsUsed ?	element	Array of colorant separation names that are requested. If not specified, the values are implied from <i>ColorStandard</i> . If specified, ColorsUsed MUST 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.

— Attribute: ColorStandard

Table 7-37: ColorStandard attribute – possible values

Value	Description
<i>CMYK</i>	Generic four color process.
<i>FIRST</i>	Flexographic Image Reproduction Specifications & Tolerances.
<i>GRACOL</i>	General Requirements for Applications in Commercial Offset Lithography
<i>Hexachrome</i>	6 Colors <i>CMYK</i> + <i>Orange</i> and <i>Green</i> .
<i>HIFI</i>	7 Colors <i>CMYK</i> + <i>Red</i> , <i>Green</i> and <i>Blue</i> .
<i>ISO12647</i>	[ISO12647-2:2004] offset standard. Deprecated in JDF 1.2
<i>JapanColor2001</i>	Japan Color 2001 standard [japancolor].
<i>Monochrome</i>	Generic single color printing condition, (e.g., black and white or one single spot color).
<i>None</i>	No marks. Used to define one-sided printing. Deprecated in JDF 1.2 (Use LayoutIntent/@Sides instead.)
<i>SNAP</i>	Specifications for Newsprint Advertising Production
<i>SWOP</i>	Specifications for Web Offset Publications. Registered by ANSI with the ICC as <i>ICC:CGATS TR001</i> pertaining to printing conditions that conform to ANSI CGATS.6 which is based on Publication printing in the US as defined by SWOP, Inc.

— Element: ColorsUsed

Table 7-38: ColorsUsed element

Name	Data Type	Description
SeparationSpec * Modified in JDF 1.2	element	<p>These can be process colors, generic spot colors or named spot colors.</p> <p>In addition, partial (spot) coating is specified by adding a SeparationSpec with anything from <i>Coatings</i> as <i>Name</i>:</p> <p><i>Aqueous</i></p> <p><i>Bronzing</i></p> <p><i>DullVarnish</i></p> <p><i>GlossVarnish</i></p> <p><i>SatinVarnish</i></p> <p><i>Silicone</i></p> <p><i>Spot</i> – Generic spot color of which the details are unknown. <i>Spot</i> MAY be specified multiple times in one ColorsUsed element. New in JDF 1.2</p> <p><i>UV</i></p> <p><i>Varnish</i> – Generic varnish including <i>DullVarnish</i>, <i>GlossVarnish</i> and <i>SatinVarnish</i>. New in JDF 1.3</p>

7.1.5 DeliveryIntent

Summarizes the options that describe pickup or delivery time and location of the physical resources 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
Resource referenced by:	—
Process Resource Pairing:	Address, DeliveryParams
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

Table 7-39: DeliveryIntent resource (Section 1 of 4)

Name	Data Type	Description
<i>Accepted</i> = "false"	boolean	The quote that is specified by this DeliveryIntent has been accepted.
<i>AdditionalAmount</i> = "1" New in JDF 1.2 Deprecated in JDF 1.3	integer	Number of components used to calculate the value of the <i>AdditionalPrice</i> 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 .
<i>BuyerAccount</i> ?	string	Account ID of the buyer with the delivery service.

Table 7-39: DeliveryIntent resource (Section 2 of 4)

Name	Data Type	Description
DeliveryCharge ? New in JDF 1.1 Modified in JDF 1.2	EnumerationSpan	Specifies who pays for a delivery being made by a third party. Possible values are: <i>Printer</i> – The <i>Printer</i> is defined as the person who creates the resource that is delivered. This includes all suppliers, (e.g., binders, prepress suppliers, etc.). <i>Buyer</i> – The customer specified in CustomerInfo . <i>Other</i> – The Contact [@ContactTypes = "DeliveryCharge"]. New in JDF 1.2
Earliest ?	TimeSpan	Specifies the earliest time after which the transfer is to be made. Within an RFQ or a Quote, at most one of <i>Earliest</i> or <i>EarliestDuration</i> MUST be specified.
EarliestDuration ?	DurationSpan	Specifies the earliest time by which the transfer is to be made relative to the date of the purchase order. Within an RFQ or a Quote, at most one of <i>Earliest</i> or <i>EarliestDuration</i> MUST be specified. Within a purchase order, <i>EarliestDuration</i> MUST NOT be specified.
Method ?	NameSpan	Specifies a delivery method, which can be a generic method. Possible values are: <i>BestWay</i> – The sender decides how to deliver. <i>CompanyTruck</i> <i>Courier</i> <i>Email</i> <i>ExpressMail</i> <i>InterofficeMail</i> <i>Storage</i> – The product is stored by the supplier. <i>OvernightService</i> Values also include a delivery service brand, for example: <i>UPS</i> <i>DHL</i> <i>FedEx</i>
Ownership = "Origin"	enumeration	Point of transfer of ownership: <i>Origin</i> – Ownership of goods is transferred upon leaving point of origin. <i>Destination</i> – Ownership is transferred upon receipt at destination.
Overage ?	NumberSpan	Percentage value that defines the acceptable upwards variation of <i>Amount</i> . Defaults to the trade custom defaults as defined by PIA, BVD, etc.
Pickup ? Deprecated in JDF 1.1	boolean	Specifies whether the delivery brings or picks up the merchandise. If <i>Pickup</i> = "false", the drop is delivered to the address specified in Company . If <i>Pickup</i> = "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 is to be made. Within an RFQ or a Quote, exactly one of <i>Required</i> or <i>RequiredDuration</i> MUST be specified.

Table 7-39: DeliveryIntent resource (Section 3 of 4)

Name	Data Type	Description
RequiredDuration ?	DurationSpan	Specifies the time by which the transfer is to be made relative to the date of the purchase order. Within an RFQ or a Quote, exactly one of <i>Required</i> or <i>RequiredDuration</i> MUST be specified. Within a purchase order, <i>RequiredDuration</i> MUST NOT be specified.
ReturnMethod ? New in JDF 1.1	NameSpan	Specifies a delivery method for returning the surplus material and MUST NOT be specified unless <i>SurplusHandling</i> = "Return". The values are the same as <i>Method</i> .
ServiceLevel ? New in JDF 1.2	StringSpan	The service level of the specific carrier. Contain values "Next Day", "2nd Day Air", "Ground", etc.
SurplusHandling ? New in JDF 1.1	EnumerationSpan	Describes what is to happen with unused or redundant parts of the transfer specified with <i>Transfer</i> = "BuyerToPrinterDeliver" or "BuyerToPrinterPickup" after the job. The return delivery or pickup address is specified by Contact [contains (@ContactTypes, "SurplusReturn")]. Possible values are: <i>ReturnWithProduct</i> – The surplus material is delivered back to the customer together with the final product. <i>Return</i> – The surplus material is delivered back independently directly after usage. <i>Pickup</i> – The customer picks up the surplus material. <i>Destroy</i> – The printer destroys the surplus material. <i>PrinterOwns</i> – The surplus material belongs to the printer. <i>Store</i> – The printer has to store the surplus material for future purposes.
Transfer ? New in JDF 1.1	EnumerationSpan	Describes the direction and responsibility of the transfer. Possible values are: <i>BuyerToPrinterDeliver</i> – The DeliveryIntent 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 is to specify in the quote a special Contact [contains (@ContactTypes, "Delivery")]. The Contact specifies where the buyer is to send the merchandise. <i>BuyerToPrinterPickup</i> – The DeliveryIntent 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. The Contact [contains (@ContactTypes, "Pickup")] specifies where the printer has to pick up the merchandise. <i>PrinterToBuyerDeliver</i> – The DeliveryIntent describes an output of the job. In this case, the printer delivers the merchandise to the buyer. The Contact [contains (@ContactTypes, "Delivery")] specifies where the printer is to send the merchandise. <i>PrinterToBuyerPickup</i> – The DeliveryIntent describes an output of the job. In this case, the buyer picks up the merchandise. The printer is to specify in the quote a special Contact [contains (@ContactTypes, "Pickup")]. The Contact specifies where the buyer is to pick up the merchandise.

Table 7-39: DeliveryIntent resource (Section 4 of 4)

Name	Data Type	Description
<i>Underage</i> ?	NumberSpan	Percentage value that defines the acceptable downwards variation of <i>Amount</i> . Defaults to the trade custom defaults as defined by PIA, BVD, etc.
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 (@ <i>ContactTypes</i> , " <i>Delivery</i> ")]/ <i>Address</i> . The actual pickup address is specified as the Contact [contains (@ <i>ContactTypes</i> , " <i>Pickup</i> ")]/ <i>Address</i> . At most one Contact [contains (@ <i>ContactTypes</i> , <i>X</i>)] MUST be specified for <i>X</i> equal to " <i>Delivery</i> ", " <i>Pickup</i> " or " <i>Billing</i> ",
DropIntent +	element	Includes all locations where the product will be delivered. Note that multiple DropIntent elements specify multiple deliveries and not options for delivery.
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.

— Element: DropIntent

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.

Table 7-40: DropIntent element (Section 1 of 2)

Name	Data Type	Description
<i>AdditionalAmount</i> ? New in JDF 1.2 Deprecated in JDF 1.3	integer	Number of components used to calculate the value of the <i>AdditionalPrice</i> 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 /@ <i>AdditionalAmount</i> . In JDF 1.3 and beyond, pricing information has been removed and will be handled by the business wrapper around JDF, e.g. PrintTalk.
<i>BuyerAccount</i> ? New in JDF 1.2	string	Account ID of the buyer with the delivery service. Defaults to the value of DeliveryIntent /@ <i>BuyerAccount</i> .
<i>Earliest</i> ?	TimeSpan	Specifies the earliest time after which the transfer is to be made. Within an RFQ or a Quote, at most one of <i>Earliest</i> or <i>EarliestDuration</i> MUST be specified.
<i>EarliestDuration</i> ?	DurationSpan	Specifies the earliest time by which the transfer is to be made relative to the date of the purchase order. Within an RFQ or a Quote, at most one of <i>Earliest</i> or <i>EarliestDuration</i> MUST be specified. Within a purchase order, <i>EarliestDuration</i> MUST NOT be specified.
<i>Method</i> ?	NameSpan	Specifies a delivery method. Possible values are the same as DeliveryIntent /@ <i>Method</i> . If <i>Method</i> is not specified, then a Consumer MUST use the value of DeliveryIntent /@ <i>Method</i> .

Table 7-40: DropIntent element (Section 2 of 2)

Name	Data Type	Description
Pickup ? Deprecated in JDF 1.1	boolean	If <i>true</i> , the merchandise is picked up. If <i>Pickup</i> = <i>false</i> , the DropIntent is delivered to the address specified in Company . If <i>Pickup</i> = <i>true</i> , 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 is to be made. Within an RFQ or a Quote, at most one of <i>Required</i> or <i>RequiredDuration</i> MUST be specified.
RequiredDuration ?	DurationSpan	Specifies the time by which the delivery is to be made relative to the date of the purchase order. Within an RFQ or a Quote, at most one of <i>Required</i> or <i>RequiredDuration</i> MUST be specified. Within a purchase order, <i>RequiredDuration</i> MUST NOT be specified.
ReturnMethod ? New in JDF 1.1	NameSpan	Specifies a delivery method for returning the surplus material, and MUST NOT be specified unless <i>SurplusHandling</i> = <i>Return</i> . Defaults to the value of DeliveryIntent/@ReturnMethod . Possible values are the same as DeliveryIntent/@ReturnMethod
ServiceLevel ? New in JDF 1.2	StringSpan	The service level of the specific carrier. Contain values <i>Next Day</i> , <i>2nd Day Air</i> , <i>Ground</i> , etc. Defaults to the value of DeliveryIntent/@ServiceLevel .
SurplusHandling ? New in JDF 1.1	Enumeration-Span	Describes what is to happen with unused or redundant parts of the transfer. The values are identical to those of <i>SurplusHandling</i> in DeliveryIntent . Defaults to the value of DeliveryIntent/@SurplusHandling .
Transfer ? New in JDF 1.1	Enumeration-Span	Describes the direction and responsibility of the transfer. The values are identical to those of <i>Transfer</i> in DeliveryIntent . Defaults to the value of DeliveryIntent/@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 .
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 (@ <i>ContactTypes</i> , <i>Delivery</i>)]/Address. The actual pickup address is specified as the Contact [contains (@ <i>ContactTypes</i> , <i>Pickup</i>)]/Address. At most one Contact [contains (@ <i>ContactTypes</i> , <i>x</i>)]/ MUST be specified for <i>x</i> equal to <i>Delivery</i> , <i>Pickup</i> or <i>Billing</i> . Defaults to the DeliveryIntent/Contact
DropItemIntent +	element	A DropIntent MAY consist of multiple products, which are represented by their respective PhysicalResource resources. Each DropItemIntent element describes a number of individual resources that is part of this DropIntent .
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.

— Element: DropItemIntent

Table 7-41: DropItemIntent element

Name	Data Type	Description
<i>AdditionalAmount</i> ? Modified in JDF 1.2 Deprecated in JDF 1.3	integer	Number of components used to calculate the value of the <i>AdditionalPrice</i> attribute in the Pricing. If not specified, defaults to the value of <i>DropIntent/@AdditionalAmount</i> . In JDF 1.3 and beyond, pricing information has been removed and will be handled by the business wrapper around JDF, e.g. PrintTalk.
<i>Amount</i> ?	integer	Specifies the final number of resources delivered. If not specified, defaults to the total amount of the resource that is specified by <i>PhysicalResource</i> or 1 if this <i>DropItemIntent</i> specifies a proof. Note that <i>DropItemIntent/@Amount</i> corresponds semantically to <i>ResourceLink/@ActualAmount</i> and <i>DropItem/@ActualAmount</i> .
<i>OrderedAmount</i> ?	integer	Specifies the original number of resources ordered. If not specified, defaults to the value of <i>Amount</i> . Note that <i>DropItemIntent/@OrderedAmount</i> corresponds semantically to <i>ResourceLink/@Amount</i> and <i>DropItem/@Amount</i> .
<i>Proof</i> ? New in JDF 1.1	string	This <i>DropItemIntent</i> refers to a proof that is specified in a <i>ProofItem</i> of the ProofingIntent of this product node. ProofingIntent/ProofItem/@ProofName MUST match <i>Proof</i> . Exactly one of <i>PhysicalResource</i> or <i>Proof</i> MUST be specified.
<i>Unit</i> ?	string	Unit of measurement for the <i>Amount</i> specified in the <i>PhysicalResource</i> . Defaults to the value of <i>Unit</i> defined in the resource described by the <i>PhysicalResource</i> .
<i>PhysicalResource</i> ? Modified in JDF 1.1	refelement	Description of the individual item that is delivered. Exactly one of <i>PhysicalResource</i> or <i>Proof</i> MUST be specified. Note that <i>PhysicalResource</i> is an abstract resource and that the element MUST be an instance of <i>PhysicalResource</i> , (e.g., Component).
<i>Pricing</i> ? Deprecated in JDF 1.3	element	Pricing element that defines the pricing of the <i>DropItemIntent</i> . In JDF 1.3 and beyond, pricing information has been removed and will be handled by the business wrapper around JDF, e.g. PrintTalk.

— Element: Pricing

[Deprecated in JDF 1.3](#)

The table defining the deprecated Pricing subelement has been moved to "DeliveryIntent Deprecated Subelements" on page 832.

— Element: Payment

[Deprecated in JDF 1.3](#)

The table defining the deprecated Payment subelement has been moved to "DeliveryIntent Deprecated Subelements" on page 832.

— Element: CreditCard

[Deprecated in JDF 1.3](#)

The table defining the deprecated CreditCard subelement has been moved to "DeliveryIntent Deprecated Subelements" on page 832.

7.1.6 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
Resource referenced by:	—
Process Resource Pairing:	EmbossingParams
Example Partition:	<i>Option, PageNumber, Side</i>
Input of processes:	Any product node
Output of processes:	—

Table 7-42: EmbossingIntent resource

Name	Data Type	Description
EmbossingItem +	element	Each embossed image is described by one EmbossingItem.

— Element: EmbossingItem

Table 7-43: EmbossingItem element (Section 1 of 2)

Name	Data Type	Description
<i>Direction</i> Modified in JDF 1.3	EnumerationSpan	The direction of the image. Possible values are: <i>Both</i> – Both debossing and embossing in one stamp. <i>Depressed</i> – Debossing. <i>Flat</i> – The embossing foil is applied flat. Used for foil stamping. New in JDF 1.3 <i>Raised</i> – Embossing.
<i>EdgeAngle</i> ?	NumberSpan	The angle of a beveled edge in degrees. Typical values are an angle of: 30, 40, 45, 50 or 60 degrees. If <i>EdgeAngle</i> is specified, <i>EdgeShape</i> = "Beveled" MUST be specified.
<i>EdgeShape</i> ?	EnumerationSpan	The transition between the embossed surface and the surrounding media can be rounded or beveled (angled). Possible values are: <i>Rounded</i> <i>Beveled</i>
<i>EmbossingType</i>	StringSpan	The strings defined in <i>EmbossingType</i> are whitespace separated combinations of the following tokens. Possible values for the tokens are: <i>BlindEmbossing</i> – Embossed forms that are not inked or foiled. The color of the image is the same as the paper. <i>FoilEmbossing</i> – Combines embossing with foil stamping in one single impression. <i>FoilStamping</i> – Using a heated die to place a metallic or pigmented image from a coated foil on the paper. <i>RegisteredEmbossing</i> – Creates an embossed image that exactly registers to a printed image.
<i>FoilColor</i> ?	EnumerationSpan	Defines the color of the foil material which is used for embossing. Allowed values are defined in Section A.3.3.2, NamedColor .
<i>Height</i> ?	NumberSpan	The height of the levels. This value specifies the <i>vertical</i> distance between the highest and lowest point of the stamp, regardless of the value of <i>Direction</i> .
<i>ImageSize</i> ?	XYPairSpan	The size of the bounding box of one single image.

Table 7-43: EmbossingItem element (Section 2 of 2)

Name	Data Type	Description
<i>Level?</i>	EnumerationSpan	The level of embossing. Possible values are: <i>SingleLevel</i> <i>MultiLevel</i> <i>Sculpted</i>
<i>Position?</i>	XYPairSpan	Position of the center of the bounding box of the embossed image in the coordinate system of the Component .

7.1.7 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. Note that the folding catalog is described in "FoldingParams" on page 438.

Resource Properties

Resource class:	Intent
Resource referenced by:	—
Process Resource Pairing:	CreasingParams, CuttingParams, Fold, FoldingParams, PerforatingParams
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

Table 7-44: FoldingIntent resource

Name	Data Type	Description
<i>FoldingCatalog</i>	NameSpan	Description of the folding scheme as specified in the folding catalog attribute in the format “Fn-i”. See JDF Folding Catalog descriptions in Figure 7-20 and Figure 7-21. 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.
<i>Folds?</i> 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 MUST be inferred from <i>FoldingCatalog</i> . The product $2*(X+1)*(Y+1)$ of <i>Folds</i> MUST always match the n of “Fn-i” of <i>FoldingCatalog</i> .
Fold * New in JDF 1.1	element	This describes the details of folding operations in the sequence described by the value of <i>FoldingCatalog</i> . Fold MUST be specified if non-symmetrical folds are requested.

7.1.8 HoleMakingIntent

This resource specifies the holmaking intent for a JDF job, using information that identifies the type of holmaking operation 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
Resource referenced by:	—
Process Resource Pairing:	Hole, HoleLine, HoleMakingParams, Media
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

Table 7-45: 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 <i>HoleType</i> . If not specified the implied default defined in "JDF/CIP4 Hole Pattern Catalog" on page 761 is assumed. Ignored when <i>HoleType</i> = " <i>Explicit</i> ".
HoleReferenceEdge = "Left" New in JDF 1.1	enumeration	The edge of the media relative to where the holes are to be punched. Use with <i>HoleType</i> . Possible values are: <i>Left</i> <i>Right</i> <i>Top</i> <i>Bottom</i> <i>Pattern</i> - Specifies that the reference edge implied by the value of <i>HoleType</i> in "JDF/CIP4 Hole Pattern Catalog" on page 761 is used.
HoleType Modified in JDF 1.1	StringSpan	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). For details of hole types and a list of additional allowed values, refer to "JDF/CIP4 Hole Pattern Catalog" on page 761. Values are: <i>Explicit</i> - Holes are defined in an array of Hole elements. Additional values defined in "JDF/CIP4 Hole Pattern Catalog" on page 761 The following values are deprecated from JDF 1.0 <i>2HoleEuro</i> - Replace by either R2m-DIN or R2m-ISO. <i>3HoleUS</i> - Replace by R3I-US <i>4HoleEuro</i> - Replace by R4m-DIN-A4 or R4m-DIN-A5.
HoleList ?	element	Array of all Hole elements. Used only when <i>HoleType</i> = " <i>Explicit</i> ", otherwise this element is not used.

7.1.9 InsertingIntent

This resource specifies the placing or inserting of one component within another, using information that identifies page location, position and attachment method. 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
Resource referenced by:	—
Process Resource Pairing:	InsertingParams, InsertSheet
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

Table 7-46: InsertingIntent resource

Name	Data Type	Description
<i>GlueType</i> ?	EnumerationSpan	Glue used to fasten the insert. Possible values are: <i>Permanent</i> <i>Removable</i>
<i>Method</i> ?	EnumerationSpan	Possible values are: <i>BindIn</i> – Apply glue to fasten the insert <i>BlowIn</i> – Loose insert.
InsertList	Element	List of individual inserts.

— Element: InsertList

Table 7-47: InsertList element

Name	Data Type	Description
<i>Insert</i> *	element	Individual insert description.

— Element: Insert

Table 7-48: Insert element

Name	Data Type	Description
<i>Folio</i>	IntegerRangeList	List of potential folios where the insert is to be placed. A <i>Folio</i> is defined by its first page in case <i>Method</i> = " <i>BlowIn</i> " and by the page that the glue is applied in case <i>Method</i> = " <i>BindIn</i> ". In general, a list of folios will only be supplied for <i>Method</i> = " <i>BlowIn</i> ". The pages are counted in the order, which is described in <i>FolioCount</i> of the parent Component .
<i>GlueType</i> ?	EnumerationSpan	Glue used to fasten the insert. Possible values are: <i>Removable</i> <i>Permanent</i> Defaults to the value of InsertingIntent / <i>@GlueType</i> .
<i>Method</i> ?	EnumerationSpan	Inserting method. Possible values are: <i>BindIn</i> – Apply glue to fasten the insert. <i>BlowIn</i> – Loose insert. Defaults to the value of InsertingIntent / <i>@Method</i> .
<i>SheetOffset</i> ? 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 <i>Transformation</i> .
<i>Transformation</i> ?	matrix	Rotation and offset between the Component to be inserted and the parent Component . If not specified, the identity matrix is applied.
<i>WrapPages</i> ? New in JDF 1.1	IntegerRangeList	List of finished pages of the cover that wrap around an Insert after all folds are correctly positioned. It is sufficient to specify the finished page of the front surface of the cover, (e.g., Cover 1 and Cover 4). Note that this key MUST NOT be specified unless the folding is ambiguous.
GlueLine * New in JDF 1.1	element	Array of all GlueLine elements used to glue in the insert. MUST NOT be specified in conjunction with <i>GlueType</i> .

7.1.10 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
Resource referenced by:	—
Process Resource Pairing:	LaminatingParams
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

Table 7-49: LaminatingIntent resource

Name	Data Type	Description
<u>Laminated ?</u> <u>Deprecated in JDF 1.1</u>	OptionSpan	If " <i>true</i> ", the product is laminated. If no LaminatingIntent is specified, the product MUST NOT be laminated.
<u>Temperature ?</u> <u>Modified in JDF 1.3</u>	EnumerationSpan	Temperature used in the lamination process. Possible values are: <i>Hot</i> <i>Cold</i>
<u>Surface ?</u>	EnumerationSpan	The surface to be laminated. One of: <i>Front</i> <i>Back</i> <i>Both</i>
<u>Texture ?</u> <u>New in JDF 1.3</u>	NameSpan	The intended texture of the laminate. Examples include: <i>Antique</i> – Rougher than vellum surface. <i>Calendared</i> – Extra-smooth or polished, uncoated paper. <i>Grain</i> <i>Linen</i> – Texture of coarse woven cloth. <i>Matte</i> <i>Smooth</i> <i>Stipple</i> – Fine pebble finish. <i>Vellum</i> – Slightly rough surface.
<u>Thickness ?</u>	NumberSpan	Thickness of the laminating material. Measured in microns [μm].

7.1.11 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
Resource referenced by:	—
Process Resource Pairing:	Layout, LayoutPreparationParams, StrippingParams
Example Partition:	<i>Option</i>
Input of processes:	Any product node

Output of processes: —

Table 7-50: LayoutIntent resource (Section 1 of 3)

Name	Data Type	Description
<i>Dimensions</i> ? New in JDF 1.1	XYPairSpan	Specifies the width (X) and height (Y) in points, respectively, of the media or product Component unfolded. For example, <i>Dimensions</i> for a Z-fold is the unfolded dimensions, while <i>FinishedDimensions</i> is the folded dimensions if known. Use <i>Dimensions</i> if <i>FinishedDimensions</i> is not known. The <i>Dimensions</i> attribute is provided for the rare case that <i>FinishedDimensions</i> does not unambiguously define the finished product, due to complex folding schemes. If both values are specified, <i>FinishedDimensions</i> takes precedence
<i>FinishedDimensions</i> ? 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 <i>FinishedDimensions</i> SHOULD be specified if both <i>FinishedDimensions</i> and <i>Dimensions</i> are known. Compatibility Warning. In JDF 1.1 height and width were erroneously switched in the description.
<i>FinishedGrainDirection</i> ? New in JDF 1.2	Enumeration-Span	Specifies the media grain direction of the finished page with respect to the binding edge. Possible values are: <i>ParallelToBind</i> – Grain direction is parallel to the binding edge. <i>PerpendicularToBind</i> – Grain direction is perpendicular to the binding edge. Note: in JDF 1.2 this was erroneously misspelled as <i>Perpendicular<u>a</u>ToBind</i>
<i>FinishedPageOrientation</i> ? Deprecated in JDF 1.1	enumeration	Indicates the desired orientation of the finished media. Possible values are: <i>Portrait</i> – The short edges of the media are the top and bottom. <i>Landscape</i> – The long edges of the media are the top and bottom. In JDF 1.1 , the finished page orientation is implied by the value of <i>Dimensions</i> and <i>FinishedDimensions</i> . If height (X) > width (Y), the product is portrait.

Table 7-50: LayoutIntent resource (Section 2 of 3)

Name	Data Type	Description
<p><i>FolioCount</i> = "Booklet" New in JDF 1.1</p>	enumeration	<p>Defines the method used when counting finished pages. One of:</p> <p><i>Booklet</i> – 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.</p> <p><i>Flat</i> – 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. <i>Flat</i> is to 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 7-20 and Figure 7-21) for the product. All sheets are counted, even if they are not included in the product, (e.g., due to a ShapeCuttingIntent).</p>
<p><i>NumberUp</i> = "1 1" Modified in JDF 1.2</p>	XYPair	<p>Specifies a regular, multi-up grid of page cells into which content pages are mapped.</p> <p>Compatibility Warning. In JDF 1.0 and 1.1 rows and columns were erroneously switched in the description.</p> <p>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).</p> <p>At most one of Layout or <i>NumberUp</i> MUST be specified.</p>
<p><i>Pages</i> ? New in JDF 1.1 Modified in JDF 1.2</p>	IntegerSpan	<p>Specifies the number of finished pages (surfaces) of the product component, including blank pages.</p> <p><i>Pages</i> multiplied with <i>Dimensions</i> then divided by two (2) identifies the amount of paper that is used in the product. <i>Pages</i> describes the paper usage regardless of document layout. This value MUST be an even number. For example, the value for <i>Pages</i> for a two-sided booklet with seven reader pages would be "8", whether the booklet were saddle stitched or glued.</p> <p>Compatibility Warning. The meaning of "pages" has been modified in JDF 1.2 to clarify an ambiguity in its definition. Prior to JDF 1.2, "pages" was ambiguously defined as the number of two-sided leaves. It is now defined as the number of surfaces and not the number of sheets which is different by a factor of two.</p>
<p><i>PageVariance</i> ? New in JDF 1.1</p>	IntegerSpan	<p>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 <i>Pages</i> is used as the default. For example, if there are ten finished pages, in which three are identical, <i>PageVariance</i> = "8" since it would take eight master copies to produce the product.</p>

Table 7-50: LayoutIntent resource (Section 3 of 3)

Name	Data Type	Description
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 <i>Dimensions</i> or <i>FinishedDimensions</i> . <i>NoRotate</i> – Do not rotate. <i>RotateOrthogonal</i> – Rotate by 90° in either direction. <i>RotateClockwise</i> – Rotate clockwise by 90°. <i>RotateCounterClockwise</i> – 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. Possible values are: <i>OneSided</i> – Page contents will only be imaged on the front side of the media. <i>OneSidedBack</i> – Page contents will only be imaged on the back side of the media. New in JDF 1.2 <i>TwoSidedHeadToHead</i> – Impose pages upon the front and back sides of media sheets so that the head (top) of page contents back up to each other. <i>TwoSidedHeadToFoot</i> – 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.
SizePolicy ? New in JDF 1.2	Enumeration-Span	Allows printing even if the container size defined in <i>Dimensions</i> or <i>FinishedDimensions</i> does not match the requirements of the data. The page container is one cell on the NUp grid of the Media defined in <i>Dimensions</i> or <i>FinishedDimensions</i> . <i>ClipToMaxPage</i> – The page contents is to be clipped to the size of the container. The printed area is centered in the source image. <i>FitToPage</i> – The page contents is to be scaled up or down to fit the container. The aspect ratio is maintained. <i>ReduceToFit</i> – The page contents is to be scaled down but not scaled up to fit the container. The aspect ratio is maintained. <i>Tile</i> – The page contents is to be split into several tiles, each printed on its own container.
Layout ? New in JDF 1.1	refelement	Specifies the details of a more complex Layout . At most one of Layout or <i>NumberUp</i> MUST 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.

7.1.12 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
Resource referenced by:	—
Process Resource Pairing:	Media
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

Table 7-51: MediaIntent resource (Section 1 of 5)

Name	Data Type	Description
<i>BackCoatings</i> ?	EnumerationSpan	Identical to <i>FrontCoatings</i> , but applied to the back surface of the media. Default = value of <i>FrontCoatings</i> .
<i>Brightness</i> ?	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.
<i>BuyerSupplied</i> ?	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.
<i>Dimensions</i> ? Deprecated in JDF 1.2	XYPairSpan	Specifies the size of the supplied media in points if <i>BuyerSupplied</i> evaluates to "true". <i>Dimensions</i> MUST be ignored if <i>BuyerSupplied</i> evaluates to "false". Note that the size of the finished product is always specified in LayoutIntent / <i>@FinishedDimensions</i> . In JDF 1.2 and beyond the specifics of <i>BuyerSupplied</i> media SHOULD be specified using a Media resource. The dimensions of the finished product are specified with LayoutIntent / <i>@Dimensions</i> or LayoutIntent / <i>@FinishedDimensions</i> .
<i>FrontCoatings</i> ? Modified in JDF 1.2	EnumerationSpan	What pre-process coating has been applied to the front surface of the media. Possible values are: <i>None</i> <i>Coated</i> – A coating of a system specified type. New in JDF 1.2 <i>Glossy</i> <i>HighGloss</i> <i>InkJet</i> – A coating intended for use with inkjet technology. New in JDF 1.2 <i>Matte</i> <i>Satin</i> <i>Semigloss</i>

Table 7-51: MediaIntent resource (Section 2 of 5)

Name	Data Type	Description
<i>Grade</i> ?	IntegerSpan	The intended grade of the media on a scale of 1 through 5. <i>Grade</i> is ignored if <i>MediaType</i> is not " <i>Paper</i> ". <i>Grade</i> of paper material is defined in accordance with the paper "types" set forth in [ISO12647-2:2004]. Offset printing paper types are defined with the following integer values: 1 "Gloss-coated paper" 2 "Matt-coated paper" 3 "Gloss-coated, web paper" 4 "Uncoated, white paper" 5 "Uncoated, yellowish paper" Note: [ISO12647-2:2004] paper type attribute values do not align with U.S. GRACOL paper grade attribute values, (i.e., [ISO12647-2:2004] type 1 does not equal U.S. GRACOL grade 1.)
<i>GrainDirection</i> ? New in JDF 1.2	EnumerationSpan	Direction of the grain in the coordinate system defined by LayoutIntent/@Dimensions or LayoutIntent/@FinishedDimensions . Possible values are: <i>ShortEdge</i> – Parallel to the shorter axis of the finished page. <i>LongEdge</i> – Parallel to the longer axis of the finished page.
<i>HoleCount</i> ? Deprecated in JDF 1.1	IntegerSpan	The intended number of holes that are to be punched in the media (either pre- or post-punched.) In JDF/1.1, use <i>HoleType</i> which includes the number of holes.
<i>HoleType</i> ? 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.) For details of hole types and a list of additional allowed values, refer to "JDF/CIP4 Hole Pattern Catalog" on page 761. Values are: <i>None</i> – no holes Additional values are defined in "JDF/CIP4 Hole Pattern Catalog" on page 761
<i>MediaColor</i> ?	EnumerationSpan	Color of the media. Allowed values are defined in Section A.3.3.2, <i>NamedColor</i> . If more-specific, specialized or site-specific media color names are needed, use <i>MediaColorDetails</i> .
<i>MediaColorDetails</i> ? New in JDF 1.2	StringSpan	A more specific, specialized or site-defined name for the media color. If <i>MediaColorDetails</i> is supplied, <i>MediaColor</i> MUST also be supplied. Note that there is a one-to-many relationship between entries in <i>MediaColor</i> and <i>MediaColorDetails</i> , (e.g., <i>MediaColorDetails</i> values of <i>Burgundy</i> and <i>Ruby</i> both correspond to a <i>MediaColor</i> of <i>DarkRed</i>).
<i>MediaSetCount</i> ?	integer	When the input media is grouped in sets, identifies the number of pieces of media in each set. For example, if the <i>UserMediaType</i> is " <i>PreCutTabs</i> ", a <i>MediaSetCount</i> of 5 would indicate that each set includes 5 tab sheets.

Table 7-51: MediaIntent resource (Section 3 of 5)

Name	Data Type	Description
MediaType ? New in JDF 1.1 Modified in JDF 1.3	EnumerationSpan	Describes the medium being employed. Possible values are: <i>CorrugatedBoard</i> New in JDF 1.3 <i>Disc</i> – CD or DVD disc to be printed on. New in JDF 1.2 <i>Other</i> – Any other media. For this value <i>MediaTypeDetails</i> SHOULD also be specified <i>Paper</i> <i>SelfAdhesive</i> New in JDF 1.3 <i>Transparency</i>
MediaTypeDetails ? New in JDF 1.3	NameSpan	Describes additional details of the medium described in <i>MediaType</i> . Possible values include: <i>Cloth</i> – Cloth, e.g., for a hard cover book case. <i>Leather</i> – Leather, e.g., for a hard cover book case. See also Media/@MediaTypeDetails for a list of further RECOMMENDED values. Note that some of the process related values of Media/@MediaTypeDetails such as <i>DryFilm</i> SHOULD NOT be specified here.
MediaUnit ? Deprecated in JDF 1.2	EnumerationSpan	Describes the format of the media as it is delivered to the device. Possible values are: <i>Roll</i> <i>Sheet</i> Reason for deprecation: Intent attributes pertain to finished product, not the raw media format. If <i>BuyerSupplied</i> = "true", then the Media resource can be used to provide this attribute.
Opacity ? Modified in JDF 1.2	EnumerationSpan	The opacity of the media. See <i>OpacityLevel</i> to specify the degree of opacity for any of these values. Possible values are: <i>Opaque</i> – The media is opaque. With two-sided printing the printing on the other side does not show through under normal incident light. <i>Translucent</i> – The media is translucent to a system specified amount. For example, translucent media can be used for back lit viewing. New in JDF 1.2 <i>Transparent</i> – The media is transparent to a system specified amount.
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.
<i>PrePrinted</i> = "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 <i>RecycledPercentage</i> .
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.

Table 7-51: MediaIntent resource (Section 4 of 5)

Name	Data Type	Description
<i>StockType ?</i>	NameSpan	Strings describing the available stock. Examples include: <i>Bristol</i> <i>Cover</i> <i>Bond</i> <i>Newsprint</i> <i>Index</i> <i>Offset</i> – This includes book stock. <i>Tag</i> <i>Text</i>
<i>Texture ?</i>	NameSpan	The intended texture of the media. Examples include: <i>Antique</i> – Rougher than vellum surface. <i>Calendared</i> – Extra-smooth or polished, uncoated paper. <i>Linen</i> – Texture of coarse woven cloth. <i>Smooth</i> <i>Stipple</i> – Fine pebble finish. <i>Vellum</i> – Slightly rough surface.
<i>Thickness ?</i> New in JDF 1.1	NumberSpan	The thickness of the chosen medium. Measured in microns [μm].

Table 7-51: MediaIntent resource (Section 5 of 5)

Name	Data Type	Description
<i>UserMediaType ?</i>	NMTOKEN	<p>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. Possible values include:</p> <p><i>Continuous</i> – Continuously connected sheets of an opaque material. Which edge is connected is not specified.</p> <p><i>ContinuousLong</i> – Continuously connected sheets of an opaque material connected along the long edge.</p> <p><i>ContinuousShort</i> – Continuously connected sheets of an opaque material connected along the short edge.</p> <p><i>Envelope</i> – Envelopes that can be used for conventional mailing purposes.</p> <p><i>EnvelopePlain</i> – Envelopes that are not preprinted and have no windows.</p> <p><i>EnvelopeWindow</i> – Envelopes that have windows for addressing purposes.</p> <p><i>FullCutTabs</i> – 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.</p> <p><i>Labels</i> – Label stock, (e.g., a sheet of peel-off labels).</p> <p><i>Letterhead</i> – Separately cut sheets of an opaque material including a letterhead.</p> <p><i>MultiLayer</i> – Form medium composed of multiple layers which are preattached to one another, (e.g., for use with impact printers).</p> <p><i>MultiPartForm</i> – Form medium composed of multiple layers not preattached to one another; each sheet MAY be drawn separately from an input source.</p> <p><i>Photographic</i> – Separately cut sheets of an opaque material to produce photographic quality images.</p> <p><i>PreCutTabs</i> – Media with tabs that are cut so that more than one tab is visible extending out beyond the edge of non-tabbed media.</p> <p><i>Stationery</i> – Separately cut sheets of an opaque material.</p> <p><i>TabStock</i> – Media with tabs (either precut or full-cut).</p> <p><i>Transparency</i> – Separately cut sheets of a transparent material.</p>
<u><i>USWeight ?</i></u> Deprecated in JDF 1.2	NumberSpan	The intended weight of the media, measured in pounds per ream of basis size. At most one of <i>Weight</i> or <i>USWeight</i> MUST be specified. If known, <i>Weight</i> SHOULD be specified in grammage (g/m ² .) In JDF 1.2 and beyond, use <i>Weight</i> .
<i>Weight ?</i>	NumberSpan	The intended weight of the media, measured in grammage (g/m ²) of the media. See "North American and Japanese Media Weight Explained" on page 721 for an explanation of how to calculate the US weight from the grammage for different stock types.

7.1.13 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.

Resource Properties

Resource class:	Intent
Resource referenced by:	—
Process Resource Pairing:	NumberingParams
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-52: NumberingIntent resource

Name	Data Type	Description
<i>ColorName</i> ?	EnumerationSpan	Defines the color of the numbering. Allowed values are defined in Section A.3.3.2, NamedColor.
ColorPool ?	refelement	Additional details about the colors used.
NumberItem +	element	Individual position of the numbers on the finished page.

— Element: NumberItem

Table 7-53: NumberItem element

Name	Data Type	Description
<i>ColorName</i> ?	EnumerationSpan	Defines the color of the numbering. Allowed values are defined in Section A.3.3.2, NamedColor. If not specified, it defaults to the values defined in NumberingIntent/@ColorName .
<i>Orientation</i> ?	NumberSpan	Rotation of the numbering machine in degrees. If <i>Orientation</i> = 0, the top of the numbers is along the leading edge.
<i>StartValue</i> = "1"	string	First value of the numbering machine.
<i>Step</i> = "1"	integer	Number that specifies the difference between two subsequent numbers of the numbering machine.
<i>XPosition</i> ?	NumberSpan	Position of the number in the X direction of the product.
<i>YPosition</i> ?	NumberSpan	Position of the number in the Y direction of the product.
SeparationSpec ?	refelement	Specifies the name of the Color in the ColorPool that is used for Numbering.

7.1.14 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
Resource referenced by:	—
Process Resource Pairing:	BoxPackingParams, Bundle, Component, PalletizingParams, Pallet, ShrinkingParams, StackingParams, Strap, StrappingParams, WrappingParams
Example Partition:	<i>Option</i>

Input of processes: Any product node

Output of processes: —

Table 7-54: PackingIntent resource

Name	Data Type	Description
<i>BoxedQuantity ?</i>	IntegerSpan	How many units of <i>product</i> in a box.
<i>BoxShape ?</i>	ShapeSpan	Describes the length, width and height of the box, in points.
<i>CartonQuantity ?</i>	IntegerSpan	How many units of <i>product</i> in a carton.
<i>CartonShape ?</i>	ShapeSpan	Describes the length, width and height of the carton, in points. For example, 288 544 1012
<i>CartonMaxWeight ?</i>	NumberSpan	Maximum weight of an individual carton, in kilograms.
<i>CartonStrength ?</i>	NumberSpan	Strength of the carton, in kilograms.
<i>FoldingCatalog ?</i>	NameSpan	Description of the folding scheme for folding the product for packaging as specified in the folding catalog attribute in the format “Foxy”. See JDF Folding Catalog descriptions in Figure 7-20 and Figure 7-21. 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.
<i>PalletCornerBoards ?</i> New in JDF 1.3	NameSpan	Additional protective corner boards for packaging on a pallet: Values include: <i>Corners</i> : Corner boards on 8 corners of the pallet. <i>VerticalEdge</i> : Corner boards along the 4 vertical edges.
<i>PalletQuantity ?</i>	IntegerSpan	Number of <i>product</i> per pallet
<i>PalletSize ?</i>	XYPairSpan	Describes the length and width of the pallet, in points, (e.g., “3500 3500”).
<i>PalletMaxHeight ?</i>	NumberSpan	Maximum height of a loaded pallet, in points.
<i>PalletMaxWeight ?</i>	NumberSpan	Maximum weight of a loaded pallet, in kilograms.
<i>PalletType ?</i>	NameSpan	Type of pallet used. Examples include: <i>2Way</i> – Two-way entry <i>4Way</i> – Four-way entry <i>Euro</i> – Standard 1*1 m Euro pallet
<i>PalletWrapping ?</i>	NameSpan	Wrapping of the completed pallet. Examples include: <i>Banding</i> <i>None</i> – explicitly requests no wrapping. <i>StretchWrap</i>
<i>WrappedQuantity ?</i>	IntegerSpan	Number of units of product per wrapped package.
<i>WrappingMaterial ?</i>	NameSpan	Examples include: <i>None</i> – explicitly requests no wrapping. <i>PaperBand</i> <i>Polyethylene</i> <i>RubberBand</i> <i>ShrinkWrap</i>

7.1.15 ProductionIntent

This resource specifies the manufacturing intent and considerations for a JDF job using information that identifies the desired result or specified manufacturing path.

Resource Properties

Resource class:	Intent
Resource referenced by:	—
Process Resource Pairing:	All
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

Table 7-55: ProductionIntent resource

Name	Data Type	Description
<i>PrintPreference ?</i>	EnumerationSpan	Intended result or goal. Possible values are: <i>Balanced</i> – Request for a manufacturing process that balances the requirements for cost, speed and quality. <i>CostEffective</i> – Request for the most cost effective manufacturing process. <i>Fastest</i> – Request for the most time effective manufacturing process. Cost and Quality can be sacrificed for a fast turnaround time. <i>HighestQuality</i> – Request for the manufacturing process which will result in the highest quality.
<i>PrintProcess ?</i>	EnumerationSpan	Print process requested. Allowed values are: <i>Electrophotography</i> <i>Flexography</i> <i>Gravure</i> <i>Inkjet</i> <i>Lithography</i> – Includes offset printing <i>Letterpress</i> <i>Screen</i> <i>Thermography</i>
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 MUST be specified for any physical resources that are physically supplied by the customer.

7.1.16 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. 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 are specified in **DeliveryIntent**.

Resource Properties

Resource class:	Intent
Resource referenced by:	—
Process Resource Pairing:	ApprovalParams, ApprovalSuccess, ColorantControl, ColorSpaceConversionParams, ExposedMedia, ImageSetterParams, InterpretingParams, Layout, Media, RenderingParams, ScreeningParams, SeparationControlParams, StrippingParams
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

Table 7-56: ProofingIntent resource

Name	Data Type	Description
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.

— Element: ProofItem

All parameters of **ProofingIntent** have been moved into **ProofItem** in JDF 1.1

Table 7-57: ProofItem element (Section 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 <i>Preferred</i> = "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. Possible values are: <i>Monochrome</i> – Generic single color printing condition, (e.g., black and white or one single spot color). <i>BasicColor</i> – Color does not match precisely. This implies the absence of a color matching system. <i>MatchedColor</i> – Color is matched to the output of the press using a color matching system.
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 is to emulate halftone screens.
ImageStrategy ? New in JDF 1.2	EnumerationSpan	Identifies which images (OPI or other) will be printed on a proof or displayed as a soft proof. <i>NoImages</i> – No images are imaged on the proof. <i>LowResolution</i> – Low resolution images are imaged on the proof. <i>HighResolution</i> – High resolution production images are imaged on the proof, resulting in proofs that accurately represent the final product.
PageIndex ? New in JDF 1.1	IntegerRangeList	List of pages in the numbering scheme given by the <i>FolioCount</i> attribute of the component that is to be proofed. Where no range is specified then all pages shall be proofed.
ProofName ? New in JDF 1.1	string	Name of the ProofItem . This field MUST be specified if delivery of a proof is specified in DeliveryIntent .
ProofTarget ? Modified in JDF 1.1	URL	Identifies a remote target for the proof output in a remote proofing environment. This can be either a soft or a hard proofing target. The file to be displayed or output is to be sent to the URL specified in <i>ProofTarget</i> .

Table 7-57: ProofItem element (Section 2 of 2)

Name	Data Type	Description
Technology ? Modified in JDF 1.1	NameSpan	Technology used for making the proof. Possible values include: <i>BlueLine</i> <i>DyeSub</i> <i>InkJet</i> <i>Laser</i> <i>PressProof</i> <i>SoftProof</i>
ProofType? Modified in JDF 1.1	EnumerationSpan	The kind of proof. Possible values are: <i>Page</i> – Page proof <i>Imposition</i> – Imposition proof <i>None</i> – No proof is needed.
SeparationSpec * New in JDF 1.1	element	Separations that are to be proofed. If not specified, all separations are proofed.
ApprovalParams ? New in JDF 1.2	refelement	List of people (e.g., a customer, printer or manager) who can sign the ApprovalSuccess .

7.1.17 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
Resource referenced by:	—
Process Resource Pairing:	—
Example Partition:	<i>Edition</i>
Input of processes:	Any product node
Output of processes:	—

Table 7-58: PublishingIntent resource

Name	Data Type	Description
<i>IssueDate</i>	TimeSpan	Publication date of the issue.
<i>IssueName</i>	StringSpan	The name of a the publication.
<i>IssueType</i>	NameSpan	Defines the product type of the issue. Values include: <i>Newspaper</i> – The publication is a newspaper <i>Magazine</i> – The publication is a magazine
<i>Circulation ?</i>	IntegerSpan	Specifies the number of copies to be published.

7.1.18 ScreeningIntent

[New in JDF 1.2](#)

This resource specifies the screening intent parameters desired for a JDF job.

Resource Properties

Resource class:	Intent
Resource referenced by:	—

Process Resource Pairing: **ScreeningParams, SeparationControlParams**
Example Partition: *Option*
Input of processes: Any product node
Output of processes: —

Table 7-59: ScreeningIntent resource

Name	Data Type	Description
<i>DotSize ?</i>	NumberSpan	Specifies the dot size of the screen in microns [μm] when FM screening is used, otherwise <i>DotSize</i> is ignored.
<i>Frequency ?</i>	NumberSpan	Specifies the line frequency of the screen in lines per inch (lpi) when AM screening is used, otherwise <i>Frequency</i> is ignored.
<i>FrequencySelection ?</i>	EnumerationSpan	Selects the AM or FM frequency range. Possible values are: <i>LowestFrequency</i> – Lowest AM or FM frequency supported. <i>MiddleFrequency</i> – Middle AM or FM frequency supported <i>HighestFrequency</i> – Highest AM or FM frequency supported
<i>ScreeningType ?</i>	EnumerationSpan	General type of screening. Possible values are: <i>AM</i> – Can be line or dot. <i>FM</i>

7.1.19 ShapeCuttingIntent

This resource specifies form and line cutting for a JDF job. The cutting processes are applied for producing special shapes like an envelope window or a heart-shaped beer mat. Information that identifies the type and shape of cuts can be described. The cutting process(es) can be performed using tools such as hollow form punching, perforating or die-cutting equipment.

Resource Properties

Resource class: Intent
Resource referenced by: —
Process Resource Pairing: **CuttingParams, ShapeCuttingParams**
Example Partition: *Option*
Input of processes: Any product node
Output of processes: —

Table 7-60: ShapeCuttingIntent resource

Name	Data Type	Description
ShapeCut *	element	Array of all ShapeCut elements. Used when each shape is exactly specified.

— Element: ShapeCut

Table 7-61: ShapeCut element (Section 1 of 2)

Name	Data Type	Description
<i>CutBox ?</i>	rectangle	Specification of a rectangular window. (See Section A.2.31, <i>rectangle</i> for a definition of the rectangle data type.)
<i>CutOut = "false"</i>	boolean	If " <i>true</i> ", the inside of a specified shape is to be removed. If " <i>false</i> ", the outside of a specified shape is to be removed. An example of an inside shape is a window, while an example of an outside shape is a shaped greeting card.
<i>CutPath ?</i> Modified in JDF 1.2	PDFPath	Specification of a complex path. This MAY be an open path in the case of a single line.

Table 7-61: ShapeCut element (Section 2 of 2)

Name	Data Type	Description
<i>Material</i> ?	StringSpan	Transparent material that fills a shape (e.g., an envelope window) that was cut out when <i>CutOut</i> = "true".
<i>CutType</i> ? Modified in JDF 1.1	EnumerationSpan	Type of cut or perforation used. Possible values are: <i>Cut</i> – Full cut. <i>Perforate</i> – Interrupted perforation that does not span the entire sheet.
<i>ShapeDepth</i> ? New in JDF 1.1	NumberSpan	Depth of the shape cut. Measured in microns [μm]. If not specified, the shape is completely cut.
<i>Pages</i> ?	IntegerRangeList	List of Finished Pages to which this shape is to be applied. Only the recto finished page of a leaf SHOULD be specified.
<i>ShapeType</i> Modified in JDF 1.3	EnumerationSpan	Describes any precision cutting other than hole making. Possible values are: <i>Rectangular</i> <i>Round</i> <i>RoundedRectangle</i> – Rectangle with rounded corners. New in JDF 1.3 <i>Path</i>
<i>TeethPerDimension</i> ?	NumberSpan	Number of teeth in a given perforation extent in teeth/point. MicroPerforation is defined by specifying a large number of teeth (n > 1000).

7.1.20 SizeIntent

[Deprecated in JDF 1.1](#)

SizeIntent has been deprecated in JDF 1.1. All contents have been moved to **LayoutIntent**.

7.2 Process Resources

The rest of the resources described in this chapter are what are known as process resources. This means that they serve as necessary components in each of the JDF processes. Section 7.2.1 describes the template for all of the sections that follow. Then every resource already defined for JDF is chronicled, in alphabetical order, below.

7.2.1 Process Resource 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. As was described in Section 3.7.2, 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 3-9, “Abstract Resource element,” on page 54. 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 sub-element 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, Quantity and Handling resource elements inherit from the PhysicalResource element, which in turn inherits from the Resource element. The Parameter and Implementation resource 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 7.2.71, *EndSheetGluingParams*. An example of the tables in this section is 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. *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 “?”. For further details, see Section 1.3.4, Specification of Cardinality

Resource Properties Template

Resource class: *Defines the resource class or specifies ResourceElement if the element only exists as a resource subelement.*

Resource referenced by: *List of parent resources that MAY contain elements of this type.*

Example Partition: *List of RECOMMENDED partitioning keys: For a complete list of partition keys, see the description of PartIDKeys in Table 3-25, “Partitionable resource element,” on page 86. 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*

Table 7-62: Template for process resource (Section 1 of 2)

Name	Data Type	Description
<i>Attribute-Name Cardinality</i>	<i>Attribute-data-type</i>	<i>Information about the attribute.</i>
<i>Element-Name Cardinality</i>	element	<i>Information about the element.</i> Note: the “element” data type means that the specified element MUST be an in-line sub-element within the resource.
<i>Element-Name Cardinality</i>	refelement	<i>Information about the element</i> Note: the “refelement” data type means that the specified element is based on other atomic resources or resource elements. The specified element MUST be either an in-line element or an instance of a ResourceRef element (see Section 3.9.2, ResourceRef – Element for Inter-Resource Linking and refElement). In case of a ResourceRef element, a “Ref” MUST be appended to the name specified in the table column entitled “Name”.
FileSpec <i>(someValue) Cardinality</i>	refelement	<i>Information about the FileSpec resource</i> Note: FileSpec/@ResourceUsage MUST match the “someValue” value specified in the parentheses. When a resource potentially contains multiple FileSpec children, the value of FileSpec@ResourceUsage distinguishes the FileSpec resources.

Table 7-62: Template for process resource (Section 2 of 2)

Name	Data Type	Description
<i>Resource-Name</i> (<i>someValue</i>) <i>Cardinality</i>	refelement	<i>Information about the resource and the attribute whose value is "someValue".</i> Note: Some specified attribute in the specified resource MUST match the " <i>someValue</i> " 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.

7.2.2 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.

Resource Properties

Resource class:	Parameter
Resource referenced by:	Contact, Location (see Table 3-13, "Location element," on page 62)
Example:	—
Input of processes:	—
Output of processes:	—

Table 7-63: Address resource

Name	Data Type	Description
<i>City ?</i>	string	City or locality of address (vCard: ADR:locality).
<i>Country ?</i>	string	Country of address (vCard: ADR:country).
<i>CountryCode ?</i>	string	Country of address. This value conforms to the [ISO3166-1:1997] standard in which countries are represented as two-character codes.
<i>PostBox ?</i>	string	Post office address (vCard: ADR:pobox. For example: P.O. Box 101).
<i>PostalCode ?</i>	string	Zip code or postal code of address (vCard: ADR:pcode).
<i>Region ?</i>	string	State or province (vCard: ADR:region).
<i>Street ?</i>	string	Street address (vCard: ADR:street).
<i>ExtendedAddress ?</i>	telem	Extended address (vCard: ADR:extadd. For example: Suite 245).

7.2.3 AdhesiveBindingParams

[Deprecated in JDF 1.1](#) See "AdhesiveBindingParams" on page 833 for details of this deprecated resource.


7.2.4 ApprovalParams

This resource provides the details of an approval process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	ConventionalPrintingParams, DigitalPrintingParams, ProofingIntent
Example Partition:	—
Input of processes:	Approval
Output of processes:	—

Table 7-64: ApprovalParams resource

Name	Data Type	Description
ApprovalPerson *	element	List of people (e.g., a customer, printer or manager) who can sign the approval.
<i>MinApprovals</i> = "1"  New in JDF 1.2	integer	Minimum number of ApprovalPerson [<i>@ ApprovalRole</i> = "Group"] whose associated person MUST sign the ApprovalSuccess for the ApprovalSuccess to be <i>Available</i> .

— Element: ApprovalPerson

Table 7-65: ApprovalPerson element

Name	Data Type	Description
<i>Obligated</i> ? Deprecated in JDF 1.2	boolean	If "true", the person has to sign this approval. In JDF 1.2 and beyond, use <i>ApprovalRole</i> .
<i>ApprovalRole</i> = "Obligated" New in JDF 1.2 Modified in JDF 1.3	enumeration	Role of the ApprovalPerson. One of: <i>Approvinator</i> – The decision of this approver immediately overrides the decisions of the other approvers and ends the approval cycle. The <i>Approvinator</i> NEED NOT sign for the approval to become valid. New in JDF 1.3 <i>Group</i> – The approver belongs to a group of which <i>MinApprovals</i> members MUST sign. <i>Informative</i> – The approver is informed of the approval process, but the approval is still valid, even without his approval. <i>Obligated</i> – The approver MUST sign the approval.
<i>ApprovalRoleDetails</i> ? New in JDF 1.3	string	Additional details on the <i>ApprovalRole</i> .
Contact	refelement	Contact (e.g., a customer, printer or manager) who MUST sign the approval. There MUST be a Contact [contains (<i>@ContactTypes</i> , "Approver")].

7.2.5 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

Resource referenced by: —

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 7-66: ApprovalSuccess resource (Section 1 of 2)

Name	Data Type	Description
ApprovalDetails * New in JDF 1.3	element	Container for details about the decision for each approver.

Table 7-66: ApprovalSuccess resource (Section 2 of 2)

Name	Data Type	Description
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.

— Element: ApprovalDetails

[New in JDF 1.3](#)

Table 7-67: ApprovalDetails element

Name	Data Type	Description
<i>ApprovalState</i>	enumeration	Decision made by the approver specified in this ApprovalDetails/ Contact . Allowed values are: <i>Approved</i> – approver approved the resource. <i>ApprovedWithComment</i> – approver approved the resource but still had some comments. <i>Rejected</i> – approver rejected the resource.
<i>ApprovalStateDetails</i> ?	string	Additional details on the decision made by the approver are specified in this ApprovalDetails/ Contact . This value provides additional machine readable details of <i>ApprovalState</i> . Hand written comments and notes MAY be specified in ApprovalDetails/Comment or ApprovalDetails/ <i>@CommentURL</i> .
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.

7.2.6 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
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>Collecting, Gathering, Stripping</i>
Output of processes:	—

Table 7-68: Assembly resource (Section 1 of 2)

Name	Data Type	Description
<i>AssemblyID</i> ? Deprecated in JDF 1.3	string	Identification of the Assembly if <i>Stripping</i> produces multiple Assembly elements.

Table 7-68: Assembly resource (Section 2 of 2)

Name	Data Type	Description
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 is to be bound. One of: <i>Left</i> <i>Right</i> <i>Top</i> <i>Bottom</i> BindingSide is ignored when Order = "None".
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. One of: <i>Left</i> <i>Right</i> <i>Top</i> <i>Bottom</i> <i>None</i>
Order = "Gathering"	enumeration	One of: <i>Collecting</i> – The sections are placed within one another. The first section is on the outside. <i>Gathering</i> – The sections are placed on top of one another. The first section is on the top. <i>None</i> – The sections are not bound. Typically used for flatwork jobs. <i>List</i> – More complex ordering of the sections
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 MUST NOT be specified unless Order = "List".
PageList ? New in JDF 1.3	refelement	Reference to the PageList that describes the pages used in this Assembly .
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 MUST NOT be specified if Order = "List".

— Element: **AssemblySection**Table 7-69: **AssemblySection** element

Name	Data Type	Description
<i>AssemblyID</i> ? 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 .
<i>AssemblyIDs</i> ? 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.
<i>JobID</i> ?	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 .
<i>Order</i> = "Gathering"	enumeration	One of: <i>Collecting</i> – The child AssemblySection elements are placed within one another. The first section is on the outside. <i>Gathering</i> – The child AssemblySection elements are placed on top of one another. The first section is on the top.
AssemblySection *	element	Additional AssemblySection elements which are collected or gathered to create this AssemblySection .
<i>PageAssignedList</i> * New in JDF 1.3	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 MUST NOT be specified if child AssemblySection elements are present in this AssemblySection .

— Element: **PageAssignedList**[New in JDF 1.3](#)**PageAssignedList** specifies the metadata related to assigned pages.Table 7-70: **PageAssignedList** element (Section 1 of 2)

Name	Data Type	Description
<i>BroadsheetNumber</i> ?	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.
<i>LogicalPrinterSection</i> ?	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.
<i>PageListIndex</i>	IntegerRangeList	List of the indices of the PageData elements of the Assembly/PageList specified in this AssemblySection .

Table 7-70: PageAssignedList element (Section 2 of 2)

Name	Data Type	Description
<i>PagePlacementName</i> ?	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 , <i>PagePlacementName</i> specifies an identifier for a single page on a web-product level. Therefore, different <i>PagePlacementName</i> elements might be assigned to one single PageList/PageData element.

7.2.7 AssetListCreationParams

[New in JDF 1.2](#)

This resource provides controls for the *AssetListCreation* process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>AssetListCreation</i>
Output of processes:	—

Table 7-71: AssetListCreationParams resource

Name	Data Types	Description
<i>AssetTypes</i> ?	regExp	Specifies what type of assets are to be listed. The regular expression represents the <i>MimeType</i> of the assets to be listed. The default behavior is to 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.
<i>ListPolicy</i> = "All"	enumeration	Policy that defines which assets MUST be added to the output RunList . Values are: <i>All</i> – List all referenced assets, including those that are unavailable. <i>Available</i> – List all referenced assets, excluding those that are unavailable.
FileSpec (<i>SearchPath</i>) *	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 MUST NOT be searched recursively.

7.2.8 AutomatedOverPrintParams

This resource 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.

Resource Properties

Resource class:	Parameter
Resource referenced by:	RenderingParams, SeparationControlParams
Example Partition:	—

Input of processes: —

Output of processes: —

Table 7-72: AutomatedOverPrintParams resource

Name	Data Types	Description
<i>KnockOutCMYKWhite</i> = "false" New in JDF 1.3	boolean	Graphic objects defined in DeviceCMYK, where all colorant values are <0.001 MUST be knocked out, even when set to overprint and when the PDF overprint mode is set to 1.
<i>OverPrintBlackLineArt</i> = "false"	boolean	Indicates whether overprint is to 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", <i>LineArtBlackLevel</i> is ignored and PDL line art overprint operators are processed.
<i>OverPrintBlackText</i> = "false"	boolean	Indicates whether overprint is to be set to "true" for black text. If "true", overprint of black line art is applied regardless of any values in the PDL. If "false", <i>TextSizeThreshold</i> and <i>TextBlackLevel</i> are ignored and PDL line art overprint operators are processed.
<i>TextSizeThreshold</i> ?	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.
<i>TextBlackLevel</i> = "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.
<i>LineArtBlackLevel</i> ?	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 <i>TextBlackLevel</i> .

7.2.9 BarcodeCompParams

[New in JDF 1.3](#)**BarcodeCompParams** specifies the technical compensation parameters for barcodes.

Resource Properties

Resource class: Parameter

Resource referenced by: **BarcodeReproParams**

Example Partition: —

Input of processes: —

Output of processes: —

Table 7-73: BarcodeCompParams resource

Name	Data Types	Description
<i>CompensationProcess</i>	enumeration	Process that is bar width spread is compensated for. Values include: <i>Printing</i> <i>Platemaking</i>
<i>CompensationValue</i> ?	double	The width of the bars is reduced by this amount in micron to compensate for technical spread.

7.2.10 BarcodeReproParams

[New in JDF 1.3](#)

BarcodeReproParams specifies the reproduction parameters for barcodes.

Resource Properties

Resource class:	Parameter
Resource referenced by:	BarcodeProductionParams
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-74: BarcodeReproParams resource

Name	Data Types	Description
<i>BearerBars ?</i>	enumeration	Indicates the policy how to generate bearer bars. (ITF). Values are: <i>None</i> <i>TopBottom</i> <i>Box</i> <i>BoxHMarks</i>
<i>Height ?</i>	double	The height of the bars of a linear barcode.
<i>Magnification ?</i>	double	The magnification factor for linear barcodes.
<i>Masking ?</i>	enumeration	Indicates the properties of the mask around the graphical content of the barcode that masks out all underlying graphics. Values are: <i>None</i> – No masking, barcode is put on top of underlying graphics. <i>WhiteBox</i> – 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.)
<i>ModuleHeight ?</i>	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.).
<i>ModuleWidth ?</i>	double	The X size in micron of an element of a 2D barcode such as DATA-MATRIX or PDF417.
<i>Ratio ?</i>	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 *	refelement	Parameters for bar width compensation. The total reduction of bar width is the sum of all BarcodeCompParams/ @CompensationValue.

7.2.11 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
Resource referenced by:	—

Example Partition: —
Input of processes: *Bending*
Output of processes: —

Table 7-75: BendingParams resource

Name	Data Types	Description
<i>Bend</i> = "true"	boolean	If "true", indicates that the device MUST bend.
<i>Punch</i> = "true"	boolean	If "true", indicates that the device MUST create registration punch holes.
<i>PunchType</i> ?	string	Name of the registration punch scheme, (e.g., Bacher).

7.2.12 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.

Resource Properties

Resource class: Parameter
Resource referenced by: **StrippingParams**
Example Partition: *WebName*
Input of processes: —
Output of processes: —

Table 7-76: BinderySignature resource (Section 1 of 4)

Name	Data Types	Description
<i>BinderySignatureType</i> = "Fold" New in JDF 1.3	enumeration	The type of BinderySignature. Possible values are: <i>Fold</i> – a folding dummy (as defined in JDF 1.2) <i>Grid</i> – a grid based layout <i>Die</i> – a layout defined by an existing die.
<i>BindingEdge</i> = "Left"	enumeration	Specifies the binding edge of this BinderySignature . <i>BindingEdge</i> defines the Spine side the folded BinderySignature . The opposite side defines the Face side. One of: <i>Left</i> <i>Right</i> <i>Top</i> <i>Bottom</i> <i>None</i> – The Spine is at the left side of the SignatureCell and the Face is at the right side of the SignatureCell.

Table 7-76: BinderySignature resource (Section 2 of 4)

Name	Data Types	Description
BindingOrientation ? New in JDF 1.3	orientation	<p>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 <i>BindingEdge</i>, the other side of the reference corner the <i>JogEdge</i>. <i>BindingOrientation</i> is the named orientation describing the transformation of the default reference corner to the new reference corner defined by <i>BindingEdge</i> and <i>JogEdge</i>.</p> <p>For BinderySignature elements defined by <i>FoldCatalog</i> or <i>Folds</i>, the default value of <i>BindingOrientation</i> = "Rotate0" if the folded BinderySignature has a closed head, otherwise <i>BindingOrientation</i> = "Flip0".</p> <p>For BinderySignature elements defined by <i>SignatureCells</i>, the default value <i>BindingOrientation</i> = "Rotate0".</p> <p>For details, See "Matrices and Orientation values for describing the orientation of a Component" on page 26.</p>
FoldCatalog ?	string	<p>Describes the type of fold according to the folding catalog in the format "Fx-y" as shown in the Fold resource (See "Fold" on page 437.) At least one of <i>SignatureCell</i>, <i>FoldCatalog</i> or Fold MUST be specified. <i>FoldCatalog</i> MUST NOT be specified unless <i>BinderySignatureType</i> = "Fold".</p>
JogEdge = "Top" New in JDF 1.3	enumeration	<p>Specifies the <i>JogEdge</i> of the folded BinderySignature. The <i>JogEdge</i> defines the Head side of the folded BinderySignature. The opposite side defines the Foot side. One of:</p> <p><i>Left</i></p> <p><i>Right</i></p> <p><i>Top</i></p> <p><i>Bottom</i></p> <p><i>None</i> – The Head side is the top of the <i>SignatureCell</i>, the Foot side is the bottom of the <i>SignatureCell</i>.</p>
NumberUp = "1 1" Modified in JDF 1.3	XYPair	<p>Specifies a regular, multi-up grid of <i>SignatureCells</i> into which content pages are mapped. The first value specifies the number of columns of <i>SignatureCells</i>, and the second value specifies the number of rows of <i>SignatureCells</i> in the multi-up grid (both numbers are integers). When the BinderySignature is partitioned (e.g., by <i>WebName</i>), <i>NumberUp</i> MAY be different from leaf to leaf.</p>
OutsideGutter ? New in JDF 1.3	boolean	<p>If <i>BinderySignatureType</i> is "Grid", this boolean defines whether the outside margins of strip cells have to be taken into account. E.g., if <i>OutsideGutter</i> is <i>false</i>, the Spine (S2) of the strip cells at the left border of the grid is considered to be 0.</p>

Table 7-76: BinderySignature resource (Section 3 of 4)

Name	Data Types	Description
StaggerColumns ? New in JDF 1.3	DoubleList	<p>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 <i>TrimSize</i>) + <i>TrimHead</i> + <i>TrimFoot</i>) by which to shift the corresponding column (can be negative). E.g., <i>StaggerColumns</i> = "0.0 0.333 0.666" specifies to shift each</p> <ul style="list-style-type: none"> "3*n column up by 0% "3*n+1 column up by 33.3% of the strip cell height "3*n column up by 66.6% of the strip cell height <p>This element MAY be present when <i>BinderySignatureType</i> = "Grid". At most one of <i>StaggerColumns</i> or <i>StaggerRows</i> MUST be specified.</p>
StaggerContinuous ? New in JDF 1.3	boolean	<p>Indicates if the BinderySignature has to be considered as a continuous repetition for staggering. This attribute MUST NOT be present unless exactly one of <i>StaggerRows</i> or <i>StaggerColumns</i> is specified. Consider a grid with <i>m</i> columns and <i>n</i> rows with <i>StaggerContinuous</i> = "true". If <i>StaggerColumns</i> is specified, the BinderySignature MUST be considered continuous with a height <i>H</i> equal to <i>n</i> multiplied by the strip cell height. If <i>StaggerColumns</i> has a value of <i>y</i> for a certain column, that column is shifted up (assuming <i>y</i> > 0) by an amount equal to <i>y</i> multiplied by the strip cell height (in the same way as described for <i>StaggerColumns</i>). All content (even partial cells) that falls above <i>H</i> (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 <i>y</i> is 0.666, then the top 66.6% of the top cell is shifted to be just below the original bottom cell. Analogous for <i>StaggerRows</i>.</p>
StaggerRows ? New in JDF 1.3	DoubleList	<p>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 <i>TrimSize</i>) + <i>TrimFace</i> + <i>Spine</i>) by which to shift the corresponding row (can be negative). E.g., "0.0 0.333 0.666" specifies to shift each</p> <ul style="list-style-type: none"> "3*n row right by 0% "3*n+1 row right by 33.3% of the strip cell width "3*n row right by 66.6% of the strip cell width <p>This element MAY be present when <i>BinderySignatureType</i> = "Grid". At most one of <i>StaggerColumns</i> or <i>StaggerRows</i> MUST be specified.</p>
DieLayout ? New in JDF 1.3	refelement	<p>The layout as defined by a pre-existing die. DieLayout MUST be present when <i>BinderySignatureType</i> = "Die".</p>
Fold *	element	<p>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 MUST NOT be specified if SignatureCell elements are present. Fold MUST NOT be specified unless <i>BinderySignatureType</i> = "Fold".</p>

Table 7-76: BinderySignature resource (Section 4 of 4)

Name	Data Types	Description
SignatureCell *	element	Describes the SignatureCells 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 MUST have precedence. SignatureCell MUST NOT be specified if Fold elements are present.

— Element: SignatureCell

SignatureCell elements describe a set of individual page cells in a **BinderySignature**.

Note: “Page number” in the table below refers to finished pages 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 MUST NOT contain final page numbers unless **Assembly/@Order = "None"**.

Table 7-77: SignatureCell element (Section 1 of 2)

Name	Data Types	Description
<i>BackFacePages</i> ?	IntegerList	Page numbers for the back finished pages forming a foldout.
<i>BackPages</i> ?	IntegerList	Page numbers of the back finished pages of a SignatureCell. The number of entries in <i>FrontPages</i> and <i>BackPages</i> MUST be identical. The entries with an identical index in <i>FrontPages</i> and <i>BackPages</i> are back-to-back in the layout. If not specified, the layout is one-sided.
<i>BottleAngle</i> ?	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.
<i>BottleAxis</i> ?	enumeration	Indicates the point around which the cell is bottled. One of: <i>FaceFoot</i> <i>FaceHead</i> <i>SpineFoot</i> <i>SpineHead</i>
<i>FrontFacePages</i> ?	IntegerList	Page numbers for the front finished pages forming a foldout.
<i>FrontPages</i> ?	IntegerList	Page numbers of the front finished pages of a SignatureCell. Multiple page cells with the same properties except for the finished pages to which they are assigned MAY be summarized as one SignatureCell with multiple entries in <i>FrontPages</i> .
<i>Orientation</i> = "Up" Modified in JDF 1.3	enumeration	Indicates the orientation of the SignatureCell. One of: <i>Down</i> – 180° rotation. <i>Left</i> – 90° counter-clockwise rotation. New in JDF 1.3 <i>Right</i> – 270° counter-clockwise rotation New in JDF 1.3 <i>Up</i> – 0° rotation.
<i>SectionIndex</i> = "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 .

Table 7-77: SignatureCell element (Section 2 of 2)

Name	Data Types	Description
StationName ? New in JDF 1.3	string	The name of the 1-up station in the die layout. This element SHOULD be present when BinderySignature/@BinderySignatureType = "Die". If BinderySignatureType = "Die" and BinderySignature/DieLayout contains more than 1 station, this attribute MUST be specified.

7.2.13 BlockPreparationParams

[New in JDF 1.1](#)

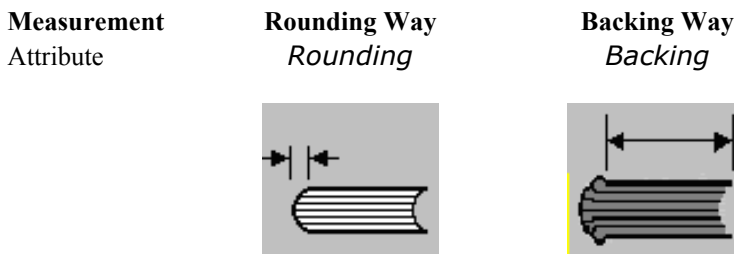
This resource describes the settings of a **BlockPreparation** process. For the tightbacking there are four different kinds of book forms.

Kinds of Book Forms
TightBacking=

Flat <i>Flat</i>	Round <i>Round</i>	Flat and Backed <i>FlatBacked</i>	Rounded and Backed <i>RoundBacked</i>
----------------------------	------------------------------	---	---



For the rounding and for the backing there are two additional measurements:



Resource Properties

Resource class: Parameter
Resource referenced by: —
Example Partition: —
Input of processes: **BlockPreparation**
Output of Processes: —

Table 7-78: BlockPreparationParams resource

Name	Data Type	Description
<i>Backing ?</i>	double	Backing distance in points.
<i>Rounding ?</i>	double	Rounding distance in points.
<i>TightBacking ?</i>	enumeration	Definition of the geometry of the back of the book block. Allowed values are: <i>Flat</i> <i>FlatBacked</i> – Backing way <i>Round</i> – Rounding way <i>RoundBacked</i> – Rounding way, backing way
RegisterRibbon *	refelement	Description of the register ribbons that are included within the book block.

7.2.14 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
Resource referenced by:	—
Example Partition:	—
Input of processes:	BoxFolding
Output of processes:	—

Table 7-79: BoxFoldingParams resource (Section 1 of 2)

Name	Data Type	Description
<i>BlankDimensionsX</i> ?	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 7-5, “BoxFoldingType attribute for values of Type00, Type01 and Type02,” on page 340 through Figure 7-8, “BoxFoldingType attribute for values of Type15 and Type20,” on page 342. The first value of <i>BlankDimensionsX</i> is the position of the fold marked by X0 in a diagram, e.g., Figure 7-5. The second value of <i>BlankDimensionsX</i> is the position of the fold marked by X1, and so on. <i>BlankDimensionsX</i> MUST NOT be specified unless <i>BoxFoldingType</i> is also specified.
<i>BlankDimensionsY</i> ?	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 7-5, “BoxFoldingType attribute for values of Type00, Type01 and Type02,” on page 340 through Figure 7-8, “BoxFoldingType attribute for values of Type15 and Type20,” on page 342. The first value of <i>BlankDimensionsY</i> is the position of the fold marked by Y0 in a diagram, e.g., Figure 7-5. The second value of <i>BlankDimensionsY</i> is the position of the fold marked by Y2, and so on. <i>BlankDimensionsY</i> MUST NOT be specified unless <i>BoxFoldingType</i> is also present.

Table 7-79: BoxFoldingParams resource (Section 2 of 2)

Name	Data Type	Description
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 MUST be described in detail. Defined values are: Type00 – Special type for boxes that are not pre-defined. See Figure 7-5 Type01 – see Figure 7-5 Type02 – see Figure 7-5 Type03 – see Figure 7-6 Type04 – see Figure 7-6 Type10 – see Figure 7-6 Type11 – see Figure 7-7 Type12 – see Figure 7-7 Type13 – see Figure 7-7 Type15 – see Figure 7-8 Type20 – see Figure 7-8
BoxApplication *	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.
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 *	refelement	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.

— Element: BoxApplication

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**.

Table 7-80: 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 MUST also be specified as in input Component to the BoxFolding process with <i>ProcessUsage</i> = "Application"
GlueLine *	refelement	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/GlueLine .

— Element: BoxFoldAction

BoxFoldAction describes an action in the folder-gluer that is perpendicular or diagonal to the movement path of the blank.

Table 7-81: BoxFoldAction element

Name	Data Type	Description
<i>FoldIndex</i>	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 <i>BlankDimensionsX</i> ; the second value of the XYPair refers to an indexed fold in <i>BlankDimensionsY</i> . If either X or Y spans multiple flaps, it MUST be set to -1.
<i>Action</i>	enumeration	Individual Action in the folder gluer: For drawings of some value specified for <i>Action</i> , see Table 7-4, “Folding examples for some values of BoxFoldAction/@Action,” on page 340. See Table 7-82 for the values.
GlueLine *	refelement	Specification of a glue lines needed to glue the Component described in this BoxApplication . The GlueLines are applied to the Component in the coordinate system of the BoxApplication/Component . The GlueLines applied to the blank are specified in BoxFoldingParams/GlueLine .

— Attribute: Action

Table 7-82: Action attribute – possible values

Value	Description
<i>LongFoldLeftToRight</i>	
<i>LongFoldRightToLeft</i>	
<i>LongPreFoldLeftToRight</i>	
<i>LongPreFoldRightToLeft</i>	
<i>FrontFoldComplete</i>	
<i>FrontFoldDiagonal</i>	
<i>FrontFoldCompleteDiagonal</i>	
<i>BackFoldComplete</i>	
<i>BackFoldDiagonal</i>	
<i>BackFoldCompleteDiagonal</i>	
<i>ReverseFold</i>	A <i>ReverseFold</i> is topologically equivalent to <i>FrontFoldDiagonal</i> but uses different equipment with other restrictions on Media weight and size and is therefore specified individually.
<i>Milling</i>	
<i>Rotate90</i>	90° counter-clockwise rotation
<i>Rotate180</i>	180° rotation
<i>Rotate270</i>	90° clockwise rotation

For instance, processing a Type01 blank (Figure 7-5, “BoxFoldingType attribute for values of Type00, Type01 and Type02,” on page 340)has the following actions:

```
<BoxFoldAction FoldIndex="0" -1"Action="LongPreFoldLeftToRight"/>
<BoxFoldAction FoldIndex="2" -1"Action="LongPreFoldRightToLeft"/>
<BoxFoldAction FoldIndex="1" -1"Action="LongFoldLeftToRight"/>
<BoxFoldAction FoldIndex="3" -1"Action="LongFoldRightToLeft"/>
```

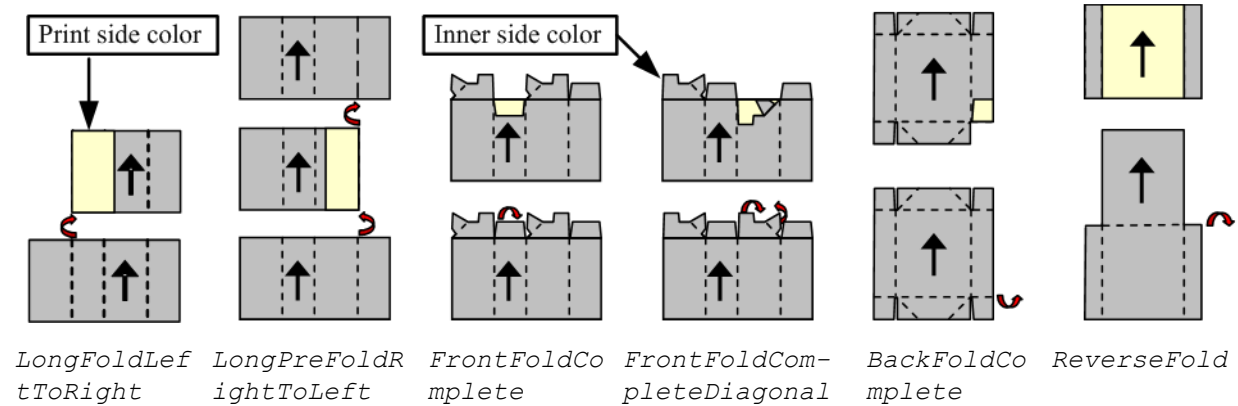


Figure 7-4: Folding examples for some values of BoxFoldAction/@Action

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

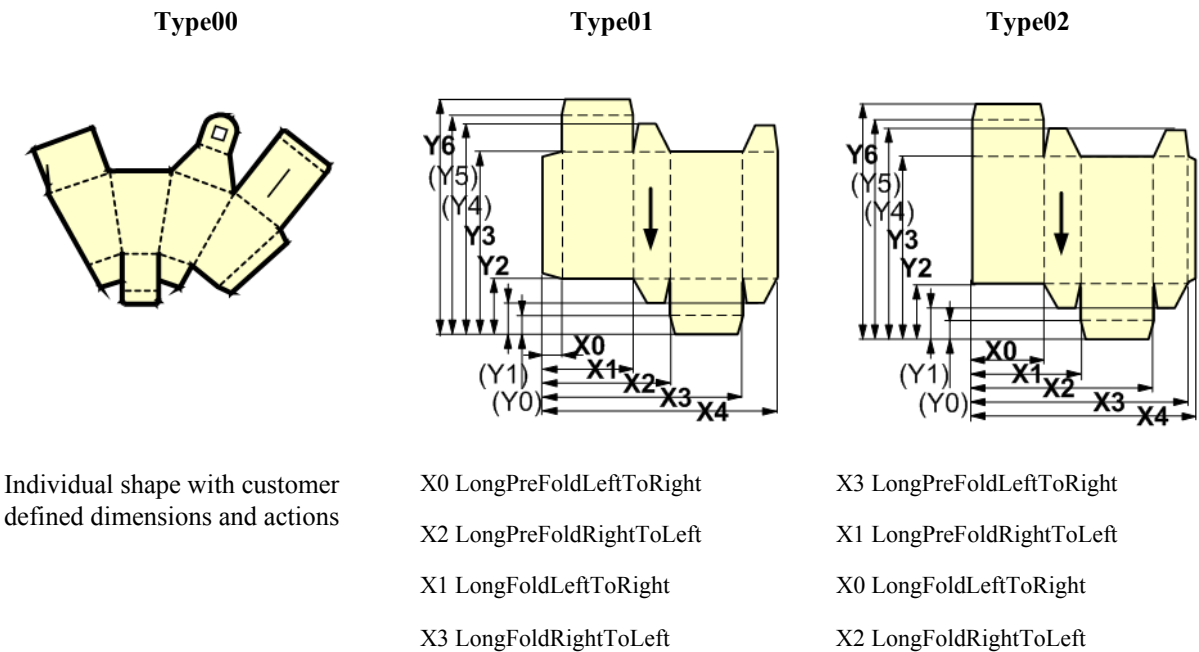


Figure 7-5: BoxFoldingType attribute for values of Type00, Type01 and Type02

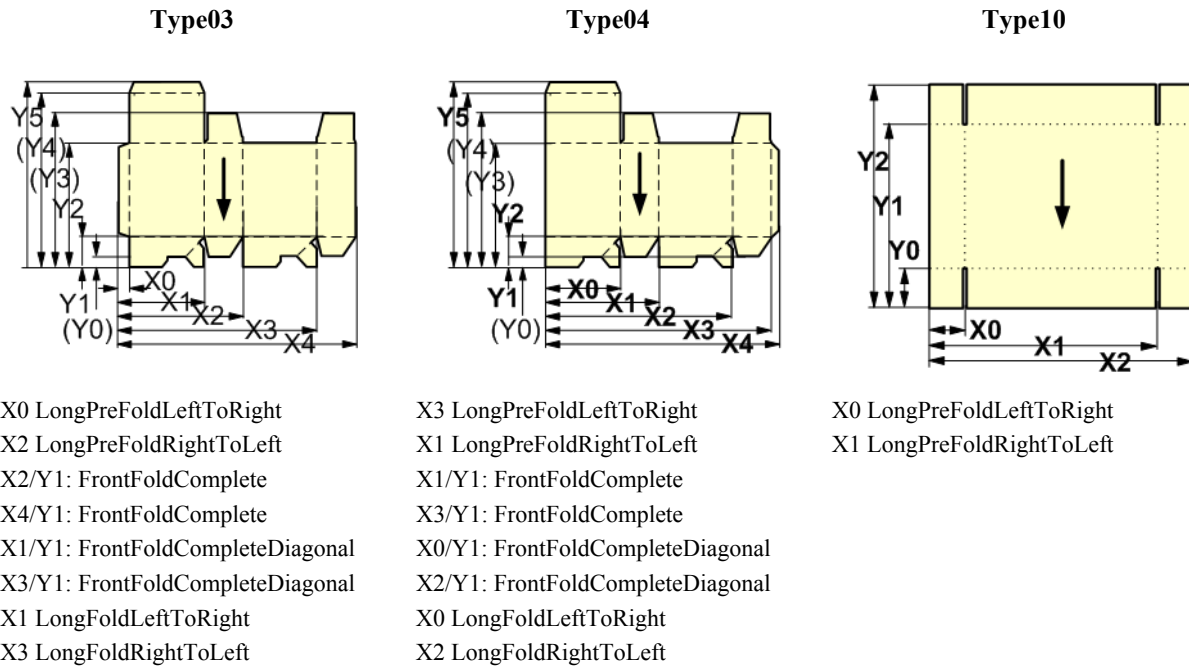


Figure 7-6: BoxFoldingType attribute for values of Type03, Type04 and Type10

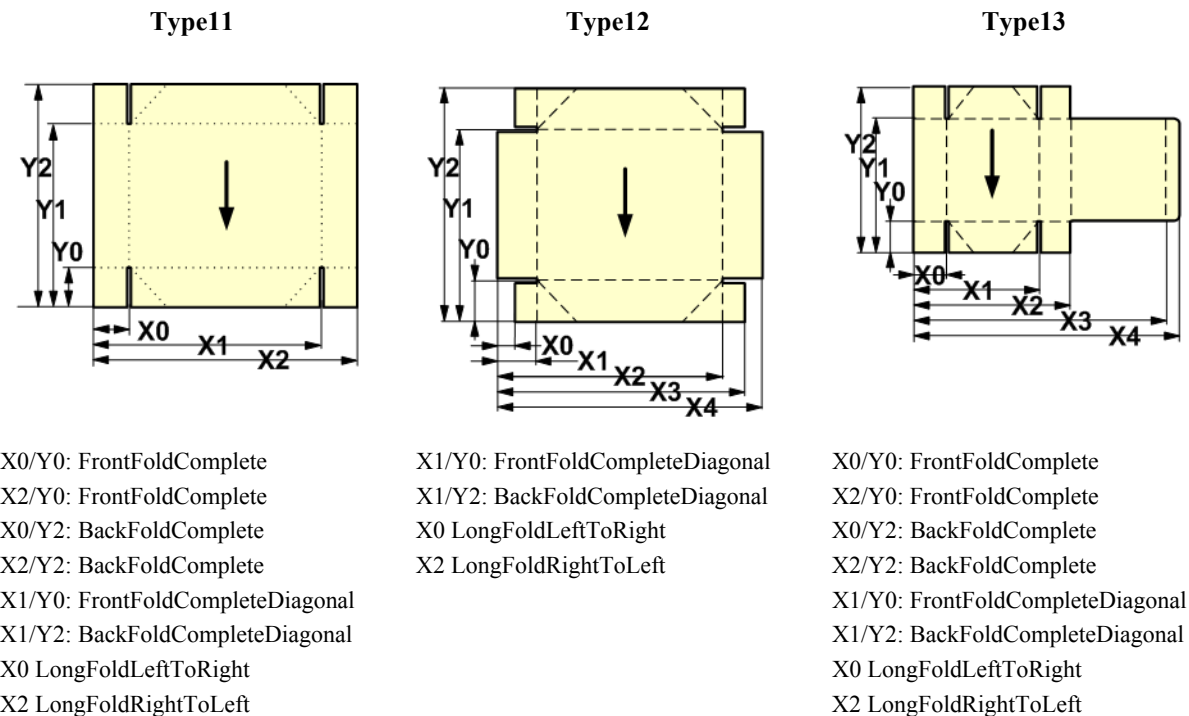
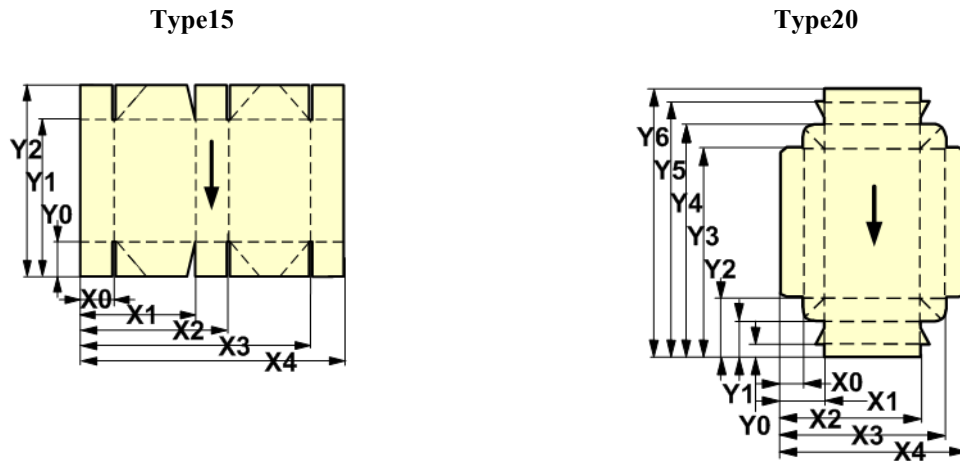


Figure 7-7: BoxFoldingType attribute for values of Type 11, Type12 and Type13



X0/Y0: FrontFoldComplete	(continued from previous column)	X0 LongFoldLeftToRight
X2/Y0 FrontFoldComplete	X3/Y0 FrontFoldCompleteDiagonal	X3 LongFoldRightToLeft
X4/Y0 FrontFoldComplete	X1/Y2 BackFoldCompleteDiagonal	
X0/Y2 BackFoldComplete	X3/Y2 BackFoldCompleteDiagonal	
X2/Y2 BackFoldComplete	X0 LongFoldLeftToRight	
X4/Y2 BackFoldComplete	X3 LongFoldRightToLeft	
X1/Y0 FrontFoldCompleteDiagonal	X2 LongFoldRightToLeft	

Figure 7-8: BoxFoldingType attribute for values of Type15 and Type20

7.2.15 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
 Resource referenced by: —
 Example Partition: —
 Input of processes: **BoxPacking**
 Output of processes: —

Table 7-83: BoxPackingParams resource (Section 1 of 2)

Name	Data Type	Description
<i>ComponentsPerRow</i> ? New in JDF 1.3	integer	Components per row in the shipping box, as illustrated by A in Figure 7-9. If the Components represent Bundles , the number of Bundles is specified.
<i>FillMaterial</i> ?	NMTOKEN	Material to fill boxes that are not completely filled, as illustrated by F in Figure 7-9. Values include: <i>Any</i> – Explicit request for system specified filling. <i>BlisterPack</i> <i>None</i> – Explicit request for no filling. <i>Paper</i> <i>Styrofoam</i>

Table 7-83: BoxPackingParams resource (Section 2 of 2)

Name	Data Type	Description
<u>Layers ?</u> New in JDF 1.3	integer	Rows per shipping box, as illustrated by L in Figure 7-9.
<u>Pattern ?</u>	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.
<u>Rows ?</u> New in JDF 1.3	integer	Rows per shipping box, as illustrated by R in Figure 7-9.
<u>Ties ?</u> 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.
<u>UnderLays ?</u> New in JDF 1.3	IntegerList	Number of underlay sheets at each layer, as illustrated by U in Figure 7-9. 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.

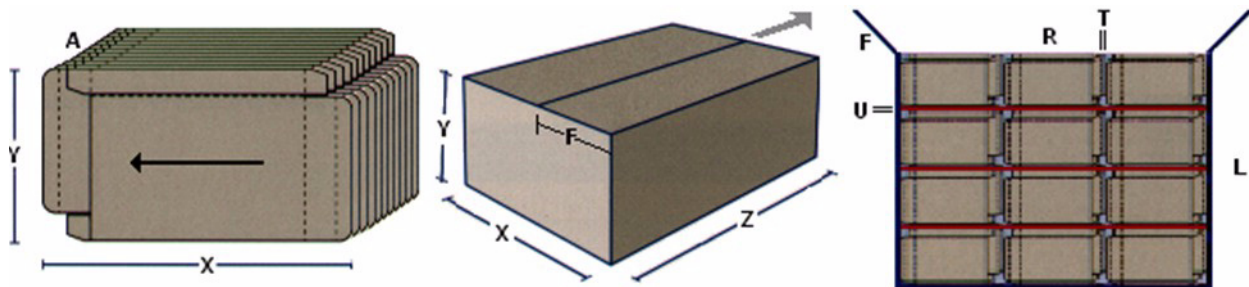


Figure 7-9: Box packing

7.2.16 BufferParams

[New in JDF 1.1](#)

This resource provides controls for **Buffer** process.

Resource Properties

- Resource class: Parameter
- Resource referenced by: —
- Example Partition: —
- Input of processes: **Buffer**
- Output of processes: —

Table 7-84: BufferParams resource

Name	Data Type	Description
<u>MinimumWait ?</u>	duration	Minimum amount of time that an individual resource MUST be buffered.

7.2.17 Bundle

[New in JDF 1.1](#)

Bundles are used to describe various kinds of sets of **Component**s. Note that **Bundle** resources can be created by many press or postpress processes and not only *Bundling*.

Resource Properties

Resource class:	Quantity
Resource referenced by:	Component
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-85: Bundle resource

Name	Data Type	Description
<i>BundleType</i> = "Stack"	enumeration	One of: <i>BoundSet</i> – Stack of components that are bound together. <i>Box</i> <i>Carton</i> <i>CollectedStack</i> – Components collected on a saddle, result of collecting process <i>CompensatedStack</i> – Loose stack of compensated components <i>Pallet</i> <i>Roll</i> – Rolled components on a print roll. <i>Sheet</i> – Multiple individual items printed onto one Sheet. <i>Stack</i> – Loose stack of equally stacked components. <i>StrappedStack</i> – Strapped stack of equally stacked components. <i>StrappedCompensatedStack</i> – Strapped stack of compensated components. <i>WrappedBundle</i>
<i>FolioCount</i> ?	integer	Total amount of individual finished pages that this bundle contains. If not specified, it MUST be calculated from the individual BundleItem elements.
<i>ReaderPageCount</i> ?	integer	Total amount of individual reader pages that this bundle contains. If not specified, it MUST be calculated from the individual BundleItem elements.
<i>TotalAmount</i> ?	integer	Total amount of individual products that this bundle contains. If the bundle contains one or more Component [contains (@ <i>ComponentType</i> , " <i>FinalProduct</i> ")], <i>TotalAmount</i> 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 MUST be calculated from the individual BundleItem elements.
BundleItem *	element	References to the individual items that form this Bundle .

— Element: **BundleItem**

A **Bundle** is described as a set of **BundleItem** elements. Since **BundleItem** elements reference **Component** resources which themselves can reference further **Bundle** resources, the structure is recursive.

Table 7-86: BundleItem element

Name	Data Type	Description
<i>Amount</i>	integer	Number of this type of items.
<i>ItemName</i> ? New in JDF 1.2	NMTOKEN	Name of the bundle item. Used for referencing individual BundleItem elements in a Bundle .
<i>Orientation</i> ?	Orientation	Named Orientation of the Component respective to the Bundle coordinate system. For details, see Table 2-4, “Matrices and Orientation values for describing the orientation of a Component,” on page 26. At most one of <i>Orientation</i> or <i>Transformation</i> MUST be specified.
<i>Transformation</i> ?	matrix	Orientation of the Component respective to the Bundle coordinate system. At most one of <i>Orientation</i> or <i>Transformation</i> MUST be specified.
Component	refelement	Reference to a Component that is part of this Bundle .

The following example code shows a JDF that describes boxing and palletizing for 4200 books. The appropriate **Bundle** elements are highlighted. The resources have not yet been completely filled in.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Bundle" Status="Waiting"
  Type="ProcessGroup" Version="1.3">
  <!-- The BoxPacking process consumes the thing to pack and the boxes-->
  <!-- The BoxPacking process creates packed boxes -->
  <JDF ID="n0235" Status="Waiting" Type="BoxPacking">
    <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"/>
    </ResourcePool>
  </JDF>
  <ResourcePool>
    <!-- This Component describes a Box with 42 Books -->
    <Component Amount="100" Class="Quantity" ID="PackedBoxID" Status="Unavailable">
      <Bundle BundleType="Box" TotalAmount="42">
        <BundleItem Amount="42">
          <ComponentRef rRef="ComponentID"/>
        </BundleItem>
      </Bundle>
    </Component>
    <Component Amount="4200" Class="Quantity" ID="ComponentID" Status="Available"/>
    <!-- This Component describes the contents of the pallet: 100 Boxes w. 42 Books -->
    <Component Amount="10" Class="Quantity" ID="palletContentsID" Status="Unavailable">
      <Bundle BundleType="Pallet" TotalAmount="420">
        <BundleItem Amount="10">
          <ComponentRef rRef="PackedBoxID"/>
        </BundleItem>
      </Bundle>
    </Component>
  </ResourcePool>
  <JDF ID="n0239" Status="Waiting" Type="Palletizing">
    <ResourceLinkPool>
      <ComponentLink Usage="Input" rRef="PackedBoxID"/>
      <PalletLink Usage="Input" rRef="palletID"/>
    </ResourceLinkPool>
  </JDF>
</JDF>
```

```

    <PalletizingParamsLink Usage="Input" rRef="palletParamsID"/>
    <ComponentLink Usage="Output" rRef="palletContentsID"/>
  </ResourceLinkPool>
  <ResourcePool>
    <Pallet Amount="10" Class="Consumable" ID="palletID" Status="Available"/>
    <PalletizingParams Class="Parameter" ID="palletParamsID" Status="Available"/>
  </ResourcePool>
</JDF>
</JDF>

```

7.2.18 BundlingParams

[New in JDF 1.2](#)

BundlingParams describes the details of a *Bundling* process.

Resource Properties

Resource class: Parameter
Resource references: —
Example Partition: —
Input of processes: *Bundling*
Output of processes: —

Table 7-87: BundlingParams resource

Name	Data Type	Description
<i>Copies</i> ?	integer	Number of copies within a bundle. <i>Copies</i> MUST NOT be specified if <i>Length</i> is present.
<i>Length</i> ?	double	Length of a bundle. <i>Length</i> MUST NOT be specified if <i>Copies</i> is present.

7.2.19 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**
Example Partition: —
Input of processes: —
Output of processes: —

Table 7-88: ByteMap resource

Name	Data Type	Description
<i>BandOrdering</i> ?	enumeration	Identifies the precedence given when ordering the produced bands. Possible values are: <i>BandMajor</i> – The position of the bands on the page is prioritized over the color. <i>ColorMajor</i> – All bands of a single color are played in order before progressing to the next plane. This is only possible with non-interleaved data. This field is REQUIRED for non-interleaved data and MUST be ignored for interleaved data if specified.
<i>FrameHeight</i>	integer	Height of the overall image that MAY be broken into multiple bands
<i>FrameWidth</i>	integer	Width of overall image that MAY be broken into multiple columns
<i>Halftoned</i>	boolean	Indicates whether or not the data has been halftoned.
<i>Interleaved</i>	boolean	If " <i>true</i> ", the data are interleaved or chunky. Otherwise the data are non-interleaved or planar.
<i>PixelSkip</i> ?	integer	Number of bits to skip between pixels of interleaved data.
<i>Resolution</i>	XYPair	Output resolution.
Band +	element	Array of bands containing raster data.
ColorPool ? New in JDF 1.2	refelement	Details of the colors represented in this ByteMap .
FileSpec (<i>RasterFileLocation</i>)?	refelement	A FileSpec resource pointing to a location where the raster is stored or is be stored shortly
PixelColorant +	element	Ordered list containing information about which colorants are represented and how many bits per pixel are used.

— Element: Band

Table 7-89: Band element

Name	Data Type	Description
<i>Data</i>	URL	Actual bytes of data.
<i>Height</i>	integer	Height in pixels of the band.
<i>Mask</i> ?	URL	1-bit mask of raster data indicating which bits of the band data to use. The mask dimensions and resolution MUST be equivalent to the contents of the band itself.
<i>WasMarked</i>	boolean	Indicates whether any rendering marks were made in this band. This attribute allows a band to be skipped if no marks were made in the band.
<i>Width</i>	integer	Width in pixels of the band.

— Element: PixelColorant

Table 7-90: PixelColorant element

Name	Data Type	Description
<i>ColorantName</i>	string	Name of colorant.
<i>PixelDepth</i>	integer	Number of bits per pixel for each colorant.

7.2.20 CaseMakingParams

[New in JDF 1.1](#)

This resource describes the settings of a **CaseMaking** process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	CaseMaking
Output of processes:	—

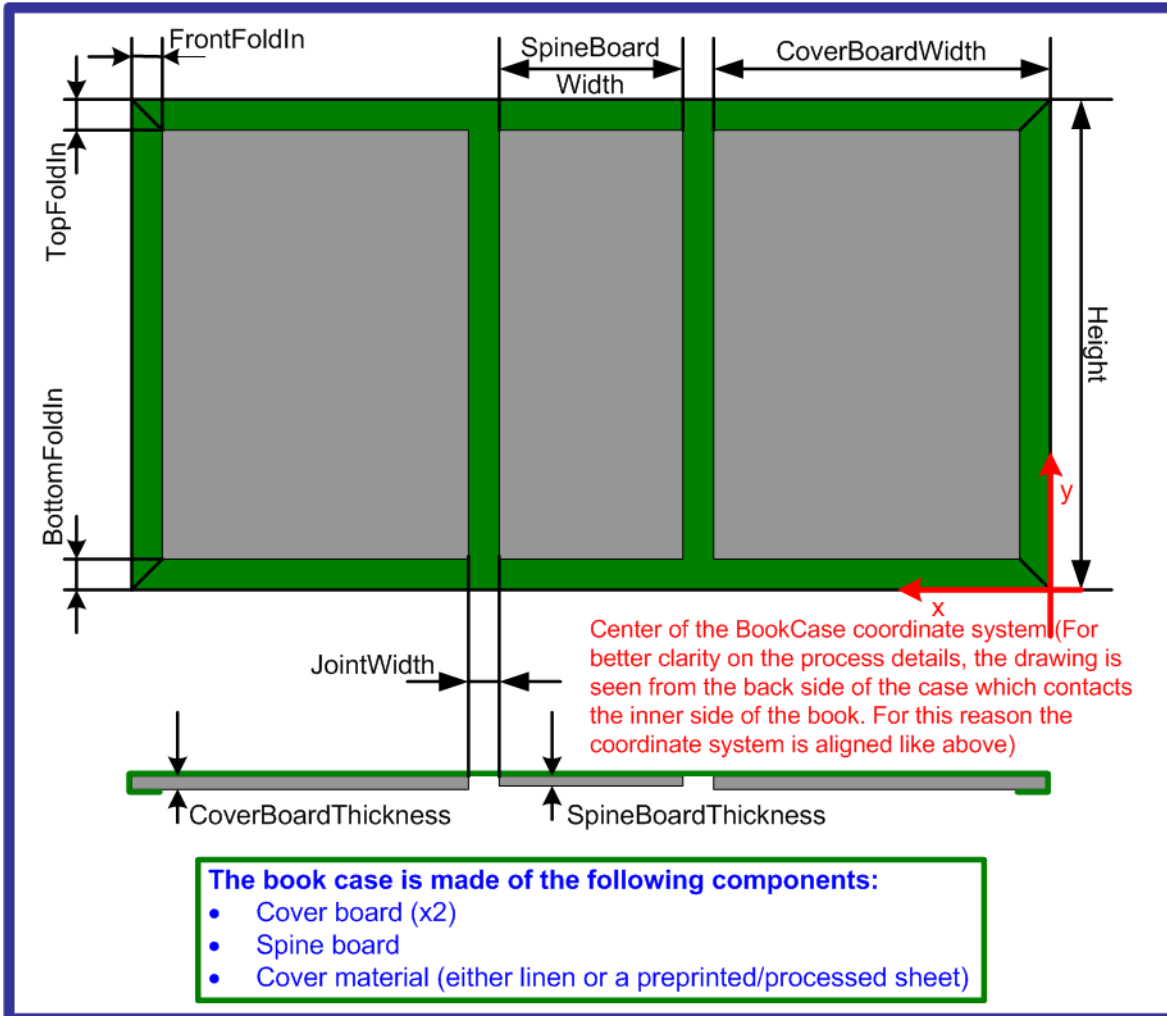


Figure 7-10: CaseMakingParams

Table 7-91: CaseMakingParams resource (Section 1 of 2)

Name	Data Type	Description
<i>BottomFoldIn</i> ?	double	Defines the width of the part of the CoverMaterial on the lower edge inside of the case. If not specified, defaults to <i>TopFoldIn</i> .
<i>CoverWidth</i> ?	double	Width of the cover cardboard in points.

Table 7-91: CaseMakingParams resource (Section 2 of 2)

Name	Data Type	Description
<i>CornerType</i> ?	NMTOKEN	Method of wrapping the corners of the cover material around the corners of the board. Possible values include: <i>LibraryCorner</i> – The American Library Corner style.
<i>FrontFoldIn</i> ?	double	Defines the width of the part of the cover material on the front edges inside of the case.
<i>Height</i> ?	double	Height of the book case, in points.
<i>JointWidth</i> ?	double	Width of the joint as seen when laying the cardboard on the cover material, in points.
<i>SpineWidth</i> ?	double	Width of the spine cardboard, in points.
<i>TopFoldIn</i> ?	double	Defines the width of the cover material on the top edge inside of the case.
GlueLine ?	refelement	Because the glue is applied to the whole back side of the cover material, GlueLine/@AreaGlue MUST be set to "true".

7.2.21 CasingInParams

[New in JDF 1.1](#)

This resource describes the settings of a *CasingIn* process. The geometry is always centered See Figure 7-11.

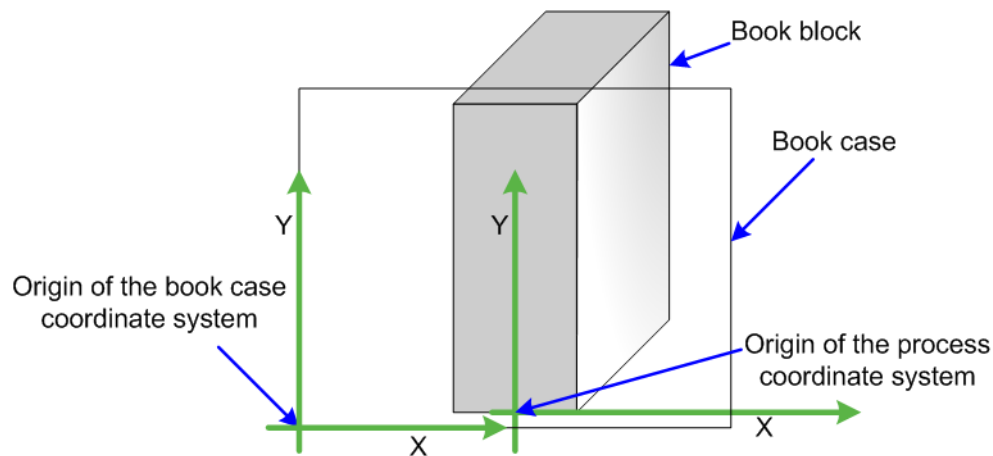


Figure 7-11: Parameters and coordinate system for CasingIn

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>CasingIn</i>
Output of processes:	—

Table 7-92: CasingInParams resource

Name	Data Type	Description
<i>CaseRadius</i> ?	double	Inner radius of the case spine rounding. If not specified, no rounding of the case spine is performed.
GlueLine +	refelement	Properties of the glue used.

7.2.22 ChannelBindingParams

This resource describes the details of the **ChannelBinding** process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	ChannelBinding
Output of processes:	—

Figure 7-12 depicts the **ChannelBinding** process.

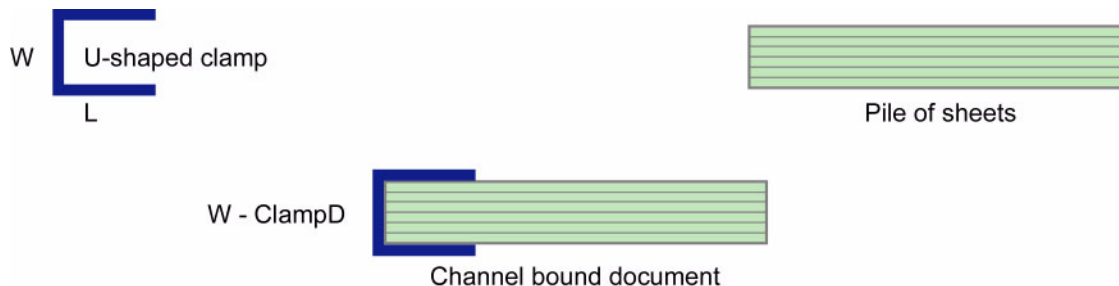


Figure 7-12: Parameters used for channel binding

The symbols W, L and ClampD of Figure 7-12 are described by the attributes *ClampD* and *ClampSize* of the table below.

Table 7-93: ChannelBindingParams resource

Name	Data Type	Description
<i>Brand</i> ?	string	The name of the clamp (or preassembled cover with clamp) manufacturer and the name of the specific item.
<i>ClampColor</i> ?	NamedColor	Determines the color of the clamp/cover. If <i>ClampSystem</i> = <i>true</i> , then the color of the cover is also meant.
<i>ClampD</i> ?	double	The distance of the clamp that was “pressed away” (see Figure 7-12 Parameters used for channel binding).
<i>ClampSize</i> ?	shape	The shape size of the clamp. The first number of the shape data type corresponds to the clamp width W (see Figure 7-12) 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 7-12). The third corresponds to the spine length (not visible in Figure 7-12). The spine length is perpendicular on the paper plane.
<i>ClampSystem</i> = “false”	boolean	If “true” the clamp is inside of a preassembled cover.

7.2.23 CIELABMeasuringField

Information about a color measuring field. The color is specified as CIE-L*a*b* value.

Resource Properties

Resource class:	Parameter
Resource referenced by:	ColorControlStrip, Surface
Example Partition:	—
Input of processes:	Any printing process
Output of processes:	—

Table 7-94: CIELABMeasuringField resource

Name	Data Type	Description
<i>Center</i>	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 , <i>Center</i> refers to the rectangle defined by <i>Center</i> and <i>Size</i> of the ColorControlStrip .
<i>CIELab</i>	LabColor	L*a*b* color specification.
<i>DensityStandard</i> ? Deprecated in JDF 1.1	enumeration	Density filter standard used during density measurements. Possible values are: <i>ANSIA</i> – ANSI Status A <i>ANSIE</i> – ANSI Status E <i>ANSII</i> – ANSI Status I <i>ANSIT</i> – ANSI Status T. <i>DIN16536</i> <i>DIN16536NB</i> In JDF 1.1 and beyond, use ColorMeasurementConditions/ @DensityStandard
<i>Diameter</i> ? Modified in JDF 1.1	double	Diameter of the measuring field.
<i>Light</i> Deprecated in JDF 1.1	NMTOKEN	Type of light. Possible values include: <i>D50</i> <i>D65</i>
<i>Observer</i> ? Deprecated in JDF 1.1	integer	Observer in degree (2 or 10). In JDF 1.1 and beyond, use ColorMeasurementConditions/ @Observer
<i>Percentages</i> ?	DoubleList	Percentage values for each separation. The number of array elements MUST match the number of separations.
<i>ScreenRuling</i> ?	DoubleList	Screen ruling values in lines per inch for each separation. The number of array elements MUST match the number of separations.
<i>ScreenShape</i> ?	string	Shape of screening dots.
<i>Setup</i> ? Deprecated in JDF 1.1	string	Description of measurement setup. In JDF 1.1 and beyond, use details from ColorMeasurementConditions
<i>Tolerance</i> ? 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.

7.2.24 CoilBindingParams

This resource describes the details of the **CoilBinding** process.

Resource Properties

Resource class: Parameter

Resource referenced by: —

Example Partition: —

Input of processes: *CoilBinding*

Output of processes: —

Table 7-95: CoilBindingParams resource

Name	Data Type	Description
<i>Brand</i> ?	string	The name of the coil manufacturer and the name of the specific item.
<i>Color</i> ?	NamedColor	Determines the color of the coil.
<i>Diameter</i> ?	double	The coil diameter to be produced is determined by the height of the block of sheets to be bound.
<i>Material</i> ?	enumeration	The material used for forming the coil binding: <i>LaqueredSteel</i> <i>NylonCoatedSteel</i> <i>PVC</i> <i>TinnedSteel</i> <i>ZincsSteel</i>
<i>Shift</i> ? 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 .
<i>Thickness</i> ?	double	The thickness of the coil.
<i>Tucked</i> = "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 .

7.2.25 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

Resource referenced by: —

Example Partition: —

Input of processes: **Collecting**

Output of processes: —

Table 7-96: CollectingParams resource

Name	Data Type	Description

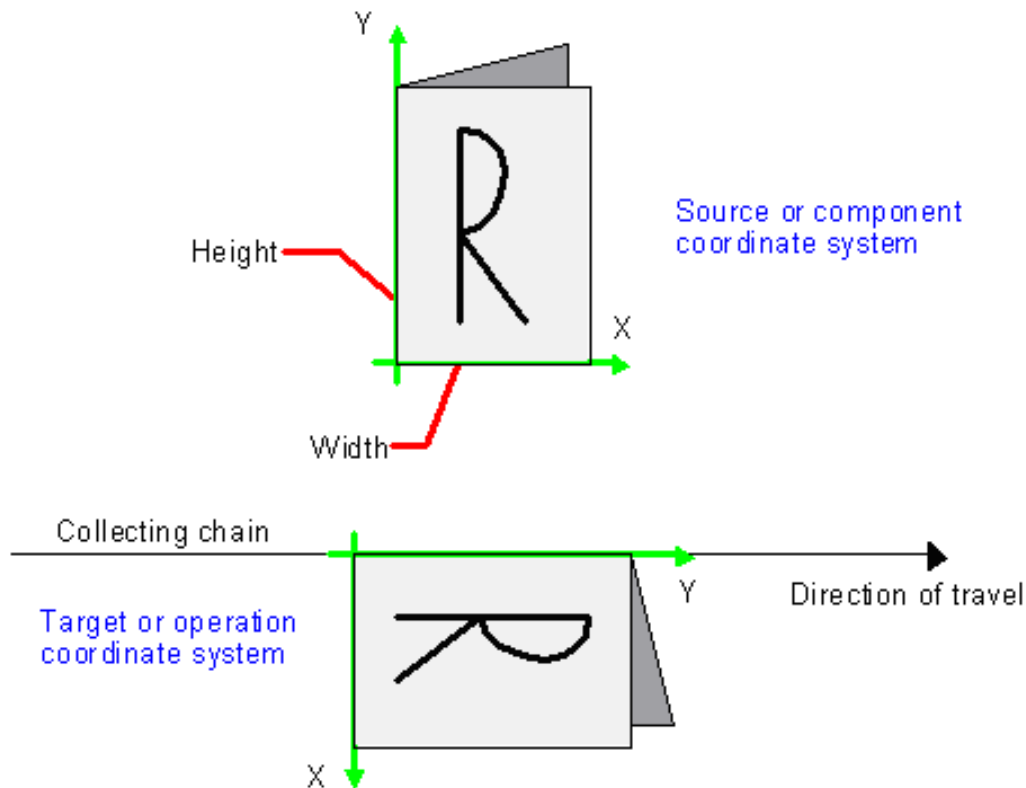


Figure 7-13: Coordinate systems used for collecting

7.2.26 Color

JDF describes spot color inks and, along that line, process color inks. 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 **MUST** 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 **MUST** match **Color/@Name**. The four names that are reserved for representing process CMYK color names are *Cyan*, *Magenta*, *Yellow* and *Black*. Every colorant can 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**
 Example Partition: —
 Input of processes: —
 Output of processes: —

Table 7-97: Color resource (Section 1 of 4)

Name	Data Type	Description
<i>CMYK</i> ? Modified in JDF 1.2	CMYKColor	CMYK value of the 100% tint value of the colorant. Although OPTIONAL, it is highly RECOMMENDED that this value be filled when the colorant is a spot colorant, (i.e., not part of the <i>ProcessColorModel</i>). This preferred CMYK MAY be associated with an ICC source profile defined in the FileSpec element with a <i>ResourceUsage</i> = "ColorProfile" when the target CMYK is different from the PDL CMYK.
<i>ColorBook</i> ? Modified in JDF 1.2	string	Definition of the color identification book name that is used to represent this color. The color book name MUST match the name defined by the color book vendor. Examples include: <i>PANTONE C</i> <i>CIP4 ColorBook Uncoated Grade 5</i> Note: The data type was modified from NMTOKEN to string in JDF 1.2.
<i>ColorBookEntry</i> ? Modified in JDF 1.2	string	Definition of the Color within the standard specified by <i>ColorBook</i> . This entry MUST exactly match the color book entry as defined by the <i>ColorBook</i> 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.
<i>ColorBookPrefix</i> ?	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.
<i>ColorBookSuffix</i> ?	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.
<i>ColorName</i> ? New in JDF 1.1	NamedColor	Mapping to a color name. Allowed values are defined in Section A.3.3.2, NamedColor.

Table 7-97: Color resource (Section 2 of 4)

Name	Data Type	Description
<p><i>ColorType</i> ? Modified in JDF 1.2</p>	enumeration	<p>A name that characterizes the colorant. Possible values are:</p> <p><i>DieLine</i> – Marks made with colorants of this type are ignored for trapping. Trapping processes need not generate a color plane for this colorant. <i>DieLine</i> can be used for auxiliary process separations. <i>DieLine</i> marks will generally appear on proof output but will not be marked on final output, (e.g., plates). Note that the ColorantControl resource MUST be correctly set up for the RIP and that <i>ColorType</i> = "<i>DieLine</i>" does not implicitly remove the <i>DieLine</i> separation from final output.</p> <p><i>Normal</i> – 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.</p> <p><i>Transparent</i> – 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.</p> <p><i>Opaque</i> – Marks covered by colorants of this type are ignored for trapping. <i>Opaque</i> can be used for metallic inks.</p> <p><i>OpaqueIgnore</i> – Marks made with colorants of this type and marks covered by colorants of this type are ignored for trapping. <i>OpaqueIgnore</i> can be used for metallic inks.</p>
<p><i>Density</i> ? New in JDF 1.2</p>	double	<p>Density value of colorant (100% tint). Whereas <i>NeutralDensity</i> describes measurements of inks on substrate with wide-band filter functions, <i>Density</i> is derived from measurements of inks on substrate with special small-band filter functions according to ANSI and DIN.</p>
<p><i>Lab</i> ?</p>	LabColor	<p>L, a, b value of the 100% tint value of the colorant.</p>
<p><i>MappingSelection</i> = "<i>UsePDLValues</i>" New in JDF 1.2</p>	enumeration	<p>This value specifies the mapping method to be used for this Color. Possible values are:</p> <p><i>UsePDLValues</i> – Use color values specified in the PDL for this color. See [ColorPS].</p> <p><i>UseLocalPrinterValues</i> – Use the Printer’s best local mapping for this Color.</p> <p><i>UseProcessColorValues</i> – Use the values defined in this Color.</p> <p>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.</p>

Table 7-97: Color resource (Section 3 of 4)

Name	Data Type	Description
MediaType ? Modified in JDF 1.2	string	Specifies the media type. Possible values include: <i>Coated</i> – Pertains to gloss coated. <i>Matte</i> – Pertains to matte or dull coated. <i>Uncoated</i>
<i>Name</i>	string	Name of the colorant. This is the value that MUST match the <i>Name</i> attribute of a SeparationSpec that references this color, (e.g., in ColorantControl/DeviceNSpace/SeparationSpec/@Name or ColorantControl/ColorantParams/SeparationSpec/@Name). This <i>Name</i> attribute MAY also be referenced from the <i>Name</i> attribute in the Ink resource. Name MAY also be referenced from ColorantAlias/@ReplacementColorantName . Only one Color with any given <i>Name</i> MUST be specified in a ColorPool .
<i>NeutralDensity</i> ?	double	A number in the range of 0.001 to 10 that represents the neutral density of the colorant, defined as $10 \cdot \log(1/Y)$. Y is the tristimulus value in CIEXYZ coordinates, normalized to 1.0.
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 <i>RawName</i> MUST be specified in a ColorPool .
<i>sRGB</i> ?	sRGBColor	sRGB value of the 100% tint value of the colorant.
UsePDALternateCS ? Deprecated in JDF 1.2	boolean	If " <i>true</i> ", the alternate color space definition defined in the PDL MUST be used for color space transformations when available. If " <i>false</i> ", the alternate color space definitions defined in <i>sRGB</i> , <i>CMYK</i> or DeviceNColor of this Color MUST be used depending on the value of ProcessColorModel in ColorantControl . In JDF 1.2 and beyond, use <i>MappingSelection</i> .
ColorMeasurementConditions ? New in JDF 1.1	reference	Detailed description of the measurement conditions for color measurements.
<i>DeviceNColor</i> *	element	Elements that define the colorant in a non-standard device-dependent process color space. <i>DeviceNColor</i> can be specified when <i>Name</i> is a spot colorant (not one of the <i>DeviceNSpace</i> colorants) and <i>ProcessColorModel</i> = " <i>DeviceN</i> " in ColorantControl .

Table 7-97: Color resource (Section 4 of 4)

Name	Data Type	Description
FileSpec (<i>ColorProfile</i>)?	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 <i>CMYK</i> attribute.
FileSpec (<i>TargetProfile</i>)?	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. <i>TargetProfile</i> applies to the alternate color defined by the value of <i>UsePDALternateCS</i> .
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”

— Element: DeviceNColor

Table 7-98: DeviceNColor element

Name	Data Type	Description
<i>ColorList</i>	DoubleList	Value of the 100% tint value of the colorant in the ordered DeviceN space. The list MUST have <i>N</i> elements. A value of 0 specifies no ink and a value of 1 specifies full ink. The mapping of indices to colors is specified in the <i>DeviceNSpace</i> element of the ColorantControl resource.
<i>N</i>	integer	Number of colors that define the color space.
<i>Name</i>	string	Color space name, (e.g., HexaChrome or HiFi). <i>Name</i> MUST match ColorantControl/DeviceNSpace/@Name .

— Element: PrintConditionColor

[New in JDF 1.2](#)

The **PrintConditionColor** 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, 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 7-99: PrintConditionColor element (Section 1 of 2)

Name	Data Type	Description
<i>CMYK</i> ?	CMYKColor	<i>CMYK</i> of the PrintConditionColor . If not specified, defaults to the parent Color/@CMYK .
<i>ColorBook</i> ?	string	<i>ColorBook</i> of the PrintConditionColor . If not specified, defaults to the parent Color/@ColorBook .
<i>ColorBookEntry</i> ?	string	<i>ColorBookEntry</i> of the PrintConditionColor . If not specified, defaults to the parent Color/@ColorBookEntry .
<i>ColorBookPrefix</i> ?	string	<i>ColorBookPrefix</i> of the PrintConditionColor . If not specified, defaults to the parent Color/@ColorBookPrefix .

Table 7-99: PrintConditionColor element (Section 2 of 2)

Name	Data Type	Description
<i>ColorBookSuffix</i> ?	string	<i>ColorBookSuffix</i> of the PrintConditionColor. If not specified, defaults to the parent Color / <i>@ColorBookSuffix</i> .
<i>Density</i> ?	double	<i>Density</i> of the PrintConditionColor. If not specified, defaults to the parent Color / <i>@Density</i> .
<i>Lab</i> ?	LabColor	<i>Lab</i> of the PrintConditionColor. If not specified, defaults to the parent Color / <i>@Lab</i> .
<i>MappingSelection</i> ? New in JDF 1.2	enumeration	This value specified the mapping method to be used for this Color. Possible values are: <i>UsePDLValues</i> – Use color values specified in the PDL for this color. See [ColorPS]. <i>UseLocalPrinterValues</i> – Use the Printer's best local mapping for this Color . <i>UseProcessColorValues</i> – Use the values defined in this Color . If not specified, defaults to the parent Color / <i>@MappingSelection</i> .
<i>MediaSide</i> = "Both"	enumeration	Media front and back surfaces can be different, affecting color results. If the Media / <i>@FrontCoatings</i> , Media / <i>@BackCoatings</i> or Media / <i>@Gloss</i> attributes indicate differences in surface then Media-Side can be used to specify the side of the media to which the PrintConditionColor attributes pertain. Values are: <i>Front</i> <i>Back</i> <i>Both</i>
<i>NeutralDensity</i> ?	double	<i>NeutralDensity</i> of the PrintConditionColor. If not specified, defaults to the parent Color / <i>@NeutralDensity</i>
<i>PrintConditionName</i> ?	NMTOKEN	<i>PrintConditionName</i> specifies a particular screening condition and printing condition that this PrintConditionColor element applies to. In order to map a PrintCondition with a PrintConditionColor, <i>PrintConditionName</i> MUST match PrintCondition / <i>@Name</i> . If not specified, this PrintConditionColor matches all PrintCondition but MAY still be dependent on Media .
<i>sRGB</i> ?	sRGBColor	<i>sRGB</i> of the PrintConditionColor. If not specified, defaults to the parent Color / <i>@sRGB</i> .
<i>DeviceNColor</i> *	element	<i>DeviceNColor</i> of the PrintConditionColor. If not specified, defaults to the parent Color / <i>@DeviceNColor</i> .
FileSpec (<i>TargetProfile</i>)	refelement	FileSpec (<i>TargetProfile</i>) of the PrintConditionColor. If not specified, defaults to the parent Color / FileSpec (<i>TargetProfile</i>)
Media *	refelement	Specifies one or more Media that this PrintConditionColor applies to. When PrintConditionColor is present, the parent attribute, Color / <i>@MediaType</i> , is ignored. If Media is not specified, PrintConditionColor applies to print processes with a matching <i>PrintConditionName</i> .
TransferCurve *	refelement	TransferCurve of the PrintConditionColor. If not specified, defaults to the parent Color / TransferCurve .

Color Example

This is an example of the structure for **Color**. The transfer curves in this example are defined for process CMYK and sRGB, independently.


```
<Color 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>
```

7.2.27 ColorantAlias

ColorantAlias is a resource 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**". Note that **ColorantAlias** was elevated from a sub-element of **ColorantControl** to a top level resource in JDF 1.2.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	ColorIntent, ColorantControl
Input of Processes:	—
Output of processes:	—

Table 7-100: ColorantAlias resource

Name	Data Type	Description
<i>ReplacementColorantName</i>	string	The value of the colorant name string to be substituted for the colorant name strings in the SeparationSpec element list.
SeparationSpec + Modified in JDF 1.2	element	The names of the colorants to be replaced in PDL files.

7.2.28 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 is to 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. 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 an 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

Resource referenced by: —

Example Partition: *DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName*

Input of Processes: ***ConventionalPrinting, ColorSpaceConversion, DigitalPrinting, ImageSetting, Interpreting, PreviewGeneration, Separation, Trapping***

Output of processes: —

Table 7-101: ColorantControl resource (Section 1 of 3)

Name	Data Type	Description
<i>ForceSeparations</i> = "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.
<i>ProcessColorModel</i> ?	NMTOKEN	Specifies the model to be used for rendering the colorants defined in color spaces into process colorants. Possible values include: <i>DeviceCMY</i> <i>DeviceCMYK</i> <i>DeviceGray</i> <i>DeviceN</i> – The specific DeviceN color space to operate on is defined in the DeviceNSpace resource. If this value is specified then the DeviceNSpace and ColorPool relements MUST also be present. <i>DeviceRGB</i>
ColorantAlias * Modified in JDF 1.2	relement	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.

Table 7-101: ColorantControl resource (Section 2 of 3)

Name	Data Type	Description
ColorantOrder ?	element	<p>The ordering of named colorants to be processed, for example in the RIP. All of the colorants named MUST either occur in the ColorantParams list or be implied by the <i>ProcessColorModel</i>. If present, then only the colorants specified by <i>ColorantOrder</i> MUST be output. Colorants listed in the ColorantParams list, or implied by the <i>ProcessColorModel</i>, but not listed in ColorantOrder, MUST NOT be output. They MUST still be processed for side effects in the colorants that are listed such as knockouts or trapping.</p> <p>If not present, then all colorants specified in ColorantParams and implied by <i>ProcessColorModel</i> 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.</p>
ColorantParams ?	element	<p>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 <i>ProcessColorModel</i> equivalents. (See ElementColorParams and <i>ColorSpaceConversion</i> process.) The colorants implied by the value of <i>ProcessColorModel</i> are assumed and MUST 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 intent to process conversion.</p>
ColorPool ?	refelement	<p>Pool of Color elements that define the specifics of the colors implied by <i>ProcessColorModel</i> and named in <i>ColorantControl</i>. <i>ColorantControl</i> uses a subset of the total ColorPool. The subset that <i>ColorantControl</i> uses from ColorPool is the subset of <i>ProcessColorModel</i> colors (possibly all), and the subset of spot colors (possibly all) designated to be processed in this instance using specific separation colorants.</p> <p>ColorPool in total includes spot colors in the job for which a JDF process color equivalent mapping is required. Those colors are used by <i>ColorSpaceConversion</i> 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.</p>
ColorSpaceSubstitute *	element	<p>Each subelement identifies a colorant that is to be replaced by another colorant.</p>

Table 7-101: ColorantControl resource (Section 3 of 3)

Name	Data Type	Description
DeviceColorantOrder ?	element	The ordering of named colorants (e.g., order of laying them down) to be output on the device ^a , such as press modules. All of the named colorants MUST occur in ColorantOrder if it is present. If ColorantOrder is not present, then all of the named colorants MUST occur in the ColorantParams list, or be implied by the <i>ProcessColorModel</i> . If the DeviceColorantOrder element is not specified, the order for laying down colorants defaults to ColorantOrder .
DeviceNSpace * Modified in JDF 1.2	refelement	Defines the colorants that make up a DeviceN color space. The DeviceNSpace attribute is REQUIRED when the <i>ProcessColorModel</i> value is <i>DeviceN</i> .

a. Note that this **MUST** be synchronized with the device output ICC profile.

— **Element: ColorantOrder, ColorantParams and DeviceColorantOrder**

Table 7-102: ColorantOrder, ColorantParams and DeviceColorantOrder element

Name	Data Type	Description
SeparationSpec * Modified in JDF 1.2	element	The names of the colorants that define the respective lists.

— **Element: ColorSpaceSubstitute**

Table 7-103: 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 <i>Separation</i> 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 7-104: Sample output for different values of ProcessColorModel, ColorantParams, ColorantOrder, ColorantControlLink and DeviceColorantOrder Elements.

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	—	Yel- low 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 Yel- low Black	Cyan Magenta
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.

7.2.29 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

j = Value of the *Rotation* attribute

Resource Properties

Resource class: Parameter

Resource referenced by: **Surface**

Example Partition: —
Input of processes: Any printing process.
Output of processes: —

Table 7-105: ColorControlStrip resource

Name	Data Type	Description
<i>Center</i>	XYPair	Position of the center of the color control strip in the coordinates of the MarkObject that contains this mark.
<i>Rotation ?</i>	double	Rotation in degrees. Positive graduation figures indicate counter-clockwise rotation; negative figures indicate clockwise rotation.
<i>Size</i>	XYPair	Size of the color control strip.
<i>StripType ?</i>	NMTOKEN	Type of color control strip. This attribute can be used for specifying a predefined, company-specific color control strip.
CIELABMeasuringField * New in JDF 1.1	refelement	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 .

7.2.30 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 see “Color Adjustment Attribute Description and Usage” on page 719 attribute description.

Note: These color adjustments are not available in any product 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
Resource referenced by: —
Example Partition: *DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName*
Input of processes: **ColorCorrection**
Output of processes: —

Table 7-106: ColorCorrectionParams resource

Name	Data Type	Description
<i>ColorManagementSystem</i> ?	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 MUST match one of the registered four-character ICC CMM Type values. See the <i>ICC Manufacturer’s Signature Registry</i> at http://www.color.org . Example values: “ADBE” for the Adobe CMM and “KODA” for the Kodak CMM.
FileSpec (<i>FinalTargetDevice</i>)?	refelement	A FileSpec resource pointing to an ICC profile that describes the characterization of the final output target device.
FileSpec (<i>WorkingColorSpace</i>)? Deprecated in JDF 1.1	refelement	A FileSpec resource pointing to an ICC profile that describes the assumed characterization of <i>CMYK</i> , <i>RGB</i> and <i>Gray</i> color spaces.
ColorCorrectionOp *	element	List of ColorCorrectionOp subelements.

— Element: ColorCorrectionOp

Table 7-107: ColorCorrectionOp element (Section 1 of 2)

Name	Data Type	Description
<i>SourceObjects</i> = “All”	enumerations	Identifies which class(es) of incoming graphical objects will be operated on. Possible values are: <i>All</i> <i>ImagePhotographic</i> – Contone images. <i>ImageScreenShot</i> – Images largely comprised of rasterized vector art. <i>Text</i> <i>LineArt</i> – Vector objects other than text. <i>SmoothShades</i> – Gradients and blends.
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.)
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.)
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.)

Table 7-107: ColorCorrectionOp element (Section 2 of 2)

Name	Data Type	Description
<i>AdjustContrast</i> ? 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.)
<i>AdjustHue</i> ? 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.)
<i>AdjustLightness</i> ? 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.)
<i>AdjustSaturation</i> ? 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.)
FileSpec (<i>AbstractProfile</i>)? 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 (<i>DeviceLinkProfile</i>)? 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.)

7.2.31 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.

Resource Properties

Resource class:	Parameter
Resource referenced by:	CIELABMeasuringField, Color, DensityMeasuringField, Media, PrintCondition
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-108: ColorMeasurementConditions resource

Name	Data Type	Description
<i>DensityStandard</i> = "ANSIT"	enumeration	Density filter standard used during density measurements. Possible values are: <i>ANSIA</i> – ANSI Status A <i>ANSIE</i> – ANSI Status E <i>ANSII</i> – ANSI Status I <i>ANSIT</i> – ANSI Status T <i>DIN16536</i> <i>DIN16536NB</i>
<i>Illumination</i> = "D50"	enumeration	Illumination used during spectral measurements. Possible values are: <i>D50</i> <i>D65</i> <i>Unknown</i>
<i>InkState</i> ?	enumeration	State of the ink during color measurements. Possible values are: <i>Dry</i> <i>Wet</i> NA Deprecated in JDF 1.2
<i>Instrumentation</i> ?	string	Specific instrumentation used for color measurements, (e.g., manufacturer, model number and serial number).
<i>MeasurementFilter</i> ?	enumeration	Optical Filter used during color measurements. Possible values are: <i>None</i> – No filter used. <i>Pol</i> – Polarization filter used <i>UV</i> – Ultraviolet cut filter used
<i>Observer</i> = "2"	integer	CIE standard observer function (2 degree and 10 degree) used during spectral measurements. Values are in degree (2 or 10).
<i>SampleBacking</i> ?	enumeration	Backing material used behind the sample during color measurements. Possible values are: <i>Black</i> <i>White</i> NA Deprecated in JDF 1.2
<i>WhiteBase</i> ?	enumeration	Reference for white calibration used for density measurements. Possible values are: <i>Absolute</i> – Means the instrument is calibrated to a device-specific calibration target (absolute white) for absolute density measurements. <i>Paper</i> – Means the instrument is calibrated relative to paper white

7.2.32 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:	ColorantControl
Example Partition:	—
Input of processes:	—

Output of processes: —

Table 7-109: ColorPool resource

Name	Data Type	Description
Color *	element	Individual named color.
<i>ColorantSetName ?</i>	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 .

7.2.33 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, described below, 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

Resource referenced by: —

Example Partition: *DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName*Input of processes: **ColorSpaceConversion**

Output of processes: —

Table 7-110: ColorSpaceConversionParams resource (Section 1 of 2)

Name	Data Type	Description
<i>ColorManagementSystem ?</i>	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 MUST match one of the registered four-character ICC CMM Type values. See the <i>ICC Manufacturer's Signature Registry</i> at http://www.color.org . Example values: “ <i>ADBE</i> ” for the Adobe CMM and “ <i>KODA</i> ” for the Kodak CMM.
<i>ConvertDevIndepColors ?</i> Deprecated in JDF 1.1	boolean	When “ <i>true</i> ”, incoming device-independent colors are processed to the selected device space. If the chosen operation is <i>Untag</i> 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 <i>ConvertDevIndepColors</i> is superseded by including one or more ColorSpaceConversionOp with <i>SourceCS</i> = “ <i>DevIndep</i> ” in JDF 1.1.

Table 7-110: ColorSpaceConversionParams resource (Section 2 of 2)

Name	Data Type	Description
<p><i>ICCProfileUsage</i> = "UsePDL" New in JDF 1.2</p>	enumeration	<p>This attribute specifies where to obtain the destination profile for the current iteration of the ColorSpaceConversion process, (i.e., either from the PDL (PDF/X job content) or supplied in the JDF ColorSpaceConversionParams resource). <i>ICCProfileUsage</i> provides an order precedence. Possible <i>ICCProfileUsage</i> values are:</p> <p><i>UsePDL</i> –</p> <ol style="list-style-type: none"> 1 Use the embedded profile. 2 Use the profile specified in the LayoutElement/ElementColorParams/FileSpec(Reference-OutputProfile). 3 Use the profile specified in the LayoutElement/ElementColorParams/FileSpec(ActualOutputProfile). 4 Use the profile specified in ColorSpaceConversionParams/FileSpec(FinalTargetDevice). 5 Use the system specified profile. <p><i>UseSupplied</i> –</p> <ol style="list-style-type: none"> 1 Use the profile specified in the LayoutElement/ElementColorParams/FileSpec(Reference-OutputProfile). 2 Use the profile specified in the LayoutElement/ElementColorParams/FileSpec(ActualOutputProfile). 3 Use the profile specified in ColorSpaceConversionParams/FileSpec(FinalTargetDevice). 4 Use the system specified profile.
<p>FileSpec <i>(FinalTargetDevice)?</i></p>	refelement	<p>A FileSpec resource pointing to an ICC profile that describes the characterization of the final output target device. FileSpec is REQUIRED when converting, but OPTIONAL for tagging or untagging.</p>
<p>FileSpec <i>(WorkingColorSpace)?</i> Deprecated in JDF 1.1</p>	refelement	<p>A FileSpec resource pointing to an ICC profile that describes the assumed characterization of <i>CMYK</i>, <i>RGB</i> and <i>Gray</i> color spaces.</p>
<p>ColorSpaceConversionOp *</p>	element	<p>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.</p> <p>If multiple, overlapping ColorSpaceConversionOp elements are specified, the operations that define individual <i>SourceCS</i> override the elements specifying groups of source color spaces.</p>

7.2.34 ColorSpaceConversionOp

Note: **ColorSpaceConversionOp** was elevated from a sub-element of **ColorSpaceConversionParams** in JDF 1.2. The **ColorSpaceConversionOp** resource 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>.

Resource Properties

Resource class:	ResourceElement
Resource referenced by:	ColorSpaceConversionParams, LayoutElement
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-111: ColorSpaceConversionOp resource (Section 1 of 3)

Name	Data Type	Description
<i>IgnoreEmbeddedICC</i> = "false"	boolean	If "true", specifies that embedded source ICC profiles MUST be ignored and that the ICC profile [ICC.1] defined by <i>SourceProfile</i> MUST be used instead.
<i>Operation</i> ? Modified in JDF 1.2	enumeration	Controls which of five functions the color space conversion utility performs. Possible values are: <i>Convert</i> – Transforms graphical elements to final target color space. <i>Tag</i> – Associates appropriate working space profile with uncharacterized graphical element. <i>Untag</i> – Removes all profiles and color characterizations from graphical elements. <i>Retag</i> – Equivalent to a sequence of <i>Untag</i> → <i>Tag</i> . <i>ConvertIgnore</i> – Equivalent to a sequence of <i>Untag</i> → <i>Convert</i> . <i>Operation</i> MUST be specified in the context of ColorSpaceConversionParams/ColorSpaceConversionOp and MUST NOT be specified in the context of ElementColorParams/ColorSpaceConversionOp . Note: The table below describes the effect of this attribute in combination with the <i>SourceCS</i> and <i>IgnoreEmbeddedICC</i> attributes.
<i>PreserveBlack</i> = "false" New in JDF 1.1	boolean	Controls how the tints of black (K in CMYK) are to be handled. If <i>PreserveBlack</i> is "false", these colors are processed through the standard ICC workflow. If <i>PreserveBlack</i> is "true", these colors are to be converted into other shades of black. The algorithm is implementation-specific.

Table 7-111: ColorSpaceConversionOp resource (Section 2 of 3)

Name	Data Type	Description
<i>RenderingIntent</i> = "ColorSpaceDependent" Modified in JDF 1.3	enumeration	Identifies the rendering intents associated with <i>SourceObjects</i> elements. Possible ICC-defined rendering intent values are: <i>Saturation</i> <i>Perceptual</i> <i>RelativeColorimetric</i> <i>AbsoluteColorimetric</i> <i>ColorSpaceDependent</i> – The rendering intent is dependent on the color space. The dependencies are implementation specific. Modified in JDF 1.3
<i>RGBGray2Black</i> = "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 <i>SourceObjects</i> . 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 <i>RGBGray2BlackThreshold</i> are not converted. <i>RGBGray2Black</i> and <i>RGBGray2BlackThreshold</i> 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.
<i>RGBGray2BlackThreshold</i> = "1" New in JDF 1.2	double	A value between "0.0" and "1.0" which specifies the threshold value above which the Device MUST NOT convert gray (R = G = B) to black (K only) when <i>RGBGray2Black</i> 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 <i>RGBGray2Black</i> = "true".
<i>SourceCS</i> Modified in JDF 1.3	enumeration	Identifies which of the incoming color spaces will be operated on. See Table 7-112, "SourceCS attribute – possible values," on page 372. See also Table 7-113, "Mapping of SourceCS enumeration values to color spaces in the most common input file formats," on page 373.
<i>SourceObjects</i> = "All"	enumerations	List of object classes that identifies which incoming graphical objects will be operated on. Possible values are: <i>All</i> <i>ImagePhotographic</i> – Contone images. <i>ImageScreenShot</i> – Images largely comprised of rasterized vector art. <i>Text</i> <i>LineArt</i> – Vector objects other than text. <i>SmoothShades</i> – Gradients and blends.

Table 7-111: ColorSpaceConversionOp resource (Section 3 of 3)

Name	Data Type	Description
<i>SourceRenderingIntent</i> ? 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 <i>SourceObjects</i> and <i>SourceCS</i> attributes. Possible ICC-defined [ICC.1] rendering intent values are: <i>Saturation</i> <i>Perceptual</i> <i>RelativeColorimetric</i> <i>AbsoluteColorimetric</i> <i>ColorSpaceDependent</i> – The rendering intent is dependent on the color space. The dependencies are implementation specific. Modified in JDF 1.3 If not specified, <i>SourceRenderingIntent</i> inherits the value of <i>RenderingIntent</i> . Note: The <i>SourceRenderingIntent</i> 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).
DeviceNSpace ? New in JDF 1.2	refelement	DeviceNSpace resource that describe the DeviceN color space on which to operate when <i>SourceCS</i> = "DeviceN". Individual colorant definitions for the colorant names given in DeviceNSpace are provided in the ColorantControl/ColorPool resource, which MUST also be present
FileSpec (<i>AbstractProfile</i>) ? New in JDF 1.2	refelement	A FileSpec resource pointing to an ICC profile [ICC.1] that describes the characterization of an Abstract Profile for specifying a preference adjustment.
FileSpec ? (<i>SourceProfile</i>)	refelement	A FileSpec resource pointing to an ICC profile [ICC.1] that describes the assumed source color space. See <i>IgnoreEmbeddedICC</i> for policies on using external profiles.
SeparationSpec * New in JDF 1.2	element	SeparationSpec element(s) defining on which separation(s) to operate when <i>SourceCS</i> = "Separation".

— Attribute: SourceCS**Table 7-112: SourceCS attribute – possible values (Section 1 of 2)**

Value	Description
<i>CalGray</i>	defines a calibrated device independent representation of Gray. New in JDF 1.3
<i>CalRGB</i>	defines a calibrated based device independent representation of RGB. New in JDF 1.3 Note: JDF 1.1 defined that <i>CalRGB</i> be treated as <i>RGB</i> , <i>CalGray</i> as <i>Gray</i> and <i>ICCBased</i> color spaces as one of <i>Gray</i> , <i>RGB</i> or <i>CMYK</i> depending on the number of channels.
<i>Calibrated</i>	Operates on <i>CalGray</i> and <i>CalRGB</i> color spaces. New in JDF 1.2
<i>CIEBased</i>	Operates on CIE-based color spaces (<i>CIEBasedA</i> , <i>CIEBasedABC</i> , <i>CIEBasedDEF</i> and <i>CIEBasedDEFG</i>). New in JDF 1.2
<i>CMYK</i>	Operates on DeviceCMYK.

Table 7-112: SourceCS attribute – possible values (Section 2 of 2)

Value	Description
<i>DeviceN</i>	Identifies the source color encoding as a <i>DeviceN</i> color space. The specific <i>DeviceN</i> color space to operate on is defined in the DeviceNSpace resource. If <i>DeviceN</i> is specified, then the DeviceNSpace and ColorantControl/ColorPool referents MUST also be present. New in JDF 1.2
<i>DevIndep</i>	Operates on device independent color spaces (equivalent to <i>Calibrated</i> , <i>CIEBased</i> , <i>ICCBased</i> , <i>Lab</i> or <i>YUV</i>).
<i>Gray</i>	Operates on <i>DeviceGray</i> .
<i>ICCBased</i>	Operates on color spaces defined using ICC profiles. The <i>ICCBased</i> value includes EPS, TIFF or PICT files with embedded ICC profiles. See [ICC.1]. If <i>IgnoreEmbeddedICC</i> is "true" then nominally <i>ICCBased</i> files or elements are to be treated as being encoded in the Alternate or underlying color space, and a ColorSpaceConversionOp where <i>SourceCS</i> = " <i>DevIndep</i> " is not to be applied unless that color space is also device independent. New in JDF 1.2
<i>ICCMYK</i>	defines an ICC based device independent representation of CMYK. New in JDF 1.3
<i>ICCGray</i>	defines an ICC based device independent representation of Gray. New in JDF 1.3
<i>ICCLAB</i>	defines an ICC based device independent representation of LAB. New in JDF 1.3
<i>ICCRGB</i>	defines an ICC based device independent representation of RGB. New in JDF 1.3
<i>Lab</i>	Operates on <i>Lab</i> . New in JDF 1.2
<i>RGB</i>	Operates on <i>DeviceRGB</i> . Modified in JDF 1.2
<i>Separation</i>	Operates on <i>Separation</i> color spaces (spot colors). The specific separation(s) to operate on are defined in the SeparationSpec element(s). If no SeparationSpec is defined, the operation will operate on all the separation color spaces in the input RunList . New in JDF 1.2
<i>YUV</i>	Operates on <i>YUV</i> (Also known as YCbCr). See [CCIR601-2] New in JDF 1.2

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 7-113: Mapping of SourceCS enumeration values to color spaces in the most common input file formats (Section 1 of 2)

SourceCS	File Format	Color Space
<i>RGB</i>	PDF (2)	DeviceRGB (1)
	PostScript	DeviceRGB
	TIFF	PhotometricInterp = 2
<i>CMYK</i>	PDF (2)	DeviceCMYK (1)
	PostScript (2)	DeviceCMYK
	TIFF	PhotometricInterp = 5 Samples per pixel = 4
<i>Gray</i>	PDF (2)	DeviceGray (1)
	PostScript (2)	DeviceGray
	TIFF	PhotometricInterp = 0 or 1

Table 7-113: Mapping of SourceCS enumeration values to color spaces in the most common input file formats (Section 2 of 2)

SourceCS	File Format	Color Space
<i>YUV</i>	PDF (2)	n/a
	PostScript (2)	n/a
	TIFF	PhotometricInterp = 6
<i>Calibrated</i>	PDF (2)	CalGray, CalRGB
	PostScript (2)	n/a
	TIFF	n/a
<i>CIEBased</i>	PDF (2)	n/a
	PostScript (2)	CIEBasedABC, CIEBasedA, CIEBasedDEF and CIEBasedDEFG
	TIFF	n/a
<i>LAB</i>	PDF (2)	LAB
	PostScript (2)	n/a
	TIFF	PhotometricInterp = 8 (CIELAB 1976 “normal” encoding) or PhotometricInterp = 9 (CIELAB 1976 using ICC profile v2 encoding).
<i>ICCBased</i> <i>ICCMYK</i> <i>ICCGray</i> <i>ICCLAB</i> <i>ICCRGB</i>	PDF (2)	ICCBased
	PostScript (2)	n/a
	PostScript/EPS	The EPS file has an embedded ICC profile.
	TIFF	The TIFF file has an embedded ICC profile.
<i>Separation</i>	PDF (2)	Separation
	PostScript (2)	Separation
	TIFF	PhotometricInterp = 5 (Applies only to one of the planes in the separated image.)
<i>DeviceN</i>	PDF (2)	DeviceN
	PostScript (2)	DeviceN
	TIFF	PhotometricInterp = 5, Samples per pixel = N

- (1) DeviceCMYK, DeviceRGB and DeviceGray in PDF files are to be mapped through DefaultCMYK, DefaultRGB or DefaultGray color spaces if present, before determining whether to apply this operation.
- (2) Where a *Pattern* or *Indexed* color space has been used, the base color space is used to determine whether to apply this operation.

Table 7-114: Effect of color space conversion operations on color spaces (Section 1 of 5)

SourceCS	Operation	Ignore Embedded ICC	FileSpec (Source-Profile)	Description
CMYK	Tag	false	CMYK ICC profile	Changes the CMYK color spaces (i.e., those without ICC profiles) in the RunList to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile.
		true	CMYK ICC profile	Changes the CMYK color spaces and all <i>ICCBased</i> color spaces with four components (CMYK) in the RunList to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile.
	Untag	n/a	n/a	n/a
	Convert	false	CMYK ICC profile	Converts the objects and/or images in CMYK color spaces (i.e., those without ICC profiles) using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile
		true	CMYK ICC profile	Converts the objects and/or images in CMYK color spaces and in four components (CMYK) <i>ICCBased</i> color spaces, using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.
RGB	Tag	false	RGB ICC profile	Changes the RGB color spaces (i.e., those without ICC profiles) in the RunList to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile.
		true	RGB ICC profile	Changes the RGB color spaces and all <i>ICCBased</i> color spaces with three components (RGB) in the RunList to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile.
	Untag	n/a	n/a	n/a
	Convert	false	RGB ICC profile	Converts the objects and/or images in RGB color spaces (i.e., those without ICC profiles) using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.
		true	RGB ICC profile	Converts the objects and/or images in RGB color spaces and in three components (RGB) <i>ICCBased</i> color spaces, using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.

Table 7-114: Effect of color space conversion operations on color spaces (Section 2 of 5)

SourceCS	Operation	Ignore Embedded ICC	FileSpec (Source-Profile)	Description
<i>Gray</i>	Tag	false	Mono-chrome ICC profile	Changes the <i>Gray</i> color spaces (i.e., those without ICC profiles) in the RunList to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile
		true	Mono-chrome ICC profile	Changes the <i>Gray</i> color spaces and all <i>ICCBased</i> color spaces with one component (Gray) in the RunList to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile.
	Untag	n/a	n/a	n/a
	Convert	false	Mono-chrome ICC profile	Converts the objects and/or images in Gray color spaces (i.e., those without ICC profiles) using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.
		true	Mono-chrome ICC profile	Converts the objects and/or images in <i>Gray</i> color spaces and in one component (Gray) <i>ICCBased</i> color spaces, using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.
<i>YUV, Lab</i>	Tag	n/a	Lab or YUV ICC profile	Changes the <i>YUV</i> or <i>Lab</i> color spaces in the RunList to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile. If <i>SourceProfile</i> is a <i>YUV</i> profile only <i>YUV</i> color spaces are affected; if <i>SourceProfile</i> is an <i>Lab</i> profile only <i>Lab</i> color spaces are affected.
	Untag	n/a	n/a	This operation does not have any effect.
	Convert	n/a	n/a	Converts the objects and/or images in the specified color spaces using the source definition embedded in the file and the <i>FinalTargetDevice</i> ICC profile as output profile.
<i>Calibrated</i>	Tag	n/a	RGB or Mono-chrome ICC profile	Changes the <i>Calibrated</i> color spaces in the RunList to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile. If <i>SourceProfile</i> is an <i>RGB</i> profile only <i>CalRGB</i> color spaces are affected; if <i>SourceProfile</i> is a monochrome profile only <i>CalGray</i> color spaces are affected.
	Untag	n/a	n/a	Changes <i>CalRGB</i> color spaces to RGB color space and <i>CalGray</i> color spaces to <i>Gray</i> color space.
	Convert	n/a	n/a	The corresponding objects in the specified color space(s) are converted using the source definition embedded in the file and the <i>FinalTargetDevice</i> ICC profile as output profile.

Table 7-114: Effect of color space conversion operations on color spaces (Section 3 of 5)

SourceCS	Operation	Ignore Embedded ICC	FileSpec (Source-Profile)	Description
<i>CIEBased</i>	Tag	n/a	n/a	This operation does not have any effect.
	Untag	n/a	n/a	This operation does not have any effect.
	Convert	n/a	n/a	The corresponding objects in the specified color space(s) are converted using the source definition embedded in the file and the <i>FinalTargetDevice</i> ICC profile as output profile.
<i>ICCBased</i> <i>ICCCMYK</i> <i>ICCGray</i> <i>ICCLAB</i> <i>ICCRGB</i>	Tag	n/a	n/a	n/a Note: In order to change the profile associated to an <i>ICCBased</i> , an <i>Untag</i> operation (see below) is to be performed before tagging. These two operations can be combined in a <i>Retag</i> operation.
	Untag	n/a	n/a	The ICC profiles in the input RunList are removed. The resulting color spaces depend on the input file format: <ul style="list-style-type: none"> PDF – Use the corresponding alternate color space. EPS – Use the PostScript file color spaces; the ICC profile comment in the EPS header is removed TIFF – Use the color space defined by the photometric interpretation tag.
	Convert	false	n/a	The <i>ICCBased</i> color spaces are converted using the corresponding embedded ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.
true		n/a	This operation does not have any effect (To ignore embedded ICC profiles when converting, the <i>CMYK</i> , <i>RGB</i> or <i>Gray</i> SourceCS enumeration values MUST be used with the <i>IgnoreEmbeddedICC</i> flag set to "true". Each <i>SourceCS</i> value will require a different ColorSpaceConversionOp instance, with the corresponding ICC profile.)	

Table 7-114: Effect of color space conversion operations on color spaces (Section 4 of 5)

SourceCS	Operation	Ignore Embedded ICC	FileSpec (Source-Profile)	Description	
<i>DevIndep</i>	Tag	n/a	n/a	This operation does not have any effect. The specific <i>SourceCS</i> enumeration values have to be used to select the color spaces to tag.	
	Untag	n/a	n/a	Untags <i>ICCBased</i> and <i>Calibrated</i> color spaces in the RunList . It does not have any effect on the other device independent color space.	
	Convert	false	n/a	Converts all the device independent color spaces (<i>CIEBased</i> , <i>Lab</i> , <i>YUV</i> , <i>Calibrated</i> and <i>ICCBased</i>) using the corresponding characterizations embedded in the file and the <i>FinalTargetDevice</i> ICC profile as output profile.	
		true	n/a	This operation does not have any effect. The specific <i>SourceCS</i> enumeration values have to be used to select the color spaces to convert.	
<i>Separation</i>	Tag	n/a	Named color ICC profile	In PostScript or PDF, it sets the alternate color space to an <i>ICCBased</i> color space with the given ICC profile.	
			No profile specified	In PostScript or PDF, it sets the alternate color space to the color definition in the ColorPool if present. If there is no color definition in the ColorPool , this operation does not have any effect.	
	Untag	n/a	n/a	This operation does not have any effect.	
	Convert	false	n/a		The specified separation(s) are converted using the alternate color space definitions in the RunList .
				Named color ICC profile	Converts the specified separation(s) using the <i>SourceProfile</i> profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.
				No profile specified	Converts the specified separation(s) using the color definition in the ColorPool and the <i>FinalTargetDevice</i> ICC profile if needed

Table 7-114: Effect of color space conversion operations on color spaces (Section 5 of 5)

SourceCS	Operation	Ignore Embedded ICC	FileSpec (Source-Profile)	Description
<i>DeviceN</i>	Tag	false	N component ICC profile	Changes the <i>DeviceN</i> color spaces in the RunList to <i>ICCBased</i> color spaces using the <i>SourceProfile</i> ICC profile. This operation only affects the selected <i>DeviceN</i> color spaces that have exactly the same number of components than the <i>SourceProfile</i> .
	Untag	n/a	n/a	This operation does not have any effect.
	Convert	false	n/a	In PostScript or PDF, the specified <i>DeviceN</i> color spaces are converted using the alternate color space.
		true	N component ICC profile	Converts the specified <i>DeviceN</i> color spaces using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile. This operation only affects the selected <i>DeviceN</i> color spaces that have exactly the same number of components than the <i>SourceProfile</i> .

Note: If the correct ICC profile is not specified for an operation that requires it, the operation does not have any effect.

7.2.35 ComChannel

A communication channel to a person or company such as an email address, phone number or fax number.

Resource Properties

Resource class:	Parameter
Resource referenced by:	Contact, Person, CustomerMessage
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-115: ComChannel resource (Section 1 of 2)

Name	Data Type	Description
<i>ChannelType</i> Modified in JDF 1.2	enumeration	Type of the communication channel. Possible values are: <i>Phone</i> – Telephone number. <i>Email</i> – Email address. <i>Fax</i> – Fax machine. <i>WWW</i> – WWW home page or form. <i>JMF</i> – JMF messaging channel. <i>PrivateDirectory</i> – Account of a registered customer of a certain service. (The list of the registered accounts is maintained by the service vendor). The <i>ChannelTypeDetails</i> attribute has the name of the private directory service vendor. <i>InstantMessaging</i> – IM service address. The <i>ChannelTypeDetails</i> attribute has the name of the IM service vendor

Table 7-115: ComChannel resource (Section 2 of 2)

Name	Data Type	Description
ChannelTypeDetails ? New in JDF 1.2	NMTOKEN	Description of the value of the <i>ChannelType</i> attribute. See Table 7-116 on page 380 for examples. Consumer treats this value as the service vendor name if <i>ChannelType</i> is <i>PrivateDirectory</i> or <i>InstantMessaging</i> .
ChannelUsage ? New in JDF 1.2	NMTOKENS	Communication channel usage. Possible values include: <i>Business</i> – Business purpose usage, (e.g., office phone number, fax). <i>Private</i> – Private purpose usage, (e.g., private phone number, fax, Email). <i>DayTime</i> – Office hours in the time zone of the recipient. <i>NightTime</i> – Out-of-office hours in the time zone of the recipient. <i>WeekEnd</i> – 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 <i>ChannelType</i> , 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: “mailto:a@b.com” instead of “a@b.com” if <i>ChannelType</i> = “Email”, “tel:+49-69-92058800” if <i>ChannelType</i> = “Phone” and “tel:+49.6151.155.299” if <i>ChannelType</i> = “Fax”.

— Attribute: ChannelTypeDetails**Table 7-116: ChannelTypeDetails attribute – predefined values for certain ChannelType values**

ChannelType value	ChannelTypeDetails value	Description
<i>Phone</i>	<i>LandLine</i>	Land line telephone number.
	<i>Mobile</i>	Mobile/Cellular telephone number.
	<i>Secure</i>	Secure phone line.
	<i>ISDN</i>	ISDN line telephone number.
<i>WWW</i>	<i>Form</i>	Upload form.
	<i>Target</i>	Upload target URL.

Examples

```
<ComChannel ChannelType="Phone" ChannelTypeDetails="Mobile" ChannelUsage="Business"
  Locator="44 07808 907 919"/>
```

```
<ComChannel ChannelType="InstantMessaging" ChannelTypeDetails="MyIMService"
  ChannelUsage="Private" Locator="123456789"/>
```

7.2.36 Company

Specifies contacts to a company including detailed information about contact persons and addresses. This structure can be used in many situations where addresses or contact persons are needed. Examples of contacts are customer, supplier, company and addressees. The structure is derived from the vCard format. It comprises the organization name and organizational units (ORG) of the organizational properties defined in the vCard format. The corresponding XML types of the vCard are quoted in the table.

Resource Properties

Resource class:	Parameter
Resource referenced by:	Contact
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-117: Company resource

Name	Data Type	Description
<i>OrganizationName</i>	string	Name of the organization or company (vCard: ORG:orgnam. For example: ABC, Inc.).
Contact * Deprecated in JDF 1.1	refelement	A contact of the company. In JDF 1.1 and beyond, Contacts reference multiple Companies.
<i>OrganizationalUnit</i> *	telem	Describes the organizational unit (vCard: ORG:orgunit. For example, if two elements are present: 1. “North American Division” and 2. “Marketing”).

7.2.37 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. **Component** resources are grouped by kind in much the same way that nodes are classified as Combined, Process or Product. The five categories of **Component** resources are: *Ribbon*, *Sheet*, *Block*, *PartialProduct* and *FinalProduct*. These categories are defined in greater detail below:

- *Ribbon* – Part of the web that enters the folder, divider etc. In case the web is not slit, the web and the ribbon are identical.
- *Sheet* – This source type is appropriate if a flat sheet, (e.g., a postcard to be glued in) is used as an input component. "Flat" in this case means that the sheet has not been folded or cut before the operation.
- *Block* – This source type is appropriate if a folded sheet, a cut portion of the sheet, or a cut and folded portion of a sheet is used as an input component.
- *PartialProduct* – This source type is appropriate if a partial product is to be used as an input **Component**.
- *FinalProduct* – This source type is appropriate if this **Component** is the final product.

Component – Terms And Definitions

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).

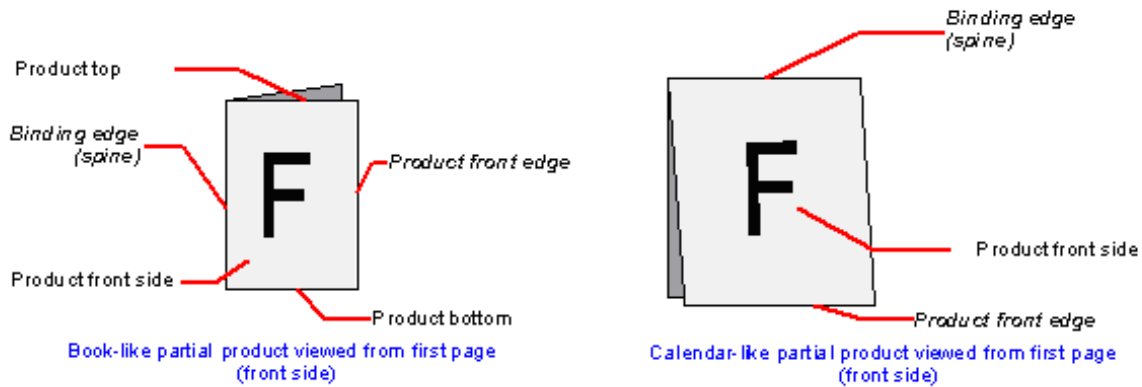


Figure 7-14: Component – terms and definitions

The table below describes the terms used to define the components.

Table 7-118: Component – terms and definitions

Edge	Description
Binding edge	The edge on which the (partial) product is glued or stitched. This edge is also often called <i>working edge</i> or <i>spine</i> .
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 MUST 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:** —
- Example Partition:** *Condition, RibbonName, SheetName, SignatureName, WebName*
- Input of processes:** Many
- Output of processes:** Many

Table 7-119: Component resource (Section 1 of 4)

Name	Data Type	Description
<i>AssemblyIDs ?</i> New in JDF 1.3	NMTOKENS	<i>AssemblyIDs</i> of the Assembly , AssemblySection or StrippingParams (BinderySignatureName) which this Component carries.
<i>CartonTopFlaps ?</i> New in JDF 1.3	XYPair	Size (F1,F2) (See Figure "Box packing" on page 343) of the two top flaps of a carton for shipping. MUST NOT be specified unless <i>ProductType</i> = "Carton".

Table 7-119: Component resource (Section 2 of 4)

Name	Data Type	Description
<u>ComponentType</u> <u>Modified in JDF 1.3</u>	enumerations	Specifies the category of the component. Possible values are: <i>Block</i> – Folded or stacked product, (e.g., book block). <i>Other</i> – 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. <u>New in JDF 1.3</u> <i>Ribbon</i> – The Component is a ribbon on a web press. <i>Sheet</i> – Single layer (sheet) of paper. <i>Web</i> – The Component is a web on a web press. Further details of the component are specified in <i>ProductType</i> . At most one of <i>FinalProduct</i> , <i>PartialProduct</i> or <i>Proof</i> MUST be specified in addition to one of the five enumerations specified above. <i>FinalProduct</i> – The Component is the final product that was ordered by the customer. <i>PartialProduct</i> – The Component is an intermediate product that will be input to a following process. <i>Proof</i> – 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. <u>Modified in JDF 1.3</u>
<i>Dimensions ?</i>	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 <i>Dimensions</i> unless they really are unknown.
<i>IsWaste = "false"</i>	boolean	If " <i>true</i> ", the Component is waste from a previous process that can be used to set up a machine.
<i>MaxHeat ?</i>	double	Maximum temperature the Component can resist (in degrees centigrade). The default setting is to impose no restriction in terms of heat, (e.g., fusers in electrophotographic process or shrink wrapping).
<u>Overfold ?</u> <u>New in JDF 1.1</u>	double	Expansion of the overfold of a Component . This attribute is needed for the Inserting or other postpress processes.
<u>OverfoldSide ?</u> <u>New in JDF 1.1</u>	enumeration	Specifies the longer side of a folded component. Values are: <i>Front</i> <i>Back</i>
<u>PageListIndex ?</u> <u>New in JDF 1.3</u>	IntegerRange-List	List of the indices of the PageData elements of the PageList specified in this Component .

Table 7-119: Component resource (Section 3 of 4)

Name	Data Type	Description
ProductType ? Modified in JDF 1.3	NMTOKEN	Type of product that this component specifies. Possible values include: <i>BackCover</i> <i>BlankBox</i> : Cut, Unfolded box, input for folder-gluer New in JDF 1.3 <i>Body</i> – Generic content inside of a cover. New in JDF 1.2 <i>Book</i> <i>BookBlock</i> <i>BookCase</i> <i>Box</i> – Convenience packaging that is not envisioned to be protection for shipping. <i>Brochure</i> <i>BusinessCard</i> <i>Carton</i> – Protection packaging for shipping. <i>Cover</i> <i>FlatBox</i> – a folded and glued blank (not opened). Output from a box folder-gluer. New in JDF 1.3 <i>FrontCover</i> <i>Insert</i> – New in JDF 1.2 <i>Jacket</i> – Hard cover case jacket. <i>Label</i> <i>Pallet</i> – Loaded pallet of Boxes, Cartons or Component resources New in JDF 1.3 <i>Poster</i> <i>Newspaper</i> – A newspaper-product New in JDF 1.3 <i>Unknown</i> – Deprecated in JDF 1.2
ProductTypeDetails ? New in JDF 1.3	string	Additional details of the product: If <i>ProductType</i> = “ <i>BlankBox</i> ” or <i>ProductType</i> = “ <i>FlatBox</i> ”, <i>ProductTypeDetails</i> specifies a box type: e.g. [ECMA], [FEFCO] or company internal box type standard. <i>NewspaperNormal</i> – Standard newspaper. <i>NewspaperMixed</i> – multiple Component resources of a newspaper are produced in parallel. <i>NewspaperCombi</i> – 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 (<i>@ComponentType</i> , “ <i>Block</i> ”) and Layout is present. Position of the block on the Layout in <i>SurfaceContentsBox</i> coordinates used in this Component .
SourceRibbon ? Deprecated in JDF 1.3	string	MUST NOT be specified unless contains (<i>@ComponentType</i> , “ <i>Ribbon</i> ”). <i>RibbonName</i> of the ribbon used in this Component . In JDF 1.3 and beyond, replaced with a direct reference to the Layout partition that represents the ribbon.

Table 7-119: Component resource (Section 4 of 4)

Name	Data Type	Description
SourceSheet ? Deprecated in JDF 1.3	string	MUST NOT be specified unless contains (<i>@ComponentType</i> , "Sheet") or contains (<i>@ComponentType</i> , "Block"). Matches the Layout/Signature/Sheet/@Name used in this Component . In JDF 1.3 and beyond, replaced with a direct reference to the Layout partition that represents the sheet.
SourceWeb ? Deprecated in JDF 1.3	string	MUST NOT be specified unless contains (<i>@ComponentType</i> , "Ribbon"). WebName of the ribbon used in this Component . In JDF 1.3 and beyond, replaced with a direct reference to the Layout partition that represents the web.
SurfaceCount ? New in JDF 1.1	integer	Total amount of individual surfaces that this Component contains.
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 <i>ResourceLink/@Transformation</i> or <i>ResourceLink/@Orientation</i> .
Assembly ? New in JDF 1.3	refelement	Specifies the assembly of the Component . In case of a newspaper-web fed, 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.
Sheet ? Deprecated in JDF 1.2	refelement	The sheet resource that describes the details of this Component if it contains (<i>@ComponentType</i> , "Sheet") or contains (<i>@ComponentType</i> , "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
Layout ? New in JDF 1.2	refelement	Specifies the original Layout of the source sheet of the Component if it contains (<i>@ComponentType</i> , "Sheet") or contains (<i>@ComponentType</i> , "Block"). The original sheet is the Layout partition element where SourceSheet matches the Layout/@SheetName used in this Component
PageList ? New in JDF 1.3	refelement	Specification of page metadata for pages described by this Component .

Example of Condition Partition Key Attribute

[New in JDF 1.2](#)

The *Condition* partition key describes whether a **Component** partition is waste and what type of waste it is. The amount of each *Condition*, in the standard manner for partitionable resources. See Section 3.9.5, Description of Partitioned Resources.)

```
<ResourcePool>
```

```
  <Component Class="Quantity" ID="Res6" PartIDKeys="Condition" Status="Available">
```

```

    <Component Condition="Good" IsWaste="false"/>
    <Component Condition="Waste" IsWaste="true"/>
  </Component>
</ResourcePool>
<ResourceLinkPool>
  <ComponentLink Usage="Output" rRef="Res6">
    <AmountPool>
      <PartAmount Amount="2970">
        <Part Condition="Good"/>
      </PartAmount>
      <PartAmount Amount="87">
        <Part Condition="Waste"/>
      </PartAmount>
    </AmountPool>
  </ComponentLink>
</ResourceLinkPool>

```

7.2.38 Contact

Element describing a contact to a person or address.

Resource Properties

Resource class:	Parameter
Resource referenced by:	ApprovalParams, ArtDeliveryIntent, CustomerInfo, DeliveryIntent, DeliveryParams, DropIntent
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-120: Contact resource (Section 1 of 2)

Name	Data Type	Description
ContactTypes Modified in JDF 1.2	NMTOKENS	Classification of the contact. Possible values include: <i>Accounting</i> – Address of where to send to the bill. <i>Administrator</i> – Person to contact for queries concerning the execution of the job. <i>Approver</i> – Person who approves this job. New in JDF 1.2 <i>ArtReturn</i> – Return delivery or pickup address for artwork of this job. <i>Billing</i> – Contact information that refers to a payment method, (e.g., credit card). <i>Customer</i> – The end customer. <i>Delivery</i> – The delivery address for all products of this job. <i>DeliveryCharge</i> – The Contact is charged for delivery of this job. <i>Owner</i> – The owner of a resource. <i>Pickup</i> – The pickup address for all products of this job. <i>Sender</i> – The source address of the delivery. New in JDF 1.2 <i>Supplier</i> – Address of a supplier of needed goods. <i>SurplusReturn</i> – Return delivery or pickup address for surplus products of this job.

Table 7-120: Contact resource (Section 2 of 2)

Name	Data Type	Description
<i>ContactTypeDetails</i> ? New in JDF 1.2	string	Details of the Contact's role or roles. For instance, if contains (<i>@ContactTypes</i> , "Delivery") this could be a description for which delivery location this Contact is responsible.
Address ?	refelement	Element describing the address.
ComChannel * Modified in JDF 1.2	refelement	Communication channels to 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 ?	refelement	Name of the contact person.

7.2.39 ContactCopyParams

[New in JDF 1.1](#)

Element describing the parameters of *ContactCopying*.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	ContactCopying
Output of processes:	—

Table 7-121: ContactCopyParams resource

Name	Data Type	Description
<i>ContactScreen</i> = "false"	boolean	<i>ContactScreen</i> = "true" if a halftone screen on film is to be used to produce halftones.
<i>Cycle</i> ?	integer	Number of exposure light units to be used. The amount depends on the subject to be exposed.
<i>Diffusion</i> ?	enumeration	The diffusion foil setting. Possible values are: <i>On</i> <i>Off</i>
<i>PolarityChange</i> = "true"	boolean	<i>PolarityChange</i> = "true" if the copy is to change polarity with respect to the original image.
<i>RepeatStep</i> = "1 1"	XYPair	Number (as integers) of copies in each direction for a Step/Repeat camera.
<i>Vacuum</i> ?	double	Amount of vacuum pressure to be used, measured in bars.
ScreeningParams ?	refelement	Properties of the halftone screen on film. Ignored if <i>ContactScreen</i> = "false".

7.2.40 ContentList

[New in JDF 1.3](#)

ContentList provides a list of ContentData elements.

Resource Properties

Resource class:	Parameter
Resource referenced by:	PageList
Example Partition:	—

Input of processes: —

Output of processes: —

Table 7-122: ContentList resource

Name	Data Type	Description
ContentData +	element	Details of the individual content element. The ContentData elements are referred to by their index in the ContentList . Therefore, ContentData elements SHOULD NOT be removed or inserted in a position other than the end of the list.

— Element: ContentData

ContentData defines the additional metadata of individual elements of 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 7-123: ContentData element

Name	Data Type	Description
<i>CatalogID</i> ?	string	Identification of the resource, (e.g., in a catalog environment).
<i>CatalogDetails</i> ?	string	Additional details of a resource in a catalog environment.
<i>ContentType</i> ?	NMTOKEN	Type of content. See Table 7-123 for values.
<i>HasBleeds</i> ?	boolean	If "true", the file has bleeds.
<i>IsBlank</i> ?	boolean	If "true", the ContentData has no content marks and is blank.
<i>IsTrapped</i> ?	boolean	If "true", the file has been trapped.
<i>JobID</i> ?	string	ID of the job that this ContentData belongs to.
<i>ProductID</i> ?	string	An ID of the ContentData as defined in the MIS system.
ElementColorParams ?	refelement	Color details of the ContentData element.
ImageCompressionParams ?	refelement	Specification of the image compression properties.
ScreeningParams ?	refelement	Specification of the screening properties.
SeparationSpec *	element	List of separation names defined in the element.

— Attribute: ContentType**Table 7-124: ContentType attribute – possible values (Section 1 of 2)**

Value	Description
<i>Ad</i>	The content represents a single ad
<i>Article</i>	The content represents a single article. Including headers, text bodies, photos etc.
<i>Barcode</i>	A barcode.
<i>ClassifiedAd</i>	Specifies a classified ad.
<i>ClassifiedsPageElement</i>	Specifies a grouping page element dealing with content of classified ads
<i>Composed</i>	Combination of elements that define an element that is not bound to a document page.
<i>Editorial</i>	Defines this element to contain editorial matter (e.g., text, photos etc.).

Table 7-124: ContentType attribute – possible values (Section 2 of 2)

Value	Description
<i>EditorialPageElement</i>	Specifies a grouping page element dealing with content of the editorial department
<i>Graphic</i>	Line art.
<i>IdentificationField</i>	A general identification field excluding bar codes.
<i>Image</i>	Bitmap image.
<i>Page</i>	Representation of one document page.
<i>PageHeader</i>	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.
<i>ROPAd</i>	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.
<i>Surface</i>	Representation of an imposed surface.
<i>Text</i>	Formatted or unformatted text.

7.2.41 ConventionalPrintingParams

This resource defines the attributes and elements of the **ConventionalPrinting** process. The specific parameters of individual printer modules are modeled by using the standard partitioning methods. These methods are described in Section 3.9.5, Description of Partitioned Resources.

Resource Properties

Resource class: Parameter

Resource referenced by: —

Example Partition: *BlockName, FountainNumber, PartVersion, ProductionRun, RibbonName, Separation, SheetName, Side, SignatureName, WebName, WebProduct*

Input of processes: **ConventionalPrinting**

Output of processes: —

Table 7-125: ConventionalPrintingParams resource (Section 1 of 3)

Name	Data Type	Description
<i>DirectProof</i> = "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 a <i>DirectProof</i> , the <i>Status</i> of the JDF node is switched to <i>Stopped</i> with the <i>StatusDetails</i> = "WaitForApproval".
<i>Drying ?</i>	enumeration	The way in which ink is dried after a print run. Possible values are: <i>UV</i> – Ultraviolet dryer <i>Heatset</i> – Heatset dryer <i>IR</i> – Infrared dryer <i>On</i> – Use the device default drying unit. <i>Off</i>

Table 7-125: ConventionalPrintingParams resource (Section 2 of 3)

Name	Data Type	Description
<u>FirstSurface ?</u> <u>Modified in JDF 1.2</u>	enumeration	Printing order of the surfaces on the sheet. Possible values are: <i>Either- <u>Deprecated in JDF 1.2</u> Omit <i>FirstSurface</i> to specify either.</i> <i>Front</i> <i>Back</i>
<u>FountainSolution ?</u>	enumeration	State of the fountain solution module in the printing units. Possible values are: <i>On</i> <i>Off</i>
<u>MediaLocation ?</u>	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 <i>Location</i> (see also Section 3.9.6.2, Locations of Physical Resources) there SHOULD be a match between one <i>Location</i> partition key and this <i>MediaLocation</i> value.
<u>ModuleAvailableIndex ?</u> <u>New in JDF 1.1</u>	Integer-RangeList	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 <i>ModuleIndex</i> .
<u>ModuleDrying ?</u>	enumeration	The way in which ink is dried in individual modules. Possible values are: <i>UV</i> – Ultraviolet dryer <i>Heatset</i> – Heatset dryer <i>IR</i> – Infrared dryer <i>On</i> – Use the device default drying unit. <i>Off</i>
<u>ModuleIndex ?</u>	Integer-RangeList	Zero-based, ordered list of print modules that are to be used. <i>ModuleIndex</i> 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 <i>ModuleAvailableIndex</i> .
<u>NonPrintableMarginBottom ?</u> <u>New in JDF 1.3</u>	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 <i>NonPrintableMarginBottom</i> .
<u>NonPrintableMarginLeft ?</u> <u>New in JDF 1.3</u>	double	Same as <i>NonPrintableMarginBottom</i> except for the left margin.
<u>NonPrintableMarginRight ?</u> <u>New in JDF 1.3</u>	double	Same as <i>NonPrintableMarginBottom</i> except for the right margin.
<u>NonPrintableMarginTop ?</u> <u>New in JDF 1.3</u>	double	Same as <i>NonPrintableMarginBottom</i> except for the top margin.
<u>PerfectingModule ?</u> <u>New in JDF 1.1</u>	integer	Index of the perfecting module if <i>WorkStyle</i> = <i>Perfecting</i> and multiple perfecting modules are installed.

Table 7-125: ConventionalPrintingParams resource (Section 3 of 3)

Name	Data Type	Description
<i>Powder?</i>	double	Quantity of powder in%.
<i>PrintingType</i> Modified in JDF 1.3	enumeration	Type of printing machine. Possible values are: <i>ContinuousFed</i> – connected sheets including fan fold. New in JDF 1.2 <i>SheetFed</i> – Separate cut sheets. <i>WebFed</i> – Paper supplied to press on rolls. Deprecated in JDF 1.3 <i>WebMultiple</i> – Web Printing with multiple plates per cylinder. Generally used with Newspaper web printing. New in JDF 1.3 <i>WebSingle</i> – Web Printing with only one plate per cylinder. Generally used in commercial and publication workflows. New in JDF 1.3
<i>SheetLay?</i>	enumeration	Lay of input media. Reference edge of where paper is placed in a feeder. Possible values are: <i>Left</i> <i>Right</i> <i>Center</i>
<i>Speed?</i> Modified in JDF 1.3	double	Maximum print speed in sheets/hour (sheet fed) or revolutions/hour (web fed). Defaults to device specific full speed.
<i>WorkStyle?</i>	enumeration	The direction in which to turn the press sheet. See Table 7-126 for values.
<i>ApprovalParams?</i> New in JDF 1.2	refelement	Details of the direct approval process, when <i>DirectProof</i> = "true".
<i>Ink*</i> Modified in JDF 1.2	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 MUST use the partition mechanism of this parameter 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 MUST NOT be specified here.

— Attribute: WorkStyle

Table 7-126: WorkStyle attribute – possible values (Section 1 of 2)

Value	Description
<i>Perfecting</i>	Many sheetfed 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 <i>WorkStyle</i> is similar to <i>WorkAndTumble</i> , but <i>Perfecting</i> is an in-line operation during the press run. Therefore, an additional plate (set) is needed during this press run.

Table 7-126: WorkStyle attribute – possible values (Section 2 of 2)

Value	Description
<i>Simplex</i>	No turning of the sheets.
<i>WorkAndBack</i>	This <i>WorkStyle</i> 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 <i>WorkAndBack</i> .
<i>WorkAndTumble</i>	The <i>WorkAndTumble</i> 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) is used again in the second press run, imaging the other sheet surface.
<i>WorkAndTurn</i>	Refers to the turning of the first-run sheet for subsequent perfecting. The front lays remain unchanged but the side lays are 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.
<i>WorkAndTwist</i>	Done between two press runs. The pallet is 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 <i>WorkStyle</i> is used for saving plate or film material. It is no longer a common <i>WorkStyle</i> .

7.2.42 CostCenter

This resource describes an individual area of a company that has separated accounting.

Resource Properties

Resource class:	ResourceElement
Resource referenced by:	Device, Employee
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-127: CostCenter resource

Name	Data Type	Description
<i>CostCenterID</i>	string	Identification of the cost center
<i>Name ?</i>	string	Name of the cost center.

7.2.43 CoverApplicationParams

[New in JDF 1.1](#)

CoverApplicationParams define the parameters for applying a cover to a book block.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	CoverApplication
Output of processes:	—

Table 7-128: CoverApplicationParams resource

Name	Data Type	Description
<i>CoverOffset</i> ? 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, <i>CoverOffset</i> is implied by the transformation matrix of the ResourceLink/ <i>@Transformation</i> 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.

— Element: Score

Table 7-129: Score element

Name	Data Type	Description
<i>Offset</i>	double	Position of scoring given in the operation coordinate system.
<i>Side</i> = "FromInside"	enumeration	Specifies the side from which the scoring tool works. Possible values are: <i>FromInside</i> <i>FromOutside</i>

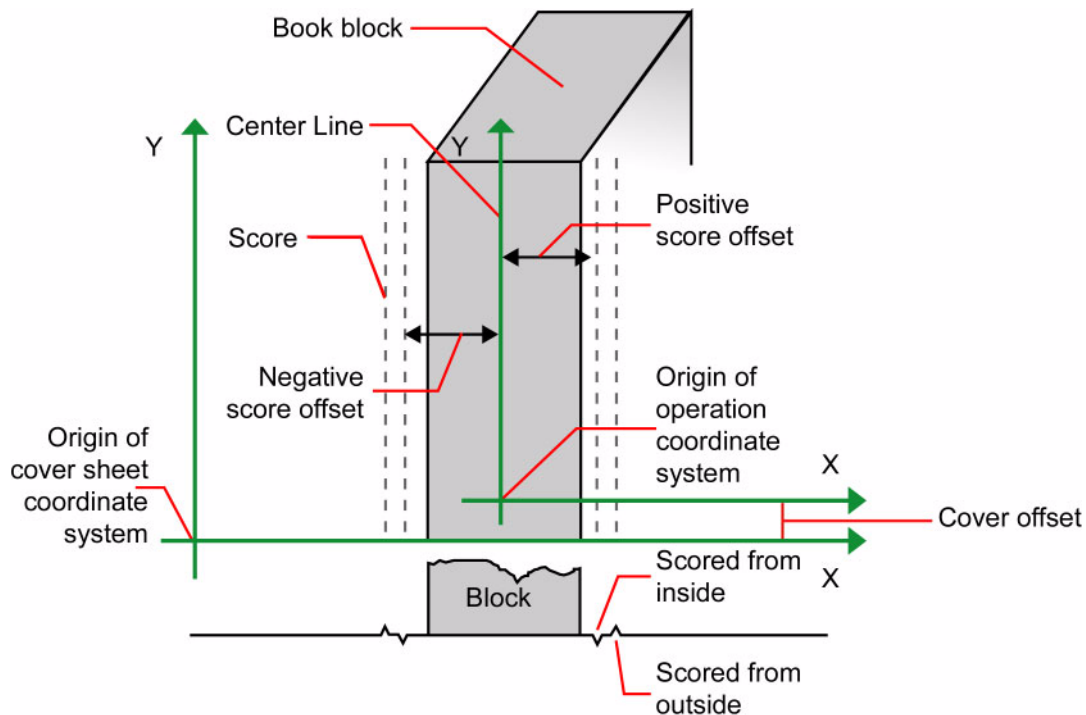


Figure 7-15: Parameters and coordinate system for cover application

7.2.44 CreasingParams

[New in JDF 1.1](#)

CreasingParams define the parameters for creasing or grooving a sheet.

Resource Properties

Resource class: Parameter

Resource referenced by: —

Example Partition: *BlockName, RibbonName, SheetName, SignatureName, WebName*Input of processes: **Creasing**

Output of processes: —

Table 7-130: CreasingParams resource

Name	Data Type	Description
Crease *	element	Defines one or more Crease lines.

— Element: CreaseCrease defines an individual crease line on a **Component**.**Table 7-131: Crease element**

Name	Data Type	Description
<i>Depth</i> ? New in JDF 1.2	double	Depth of the Crease, measured in microns [µm].
<i>RelativeTravel</i> ? New in JDF 1.2	double	Relative distance of the reference edge relative to <i>From</i> in the coordinates of the incoming Component . The <i>RelativeTravel</i> 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.
<i>RelativeStartPosition</i> ? New in JDF 1.2	XYPair	Relative starting position of the tool. The <i>RelativeStartPosition</i> 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 .
<i>RelativeWorkingPath</i> ? New in JDF 1.2	XYPair	Relative working path of the tool beginning at <i>RelativeStartPosition</i> . Since the tools can only work parallel to the edges, one coordinate MUST be zero. The <i>RelativeWorkingPath</i> 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 .
<i>StartPosition</i> ? Modified in JDF 1.2	XYPair	Starting position of the tool. If both <i>StartPosition</i> and <i>RelativeStartPosition</i> are specified, <i>RelativeStartPosition</i> is ignored. At least one of <i>StartPosition</i> or <i>RelativeStartPosition</i> MUST be specified.
<i>Travel</i> ? New in JDF 1.2	double	Distance of the reference edge relative to <i>From</i> . If both <i>Travel</i> and <i>RelativeTravel</i> are specified, <i>RelativeTravel</i> is ignored. At least one of <i>Travel</i> or <i>RelativeTravel</i> MUST be specified.
<i>WorkingPath</i> ?	XYPair	Working path of the tool beginning at <i>StartPosition</i> . Since the tools can only work parallel to the edges, one coordinate MUST be zero. If both <i>WorkingPath</i> and <i>RelativeWorkingPath</i> are specified, <i>RelativeWorkingPath</i> is ignored. At least one of <i>WorkingPath</i> or <i>RelativeWorkingPath</i> MUST be specified.
<i>WorkingDirection</i>	enumeration	Direction from which the tool is working. Possible values are: <i>Top</i> – From above. <i>Bottom</i> – From below.

7.2.45 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.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>all</i>
Output of processes:	—



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?

Table 7-132: CustomerInfo resource

Name	Data Type	Description
<i>BillingCode</i> ?	string	A code to bill charges incurred while executing the node.
<i>CustomerID</i> ?	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.
<i>CustomerJobName</i> ?	string	The name that the customer uses to refer to the job.
<i>CustomerOrderID</i> ?	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.
<i>CustomerProjectID</i> ? New in JDF 1.2	string	The internal project id in the system of the customer. This number might be provided when the order is placed and then referenced on the order confirmation or the bill.
<i>rRefs</i> ? Deprecated in JDF 1.2	IDREFS	Array of <i>IDs</i> 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 Contacts 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 , " <i>Customer</i> ")]. Such a Contact specifies the primary customer's name, address etc.
<i>CustomerMessage</i> * New in JDF 1.2	element	Element that describes messages to the customer.

— Element: CustomerMessage

[New in JDF 1.2](#)

CustomerMessage is an abstract definition of messages to the customer. Formatting and details of the content generation of the message are system dependent.

Table 7-133: CustomerMessage element

Name	Data Type	Description
ComChannel *	refelement	Communication channel for the desired CustomerMessage. In case it is not specified, the CustomerMessage will be provided according to system predefined information. If multiple ComChannel elements are specified, the CustomerMessage SHOULD be sent to all specified communication channels.
<i>Language</i> ?	language	Language to be used for the CustomerMessage.
<i>MessageEvents</i>	NMTOKENS	Defines the set of events that trigger a message that is defined or specified by the system. A list of predefined values is provided in "MessageEvents and Milestone Values" on page 712.
<i>ShowList</i> ?	NMTOKENS	List of parameters to display in the CustomerMessage. Values include: <i>Amount</i> – Amount of the product that was produced. <i>DeviceID</i> – <i>ID</i> of the device. This is a unique name within the workflow. <i>EndTime</i> – Actual <i>EndTime</i> of the job. <i>Error</i> – Errors that happened during the job. <i>FriendlyName</i> – The <i>FriendlyName</i> of the device. <i>JobName</i> – <i>DescriptiveName</i> of the node that is executing. <i>JobRecipientName</i> – Name of the recipient of the job. <i>JobSubmitterName</i> – Name of the submitter of the job. <i>StartTime</i> – Actual <i>StartTime</i> of the job. <i>MediaBrand</i> – <i>Brand</i> of the media that is being printed. <i>MediaType</i> – <i>DescriptiveName</i> of the media that is being printed. <i>Operator</i> – Name of the Operator. <i>Resolution</i> – Output resolution. <i>ResolutionX</i> – Output resolution in X direction. <i>ResolutionY</i> – Output resolution in Y direction. <i>ScreeningFamily</i> – Name of the screening family of the output. <i>UserText</i> – User defined text in a Comment [<i>@Name</i> = " <i>UserText</i> ".] <i>Warning</i> – Warnings that happened during the job.

7.2.46 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
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-134: CutBlock resource

Name	Data Type	Description
AssemblyIDs ? New in JDF 1.3	NMTOKENS	The <i>AssemblyIDs</i> of the Assembly , AssemblySection or StrippingParams [@ <i>BinderySignatureName</i>] which are contained in this CutBlock .
<i>BlockElementSize</i> ?	XYPair	Element dimension in X and Y direction. The default value is equivalent to the XYPair value in <i>BlockSize</i> .
<i>BlockElementType</i> ?	enumeration	Element type. Possible values are: <i>CutElement</i> – Cutting element. <i>PunchElement</i> – Punching element.
<i>BlockName</i>	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.
<i>BlockSize</i>	XYPair	Size of the block.
<i>BlockSubdivision</i> = "1 1"	XYPair	Number (as integers) of elements in X and Y direction.
<i>BlockTrf</i> = "1 0 0 1 0 0"	matrix	Block transformation matrix. Defines the position and orientation of the block relative to the Component coordinate system.
<i>BlockType</i>	enumeration	Block type. Possible values are: <i>CutBlock</i> – Block to be cut. <i>SaveBlock</i> – Protected block, cut only via outer contour. <i>TempBlock</i> – Auxiliary block that is not taken into account during cutting. <i>MarkBlock</i> – Contains no elements, only marks.
Assembly ? New in JDF 1.3	refelement	Assembly that is referred to by <i>AssemblyIDs</i> or contains the AssemblySection that is referred to by <i>AssemblyIDs</i> .

7.2.47 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
Example Partition:	—
Input of processes:	—
Output of processes:	—










Table 7-135: CutMark resource (Section 1 of 2)

Name	Data Type	Description
Blocks ? Modified in JDF 1.1	NMTOKENS	Values of the <i>BlockName</i> partition attributes of the blocks defined by the CutMark resource.

Table 7-135: CutMark resource (Section 2 of 2)

Name	Data Type	Description
<i>MarkType</i>	enumeration	Cut mark type. Possible values are: <i>CrossCutMark</i> <i>TopVerticalCutMark</i> <i>BottomVerticalCutMark</i> <i>LeftHorizontalCutMark</i> <i>RightHorizontalCutMark</i> <i>LowerLeftCutMark</i> <i>UpperLeftCutMark</i> <i>LowerRightCutMark</i> <i>UpperRightCutMark</i>
<i>Position</i>	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 7-136: Cut mark types as specified by CutMark/@MarkType

Symbol	Name	Position of Symbol
	<i>CrossCutMark</i>	Centered at logical position
	<i>TopVerticalCutMark</i>	Slightly above logical position
	<i>BottomVerticalCutMark</i>	Slightly below logical position
	<i>LeftHorizontalCutMark</i>	Slightly to the left of logical position
	<i>RighHorizontalCutMark</i>	Slightly to the right of logical position
	<i>LowerLeftCutMark</i>	Corner at logical position
	<i>UpperLeftCutMark</i>	Corner at logical position
	<i>LowerRightCutMark</i>	Corner at logical position
	<i>UpperRightCutMark</i>	Corner at logical position

7.2.48 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
Resource referenced by:	—
Example Partition:	<i>BlockName, RibbonName, SheetName, SignatureName, WebName</i>
Input of processes:	Cutting
Output of processes:	—

Table 7-137: CuttingParams resource

Name	Data Type	Description
CutBlock *	refelement	One or several CutBlock elements can be used to find the Cutting sequence. The CutBlock elements MUST NOT be written if Cuts 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.
Cut *	element	Cut elements describe an individual cut. Cuts MUST NOT be specified if CutBlock elements are specified.

— Element: Cut

Cut describes one straight cut with an arbitrary tool.

Table 7-138: Cut element (Section 1 of 2)

Name	Data Type	Description
<i>RelativeStartPosition</i> ? New in JDF 1.2	XYPair	Relative starting position of the tool. The <i>RelativeStartPosition</i> 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 .
<i>RelativeWorkingPath</i> ? New in JDF 1.2	XYPair	Relative working path of the tool beginning at <i>RelativeStartPosition</i> . Since the tools can only work parallel to the edges, one coordinate MUST be zero. <i>RelativeWorkingPath</i> 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 .
<i>StartPosition</i> ? Modified in JDF 1.2	XYPair	Starting position of the tool. If both <i>StartPosition</i> and <i>RelativeStartPosition</i> are specified, <i>RelativeStartPosition</i> is ignored. At least one of <i>StartPosition</i> or <i>RelativeStartPosition</i> MUST be specified.
<i>WorkingPath</i> ? Modified in JDF 1.2	XYPair	Working path of the tool beginning at <i>StartPosition</i> . Since the tools can only work parallel to the edges, one coordinate MUST be zero. If both <i>WorkingPath</i> and <i>RelativeWorkingPath</i> are specified, <i>RelativeWorkingPath</i> is ignored. At least one of <i>WorkingPath</i> or <i>RelativeWorkingPath</i> MUST be specified.

Table 7-138: Cut element (Section 2 of 2)

Name	Data Type	Description
<i>WorkingDirection</i>	enumeration	Direction from which the tool is working. Possible values are: <i>Top</i> – From above. <i>Bottom</i> – From below.

7.2.49 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
Resource references:	—
Resource inheritance:	—
Example Partition:	<i>PlateLayout, ProductionRun, Separation</i>
Input of processes:	—
Output of processes:	CylinderLayoutPreparation

Table 7-139: CylinderLayout resource

Name	Data Type	Description
<i>DeviceID</i> ?	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.

— Element: CylinderPosition

Table 7-140: CylinderPosition element (Section 1 of 2)

Name	Data Type	Description
<i>PlatePosition</i>	XYPairRangeList	Specifies where to mount this plate onto the cylinder. See figure below for details.
<i>PlateType</i> = "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. <i>Dummy</i> – Indicates that the plate is a dummy plate. It MUST be bent by a Bending process. But it is unlikely to be exposed by an ImageSetting process. <i>Exposed</i> – Indicates that the plate contains content data and MUST be exposed by an ImageSetting process.
<i>PlateUsage</i> = "Original"	enumeration	Specifies, whether a plate has to be produced for a specific web run or not. Allowed values are: <i>Original</i> – indicates that the plate is to be produced specifically for this run. <i>Reuse</i> – 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.

Table 7-140: CylinderPosition element (Section 2 of 2)

Name	Data Type	Description
<i>DeviceModuleIndex</i>	integer	Defines a Module with <i>ModuleType</i> = " <i>PrintModule</i> " within the Device specified by CylinderLayout/@DeviceID . In a newspaper-web press, <i>PrintModule</i> corresponds to a single cylinder.

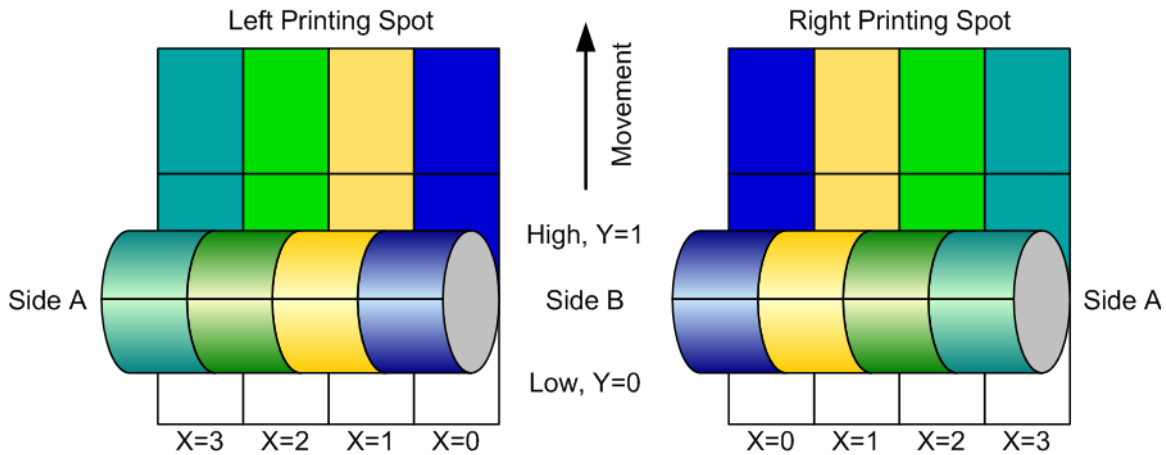


Figure 7-16: Definition of the PlatePosition attribute on a newspaper-web press

In Figure 7-16, 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.

The sketch in Figure 7-17 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.

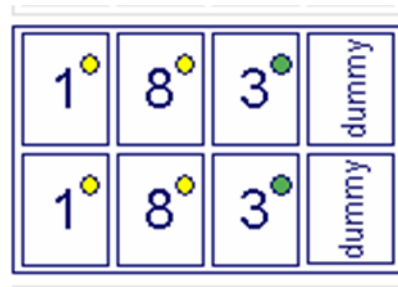


Figure 7-17: Example of a single physical section of eight pages

The following **CylinderLayout** is an example representation of the cylinder layout as shown in the sketch.

```
<JDF>
  <ResourcePool>
    <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" PartID=
Keys="ProductionRun PlateLayout Separation" DeviceID="DEV-001">
    <LayoutRef rRef="LAY-001"/>
    <CylinderLayout ProductionRun="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>

```

```
</JDF>
```

In case of a double-spread-page plate (or double-truck-page plate) the `CylinderPosition` MAY be set as:

```
<!-- PlatePosition (XYPairRangeList)-->
<CylinderPosition DeviceModuleIndex="2" PlatePosition="0 0 ~ 1 0" PlateType="Exposed"
PlateUsage="Original"/>
```

7.2.50 CylinderLayoutPreparationParams

[New in JDF 1.3](#)

This resource specifies the parameters of the *CylinderLayoutPreparation* process.

Resource Properties

Resource class:	Parameter
Resource references:	—
Resource inheritance:	—
Example Partition:	<i>WebName, ProductionRun</i>
Input of processes:	<i>CylinderLayoutPreparation</i>
Output of processes:	—

Table 7-141: CylinderLayoutPreparationParams resource

Name	Data Type	Description
ProductionPath	refelement	ProductionPath describes the individual paper path through the different modules of a web-press.

7.2.51 DBMergeParams

This resource specifies the parameters of the *DBTemplateMerging* process.

Resource Properties

Resource class:	Parameter
Resource references:	—
Resource inheritance:	—
Example Partition:	—
Input of processes:	<i>DBTemplateMerging</i>
Output of processes:	—

Table 7-142: DBMergeParams resource

Name	Data Type	Description
<i>SplitDocuments ?</i>	integer	Indicates how often to split documents to create a new file.
FileSpec ?	refelement	URL of the generated destination file. This is most often a printable file type, (e.g., PDF or PPML). If FileSpec is not specified, DBMergeParams MUST be a Pipe resource.

7.2.52 DBRules

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
Resource references:	—
Resource inheritance:	—
Example Partition:	—
Input of processes:	<i>DBDocTemplateLayout, Inserting, Collecting, Gathering</i>
Output of processes:	—

Table 7-143: DBRules resource

Name	Data Type	Description
Comment +	telem	Human-readable description of the database rules that map database fields to image or text content.

7.2.53 DBSchema

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
Resource references:	—
Resource inheritance:	—
Example Partition:	—
Input of processes:	<i>DBDocTemplateLayout, Verification</i>
Output of processes:	—

Table 7-144: DBSchema resource

Name	Data Type	Description
<i>DBSchemaType</i>	enumeration	Database type. Possible values are: <i>CommaDelimited</i> <i>SQL</i> <i>XML</i>
Comment +	telem	Human-readable description of the database schema.

7.2.54 DBSelection

This resource specifies a selection of records from a database.

Resource Properties

Resource class:	Parameter
Resource references:	—
Resource inheritance:	—
Example Partition:	—
Input of processes:	<i>DBTemplateMerging, Inserting, Collecting, Gathering, Verification</i>
Output of processes:	<i>Verification</i>

Table 7-145: DBSelection resource (Section 1 of 2)

Name	Data Type	Description
<i>DataBase</i>	URL	URL of the database

Table 7-145: DBSelection resource (Section 2 of 2)

Name	Data Type	Description
<i>Records ?</i>	IntegerRangeList	The indices of the database records.
<i>Select ?</i>	string	Database selection criteria in the native language of the database, (e.g., SQL).

7.2.55 DeliveryParams

Provides information needed by a **Delivery** process. A **Delivery** process consists of sending a quantity of a product to a specific location at, in some cases, a specified date and time.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	Delivery
Output of processes:	—

Table 7-146: DeliveryParams resource (Section 1 of 2)

Name	Data Type	Description
<i>Earliest ?</i>	dateTime	Specifies the earliest time after which the delivery is intended to be made.
<i>Method ?</i>	string	Specifies a delivery method, (e.g., <i>ExpressMail</i> or <i>InterofficeMail</i>). Note that it is strongly RECOMMENDED to use an NMTOKEN compatible string in this attribute, without blanks. The values are the same as DeliveryIntent/@Method .
<i>Pickup ?</i> Deprecated in JDF 1.2	boolean	If " <i>true</i> ", the merchandise is picked up. If " <i>false</i> ", the merchandise is delivered. Replaced with <i>Transfer</i> in JDF 1.2.
<i>Required ?</i>	dateTime	Specifies the time by which the delivery is intended to be made.
<i>ServiceLevel ?</i> New in JDF 1.2	string	The service level of the specific carrier. Contain values " <i>Next Day</i> ", " <i>2nd Day Air</i> ", " <i>Ground</i> ", etc.

Table 7-146: DeliveryParams resource (Section 2 of 2)

Name	Data Type	Description
Transfer ? New in JDF 1.2	enumeration	Describes the direction and responsibility of the transfer. Possible values are: <i>BuyerToPrinterDeliver</i> – The DeliveryIntent 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 can choose to specify in the quote a special Contact [contains (@ <i>ContactTypes</i> , <i>Delivery</i>)]. This Contact specifies where the buyer is to send the merchandise. <i>BuyerToPrinterPickup</i> – The DeliveryIntent 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. The Contact [contains (@ <i>ContactTypes</i> , " <i>Pickup</i> ") specifies where the printer is to pick up the merchandise. <i>PrinterToBuyerDeliver</i> – The DeliveryIntent describes an output of the job. In this, case the printer delivers the merchandise to the buyer. The Contact [contains (@ <i>ContactTypes</i> , " <i>Delivery</i> ") specifies where the printer is to send the merchandise. <i>PrinterToBuyerPickup</i> – the DeliveryIntent describes an output of the job. In this case, the buyer picks up the merchandise. The printer can choose to specify in the quote a special Contact [contains (@ <i>ContactTypes</i> , " <i>Pickup</i> ")]. This Contact specifies where the buyer is to 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
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.

— Element: Drop

Table 7-147: Drop element (Section 1 of 2)

Name	Data Type	Description
Earliest ?	dateTime	Specified the earliest time after which the delivery is to be made. Default = DeliveryParams/@Earliest .
Method ?	string	Specifies a delivery method, (e.g., <i>ExpressMail</i> or <i>InterofficeMail</i>). Default is DeliveryParams/@Method . Note that it is strongly RECOMMENDED to use an NMTOKEN compatible string without blank spaces in this attribute. The values are the same as DeliveryIntent/@Method .
Pickup ? Deprecated in JDF 1.2	boolean	If " <i>true</i> ", the merchandise is picked up. If " <i>false</i> ", the merchandise is delivered. Default = DeliveryParams/@Pickup . Replaced with <i>Transfer</i> in JDF 1.2.
Required ?	dateTime	Specifies the time by which the delivery is intended to be made. Default is DeliveryParams/@Required .
ServiceLevel ? New in JDF 1.2	string	The service level of the specific carrier. Contain values " <i>Next Day</i> ", " <i>2nd Day Air</i> ", " <i>Ground</i> ", etc. Default = DeliveryParams/@ServiceLevel .

Table 7-147: Drop element (Section 2 of 2)

Name	Data Type	Description
TrackingID ? New in JDF 1.2	string	The string that can help in tracking the delivery. The value of the <i>TrackingID</i> 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. The default is and possible values are defined in 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 Contact responsible for this delivery. Default = DeliveryParams/Contact .
DropItem +	element	A Drop MAY consist of multiple products, which are represented by their respective PhysicalResource resources. Each DropItem describes an individual resource that is part of this Drop.

— Element: DropItem

Table 7-148: DropItem element (Section 1 of 2)

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 . <i>ActualAmount</i> 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 <i>TotalAmount</i> of items delivered in this drop. Note that this logs the information after the fact in a way that is similar to an Audit . <i>ActualTotalAmount</i> 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 <i>Amount</i> 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 <i>TotalAmount</i> and <i>Amount</i> differ if the PhysicalResource is a Bundle of multiple resources. The <i>Amount</i> specifies the number of Bundles (e.g., boxes, palettes etc.). Whereas <i>TotalAmount</i> 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 <i>TrackingID</i> attribute will depend on the carrier chosen to ship the products. Defaults to Drop/@TrackingID .
Unit ?	string	Unit of measurement for the <i>Amount</i> of the resource that is referenced by PhysicalResource . Defaults to the value of <i>Unit</i> defined in PhysicalResource .

Table 7-148: Dropltem element (Section 2 of 2)

Name	Data Type	Description
PhysicalResource ? Modified in JDF 1.2	refelement	Description of the individual item to be delivered. It can be any kind of physical resource. This was Component prior to JDF 1.2.

7.2.56 DensityMeasuringField

This resource contains information about a density measuring field.

Resource Properties

Resource class:	Parameter
Resource referenced by:	ColorControlStrip, Layout/PlacedObject
Example Partition:	—
Input of processes:	Any printing process
Output of processes:	—

Table 7-149: DensityMeasuringField resource (Section 1 of 2)

Name	Data Type	Description
<i>Center</i>	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 , <i>Center</i> refers to the rectangle defined by <i>Center</i> and <i>Size</i> 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 remains C M Y K, as in the data type CMYKColor .
<i>Diameter</i>	double	Diameter of measuring field.
<i>DotGain</i>	double	Percentage of dot gain.
<i>Percentage</i>	double	Film percentage or equivalent.
<i>Screen</i>	string	Description of the screen.
<i>Separation</i>	string	Reference to a <i>Separation</i> that this applies DensityMeasuringField to. When DensityMeasuringField is use as an element, it is a standard attribute, otherwise when DensityMeasuringField is used as a resource, <i>Separation</i> MUST be defined as a <i>Separation</i> partition key.
<i>Setup ?</i>	string	Description of measurement setup.
<i>ToleranceCyan</i>	XYPair	Upper and lower cyan measurement limits (in density units).
<i>ToleranceMagenta</i>	XYPair	Upper and lower magenta measurement limits (in density units).
<i>ToleranceYellow</i>	XYPair	Upper and lower yellow measurement limits (in density units).

Table 7-149: DensityMeasuringField resource (Section 2 of 2)

Name	Data Type	Description
<i>ToleranceBlack</i>	XYPair	Upper and lower black measurement limits (in density units).
<i>ToleranceDotGain</i>	XYPair	Upper and lower measurement limits (in%).
ColorMeasurementConditions ? New in JDF 1.1	refelement	Detailed description of the measurement conditions for color measurements.

7.2.57 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
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>ContactCopying, FilmToPlateCopying, ImageSetting</i>
Output of processes:	—

Table 7-150: DevelopingParams resource

Name	Data Type	Description
<i>PreHeatTemp</i> ?	double	Temperature of the preheating process in °C.
<i>PreHeatTime</i> ?	duration	Duration of the preheating process.
<i>PostBakeTemp</i> ?	double	Temperature of the postbaking process in °C.
<i>PostBakeTime</i> ?	duration	Duration of the postbaking process. <i>PostBakeTime</i> MUST NOT be specified if <i>PostExposeTime</i> is present.
<i>PostExposeTime</i> ?	duration	Duration of the postexposing process. <i>PostExposeTime</i> MUST NOT be specified if <i>PostBakeTime</i> is present.

7.2.58 Device

Information about a specific device. This can include information about the devices capabilities. For more information, see Section 3.7.2.3, Implementation Resources and Section 4.8, Capability and Constraint Definitions.

Resource Properties

Resource class:	Implementation
Resource referenced by:	—
Example Partition:	—
Input of processes:	Any process
Output of processes:	—

Table 7-151: Device resource (Section 1 of 2)

Name	Data Type	Description
DeviceFamily ? Deprecated in JDF 1.1	string	Manufacturer family type ID. The <i>DeviceFamily</i> is replaced by the appropriate <i>ModelXXX</i> attributes in this list.
DeviceID ?	string	Name of the device. This is a unique name within the workflow. <i>DeviceID</i> MUST be the same over time for a specific device instance, (i.e., MUST survive reboots). If the device sends JMF messages, this value MUST also be used for <i>JMF/@SenderID</i> . For UPNP devices, this MUST match UPNP:UDN. See [UPNP]. <i>DeviceID</i> 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.
Directory ? New in JDF 1.1	URL	Defines a directory where the URLs that are associated with this Device can be located. If <i>Directory</i> is specified, it MUST be an Absolute URI [RFC3986] that implicitly also specifies a Base URI which is used to resolve any relative URL of Device . See "Resolving RunList/@Directory and FileSpec/@URL URI references" on page 745 and [FileURL].
FriendlyName ? New in JDF 1.1	string	Short user-friendly title.
ICSVersions ? New in JDF 1.3	NMTOKENS	CIP4 Interoperability Conformance Specification (ICS) Versions that this Device complies with. The semantics are identical to <i>JDF/ICSVersions</i> . For details, see Table 3-4, "JDF node," on page 41
JDFErrorURL ? New in JDF 1.2	URL	URL where, by default, the device can copy JDF output job tickets that are aborted or in error. If this is a directory, it specifies the device default error output folder. If not specified, it defaults to the value of <i>JDFOutputURL</i> .
JDFInputURL ? New in JDF 1.2	URL	URL where, by default, the device can accept JDF input job tickets. If this is a directory, it specifies the device default 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 can copy successfully completed JDF output job tickets. If this is a directory, it specifies the device default output folder.
JDFVersions ? New in JDF 1.1	JDFJMFVersions	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 version are supported).
JMFSenderID ? New in JDF 1.1	string	ID of the controller will process JMF messages for the device. This corresponds to the <i>SenderID</i> attribute that is specified for the device in JMF messages. If a Device emits it's own JMF messages, this value MUST match the <i>DeviceID</i> .
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 <i>JMFURL</i> when responding to <i>KnownDevices</i> 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.

Table 7-151: Device resource (Section 2 of 2)

Name	Data Type	Description
<i>ModelDescription</i> ? New in JDF 1.1	string	Long description for end user.
<i>ModelName</i> ? New in JDF 1.1	string	Model name.
<i>ModelNumber</i> ? New in JDF 1.1	string	Model number.
<i>ModelURL</i> ? New in JDF 1.1	string	Web site for model.
<i>SerialNumber</i> ? New in JDF 1.1	string	Serial number of the device.
<i>PresentationURL</i> ? New in JDF 1.1	string	URL to presentation for device It is a URL to a device-provided user interface for configuration, status, etc. Thus, if the device has an embedded Web server, this is a URL to the configuration page hosted on that Web server.
<i>SecureJMFURL</i>? New in JDF 1.3	URL	URL of the device port that will accept JMF messages via the https protocol.
<i>UPC</i> ? 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.
<i>CostCenter</i> ?	element	MIS cost center ID.
<i>DeviceCap</i> * 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 7.3, <i>Device Capability Definitions</i> .
<i>IconList</i> ? New in JDF 1.1	element	List of locations of icons that can be used to represent the device.
<i>Module</i> * New in JDF 1.3	element	Individual Modules that are represented by this Device .

— Element: IconList[New in JDF 1.1](#)The **IconList** is a list of individual icon descriptions.**Table 7-152: IconList element**

Name	Data Type	Description
Icon +	element	Individual icon description.

— Element: Icon[New in JDF 1.1](#)An **Icon** represents a device in the user interface.**Table 7-153: Icon element (Section 1 of 2)**

Name	Data Type	Description
<i>Size</i>	XYPair	Height and width of the icon.
<i>BitDepth</i>	integer	Bit depth of one color.

Table 7-153: Icon element (Section 2 of 2)

Name	Data Type	Description
<i>IconUsage</i> ?	enumerations	Definition of the <i>Status</i> of the device that this ICON represents. Any combination of: <i>Unknown</i> – No link to the device exists <i>Idle</i> <i>Down</i> <i>Setup</i> <i>Running</i> <i>Cleanup</i> <i>Stopped</i> If not specified, the ICON is used for all of the above. The meaning of the individual enumerations is described in the DeviceInfo message element. See Section 5.7.3, KnownDevices .
FileSpec	element	Details of the file containing the icon data.

— Element: Module[New in JDF L.3](#)A **Module** represents a physical machine or part of a **Device**.**Table 7-154: Module element**

Name	Data Type	Description
<i>DeviceType</i> ?	string	Manufacturer type ID, including a revision stamp.
<i>Manufacturer</i> ?	string	Manufacturer name.
<i>ManufacturerURL</i> ?	string	Web site for manufacturer.
<i>ModelDescription</i> ?	string	Long description for end user.
<i>ModelName</i> ?	string	Model name.
<i>ModelNumber</i> ?	string	Model number.
<i>ModelURL</i> ?	string	Web site for model.
<i>ModuleID</i> ?	string	Name of the Module. This is a unique identifier within the workflow. ModuleID MUST be the same over time for a specific device instance, (i.e., MUST survive reboots). At least one of ModuleID or ModuleIndex MUST be specified. If multiple logical Devices share a physical Module , ModuleID MUST be identical. ModuleID SHOULD be used to specify Machines that comprise a Device .
<i>ModuleIndex</i> ?	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 MUST be specified. ModuleIndex SHOULD be used to specify identical modules, e.g., print modules in a complex Device.
<i>ModuleType</i> ?	NMTOKEN	Type of Module . For a list of predefined ModuleType values, See “ModuleType Supported Strings” on page 707.
<i>SerialNumber</i> ?	string	Serial number of the device.
<i>SubModuleIndex</i> ?	integer	Zero-based index of the Module in the unit as specified by the parent Module . MUST NOT be specified if Module is a direct child of Device .
Module *	element	Recursive modules that are part of this module.

7.2.59 DeviceMark

[New in JDF 1.1](#)

Promoted from subelement status in the **Layout** resource with new attributes defined below.

Resource Properties

Resource class:	Parameter
Resource referenced by:	Layout
Example Partition:	<i>Side</i>
Input of processes:	—
Output of processes:	—

Table 7-155: DeviceMark resource

Name	Data Type	Description
<i>Font ?</i>	NMTOKEN	The name of the font that is to be used for the DeviceMark . Values include: <i>Courier</i> <i>Helvetica</i> <i>Helvetica-Condensed</i> <i>Times-Roman</i>
<i>FontSize ?</i>	integer	The size of the font that is to be used for the DeviceMark , in points ≥ 0 .
<i>MarkJustification ?</i>	enumeration	Description of the preferred DeviceMark justification. Interpreted in context of the <i>MarkOrientation</i> . One of: <i>Center</i> <i>Left</i> <i>Right</i>
<i>MarkOffset ?</i>	XYPair	Description of the preferred DeviceMark offset. Interpreted in context of the device dependent default position in the coordinate system defined by <i>MarkOrientation</i> .
<i>MarkOrientation ?</i>	enumeration	Description of the preferred DeviceMark orientation. One of: <i>Horizontal</i> <i>Vertical</i>
<i>MarkPosition ?</i>	enumeration	Description of the preferred DeviceMark position. One of: <i>Top</i> <i>Bottom</i> <i>Left</i> <i>Right</i>

7.2.60 DeviceNSpace

Note: **DeviceNSpace** was elevated to a resource in JDF 1.2. 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/DeviceNColor/@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.

Resource Properties

Resource class:	Parameter
Resource referenced by:	ColorantControl, ColorSpaceConversionParams
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-156: DeviceNSpace resource

Name	Data Type	Description
<i>Name</i> ?	string	Color space name, (e.g., HexaChrome or HiFi).
<i>N</i>	integer	The number of colors that define the color space.
SeparationSpec * Modified in JDF 1.2	element	Ordered list of colorant names that define the DeviceN color space. Note that these colorants MUST be specified in a corresponding ColorantParams element of the ColorantControl or be implied by <i>ProcessColorModel</i> . In other words, they MUST be real, physical colorants.

7.2.61 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 (ANSI® IT8.6-2002). 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
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-157: DieLayout resource

Name	Data Type	Description
FileSpec	refelement	Reference to an external URL that represents the die.
Station *	element	Description of the stations in a DieLayout. One Station produces one shape.

— Element: Station

Table 7-158: Station element

Name	Data Type	Description
<i>StationAmount</i> = "1"	integer	The number of stations in the DieLayout with this <i>StationName</i> .
<i>StationName</i> ?	string	The name of the 1-up design in the DieLayout .

7.2.62 DigitalDeliveryParams

[New in JDF 1.2](#)

This resource specifies the parameters of the DigitalDelivery process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>Location</i>
Input of processes:	DigitalDelivery
Output of processes:	—

Table 7-159: DigitalDeliveryParams resource

Name	Data Type	Description
<i>DigitalDeliveryDirection</i> ?	enumeration	Describes which side activates the delivery. <i>Push</i> – The artwork will be sent (the source end is active). <i>Pull</i> – The artwork will be retrieved (the destination end is active).
<i>DigitalDeliveryProtocol</i> ?	NMTOKEN	Identifies the delivery network protocol. Values include: <i>FTP</i> <i>HTTP</i> <i>HTTPS</i> <i>SMTP</i>
<i>Method</i> ?	NMTOKEN	Identifies the delivery method. Values include: <i>Email</i> <i>ISDNSoftware</i> <i>NetworkCopy</i> – This includes LAN and VPN. <i>WebServer</i> – Upload / Download from HTTP / FTP server. <i>InstantMessaging</i> Values can be a digital delivery service brand, and include: <i>Vio</i> <i>WAMNET</i>
Contact *	refelement	Source and destination address for the transfer of the artwork. The destination delivery address is specified as the Contact [contains (@ <i>ContactTypes</i> , "Delivery")]/ ComChannel . Exactly one such Contact MUST be specified per destination. If multiple delivery destinations are specified within one DigitalDelivery process, such a Contact MUST be partitioned with the partition key " <i>Location</i> ". If the output RunList completely specifies the destination, a Contact [contains (@ <i>ContactTypes</i> , "Delivery")] SHOULD be omitted. This is generally the case if <i>Method</i> = " <i>NetworkCopy</i> " or " <i>WebServer</i> ". A Contact [contains (@ <i>ContactTypes</i> , "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 "DigitalDelivery Examples" on page 790. for a set of examples.

7.2.63 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, DeliveryParams
Example Partition:	—
Input of processes:	—
Output of processes:	Delivery

Table 7-160: DigitalMedia resource

Name	Data Type	Description
<i>Capacity</i> ?	integer	Size of the digital media in megabytes.
<i>MediaLabel</i> ?	string	Electronic label of the media.
<i>MediaType</i>	NMTOKEN	The digital media type. Values include: <i>CD</i> – Recordable compact disc. <i>DAT</i> – DAT tape backup media. <i>DLT</i> – DLT tape backup media. <i>DVD</i> – DVD disc. <i>Exabyte</i> – Exabyte tape backup media. <i>HardDrive</i> – Removable hard drives from a rack. <i>Jaz</i> – Jaz removable disk drive. <i>Optical</i> – Optical removable disk drive. Excluding CDs and DVDs. <i>Tape</i> – Tape backup media. Use only when the explicit tape type is not listed here. <i>Zip</i> – Zip removable disk drive.
<i>MediaTypeDetails</i> ?	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.

7.2.64 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-fed presses. *SheetFed* printing, on the other hand, accepts pre-cut substrates.

7.2.64.1 Coordinate systems in DigitalPrinting

[New in JDF 1.2](#)

Figure 2-11 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
Resource referenced by:	—

Example Partition: *BlockName, DocRunIndex, DocSheetIndex, PartVersion, Run, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetIndex, Separation, SheetName, Side, SignatureName, DocIndex*

Input of processes: *DigitalPrinting*

Output of processes: —

Table 7-161: DigitalPrintingParams resource (Section 1 of 3)

Name	Data Type	Description
<u>Collate ?</u> <u>New in JDF 1.1</u>	enumeration	<p>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.</p> <p><i>None</i> – Do not collate sheets in the document or document(s) in the job.</p> <p><i>Sheet</i> – Collate the sheets in each document; do not collate the documents in the job. The result of <i>Sheet</i> and <i>SheetAndSet</i> is the same when there is one document in the set. The result of <i>Sheet</i> and <i>SheetSetAndJob</i> is the same when there is one document in the set and one set in the job.</p> <p><i>SheetAndSet</i> – Collate the sheets in the document and collate the documents in the set. Do not collate the sets in the job. The result of <i>SheetAndSet</i> and <i>SheetSetAndJob</i> is the same when there is one set in the job.</p> <p><i>SheetSetAndJob</i> – Collate the sheets in the document and collate the documents in the set and collate the sets in the job.</p> <p>The following example consists of two documents, A and B, each having 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.</p> <p>If Collate = "<i>None</i>", the sheet order will be: A1A1A1 A2A2A2 B1B1B1 B2B2B2</p> <p>If Collate = "<i>Sheet</i>", the sheet order will be: A1A2 A1A2 A1A2 B1B2 B1B2 B1B2</p> <p>If Collate = "<i>SheetAndSet</i>" or "<i>SheetSetAndJob</i>", the sheet order will be: A1A2 B1B2 A1A2 B1B2 A1A2 B1B2</p>
<u>DirectProofAmount = "0"</u> <u>New in JDF 1.2</u>	integer	<p>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 a <i>DirectProofAmount</i>, the JDF node's <i>Status</i> is switched to "<i>Stopped</i>" with the <i>StatusDetails</i> = "<i>WaitForApproval</i>".</p>
<u>ManualFeed = "false"</u> <u>New in JDF 1.1</u>	boolean	<p>Indicates whether the media will be fed manually.</p>

Table 7-161: DigitalPrintingParams resource (Section 2 of 3)

Name	Data Type	Description
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 MUST put marks up to, but not in, the non-printable margin area. The Media 's origin is unaffected by <i>NonPrintableMarginBottom</i> . These margins are independent of the PDL content.
NonPrintableMarginLeft ? New in JDF 1.2	double	Same as <i>NonPrintableMarginBottom</i> except for the left margin.
NonPrintableMarginRight ? New in JDF 1.2	double	Same as <i>NonPrintableMarginBottom</i> except for the right margin.
NonPrintableMarginTop ? New in JDF 1.2	double	Same as <i>NonPrintableMarginBottom</i> except for the top margin.
OutputBin ? New in JDF 1.1 Modified in JDF 1.2	NMTOKEN	Specifies the bin to which the finished document is to be output. Suggested values can be found in "Input Tray and Output Bin Names" on page 714.
PageDelivery ? New in JDF 1.1	enumeration	Indicates how pages are to be delivered to the output bin or finisher. Possible values are: <i>FanFold</i> – The output is alternating face-up, face down. <i>SameOrderFaceUp</i> – Order as defined by the RunList , with the “ <i>Front</i> ” sides of the media up and the first sheet on top. <i>SameOrderFaceDown</i> – Order as defined by the RunList , with the “ <i>Front</i> ” sides of the media down and the first sheet on the bottom. <i>ReverseOrderFaceUp</i> – Sheet order reversed compared to <i>SameOrderFaceUp</i> , with the “ <i>Front</i> ” sides of the media up and the last sheet on top. <i>ReverseOrderFaceDown</i> – Sheet order reversed compared to <i>SameOrderFaceDown</i> , with the “ <i>Front</i> ” sides of the media down and the last sheet on the bottom.
PrintingType ? Modified in JDF 1.2	enumeration	Type of printing machine. Possible values are: <i>ContinuousFed</i> – connected sheets including fan fold. New in JDF 1.2 <i>SheetFed</i> <i>WebFed</i>
PrintQuality ? Deprecated in JDF 1.1	enumeration	Indicates how pages are to be delivered to the output bin or finisher. Possible values are: <i>High</i> – Highest quality available on the printer. <i>Normal</i> – The default quality provided by the printer. <i>Draft</i> – Lowest quality available on the printer Replaced by InterpretingParams/@PrintQuality
SheetLay ?	enumeration	Lay of input media. Reference edge of where paper is placed in feeder. Possible values are: <i>Left</i> <i>Right</i> <i>Center</i>

Table 7-161: DigitalPrintingParams resource (Section 3 of 3)

Name	Data Type	Description
Sides ? New in JDF 1.3	enumeration	Indicates whether the content layout MUST be imaged on one or both sides of the media. This attribute is needed when the DigitalPrinting process is a single process in a process network and the single process is accepting a RunList of byte maps consisting of page image data output from another process. When a different value for this attribute is encountered, it MUST 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. For a list of possible values, see LayoutPreparationParams/@Sides
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 MUST be specified per partition.
Disjointing ? New in JDF 1.1	element	Describes how individual components are separated from one another in the output bin.
Ink ? New in JDF 1.3	refelement	If present indicates that overcoating is to be applied to the surface(s) of printed sheets and specifies the ink to be used for overcoating. Overcoating ink MUST be applied after imaging colorants have been printed. Note: for selective image-wise overcoating (e.g., spot varnishing) a separate separation utilizing overcoating ink MUST be specified.
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 MUST be specified per partition.
MediaSource ? Deprecated in JDF 1.1	refelement	Describes the media to be used. At most one of MediaSource and Component MUST be specified. Replaced with Media in JDF 1.1.

7.2.65 Disjointing

The **Disjointing** resource describes how individual components are separated from one another on a stack.

Resource Properties

Resource class:	ResourceElement
Resource referenced by:	Component, DigitalPrintingParams, GatheringParams
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-162: Disjointing resource (Section 1 of 2)

Name	Data Type	Description
Number ?	integer	Number of sheets that make up one component.
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.

Table 7-162: Disjoining resource (Section 2 of 2)

Name	Data Type	Description
<i>OffsetDirection</i> ?	enumeration	Offset-shift action for the first component. A component can be offset to one of two positions—left or right. Possible values are: <i>Alternate</i> – The position of the first component 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. <i>Left</i> – Offset consecutive components sideways to the left, next to the right. <i>None</i> – Do not offset consecutive components. The position of all components is the same as the position of the previous component. <i>Right</i> – Offset consecutive components sideways to the right, next to the left. <i>Straight</i> – Same as <i>None</i> . Deprecated in JDF 1.2
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.

7.2.66 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 either as:

$$Until \leq "Disposition\ time" \leq Until + ExtraDuration$$

$$ProcessCompleteTime + MinDuration \leq "Disposition\ time" \leq$$

$$ProcessCompleteTime + MinDuration + ExtraDuration$$

Resource Properties

Resource class:	ResourceElement
Resource referenced by:	FileSpec, SubmitQueueEntry/QueueSubmissionParams, RunList
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-163: Disposition resource (Section 1 of 2)

Name	Data Type	Description
<i>DispositionAction</i> = "Delete"	enumeration	<i>Delete</i> – The asset is deleted when disposition time occurs. <i>Archive</i> – The asset is archived when disposition time occurs.

Table 7-163: Disposition resource (Section 2 of 2)

Name	Data Type	Description
<i>DispositionUsage</i> ?	enumeration	Specifies the usage of the asset by the process. If not specified, Disposition applies to all processes that link to the resource. <i>Input</i> – Disposition applies only to processes that use the asset as an input resource. <i>Output</i> – Disposition applies only to processes that use the asset as an output resource.
<i>ExtraDuration</i> ?	duration	Indicates the maximum duration that the device is allowed to retain the asset after the time specified by <i>MinDuration</i> or <i>Until</i> . If <i>ExtraDuration</i> , <i>MinDuration</i> and <i>Until</i> are all unspecified, the asset is retained for a system specified time.
<i>MinDuration</i> ?	duration	Indicates the minimum duration that the device SHOULD retain the asset after the process that uses the asset completes.
<i>Priority</i> = "0"	integer	Value between 0 and 100 that specifies the order in which assets are deleted or archived when the values of <i>ExtraDuration</i> , <i>MinDuration</i> and <i>Until</i> cannot be honored, (e.g., when local storage runs low). Assets with <i>Priority</i> = "0" will be deleted first.
<i>Until</i> ?	dateTime	Indicates an absolute point in time when the device or application SHOULD stop the asset retention. If <i>Until</i> is specified, <i>MinDuration</i> MUST be ignored.

7.2.67 DividingParams

[Deprecated in JDF 1.1.](#)

Since the **Dividing** process has been replaced by **Cutting**, this resource is no longer needed. See "DividingParams" on page 834 for details of this deprecated resource.

7.2.68 ElementColorParams

[New in JDF 1.2](#)

This resource provides a container for color management related metadata applicable to a **LayoutElement**.

Resource Properties

Resource class:	Parameter
Resource referenced by:	ContentList,LayoutElement, PageList
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-164: ElementColorParams resource (Section 1 of 2)

Name	Data Type	Description
<i>ColorManagementSystem</i> ?	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 MUST match one of the registered four-character ICC CMM Type values. See the ICC Manufacturer's Signature Registry at http://www.color.org . Example values: "ACME" for the Acme Corp. CMM.

Table 7-164: ElementColorParams resource (Section 2 of 2)

Name	Data Type	Description
<i>ICCOutputProfileUsage</i> ?	enumeration	This attribute specifies the usage of the output intent profile or specified printing condition from the PDL. Possible values are: <i>PDLActual</i> – The embedded PDL output printing condition defines the actual output intent profile, (e.g., the final press output). <i>PDLReference</i> – The embedded PDL output printing condition defines the reference output intent profile, (e.g., the press profile for proofing). <i>IgnorePDL</i> – The embedded ICC output profile is incorrect and is to be ignored.
AutomatedOverPrintParams ?	refelement	A resource that provides controls for the automated selection of overprinting of black text or graphics.
ColorantAlias *	refelement	Each resource instance specifies a replacement colorant name string to be used instead of one or more named colorant strings found in the layout element.
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 <i>ColorStandard</i> . ColorSpaceConversionOp/@Operation is ignored in the context of ElementColorParams .
FileSpec (<i>ActualOutputProfile</i>) ?	refelement	A FileSpec resource pointing to an ICC profile that describes the characterization of an actual output target device.
FileSpec (<i>ReferenceOutputProfile</i>) ?	refelement	A FileSpec resource pointing to an ICC profile that describes a reference output print condition behavior that is to 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 .

7.2.69 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
Resource referenced by:	—
Example Partition:	<i>BlockName, RibbonName, SheetName, SignatureName, WebName</i>
Input of processes:	Embossing
Output of processes:	—

Table 7-165: EmbossingParams resource

Name	Data Type	Description
<i>Emboss</i> *	element	One <i>Emboss</i> element is specified for each impression.

— Element: Emboss

Table 7-166: Emboss element

Name	Data Type	Description
Direction Modified in JDF 1.3	enumeration	The direction of the image. Possible values are: <i>Both</i> – Both debossing and embossing in one stamp. <i>Flat</i> – The embossing foil is applied flat. Used for foil stamping. New in JDF 1.3 <i>Raised</i> – Embossing. <i>Depressed</i> – Debossing.
<i>EdgeAngle</i> ?	double	The angle of a beveled edge in degrees. Typical values are an angle of: 30, 40, 45, 50 or 60 degrees. If <i>EdgeAngle</i> is specified, <i>EdgeShape</i> = <i>Beveled</i> MUST be specified.
<i>EdgeShape</i> = "Rounded"	enumeration	The transition between the embossed surface and the surrounding media can be rounded or beveled (angled). Possible values are: <i>Rounded</i> <i>Beveled</i>
<i>EmbossingType</i>	enumeration	Possible values include <i>BlindEmbossing</i> – Embossed forms that are not inked or foiled. The color of the image is the same as the paper. <i>EmbossedFinish</i> – 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. <i>FoilEmbossing</i> – Combines embossing with foil stamping in one single impression. <i>FoilStamping</i> – Using a heated die to place a metallic or pigmented image from a coated foil on the paper. <i>RegisteredEmbossing</i> – Creates an embossed image that exactly registers to a printed image.
<i>Height</i> ?	double	The height of the levels. This value specifies the <i>vertical</i> distance between the highest and lowest point of the stamp, regardless of the value of <i>Direction</i> .
<i>ImageSize</i> ?	XYPair	The size of the bounding box of one single image.
<i>Level</i> ?	enumeration	The level of embossing. Possible values are: <i>SingleLevel</i> <i>MultiLevel</i> <i>Sculpted</i>
<i>Position</i> ?	XYPair	Position of the lower left corner of the bounding box of the embossed image in the coordinate system of the Component .

7.2.70 Employee

Information about a specific device or machine operator (see Section 3.7.2.3, Implementation Resources). **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: **NodeInfo**
Example Partition: —

Input of processes: Any process

Output of processes: —

Table 7-167: Employee resource

Name	Data Type	Description
<i>PersonalID</i> ?	string	ID of the relevant MIS employee.
<i>Roles</i> ? New in JDF 1.2	NMTOKENS	Defines the list of roles that the employee fills. Values include: <i>Apprentice</i> – Employee that is in training, (“Auszubildender” / “Auszubildende” in German). <i>Assistant</i> – Assistant operator. <i>Craftsman</i> – Trained employee, (“Geselle” / “Facharbeiter” in German). <i>CSR</i> – Customer Service Representative <i>Manager</i> – Manager. <i>Master</i> – Highly trained employee, (“Meister” in German). <i>Operator</i> – Operator. <i>ShiftLeader</i> – The leader of the shift.
<i>Shift</i> ?	string	Defines the shift to which the employee belongs.
<i>CostCenter</i> ?	element	MIS cost center ID.
<i>Person</i> ?	refelement	Describes the employee. If no Person element is specified, the Employee resource represents any employee who fulfills the selection criteria.

7.2.71 EndSheetGluingParams

This resource describes the attributes and elements used in executing the **EndSheetGluing** process.

Resource Properties

Resource class: Parameter

Resource referenced by: —

Example Partition: —

Input of processes: **EndSheetGluing**

Output of processes: —

Table 7-168: EndSheetGluingParams resource

Name	Data Type	Description
EndSheet (Front)	element	Information about the front-end sheet. The <i>Side</i> attribute of this element MUST be “Front”.
EndSheet (Back)	element	Information about the back-end sheet. The <i>Side</i> attribute of this element MUST be “Back”.

— Element: EndSheet

Table 7-169: EndSheet element (Section 1 of 2)

Name	Data Type	Description
<i>Offset</i> ? Deprecated in JDF 1.2	XYPair	Offset of end sheet in X and Y direction. In JDF 1.2 and beyond, <i>Offset</i> is implied by the Transformation matrix in ResourceLink/ <i>@Transformation</i> of the EndSheet’s ComponentLink.

Table 7-169: EndSheet element (Section 2 of 2)

Name	Data Type	Description
<i>Side</i>	enumeration	Location of the end sheet. Possible values are: <i>Front</i> <i>Back</i>
GlueLine	element	Description of the glue line.

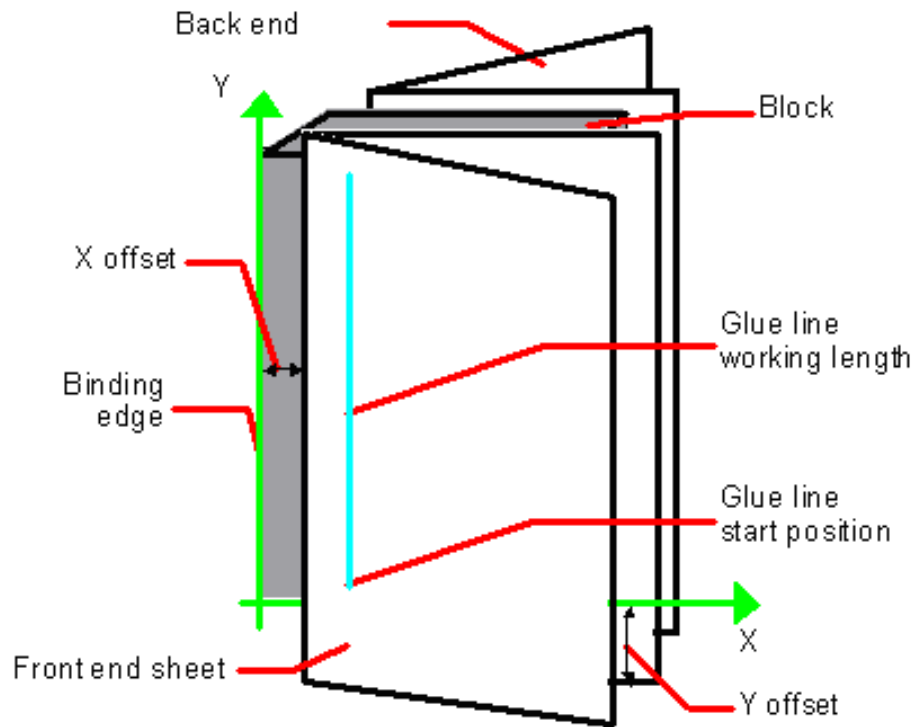


Figure 7-18: Parameters and coordinate system used for end-sheet gluing

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).

7.2.72 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.

Resource Properties

Resource class: Handling

Resource referenced by: —

Example Partition: *DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, Separation, SheetName, Side, SignatureName, TileID, WebName*

Input of processes: **Bending, ContactCopying, ConventionalPrinting, PreviewGeneration, DigitalPrinting, Scanning**

Output of processes: **Bending, ContactCopying, ImageSetting, Proofing**

Table 7-170: ExposedMedia resource

Name	Data Type	Description
<i>ColorType</i> ?	enumeration	Possible values are: <i>Color</i> <i>GrayScale</i> <i>Monochrome</i> – Black and white.
<i>PageListIndex</i> ? New in JDF 1.3	IntegerRangeList	List of the indices of the PageData elements of the PageList specified in this ExposedMedia .
<i>PlateType</i> ? New in JDF 1.3	enumeration	Specifies whether a plate is exposed or a dummy plate. <i>Exposed</i> – The plate has been imaged. <i>Dummy</i> – Specifies a dummy plate that has not been imaged. Usually, dummy plates are only needed on newspaper-web presses.
<i>Polarity</i> = "true"	boolean	<i>false</i> if the media contains a negative image.
<i>ProofName</i> ? New in JDF 1.2	string	When this ExposedMedia specifies a proof, <i>ProofName</i> is the name of the ProofingIntent/ProofItem that specified this proof in the product intent section.
<i>ProofQuality</i> ? Modified in JDF 1.2	enumeration	This attribute is present if the ExposedMedia resource describes a proof. Possible values are: <i>None</i> – Not a proof or the quality is unknown. Deprecated in JDF 1.2 <i>Halftone</i> – Halftones are emulated. <i>Contone</i> – No halftones, but exact color. <i>Conceptual</i> – Color does not match precisely.
<i>ProofType</i> ? Modified in JDF 1.2	enumeration	<i>None</i> – Not a proof or the type is unknown. Deprecated in JDF 1.2 <i>Page</i> – A page proof. <i>Imposition</i> – An imposition proof.
<i>PunchType</i> ?	string	Name of the registration punch scheme. Possible values include, but are not limited to: <i>Bacher</i> <i>Stoesser</i> If not specified, no holes are punched.
<i>Resolution</i> ?	XYPair	Resolution of the output.
FileSpec (<i>OutputProfile</i>)?	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.

7.2.73 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
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-171: 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.

7.2.74 FeedingParams

[New in JDF 1.2](#)

The parameters for any JDF Feeder processing device.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, Separation, SheetName, Side, SignatureName, TileID, WebName</i>
Input of processes:	Feeding
Output of processes:	—

Table 7-172: FeedingParams resource

Name	Data Type	Description
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.
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.

— Element: Feeder

Table 7-173: Feeder element (Section 1 of 2)

Name	Data Type	Description
<i>AlternatePositions ?</i>	IntegerList	Positions of alternate feeders including the feeder specified in <i>Position</i> on a feeding chain. Alternate feeders share the load according to the policy defined in <i>FeederSynchronization</i> . If not specified, it defaults to the value of <i>Position</i> . <i>AlternatePositions</i> MUST be non-negative.
<i>Position ?</i>	integer	<i>Position</i> of feeder on a collecting and gathering chain in chain movement direction. <i>Position</i> = "0" is first feeder feeding to the collecting and gathering chain. Only one Feeder can be specified for any given <i>Position</i> . If <i>Position</i> is negative, it specifies the position counted from the back of the chain, (e.g., "-1" = last position, "-2" = next to last position, etc.).

Table 7-173: Feeder element (Section 2 of 2)

Name	Data Type	Description
<i>FeederSynchronization</i> = "Primary"	enumeration	Specifies the synchronization of multiple Feeder(s) with identical Component (s): Values include: <i>Alternate</i> – The feeders specified in <i>Position</i> alternate. <i>Backup</i> – This Feeder is the backup feeder for the Component in case of a misfeed or malfunction. The priority of backup feeders is defined by their position in <i>AlternatePositions</i> . <i>Chain</i> – This feeder is activated as soon as the feeder prior to it in the list is empty. <i>Primary</i> – This Feeder is the primary feeder for the Component .
<i>FeederType ?</i>	NMTOKEN	Specifies the feeder type. Values include: <i>Sheet</i> – Single sheet feeder. <i>Signature</i> – Single signature feeder. <i>Folding</i> – A folding feeder that folds the input Component or Media . <i>Gluing</i> – A gluing feeder <i>AddOn</i> – Add on feeder, (e.g., CDs).
<i>Loading ?</i>	NMTOKEN	Specifies the feeder loading. Values include: <i>Bundle</i> – Stream feeder, using the output of the Bundling process. <i>FanFold</i> – Automatic loading of <i>FanFold</i> Media . <i>Manual</i> – Manual loading of stacks <i>Online</i> – Loaded by a gripper or conveyor. <i>PrintRoll</i> – Automatic loading of single products from a print roll, using the output of the PrintRolling process.
<i>Opening</i> = "None"	enumeration	Specifies the opening of signatures: <i>Back</i> – Overfold on back. <i>Front</i> – Overfold on front. <i>None</i> – Signatures are not opened. <i>Sucker</i> – Sucker opening, no overfold.
Component ?	refelement	Specifies the Component that is to be loaded into this Feeder . This Component MUST be an input of the Feeding process. Exactly one of Component or Media MUST be specified.
<i>FeederQualityParams ?</i>	element	Definition of the setup and policy for feeding quality.
Media ?	refelement	Specifies the Media that is to be loaded into this Feeder . This Media MUST be an input of the Feeding process. Exactly one of Component or Media MUST be specified.

— Element: FeederQualityParams

The *FeederQualityParams* element defines the setup and policy for feeding quality control. It can be specified individually for each *Feeder*.

Table 7-174: FeederQualityParams element

Name	Data Type	Description
<i>IncorrectComponentQuality</i> ?	enumeration	Defines the operation of the incorrect components quality control: Supported values are: <i>NotActive</i> – Quality control is not active. <i>Check</i> – Check the quality and register. <i>Waste</i> – Check the quality and register. A component failing the test is waste. <i>StopNoWaste</i> – Check the quality and register. Device will stop after the defined number of consecutive errors. The error will be corrected, (e.g., manually). <i>StopWaste</i> – Check the quality and register. A component failing the test is waste, and the device will stop after the defined number of consecutive errors.
<i>IncorrectComponents</i> ?	integer	Number of consecutive incorrect components until the device stops.
<i>DoubleFeedQuality</i> ?	enumeration	Defines the operation of the double feed quality control. For a list of supported values see: <i>IncorrectComponentQuality</i> .
<i>DoubleFeeds</i> ?	integer	Number of consecutive double feeds until the device stops.
<i>BadFeedQuality</i> ?	enumeration	Defines the operation of the bad feed quality control. For a list of supported values see: <i>IncorrectComponentQuality</i> .
<i>BadFeeds</i> ?	integer	Number of consecutive bad feeds until the device stops.

— Element: CollatingItem

Table 7-175: CollatingItem element (Section 1 of 2)

Name	Data Type	Description
<i>Amount</i> = "1"	integer	Determines, how many consecutive items shall be consumed.
<i>BundleDepth</i> ?	integer	In case of nested bundles with <i>BundleType</i> = "Stack", this parameter addresses the element to be consumed within the "tree" of such bundles. If the real bundle depth level (<i>BundleType</i> = "Stack") is smaller than the value of <i>BundleDepth</i> , 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.
<i>Orientation</i> ?	Orientation	Named <i>Orientation</i> of the <i>CollatingItem</i> relative to the input coordinate system. For details see Table 2-4 on page 26. At most one of <i>Orientation</i> or <i>Transformation</i> MUST 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 <i>ResourceLink</i> .
<i>Transformation</i> ?	matrix	Orientation of the Component respective to the input coordinate system. This <i>Transformation</i> specified here is applied in addition to orientation/transformation specified in the respective <i>ResourceLink</i> . At most one of <i>Orientation</i> and <i>Transformation</i> MUST be specified. If neither is specified, no transformation is applied.

Table 7-175: CollatingItem element (Section 2 of 2)

Name	Data Type	Description
<i>TransformationContext</i> = " <i>StackItem</i> "	enumeration	This parameter specifies the object, which is to be manipulated in orientation/transformation, and it is important to determine the sequence of stack items after flipping. <i>StackItem</i> – 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). <i>Component</i> – Apply to each single element of a <i>CollatingItem</i> individually. <i>CollateItem</i> – apply to a <i>CollatingItem</i> as a whole. Note: If <i>Amount</i> = "1", Component and <i>CollatingItem</i> are referring to the same object and, therefore, result in the same output.
Component ?	reference	References one of the input components to the process to be (partially) consumed by the <i>CollatingItem</i> element. This Component MUST be an input of the Feeding process. Exactly one of Component or Media MUST be specified.
Media ?	reference	References one of the input media to the process to be consumed by the <i>CollatingItem</i> element. This Media MUST be an input of the Feeding process. Exactly one of Component or Media MUST 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 **PrintRoll** process.

7.2.75 FileSpec

Specification of a file or a set of files. If a single **FileSpec** instance specifies a set of files, it MUST 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 MUST define a *Container* subelement which defines another **FileSpec** instance that specifies the container file. In such a case, the attributes of each **FileSpec** instance MUST apply only to the properties of the file at that level.

Resource Properties

Resource class:	Parameter
Resource referenced by:	DBMergeParams, LayoutElement, PDLResourceAlias, ScanParams
Example Partition:	<i>Separation</i>
Input of processes:	—
Output of processes:	—

Table 7-176: FileSpec resource (Section 1 of 6)

Name	Data Type	Description
<i>Application</i> ?	string	Creator application. See <i>AppVersion</i> for the application version number.

Table 7-176: FileSpec resource (Section 2 of 6)

Name	Data Type	Description
AppOS ? Modified in JDF 1.2	string	Operating system of the application that created the file. Possible values include: <i>DG_UX</i> <i>HP_UX</i> <i>IRIX</i> <i>Linux</i> <i>Mac</i> <i>Solaris</i> <i>Windows</i> 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 “AppOS and OSVersion Attributes” on page 739 for combinations of <i>AppOS</i> and <i>OSVersion</i> values.
AppVersion ?	string	Version of the value of the <i>Application</i> attribute. The following are some examples: “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 <i>Checksum</i> MUST 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. Possible values include: <i>Base64</i> – A format for encoding arbitrary binary information for transmission by electronic mail. [RFC3548] <i>BinHex</i> – BinHex encoding converts an 8-bit file into a 7-bit format, similar to Uuencoding [RFC1741]. <i>Compress</i> – UNIX compression [RFC1977]. <i>Deflate</i> – The file is compressed using Zip public domain compression format [RFC1951]. <i>Gzip</i> – GNU Zip compression technology [RFC1952]. <i>MacBinary</i> – 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] <i>None</i> – The file is neither compressed nor encoded. <i>UUencode</i> – A set of algorithms for converting files into a series of 7-bit ASCII characters that can be transmitted over the Internet. [uuencode] <i>ZLIB</i> – ZLIB compression [RFC1950].

Table 7-176: FileSpec resource (Section 3 of 6)

Name	Data Type	Description
<i>Disposition</i> ? Deprecated in JDF 1.2	enumeration	Indicates what the device is to do with the file when the process that uses this resource as an input resource completes. Possible values include: <i>Unlink</i> – The device is to release the file. <i>Delete</i> – The device is to attempt to delete the file. <i>Retain</i> – The device is to do nothing with the file. In JDF 1.2 and beyond, retention of assets is specified in the Disposition element.
<i>DocumentNaturalLang</i> ?	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.
<i>FileFormat</i> ?	string	A formatting string used with the <i>FileTemplate</i> attribute to define a sequence of URLs in a batch process, each of which has the same semantics as the <i>URL</i> attribute. If neither <i>URL</i> nor <i>UID</i> is present, both <i>FileFormat</i> and <i>FileTemplate</i> MUST be present, unless the resource is a pipe. If either <i>URL</i> or <i>UID</i> is specified, then <i>FileFormat</i> and <i>FileTemplate</i> MUST NOT be specified. For more information, see "Generating strings with Format and Template" on page 743
<i>FileSize</i> ? 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.
<i>FileTargetDeviceModel</i> ? 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. The value of this attribute MUST exactly match the IEEE 1284-2000 Device ID string, except the length field MUST NOT be specified. See the Microsoft Universal Plug-and-Play [UPNP] section 2.2.6 <i>DeviceId</i> parameter for details. Here is an example showing only the REQUIRED fields for a PostScript document formatted for a <i>LaserBeam 9</i> : MANUFACTURER:ACME Co.;COMMAND SET:PS;MODEL:LaserBeam 9; (See [IEEE1284] clause 7.6) If this attribute is not present, it is assumed that the file is device independent.
<i>FileTemplate</i> ?	string	A template, used with <i>FileFormat</i> , to define a sequence of URLs in a batch process, each of which has the same semantics as the <i>URL</i> attribute. If neither <i>URL</i> nor <i>UID</i> is present, both <i>FileFormat</i> and <i>FileTemplate</i> MUST be present, unless the resource is a pipe. For more information, see "Generating strings with Format and Template" on page 743
<i>FileVersion</i> ? New in JDF 1.1	string	Version of the file referenced by this FileSpec.

Table 7-176: FileSpec resource (Section 4 of 6)

Name	Data Type	Description
MimeType ? Modified in JDF 1.2	string	<p>MIME type or file type of the file (or files of identical type when specifying a sequence of file names using the <i>FileFormat</i> and <i>FileTemplate</i> attributes). See <i>Compression</i> for the indication of compression or encoding of the file. See <i>MimeTypeVersion</i> for the format version.</p> <p>If the file format has a MIME Media Type [iana-mt] registered with IANA, that value MUST be used. The [RFC2046] defines that MIME Media Types are case-insensitive.</p> <p>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 MUST be used.</p> <p>See "FileSpec Attribute Examples for MimeType and MimeTypeVersion Attributes" on page 729 for examples in common use by JDF applications.</p>
MimeTypeVersion ? New in JDF 1.2	string	<p>The level or version of the file format identified by <i>MimeType</i>, whether the value of <i>MimeType</i> is a MIME Media Type or a file type value defined by the JDF spec. Example values include:</p> <p>"PDF/1.3", "PDF/1.4" and "PDF/X-1a:2001" for <i>MimeType</i> = "application/pdf"</p> <p>"TIFF-IT/FP:1998", "TIFF-IT/CT:1998" and "TIFF-IT/LW/P1:1998" for <i>MimeType</i> = "TIFF/IT"</p> <p>See "FileSpec Attribute Examples for MimeType and MimeTypeVersion Attributes" on page 729 for examples in common use by JDF applications.</p>
OSVersion ? Modified in JDF 1.2	string	<p>Version of the operating system specified by <i>AppOS</i>. The IANA Registry provides a list. See "AppOS and OSVersion Attributes" on page 739, for combinations of <i>AppOS</i> and <i>OSVersion</i> values.</p>
OverwritePolicy ? New in JDF 1.2	enumeration	<p>Policy that specifies the policy to follow when a file already exists and the FileSpec is used as an output resource. One of:</p> <p><i>Overwrite</i> – Overwrite the old file.</p> <p><i>RenameNew</i> – Rename the new file.</p> <p><i>RenameOld</i> – Rename the old file.</p> <p><i>NewVersion</i> – Create a new file version. Only valid when the FileSpec references a file on a version aware file system.</p> <p><i>OperatorIntervention</i> – Present a dialog to an operator.</p> <p><i>Abort</i> – Abort the process without modifying the old file.</p>
PageOrder ?	enumeration	<p>Indicates the order of pages in the file containing pages. Possible values are:</p> <p><i>Ascending</i> – The first page in the file is the lowest numbered page.</p> <p><i>Descending</i> – The first page in the file is the highest numbered page.</p>
Password ? New in JDF 1.3	string	<p>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.</p>

Table 7-176: FileSpec resource (Section 5 of 6)

Name	Data Type	Description
<i>RequestQuality</i> ? New in JDF 1.3	double	<p><i>RequestQuality</i> specifies a requested quality of the encoded data when reading image data with selected <i>MimeType</i> values which support variable quality. <i>RequestQuality</i> is ignored when the FileSpec is referenced from an output resource or the FileSpec does not reference image data which support variable quality.</p> <p>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 <i>MimeType</i> values:</p> <p><i>image/jp2</i>, <i>image/jpx</i> – The value represents the ratio of the encoding bitrate of the maximum bitrate layer encoded in the file.</p> <p><i>image/gif</i> – (Note: Only interleaved GIF) The number represents a ratio of the total interleaved layers of the file.</p> <p><i>image/tiff</i> – (Note: Only pyramid TIFF) The number represents the ratio of the total resolution of the complete image.</p>
<i>ResourceUsage</i> ?	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 .
<i>SearchDepth</i> ? New in JDF 1.2	integer	Used when <i>ResourceUsage</i> = “ <i>SearchPath</i> ” to specify the maximum directory depth that will be recursively searched. 0 specifies this directory only, <i>INF</i> specifies an unlimited search.
<i>UID</i> ? New in JDF 1.1	string	<p>Unique internal ID of the referenced file. This attribute is dependent on the type of file that is referenced:</p> <p>PDF – Variable unique identifier in the ID field of the PDF file’s trailer.</p> <p>ICC Profile – The Profile ID in bytes 84-99 of the ICC profile header.</p> <p>Others – Format specific.</p> <p>If neither <i>URL</i> nor <i>UID</i> is present on an input FileSpec, and neither <i>FileFormat</i> nor <i>FileTemplate</i> is present, the referencing resource MUST be a pipe. If either <i>URL</i> or <i>UID</i> is specified, then <i>FileFormat</i> and <i>FileTemplate</i> MUST NOT be specified.</p>

Table 7-176: FileSpec resource (Section 6 of 6)

Name	Data Type	Description
<i>URL</i> ?	URL	Location of the file specified as either an Absolute URI or a Relative URI. If neither <i>URL</i> nor <i>UID</i> is present on an input FileSpec , and neither <i>FileFormat</i> nor <i>FileTemplate</i> is present, the referencing resource MUST be a pipe. If either <i>URL</i> or <i>UID</i> is specified, then <i>FileFormat</i> and <i>FileTemplate</i> MUST NOT be specified. If <i>URL</i> is not specified in an output resource, the system-specified location will be assumed, but this value MUST 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 <i>Compression</i> attribute without the <i>URL</i> attribute. See [RFC3986] and "Resolving RunList/@Directory and FileSpec/@URL URI references" on page 745 and "FileSpec Attributes and Container Subelement" on page 735 for the syntax and examples. For the "file:" URL scheme see also [RFC1738] and [FileURL].
<i>UserFileName</i> ?	string	A user-friendly name which can be used to identify the file.
<i>Container</i> ? New in JDF 1.2	element	Specifies the container for this file. When a container FileSpec is pointed to by <i>Container</i> , that FileSpec MUST NOT also specify <i>FileFormat</i> and <i>FileTemplate</i> 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 "Resolving RunList/@Directory and FileSpec/@URL URI references" on page 745 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 .
<i>FileAlias</i> *	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).

— Element: 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 7-177: 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 "Resolving RunList/@Directory and FileSpec/@URL URI references" on page 745 for details.

— Element: FileAlias

Table 7-178: FileAlias element

Name	Data Type	Description
<i>Alias</i>	string	The filename which is expected to occur in the file.
<i>Disposition</i> ? Deprecated in JDF 1.2	enumeration	Indicates what the device is to do with the file referenced by this alias when the process that uses this resource as an input resource completes. Possible values are: <i>Unlink</i> – The device is to release the file. <i>Delete</i> – The device is to attempt to delete the file. <i>Retain</i> – The device is to do nothing with the file. In JDF/1.2 and beyond, use FileSpec/Disposition .
<i>MimeType</i> ? Deprecated in JDF 1.2	string	MIME type of the file. In JDF/1.2 and beyond, use FileSpec/@MimeType .
<i>RawAlias</i> ? 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.
<i>URL</i> ? 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 MUST be present, and MUST contain a <i>URL</i> attribute. FileSpec MAY contain additional properties of the file, (e.g., Disposition , <i>MimeType</i> , <i>MimeTypeVersion</i> , etc.).

7.2.76 FitPolicy

[New in JDF 1.1](#)

This resource specifies how to fit content into a receiving container e.g., a **RunList** entry into a **PlacedObject** or content into either a PageCell or a PageCell grid in a *SurfaceContentsBox*. See the description of each reference to **FitPolicy** to determine what the context-specific "content" is and what the "receiving container" is.

Resource Properties

Resource class: Parameter

Resource referenced by: **ImageSetterParams, InterpretingParams, LayoutPreparationParams, LayoutPreparationParams/PageCell, RasterReadingParams**

Example Partition: —

Input of processes: —

Output of processes: —

Table 7-179: FitPolicy resource (Section 1 of 2)

Name	Data Type	Description
<i>ClipOffset</i> ?	XYPair	Defines the offset (position) of the imaged area in the non-rotated source image when <i>SizePolicy</i> is <i>ClipToMaxPage</i> . 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 <i>PageCell/@ImageShift</i> is specified.

Table 7-179: FitPolicy resource (Section 2 of 2)

Name	Data Type	Description
<i>GutterPolicy</i> = "Fixed"	enumeration	Allows printing of NUp grids even if the media size does not match the requirements of the data. One of: <i>Distribute</i> – The gutters can grow or shrink to the value specified in <i>MinGutter</i> . <i>Fixed</i> – The gutters are fixed.
<i>MinGutter</i> ?	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.
<i>RotatePolicy</i> ?	enumeration	Specifies the policy for the device to automatically rotate the content to optimize the fit of the content to the receiving container. <i>NoRotate</i> – Do not rotate. <i>RotateOrthogonal</i> – Rotate by 90° in either direction. <i>RotateClockwise</i> – Rotate clockwise by 90°. <i>RotateCounterClockwise</i> – Rotate counterclockwise by 90°.
<i>SizePolicy</i> ? Modified in JDF 1.1A	enumeration	Allows printing even if the container size does not match the requirements of the data. <i>ClipToMaxPage</i> – 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 <i>ClipOffset</i> key is given, or from that position which is determined by <i>ClipOffset</i> . <i>Abort</i> – Emit an error and abort printing. <i>FitToPage</i> – The page contents are to be scaled up or down to fit the container. The aspect ratio is maintained. <i>ReduceToFit</i> – The page contents are to be scaled down but not scaled up to fit the container. The aspect ratio is maintained. <i>Tile</i> – the page contents are to be split into several tiles, each printed on its own surface.

7.2.77 Fold

[New in JDF 1.1](#)

Fold describes an individual folding operation of the **Component**.

Resource Properties

Resource class:	Parameter
Resource referenced by:	FoldingIntent, FoldingParams
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-180: Fold resource (Section 1 of 2)

Name	Data Type	Description
<i>From</i>	enumeration	Edge from which the page is folded. Possible values are: <i>Front</i> <i>Left</i>

Table 7-180: Fold resource (Section 2 of 2)

Name	Data Type	Description
<i>To</i>	enumeration	Direction in which it is folded. Possible values are: <i>Up</i> – Upwards; corresponds to a valley fold with the left/bottom side coming over the opposite side. <i>Down</i> – Downwards; corresponds to a mountain or peak fold with the left/bottom side coming under the opposite side.
<i>Travel</i> ? Modified in JDF 1.2	double	Distance of the reference edge relative to <i>From</i> . If both <i>Travel</i> and <i>RelativeTravel</i> are specified, <i>RelativeTravel</i> is ignored. At least one of <i>Travel</i> or <i>RelativeTravel</i> MUST be specified.
<i>RelativeTravel</i> ? New in JDF 1.2	double	Relative distance of the reference edge relative to <i>From</i> in the coordinates of the incoming Component . The <i>RelativeTravel</i> 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 <i>Travel</i> or <i>RelativeTravel</i> MUST be specified.

7.2.78 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 is moved to the lower left corner of the intermediate folding product. For details see "Product Example: Simple Brochure" on page 27.

The specification of 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 7-19, below.

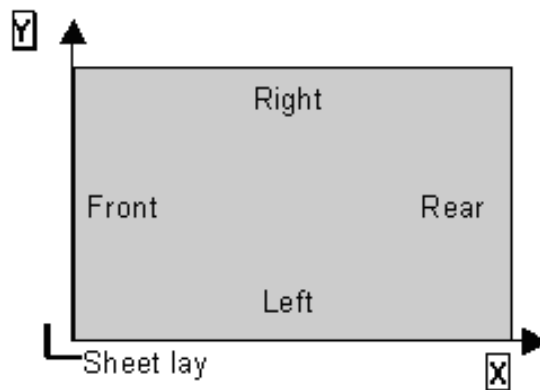


Figure 7-19: Names of the reference edges of a sheet in the FoldingParams resource

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>BlockName</i> , <i>RibbonName</i> , <i>SheetName</i> , <i>SignatureName</i> , <i>WebName</i>
Input of processes:	Folding
Output of processes:	—

Table 7-181: FoldingParams resource

Name	Data Type	Description
DescriptionType ? Deprecated in JDF 1.2	enumeration	How the folding operations are described. Possible values are: <i>FoldProc</i> – Detailed description of each individual fold. <i>FoldCatalog</i> – Selection of fold procedure from <i>FoldCatalog</i> . In JDF 1.2 and beyond, the <i>FoldCatalog</i> defines the topology of the folding scheme. The specifics of each individual Fold can be described using Fold elements. If both <i>FoldCatalog</i> and Fold are specified, Fold takes precedence
FoldCatalog ?	string	Describes the type of fold according to the folding catalog in the format “ <i>Fx-y</i> ” as shown in Table 7-20 and Table 7-21 on page 441. Imposition folds define finished pages. Thus, a sheet with a “ <i>F6-2</i> ” Z-fold is comprised of six (6) finished pages.
FoldSheetIn ? Deprecated in JDF 1.1	XYPair	Input sheet format. If the specified size does not match the size of the <i>X</i> and <i>Y</i> dimensions of the input Component , all coordinates of the folding procedure are scaled accordingly. The scaling factors in <i>X</i> and <i>Y</i> direction MAY differ. Implementation Note: This attribute SHOULD always match the <i>Size</i> attribute of the input Component , which is the default.
SheetLay = “Left”	enumeration	Lay of input media. Possible values are: <i>Left</i> <i>Right</i> Note that <i>SheetLay</i> does not modify the coordinate references of the Folding process.
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 <i>FoldCatalog</i> and Fold elements are specified, the Fold elements have precedence, and the <i>FoldCatalog</i> 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 “ <i>F4-1</i> ”.
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 element Fold * in JDF 1.1 and beyond.

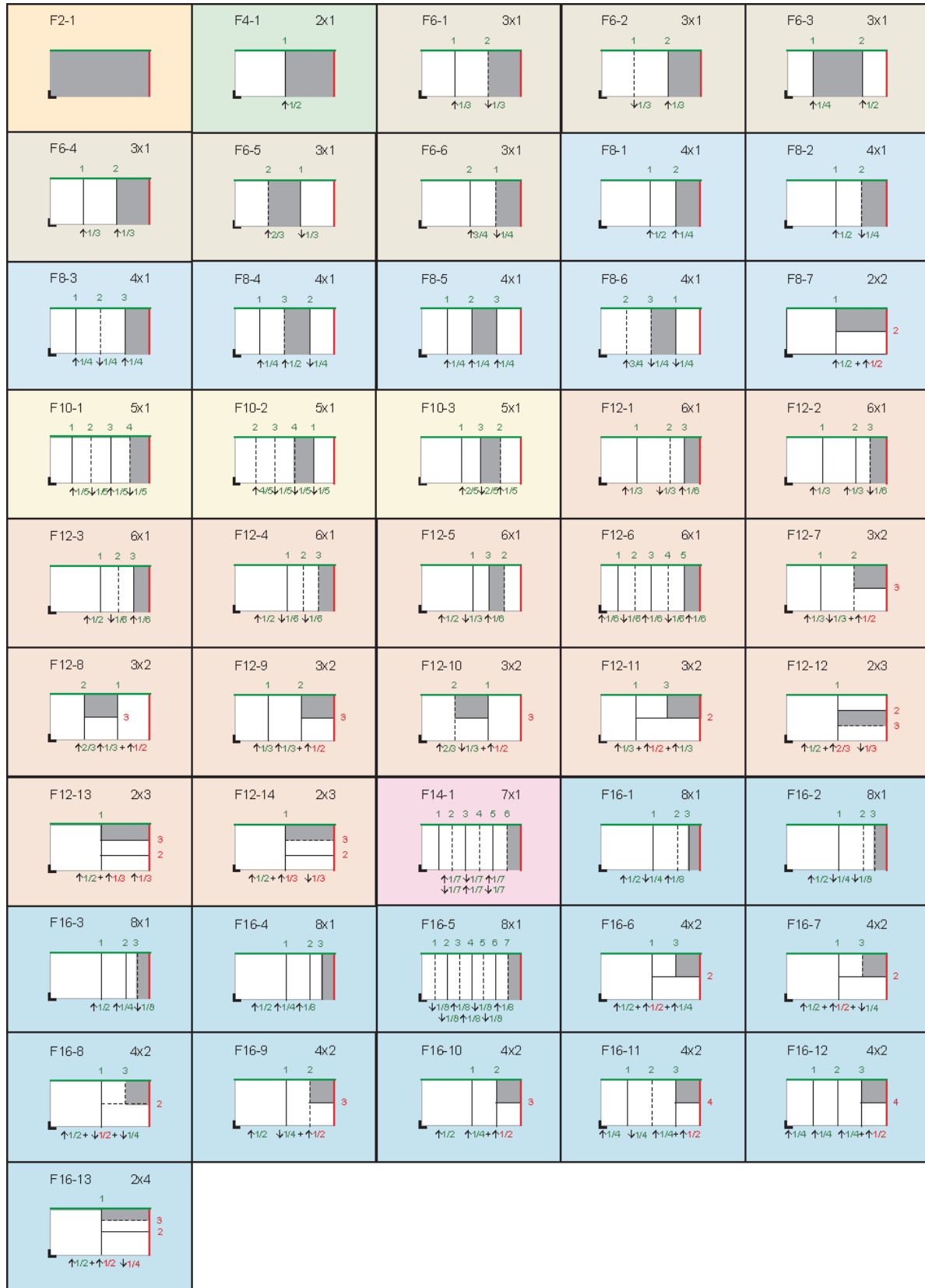


Figure 7-20: Fold catalog part 1

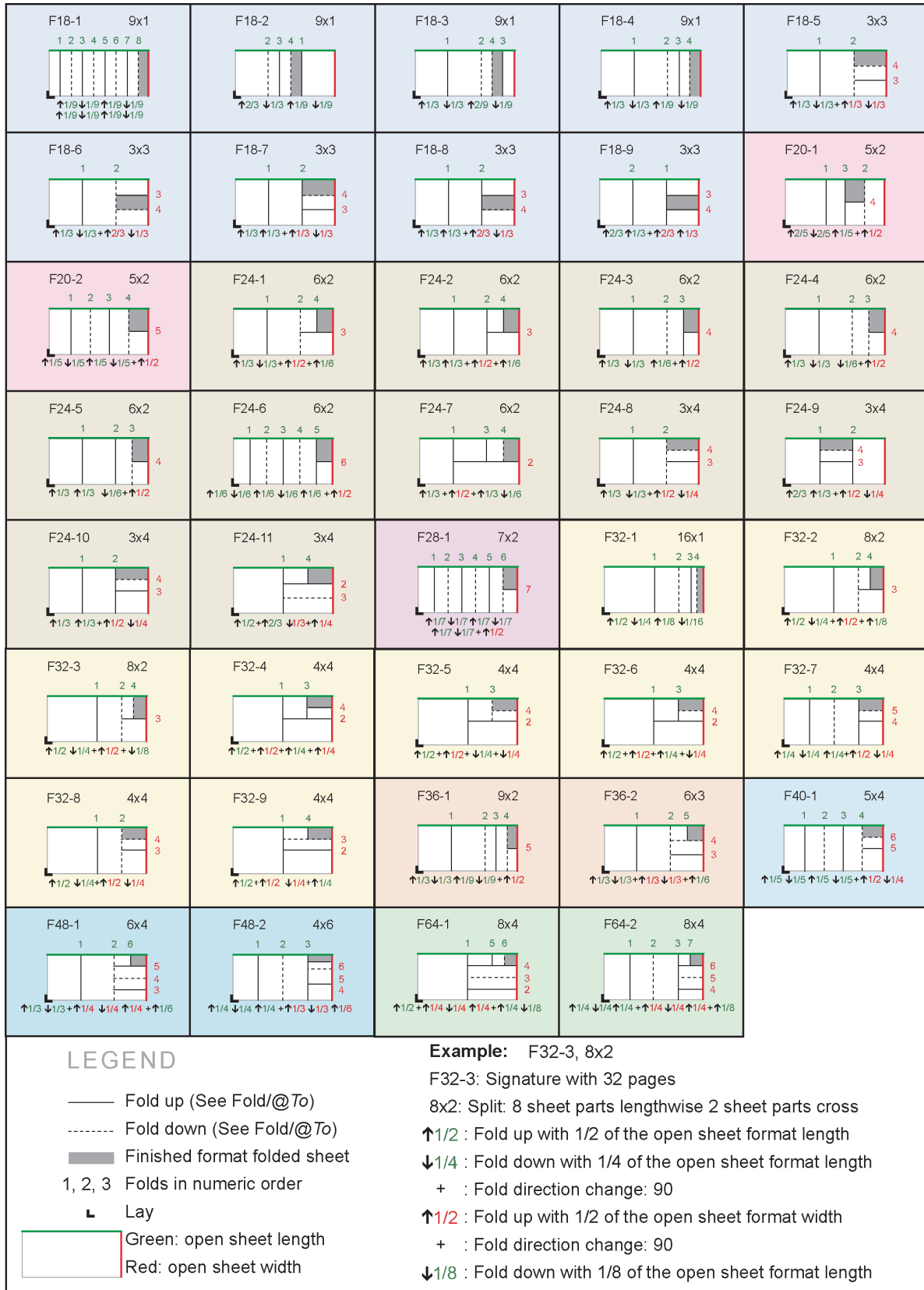


Figure 7-21: Fold catalog part 2

7.2.79 FontParams

This resource describes how fonts are handled when converting PostScript or other PDL files to PDF.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName</i>
Input of processes:	PSToPDFConversionParams
Output of processes:	—

Table 7-182: FontParams resource

Name	Data Type	Description
<i>AlwaysEmbed</i> ?	NMTOKENS	One or more names of fonts that are always to be embedded in the PDF file. Each name MUST be the PostScript language name of the font. An entry that occurs in both the <i>AlwaysEmbed</i> and <i>NeverEmbed</i> lists constitutes an error.
<i>CannotEmbedFontPolicy</i> = "Warning"	enumeration	Determines what occurs when a font cannot be embedded. Possible values are: <i>Error</i> – Log an error and abort the process if any font can not be found or embedded. <i>Warning</i> – Warn and continue if any font cannot be found or embedded. <i>OK</i> – Continue without warning or error if any font can not be found or embedded.
<i>EmbedAllFonts</i> = "false"	boolean	If "true", specifies that all fonts, except those in the <i>NeverEmbed</i> list, are to be embedded in the PDF file.
<i>MaxSubsetPct</i> ?	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 <i>SubsetFonts</i> = "true".
<i>NeverEmbed</i> ?	NMTOKENS	One or more names of fonts that are never to be embedded in the PDF file. Each name MUST be the PostScript language name of the font. An entry that occurs in both the <i>AlwaysEmbed</i> and <i>NeverEmbed</i> lists constitutes an error.
<i>SubsetFonts</i> ?	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 <i>MaxSubsetPct</i> . Note: Embedded instances of multiple master fonts are always subsetted, regardless of the setting of <i>SubsetFonts</i> .

7.2.80 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 MUST 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
Resource referenced by:	—
Example Partition:	<i>DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName</i>
Input of processes:	<i>IDPrinting, Interpreting, Trapping</i>
Output of processes:	—

Table 7-183: FontPolicy resource

Name	Data Type	Description
<i>PreferredFont</i>	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.
<i>UseDefaultFont</i>	boolean	If <i>true</i> , the device MUST 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.
<i>UseFontEmulation</i>	boolean	If <i>true</i> , the device MUST emulate a requested font if a font cannot be found.

7.2.81 FormatConversionParams

[New in JDF 1.1](#)

This resource defines the parameters needed for generic **FormatConversion** of digital files.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>DocIndex, RunIndex, RunTags</i>
Input of processes:	<i>FormatConversion</i>
Output of processes:	—

Table 7-184: FormatConversionParams resource (Section 1 of 2)

Name	Data Type	Description
FileSpec (<i>InputFormat</i>)? Deprecated in JDF 1.2	refelement	The format of the original file is specified in a FileSpec with <i>ResourceUsage = InputFormat</i> . 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.

Table 7-184: FormatConversionParams resource (Section 2 of 2)

Name	Data Type	Description
FileSpec (<i>OutputFormat</i>)? Deprecated in JDF 1.2	refelement	The format of the converted file is specified in a FileSpec with <i>ResourceUsage</i> = <i>OutputFormat</i> . 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 .
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.
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
ColorPool ? New in JDF 1.2	refelement	Additional detail about the colors used in the file to be converted.

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**.

— Element: **TIFFFormatParams**

[New in JDF 1.2](#)

Table 7-185: TIFFFormatParams element (Section 1 of 2)

Name	Data Type	Description
<i>ByteOrder</i> ?	enumeration	Byte order of the TIFF file. Possible values are: <i>II</i> – Low byte first. <i>MM</i> – high byte first. The identifiers have been selected to match the identifier with the same purpose within the TIFF file itself.
<i>Interleaving</i> = "1"	integer	How the components of each pixel are stored. The values are taken from TIFF tag 284— <i>PlanarConfiguration</i> : <i>1</i> – “Chunky” format, which is pixel interleaved. <i>2</i> – “Planar” format, which is strip interleaved.
<i>WhiteIsZero</i> = "true"	boolean	When writing monochrome or grayscale files, this flag indicates whether the data is to be written as “WhiteIsZero” or “BlackIsZero.”

Table 7-185: TIFFFormatParams element (Section 2 of 2)

Name	Data Type	Description
<i>Segmentation</i> ?	enumeration	How the image data are segmented. Possible values are: <i>SingleStrip</i> —all data are included in one segment. This is encoded in the TIFF file by setting <i>RowsPerStrip</i> to a number equal to or larger than the number of pixel rows in the image. <i>Stripped</i> —Data are segmented into strips. <i>Tiled</i> —Data are segmented into tiles.
<i>RowsPerStrip</i> ?	integer	The number of image scan lines per strip, encoded in the TIFF file as <i>RowsPerStrip</i> . This attribute is ignored if <i>Segmentation</i> = " <i>Stripped</i> ". 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.
<i>TileSize</i> ?	XYPair	Two integers. The X value provides width of tiles, and the Y value provides height of tiles. This attribute is ignored if <i>Segmentation</i> is not <i>Tiled</i> .
<i>SeparationNameTag</i> = "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 sub element, then the TIFFtag element takes priority over <i>SeparationNameTag</i> . The tag SHOULD only be put in the resulting TIFF files if the name of the separation is known, (e.g., from a ColorPool resource supplied to FormatConversion , or because the FormatConversion process forms a part of a compound process with a Separation process). The default of "270" is the <i>TIFF</i> ImageDescription tag.
TIFFtag *	element	Specific tag values for inclusion in the TIFF file.
TIFFEmbeddedFile *	element	Files to be embedded within the created TIFF file. These might include an ICC profile, XMP data, etc.

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 supplied to **FormatConversion**.

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.

— Element: TIFFtag

[New in JDF 1.2](#)

Table 7-186: TIFFtag element (Section 1 of 2)

Name	Data Type	Description
<i>BinaryValue</i> ?	hexBinary	If the type of the tag is UNDEFINED, then <i>BinaryValue</i> is used to encode the data

Table 7-186: TIFFtag element (Section 2 of 2)

Name	Data Type	Description
<i>IntegerValue</i> ?	IntegerList	If the type of the tag is BYTE, SHORT, LONG, SBYTE, SSHORT or SLONG, then <i>IntegerValue</i> is used to encode that data
<i>NumberValue</i> ?	DoubleList	If the type of the tag is RATIONAL, SRATIONAL, FLOAT or DOUBLE, then <i>NumberValue</i> is used to encode that data
<i>StringValue</i> ?	string	If the type of the tag is ASCII, then <i>StringValue</i> is used to encode the data.
<i>TagNumber</i>	integer	Tag number of the specified tag, (e.g., 270 (decimal) for ImageDescription).
<i>TagType</i>	integer	The type of the tag as defined in [TIFF6] (1 = BYTE, 2 = SHORT, etc.)

Exactly one of *IntegerValue*, *NumberValue*, *StringValue* or *BinaryValue* MUST be present, depending on the type of the TIFF tag to be carried. TIFFtag elements MUST 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.

— Element: TIFFEmbeddedFile

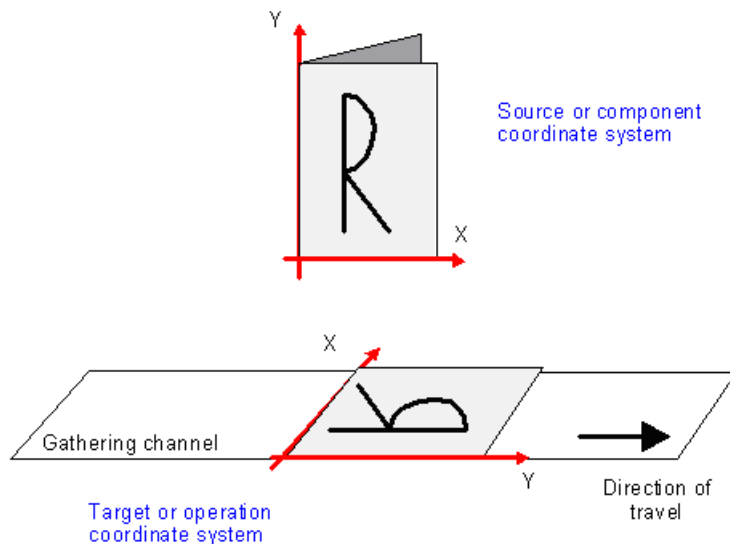
[New in JDF 1.2](#)

Table 7-187: TIFFEmbeddedFile element

Name	Data Type	Description
<i>TagNumber</i>	integer	Tag number of the specified tag, (e.g., 34675 (decimal) for an ICC profile or 700 for XMP).
<i>TagType</i>	integer	The type of the tag as defined in [TIFF6]. This will usually be 1 (BYTE) or 7 (UNDEFINED).
FileSpec	reference	Reference to the file to be embedded.

7.2.82 GatheringParams

This resource contains the attributes of the **Gathering** process.

**Figure 7-22: Coordinate system used for gathering**

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	Gathering
Output of processes:	—

Table 7-188: GatheringParams resource

Name	Data Type	Description
Disjointing ?	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.

7.2.83 GeneralID

[New in JDF 1.3](#)

GeneralID describes a generic identifier. The name or usage of the identifier is specified in **GeneralID/@IDUsage** and the specific value of the identifier is specified in **GeneralID/@IDValue**.

Resource Properties

Resource class:	ResourceElement
Resource referenced by:	All Resources
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-189: GeneralID element

Name	Data Type	Description
<i>IDUsage</i>	NMTOKEN	Usage of the GeneralID . There are no predefined values in JDF. The following values are defined in [AdsML]. <i>AdsML:AdBuyer_BookingTransactionID</i> —an ID for the booking transaction that was assigned by a party acting on behalf of the advertiser <i>AdsML:AdSeller_BookingTransactionID</i> —an ID for the booking transaction that was assigned by the publisher or a party acting on its behalf <i>AdsML:AdBuyer_AdMaterialID</i> —an ID for the artwork that was assigned by a party acting on behalf of the advertiser <i>AdsML:AdSeller_AdMaterialID</i> —an ID for the artwork that was assigned by the publisher or a party acting on its behalf
<i>IDValue</i>	string	Value of the GeneralID .

7.2.84 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:	CoverApplicationParams, GluingParams, SpineTapingParams
Input of processes:	—
Output of processes:	—

Table 7-190: GlueApplication resource

Name	Data Type	Description
<i>GluingTechnique</i>	enumeration	Type or technique of gluing application. Possible values are: <i>SpineGluing</i> <i>SideGluingFront</i> <i>SideGluingBack</i>
GlueLine	reference	Structure of the glue line.

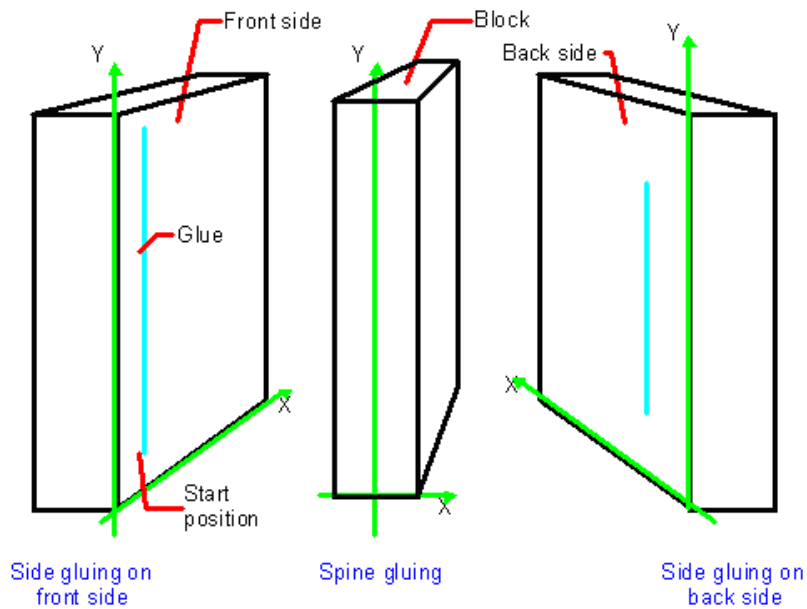


Figure 7-23: Parameters and coordinate system for glue application

7.2.85 GlueLine

This resource provides the information to determine where and how to apply glue.

Resource Properties

- Resource class: Parameter
- Resource referenced by: **CaseMakingParams, EndSheetGluingParams, FoldingParams, CoverApplicationParams, InsertingParams, SpineTapingParams, ThreadSealingParams**
- Example Partition: —
- Input of processes: —
- Output of processes: —

Table 7-191: GlueLine resource (Section 1 of 2)

Name	Data Type	Description
<i>AreaGlue = "false"</i> New in JDF 1.1	boolean	Specifies that this GlueLine is to cover the complete width of the Component it is applied to.
<i>GlueBrand ?</i>	string	Glue brand.

Table 7-191: GlueLine resource (Section 2 of 2)

Name	Data Type	Description
<i>GlueLineWidth</i> ?	double	Width of the glue line. Note that in extreme cases, the glue line could cover the input component over the whole width.
<i>GlueType</i> ?	enumeration	Glue type. Possible values are: <i>ColdGlue</i> – Any type of glue that needs no heat treatment. <i>Hotmelt</i> – Hotmelt EVA (Ethyl-Vinyl-Acetate-Copolymere) <i>PUR</i> – Polyurethane
<i>GluingPattern</i> ? 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). <i>GluingPattern</i> MUST contain an even number of entries. If the total length of <i>GluingPattern</i> is less than <i>WorkingPath</i> or the length implied by <i>RelativeWorkingPath</i> , the pattern restarts after the last gap. If the total length of <i>GluingPattern</i> is larger than <i>WorkingPath</i> or the length implied by <i>RelativeWorkingPath</i> , the pattern is clipped at the end.
<i>MeltingTemperature</i> ?	integer	Temperature needed for melting the glue, in degrees centigrade. Used only when <i>GlueType</i> = "Hotmelt" or <i>GlueType</i> = "PUR".
<i>RelativeStartPosition</i> ? New in JDF 1.2	XYPair	Relative starting position of the tool. The <i>RelativeStartPosition</i> 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 .
<i>RelativeWorkingPath</i> ? New in JDF 1.2	XYPair	Relative working path of the tool beginning at <i>RelativeStartPosition</i> . The <i>RelativeWorkingPath</i> 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 .
<i>StartPosition</i> ? 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 <i>StartPosition</i> and <i>RelativeStartPosition</i> are specified, <i>RelativeStartPosition</i> is ignored.
<i>WorkingPath</i> ? Modified in JDF 1.2	XYPair	Relative working path of the gluing tool. If both <i>WorkingPath</i> and <i>RelativeWorkingPath</i> are specified, <i>RelativeWorkingPath</i> is ignored.

7.2.86 GluingParams

[New in JDF 1.1](#)

GluingParams define the parameters applying a generic line of glue to a component.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>ProductionRun, WebName, WebProduct</i>
Input of processes:	Gluing
Output of processes:	—

Table 7-192: GluingParams resource

Name	Data Type	Description
<i>GluingProductionID</i> ? New in JDF 1.3	string	Defines a gluing scheme for production.
Glue *	element	Definition of one or more Glue line applications.

— Element: Glue

The Glue element describes how to apply a line of glue.

Table 7-193: Glue element

Name	Data Type	Description
<i>WorkingDirection</i>	enumeration	Direction from which the tool is working. Possible values are: <i>Top</i> – From above. <i>Bottom</i> – From below.
GlueApplication ? Modified in JDF 1.3	refelement	Describes the glue application. Exactly one of GlueApplication or GlueLine MUST be specified.
GlueLine ? New in JDF 1.3	refelement	Structure of the GlueLine used for generic gluing. Exactly one of GlueApplication or GlueLine MUST be specified.

7.2.87 HeadBandApplicationParams

[New in JDF 1.1](#)

This resource specifies how to apply headbands in hard cover book production.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	HeadBandApplication
Output of processes:	—

Table 7-194: HeadBandApplicationParams resource (Section 1 of 2)

Name	Data Type	Description
<i>BottomBrand</i> ?	string	Bottom head band brand. If not specified, defaults to the value of <i>TopBrand</i> .
<i>BottomColor</i> ?	NamedColor	Color of the bottom head band. If not specified, defaults to the value of <i>TopColor</i> .
<i>BottomLength</i> ?	double	Length of the carrier material of the bottom head band along binding edge. If not specified, both head bands are on one carrier.
<i>TopBrand</i> ?	string	Top head band brand.
<i>TopColor</i> ?	NamedColor	Color of the top head band.
<i>TopLength</i> ?	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.

Table 7-194: HeadBandApplicationParams resource (Section 2 of 2)

Name	Data Type	Description
<i>StripMaterial</i> ?	enumeration	Strip material. Possible values are: <i>Calico</i> <i>Cardboard</i> <i>CrepePaper</i> <i>Gauze</i> <i>Paper</i> <i>PaperlinedMules</i> <i>Tape</i>
<i>Width</i> ?	double	Width of the head bands and carrier.
GlueLine *	refelement	The carrier can be applied to the book block with glue. The coordinate system for the GlueLine is defined in the Section 7.2.71, EndSheetGluingParams.

7.2.88 Hole

The Hole element describes an individual hole.

Resource Properties

Resource class:	Parameter
Resource referenced by:	HoleLine, HoleMakingIntent, HoleMakingParams
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-195: Hole resource

Name	Data Type	Description
<i>Center</i>	XYPair	Position of the center of the hole relative to the Component coordinate system. For more information, see Section 6.7.2, HoleMaking.
<i>Extent</i>	XYPair	Size (Bounding Box) of the hole, in points. If <i>Shape</i> is <i>Round</i> , only the first entry of <i>Extent</i> is evaluated and defines the hole diameter.
<i>Shape</i> Modified in JDF 1.1	enumeration	Shape of the hole. Possible values are: <i>Elliptic</i> <i>Round</i> <i>Rectangular</i>

7.2.89 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 element. The distance between holes within each line of holes is identical (constant pitch).

Resource Properties

Resource class:	Parameter
Resource referenced by:	HoleMakingIntent, HoleMakingParams
Example Partition:	—

Input of processes: —

Output of processes: —

Table 7-196: HoleLine resource

Name	Data Type	Description
<i>Pitch</i>	double	Center-hole to center-hole distance within a line of holes.
Hole	element	Size and position of the first hole in the HoleLine.

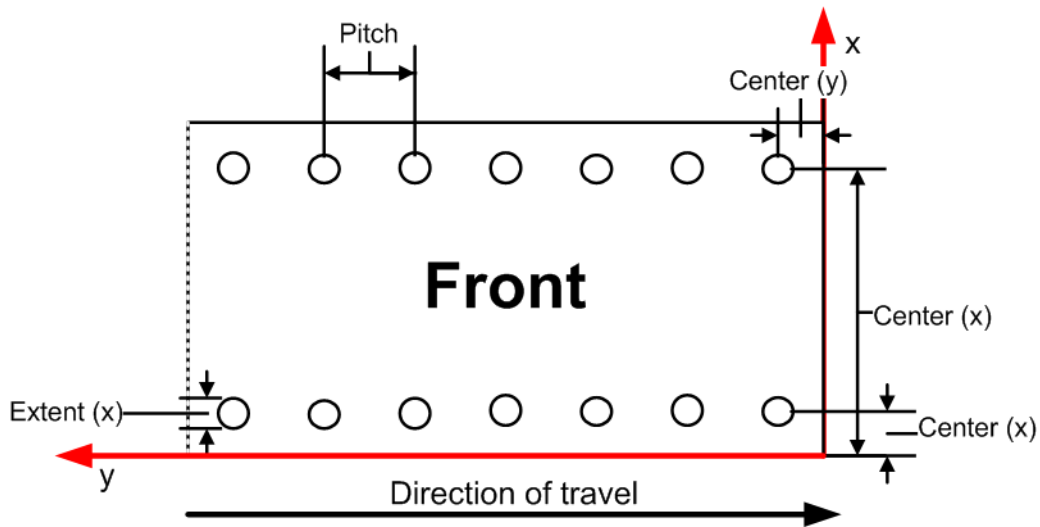


Figure 7-24: Hole line parameters

However, sometimes Line Hole Punching is performed for multiple webs before dividing the web after the HoleMaking process as illustrated in Figure 7-25 below:

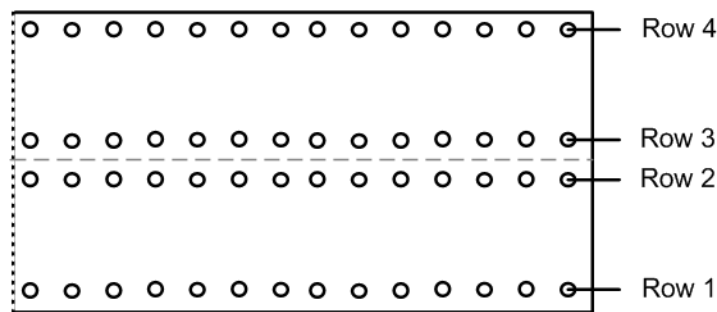


Figure 7-25: Line hole punching for multiple webs

7.2.90 HoleList

This resource is used to describe holes or rows of holes in intent resources. Note that it was an intent resource sub element prior to JDF 1.2.

Resource Properties

Resource class: Parameter

Resource referenced by: **BindingIntent, HoleMakingIntent**

Example Partition: —

Input of processes: —

Output of processes: —

Table 7-197: HoleList resource

Name	Data Type	Description
Hole * Modified in JDF 1.1	refelement	Description of individual holes. See Section 7.2.88, Hole.
HoleLine * New in JDF 1.1	refelement	Array of all HoleLine elements. See Section 7.2.89, HoleLine.

7.2.91 HoleMakingParams

This resource specifies where to make a hole of what shape in components. This information is used by the **HoleMaking** process.

Resource Properties

Resource class: Parameter

Resource referenced by: —

Example Partition: *SheetName, SignatureName*Input of processes: **HoleMaking**

Output of processes: —

Table 7-198: HoleMakingParams resource (Section 1 of 2)

Name	Data Type	Description
<i>Center</i> ? Modified in JDF 1.1	XYPair	Position of the center of the hole pattern relative to the Component coordinate system if <i>HoleType</i> is not <i>Explicit</i> . If not specified, it defaults to the value implied by <i>HoleType</i> .
<i>CenterReference</i> = "TrailingEdge" New in JDF 1.1	enumeration	Defines the reference coordinate system for <i>Center</i> . One of: <i>TrailingEdge</i> – Physical coordinate system of the component. <i>RegistrationMark</i> – The center is relative to a registration mark.
<i>Extent</i> ?	XYPair	Size (Bounding Box) of the hole in points if <i>HoleType</i> is not <i>Explicit</i> . If <i>Shape</i> is <i>Round</i> , only the first entry of <i>Extent</i> is evaluated and defines the hole diameter. If not specified, it defaults to the value implied by <i>HoleType</i> .
<i>HoleCount</i> ? New in JDF 1.2	IntegerList	For patterns with <i>HoleType</i> whose enumeration values begin with a "P", "W" or "C", this parameter specifies the number of consecutive holes and spaces. ^a 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".

Table 7-198: HoleMakingParams resource (Section 2 of 2)

Name	Data Type	Description
HoleReferenceEdge ? New in JDF 1.1 Deprecated in JDF 1.2	enumeration	<p>The edge of the media relative to where the holes are to be punched. Use with <i>HoleType</i>. Possible values are:</p> <p><i>Left</i></p> <p><i>Right</i></p> <p><i>Top</i></p> <p><i>Bottom</i></p> <p><i>Pattern</i> – Specifies that the reference edge implied by the value of <i>HoleType</i> in "JDF/CIP4 Hole Pattern Catalog" on page 761 is used.</p> <p>The default if <i>HoleType</i> is <i>Explicit</i>, otherwise <i>Left</i>.</p> <p><i>HoleReferenceEdge</i> has been replaced with an explicit <i>Transformation</i> or <i>Orientation</i> of the input Component. If either <i>Transformation</i> or <i>Orientation</i> along with <i>HoleReferenceEdge</i> is specified, the result is the matrix product of both transformations. <i>Transformation</i> or <i>Orientation</i> MUST be applied first.</p>
HoleType New in JDF 1.1	enumerations	<p>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). For details of hole types and a list of additional allowed values, refer to "JDF/CIP4 Hole Pattern Catalog" on page 761. Values are:</p> <p><i>Explicit</i> – Holes are defined in an array of Hole elements.</p> <p>Additional values defined in "JDF/CIP4 Hole Pattern Catalog" on page 761</p> <p>The following values are deprecated from JDF 1.0</p> <p><i>2HoleEuro</i> – Replaced by either <i>R2m-DIN</i> or <i>R2m-ISO</i>.</p> <p><i>3HoleUS</i> – Replaced by <i>R3I-US</i>.</p> <p><i>4HoleEuro</i> – Replaced by either <i>R4m-DIN-A4</i> or <i>R4m-DIN-A5</i>.</p>
Shape ? Modified in JDF 1.1	enumeration	<p>Shape of the holes if <i>HoleType</i> is not <i>Explicit</i>. Possible values are:</p> <p><i>Elliptic</i></p> <p><i>Round</i></p> <p><i>Rectangular</i></p> <p>If not specified, it defaults to the value implied by <i>HoleType</i>.</p>
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 <i>HoleMaking</i> .

- a. **Application Note:** For dealing with the “default” case, 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.

7.2.92 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:	Disjointing, Layout, LayoutElementProductionParams and any physical resource
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-199: IdentificationField resource (Section 1 of 2)

Name	Data Type	Description
<i>BoundingBox</i> ?	rectangle	Box that provides the boundaries in the coordinate system of the mark that indicates where the IdentificationField is placed. If no <i>BoundingBox</i> is defined, the complete visible surface MUST be scanned for an appropriate bar code. The <i>BoundingBox</i> is used only as metadata when searching or scanning IdentificationField elements and not used when generating IdentificationField elements in a <i>LayoutElementProduction</i> process.
<i>Encoding</i> Modified in JDF 1.3	enumeration	Encoding of the information. Possible values are: <i>ASCII</i> – Plain-text font. <i>Barcode</i> – Any bar code. New in JDF 1.3 <i>BarCode1D</i> – One-dimensional bar code. Deprecated in JDF 1.3 <i>BarCode2D</i> – Two-dimensional bar code. Deprecated in JDF 1.3 <i>RFID</i> – Radio Frequency Identification tag. New in JDF 1.3
<i>EncodingDetails</i>	NMTOKEN	Details about the encoding type. An example is the bar code scheme. See "EncodingDetails attribute – possible values" on page 457 for values of <i>EncodingDetails</i> .
<i>Format</i> ? 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 <i>Format</i> , <i>Value</i> or the pair <i>ValueFormat</i> and <i>ValueTemplate</i> MUST be specified.
<i>Orientation</i> ?	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 <i>Orientation</i> is used only as metadata when searching or scanning IdentificationField elements and not used when generating IdentificationField elements in a <i>LayoutElementProduction</i> process.

Table 7-199: IdentificationField resource (Section 2 of 2)

Name	Data Type	Description
<i>Position ?</i>	enumeration	Position with respect to the instance document or physical resource to which the resource refers. Possible values are: <i>Header</i> – Sheet before the document. <i>Trailer</i> – Sheet after the document. <i>Page</i> – A page of the document. <i>Top</i> – The top of the resource. <i>Bottom</i> – The bottom of the resource. <i>Left</i> – The left side of the resource. <i>Right</i> – The right side of the resource. <i>Front</i> – The front side of the resource. <i>Back</i> – The back side of the resource. <i>Any</i> – Deprecated in JDF 1.2
<i>Page ?</i>	integer	If <i>Position = Page</i> , 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.
<i>Purpose ?</i>	enumeration	Purpose defines the usage of the field. Possible values are: <i>Label</i> – Used to mark a product or component. <i>Separation</i> – Used to separate documents. <i>Verification</i> – Used for verification of documents.
<i>PurposeDetails ?</i> New in JDF 1.3	NMTOKEN	More detail about the usage of the barcode. Values include: <i>ProductIdentification</i> – End product identification e.g., scanning in the super market.
<i>Value ?</i> New in JDF 1.1	string	Fixed value of the IdentificationField , (e.g., on a label). Exactly one of <i>Format</i> , <i>Value</i> or the pair <i>ValueFormat</i> and <i>ValueTemplate</i> MUST be specified.
<i>ValueFormat ?</i> New in JDF 1.3	string	A formatting string used with <i>ValueTemplate</i> to define fixed and/or variable content of barcodes or text. For more information, see "Generating strings with Format and Template" on page 743. Exactly one of <i>Format</i> , <i>Value</i> or the pair <i>ValueFormat</i> and <i>ValueTemplate</i> MUST be specified.
<i>ValueTemplate ?</i> New in JDF 1.3	string	A list of values used with <i>ValueTemplate</i> to define fixed and/or variable content of barcodes or text. For more information, see "Generating strings with Format and Template" on page 743. Exactly one of <i>Format</i> , <i>Value</i> or the pair <i>ValueFormat</i> and <i>ValueTemplate</i> MUST 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 .

— Attribute: EncodingDetails

Table 7-200: EncodingDetails attribute – possible values

Value	Description	Value	Description
<i>BOBST</i>		<i>ITF_6</i>	
<i>CODABAR</i>		<i>ITF_16</i>	
<i>CODABAR_Tradional</i>		<i>KURANDT</i>	
<i>CODABLOCK</i>		<i>LAETUS_PHARMA</i>	
<i>CODABLOCK_F</i>		<i>MSI</i>	
<i>Code128</i>		<i>NDC_HRI</i>	
<i>Code25</i>		<i>PARAF</i>	
<i>Code39</i>		<i>Plessey</i>	
<i>Code39_Extended</i>		<i>PDF417</i>	
<i>DATAMATRIX</i>		<i>PZN</i>	
<i>EAN</i>	Deprecated in JDF 1.3	<i>QR</i>	
<i>EAN_13</i>	includes Bookland_EAN and ISSN.	<i>RSS_14</i>	
<i>EAN_8</i>		<i>RSS_14_Stacked</i>	
<i>EAN_Coupon</i>		<i>RSS_14_Stacked_Omnidir</i>	
<i>EAN_128</i>		<i>RSS_14_Truncated</i>	
<i>HIBC_Code39</i>		<i>RSS_Limited</i>	
<i>HIBC_Code128</i>		<i>RSS_Expanded</i>	
<i>HIBC_Code39_2</i>		<i>RSS_Expanded_Stacked</i>	
<i>HIBC_CODABLOCK_F</i>		<i>UPC_A</i>	
<i>HIBC_QR</i>		<i>UPC_Coupon</i>	
<i>HIBC_DATAMATRIX</i>		<i>UPC_E</i>	
<i>Interleave25</i>		<i>UPC_SCS</i>	
<i>ITF_14</i>			

— Element: BarcodeDetails

Table 7-201: BarcodeDetails element (Section 1 of 2)

Name	Data Type	Description
<i>BarcodeVersion ?</i>	NMTOKEN	The version of a barcode. For <i>DATAMATRIX</i> values, see Table 7-204. For <i>QR</i> barcodes, the values are: <i>QR_1</i> ... <i>QR_40</i>

Table 7-201: BarcodeDetails element (Section 2 of 2)

Name	Data Type	Description
<i>ErrorCorrectionLevel?</i>	NMTOKEN	Error correction level for barcodes having a separately definable error correction level. Values include: For <i>EncodingDetails</i> = <i>PDF417</i> : <i>PDF417_EC_0</i> ... <i>PDF417_EC_8</i> For <i>EncodingDetails</i> = <i>QR</i> : <i>QR_EC_L</i> <i>QR_EC_M</i> <i>QR_EC_Q</i> <i>QR_EC_H</i>
<i>XCells?</i>	integer	The number of cells in x direction of a matrix barcode. For <i>DATAMATRIX</i> this field can be omitted since <i>BarcodeVersion</i> already defines this. For <i>PDF417</i> this is the number of codewords/row.
<i>YCells?</i>	integer	The number of cells in y direction of a matrix barcode For <i>DATAMATRIX</i> this field can be omitted since <i>BarcodeVersion</i> already defines this. For <i>PDF417</i> this is the number of rows.

— Element: ExtraValues

Table 7-202: ExtraValues element

Name	Data Type	Description
<i>Usage</i>	NMTOKEN	The usage of the value. Defined values include: <i>Supplemental</i> – UPC supplemental 2/5 digit symbology <i>CompositeCode</i> – This is applicable for barcodes like RSS-14 that have an optional composite code part. <i>Coupon</i> – The additional message for the EAN128 part of a UPC or EAN coupon.
<i>Value</i>	string	Additional value of the IdentificationField as specified in <i>Usage</i> .

The following table specifies whether the attributes *Height*, *Magnification* and *Ratio* are applicable for a given barcode type that is specified by *EncodingDetails*.

Table 7-203: Usage of barcode attributes for certain barcode types (Section 1 of 2)

EncodingDetails values (barcode types)	Height	Magnification	Ratio
<i>Code25</i>	Used	Used	Used
<i>Code39</i>			
<i>Code39_Extended</i>			
<i>Interleave25</i>			
<i>MSI</i>			
<i>Plessey</i>			

Table 7-203: Usage of barcode attributes for certain barcode types (Section 2 of 2)

EncodingDetails values (barcode types)	Height	Magnification	Ratio
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

— Attribute: BarcodeVersion for DATAMATRIX

The following table specifies valid values of BarcodeDetails/@BarcodeVersion for a DATAMATRIX barcode:

Table 7-204: BarcodeDetails/@BarcodeVersion – possible values

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
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

Example of a barcode description

The following example illustrates the description of a barcode in a *LayoutElementProduction* process:

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" DescriptiveName="Barcode Creation"
  ID="n001" Status="Waiting" Type="LayoutElementProduction" Version="1.3">
  <ResourcePool>
    <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>
  <LayoutElement Class="Parameter" ID="LayElOut" Status="Unavailable"/>
</ResourcePool>
<ResourceLinkPool>
  <LayoutElementProductionParamsLink Usage="Input" rRef="BarcodeParams"/>
  <LayoutElementLink Usage="Output" rRef="LayElOut"/>
</ResourceLinkPool>
</JDF>

```

7.2.93 IDPrintingParams

[Deprecated in JDF 1.1](#) See "IDPrintingParams" on page 835 for details of this deprecated resource.

7.2.94 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:	Layout, LayoutElement, PageList, PageList/PageData
Example Partition:	<i>DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName</i>
Input of processes:	<i>ImageReplacement, PSToPDFConversion,</i>
Output of processes:	—

Table 7-205: ImageCompressionParams resource

Name	Data Type	Description
ImageCompression *	element	Specifies how images are to be compressed.

— Element: ImageCompression

Table 7-206: ImageCompression element (Section 1 of 3)

Name	Data Type	Description
<i>AntiAliasImages</i> = "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 <i>ImageDepth</i> has a value greater than the number of bits per color component in the input image.
<i>AutoFilterImages</i> = "true" Modified in JDF 1.2	boolean	MUST NOT be specified unless <i>EncodeImages</i> is "true". This attribute is not used if <i>ImageType</i> = <i>Monochrome</i> . If "true", the filter defined by <i>ImageAutoFilterStrategy</i> is applied to photos and the <i>FlateEncode</i> filter is applied to screen shots. If "false", the <i>ImageFilter</i> compression method is applied to all images.
<i>ConvertImagesToIndexed</i> ?	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 <i>ImageType</i> = <i>Color</i> .
<i>DCTQuality</i> = "0"	double	A value between 0 and 1 that indicates "how much" the process is to compress images when using a <i>DCTEncode</i> filter. 0.0 means "do as loss-less compression as possible." 1.0 means "do the maximum compression possible."
<i>DownsampleImages</i> = "false" Modified in JDF 1.1A	boolean	If "true", sampled color images are downsampled using the resolution specified by <i>ImageResolution</i> . If "false", downsampling is not carried out and the image resolution in the PDF file is the same as that in the source file.
<i>EncodeColorImages</i> ? Deprecated in JDF 1.1	boolean	If "true", color images are encoded using the compression filter specified by the value of the <i>ImageFilter</i> key. If "false", no compression filters are applied to color sampled images.
<i>EncodeImages</i> = "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 <i>ImageFilter</i> key. If "false", no compression filters are applied to sampled images.
<i>ImageAutoFilterStrategy</i> ? New in JDF 1.2	NMTOKEN	Selects what image compression strategy to employ if passing through an image that is not already compressed. Possible values include: <i>JPEG</i> – Lossy JPEG compression for low-frequency images and lossless Flate compression for high-frequency images. <i>JPEG2000</i> – Lossy JPEG2000 compression for low-frequency images and lossless JPEG2000 compression for high-frequency images.

Table 7-206: ImageCompression element (Section 2 of 3)

Name	Data Type	Description
<i>ImageDepth</i> ?	integer	Specifies the number of bits per component in the downsampled image when <i>DownsampleImages</i> = "true". If not specified, the downsampled image has the same number of bits per sample as the original image.
<i>ImageDownsampleThreshold</i> = "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 <i>ImageDownsampleThreshold</i> , set the following attributes to the values indicated: <i>ImageResolution</i> = 72 <i>ImageDownsampleThreshold</i> = 1.5 The input image would not be downsampled unless it has a resolution greater than $(72 * 1.5) = 108$ dpi
<i>ImageDownsampleType</i> ?	enumeration	Downsampling algorithm for images. Possible values are: <i>Average</i> – The program averages groups of samples to get the new downsampled value. <i>Bicubic</i> – The program uses bicubic interpolation on a group of samples to get a new downsampled value. <i>Subsample</i> – The program picks the middle sample from a group of samples to get the new downsampled value.
<i>ImageFilter</i> ? Modified in JDF 1.3	NMTOKEN	Specifies the compression filter to be used for images. Ignored if <i>AutoFilterImages</i> = "true" or if <i>EncodeImages</i> = "false". Possible values are: <i>CCITTFaxEncode</i> – Used to select CCITT Group 3 or 4 facsimile encoding. Used only if <i>ImageType</i> = <i>monochrome</i> . <i>DCTEncode</i> – Used to select JPEG compression. <i>FlateEncode</i> – Used to select ZIP compression. <i>JBIG2Encode</i> – Used to select JBIG2 encoding. Used only if <i>ImageType</i> = "Monochrome". New in JDF 1.3 <i>JPEG2000</i> – Used to select JPEG2000/Wavelet compression. New in JDF 1.2 <i>LZWEncode</i> – LZW Compression. <i>PackBits</i> – A simple byte-oriented run length scheme. In JDF 1.1 and below, the data type was enumeration. It has been extended to NMTOKEN in order to allow for extensions.
<i>ImageResolution</i> ?	double	Specifies the minimum resolution for downsampled color images in dots per inch. This value is used only when <i>DownsampleImages</i> = "true". The application down-samples only images that are above that resolution to that actual resolution.

Table 7-206: ImageCompression element (Section 3 of 3)

Name	Data Type	Description
<i>ImageType</i>	enumeration	Specifies the kind of images that are to be manipulated. Possible values are: <i>Color</i> <i>Grayscale</i> <i>Monochrome</i>
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 <i>Rows</i> and <i>BlackIs1</i> parameters, which are implicit in the raster data to be compressed.
DCTParams ? New in JDF 1.2	element	Provides the equivalents of the PostScript <i>Columns</i> , <i>Rows</i> and <i>Colors</i> 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 <i>Columns</i> , <i>BitsPerComponent</i> and <i>Colors</i> parameters, which are implicit in the raster data to be compressed.
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 <i>Columns</i> , <i>BitsPerComponent</i> and <i>Colors</i> parameters, which are implicit in the raster data to be compressed

— Element: CCITTFaxParams[New in JDF 1.2](#)**Table 7-207: CCITTFaxParams element**

Name	Data Type	Description
<i>Uncompressed</i> = "false"	boolean	A flag to indicate whether the file generated can use uncompressed encoding when advantageous.
<i>K</i> = "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
<i>EndOfLine</i> ?	boolean	A flag indicating whether the CCITTFaxEncode filter prefixes an end-of-line bit pattern to each line of encoded data.
<i>EncodedByteAlign</i> ?	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.
<i>EndOfBlock</i> ?	boolean	A flag indicating whether the CCITTFaxEncode filter appends an end-of-block pattern to the encoded data

— Element: DCTParams

[New in JDF 1.2](#)

Table 7-208: DCTParams element

Name	Data Type	Description
<i>SourceCSs</i>	enumerations	Identifies which of the incoming color spaces will be operated on. See Table 7-209 for values of <i>SourceCSs</i> . 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.
<i>HSamples</i> ?	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.
<i>VSamples</i> ?	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.
<i>QFactor</i> = "1.0"	double	A scale factor applied to the elements of <i>QuantTable</i> .
<i>QuantTable</i> ?	DoubleList	Quantization tables. If present, there MUST be one <i>QuantTable</i> entry for each color channel.
<i>HuffTable</i> ?	DoubleList	Huffman tables for DC and AC components. If present, there MUST be at least one <i>HuffTable</i> element for each color channel.
<i>ColorTransform</i> = "Automatic"	enumeration	Color transformation algorithm: Values are: <i>None</i> – Colors are not to be transformed. <i>YUV</i> – 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. <i>Automatic</i> – "YUV" for 3-channel raster data, "None" otherwise. Note that YUV is equivalent to YCbCr in TIFF terminology.

When the DCTParams element is a subelement of **ImageCompressionParams** used in a **FormatConversion** 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 MUST be 1 in each case; that the fourth element MUST be 1 where CMYK data is to be compressed; and that the second and third elements MUST equal each other.

— Attribute: SourceCSs

Table 7-209: SourceCSs attribute – possible values (Section 1 of 2)

Value	Description
<i>Calibrated</i>	Operates on CalGray and CalRGB color spaces. New in JDF 1.2
<i>CIEBased</i>	Operates on CIE-Based color spaces (CIEBasedA, CIEBasedABC, CIEBasedDEF, CIE-BasedDEFG).
<i>CMYK</i>	Operates on DeviceCMYK.

Table 7-209: SourceCSs attribute – possible values (Section 2 of 2)

Value	Description
<i>DeviceN</i>	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 this value is specified then DeviceNSpace and ColorPool MUST also be present.
<i>DevIndep</i>	Operates on device independent color spaces (equivalent to Calibrated or CIE-Based or ICC-Based or Lab or YUV).
<i>Gray</i>	Operates on DeviceGray.
<i>ICCBased</i>	Operates on color spaces defined using ICC profiles. ICC-Based includes EPS, TIFF or PICT files with embedded ICC profiles. See [ICC.1]. If IgnoreEmbeddedICC is "true", then nominally ICC-Based files or elements is to be treated as being encoded in the alternate or underlying color space, and a ColorSpaceConversionOp where <i>SourceCS</i> = "DevIndep" is not to be applied, unless that color space is also device independent.
<i>Lab</i>	Operates on Lab.
<i>RGB</i>	Operates on DeviceRGB
<i>Separation</i>	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 .
<i>YUV</i>	Operates on YUV (Also known as YCbCr). See [CCIR601-2].

— Element: FlateParams[New in JDF 1.2](#)**Table 7-210: FlateParams element**

Name	Data Type	Description
<i>Effort ?</i>	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.
<i>Predictor = "1"</i>	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.

— Element: JBIG2Params

[New in JDF 1.3](#)

Table 7-211: JBIG2Params element

Name	Data Type	Description
<i>JBIG2Lossless</i> ?	boolean	If "true" requires JBIG2 compressed images to retain the exact representation of the original image without loss.

— Element: JPEG2000Params

[New in JDF 1.3](#)

Table 7-212: JPEG2000Params element

Name	Data Type	Description
<i>CodeBlockSize</i> ?	integer	The nominal code block width and height. MUST be a power of 2.
<i>LayersPerTile</i> = "1"	integer	Specifies the number of quality layers per tile at the same resolution.
<i>LayerRates</i> ?	DoubleList	Compression bit ratio for each layer. If specified, there MUST be the same number of doubles in this list as <i>LayersPerTile</i> in ascending order. Small values correspond to maximum compression and 1.0 corresponds to no compression (lossless). If available, <i>LayerRates</i> SHOULD be supplied.
<i>NumResolutions</i> ?	integer	The number of resolution levels encoded in the file.
<i>ProgressionOrder</i> ?	enumeration	Per tile progression order. <i>LRCP</i> – layer-resolution-component-position progressive (i.e., rate scalable). <i>RLCP</i> – Resolution-layer-component-position progressive (i.e., resolution scalable). <i>RPCL</i> – Resolution-position-component-layer progressive. <i>PCRL</i> – Position-component-resolution-layer progressive. <i>CPRL</i> – Component-position-resolution-layer progressive.
<i>TileSize</i> ?	XYPair	The width and height of each encoding tile. If not specified the image is considered to be a single tile.

— Element: LZWParams

[New in JDF 1.2](#)

Table 7-213: LZWParams element (Section 1 of 2)

Name	Data Type	Description
<i>EarlyChange</i> = "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 <i>EarlyChange</i> 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.

Table 7-213: LZWParams element (Section 2 of 2)

Name	Data Type	Description
<i>Predictor</i> = "1"	integer	<p>A code that selects the predictor function:</p> <p>1 – No predictor (normal encoding or decoding).</p> <p>2 – TIFF Predictor 2.</p> <p>10 – PNG predictor, None function.</p> <p>11 – PNG predictor, Sub function.</p> <p>12 – PNG predictor, Up function.</p> <p>13 – PNG predictor, Average function.</p> <p>14 – PNG predictor, Path function.</p> <p>15 – PNG predictor in which the encoding filter automatically chooses the optimum function separately for each row.</p> <p>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.</p>

7.2.95 ImageReplacementParams

This resource specifies parameters to control image replacement within production workflows.

Resource Properties

Resource class: Parameter

Resource referenced by: —

Example Partition: *DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName*

Input of processes: *ImageReplacement*

Output of processes: —

Table 7-214: ImageReplacementParams resource (Section 1 of 2)

Name	Data Type	Description
<i>ImagePreScanStrategy</i> ? New in JDF 1.2	NMTOKEN	<p>Specifies the image pre-scanning strategy to be used on the input document data before starting the RIPing process. Possible values are:</p> <p><i>NoPreScan</i> – Do not pre-scan the document looking for references to images.</p> <p><i>PreScan</i> – 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.</p> <p><i>PreScanAndGather</i> – 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.</p>

Table 7-214: ImageReplacementParams resource (Section 2 of 2)

Name	Data Type	Description
<i>ImageReplacementStrategy</i>	enumeration	Identifies how externally referenced images will be handled within the associated process. Possible values are: <i>Omit</i> – Complete process maintaining only references to external data. <i>Proxy</i> – Complete process using available proxy images. <i>Replace</i> – Replace external references with image data during processing. <i>AttemptReplacement</i> – Attempt to replace external references with image data during processing. If replacement fails, complete the process using available proxy images.
<u>MaxResolution ?</u> <u>Deprecated in JDF 1.1</u>	double	Reduces the resolution of images with a resolution higher than <i>MaxResolution</i> . Replaced with a link to ImageCompressionParams in the process.
<i>MinResolution ?</i>	double	Specifies the minimum resolution that an image MUST have in order to be embedded. If not specified, images of any resolution can be embedded.
<u>ResolutionReductionStrategy ?</u> <u>Deprecated in JDF 1.1</u>	enumeration	Identifies the mechanism used for reducing the image resolution. Possible values are: <i>Downsample</i> <i>Subsample</i> <i>Bicubic</i> Replaced with a link to ImageCompressionParams in the process.
<i>IgnoreExtensions ?</i>	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. Examples include: <i>.lay</i> <i>.e</i> <i>.samp</i>
<i>MaxSearchRecursion ?</i>	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.
FileSpec (<i>SearchPath</i>) + <u>New in JDF 1.1</u>	refelement	Specification of the paths to search when trying to locate the referenced data. The Filespec replaces the SearchPath text element.
SearchPath * <u>Deprecated in JDF 1.1</u>	telem	String that identifies the paths to search when trying to locate the referenced data.

7.2.96 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:	—
Example Partition:	—
Input of processes:	<i>ImageSetting</i>
Output of processes:	—

Table 7-215: ImageSetterParams resource (Section 1 of 2)

Name	Data Type	Description
<i>AdvanceDistance</i> ?	double	Additional media advancement beyond the media dimensions on a roll-fed device.
<i>BurnOutArea</i> ? New in JDF 1.1	XYPair	Size of the burnout area. The area defined by <i>BurnOutArea</i> is exposed, regardless of the size of the image. If not specified or "0 0", only the area defined by the image is exposed.
<i>CenterAcross</i> ?	enumeration	Specifies the axis around which a device is to center an image if the device is capable of doing so. Possible values are: <i>None</i> – Do not center. <i>FeedDirection</i> – Image is centered around the feed-direction axis. <i>MediaWidth</i> – Image is centered around the media-width axis. <i>Both</i> – Image is centered around both axes.
<i>CutMedia</i> ?	boolean	Indicates whether or not to cut the media (roll-fed).
<i>ManualFeed</i> ? New in JDF 1.2	boolean	Indicates whether the media will be fed manually.
<i>MirrorAround</i> = "None"	enumeration	This attribute specifies the axis around which a device MUST mirror an image if the device is capable of doing so. Possible values are: <i>None</i> – Do not mirror the image. <i>FeedDirection</i> – Image is mirrored around the feed-direction axis. <i>MediaWidth</i> – Image is mirrored around the media-width axis. <i>Both</i> – Image is mirrored around both possible axes.
<i>NonPrintableMarginBottom</i> ? 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 <i>NonPrintableMarginBottom</i> . These margins are independent of the PDL content.
<i>NonPrintableMarginLeft</i> ? New in JDF 1.3	double	Same as <i>NonPrintableMarginBottom</i> except for the left margin.
<i>NonPrintableMarginRight</i> ? New in JDF 1.3	double	Same as <i>NonPrintableMarginBottom</i> except for the right margin.
<i>NonPrintableMarginTop</i> ? New in JDF 1.3	double	Same as <i>NonPrintableMarginBottom</i> except for the top margin.
<i>Polarity</i> = "Positive"	enumeration	Some devices can invert the image (in hardware). Possible values are: <i>Positive</i> <i>Negative</i>

Table 7-215: ImageSetterParams resource (Section 2 of 2)

Name	Data Type	Description
<i>Punch ?</i> Deprecated in JDF 1.3	boolean	If " <i>true</i> ", indicates that the device MUST create registration punch holes. Use a combined Bending process to specify punching in JDF 1.3 and beyond.
<i>PunchType ?</i> Deprecated in JDF 1.3	string	Name of the registration punch scheme, (e.g., <i>Bacher</i>). Use a combined Bending process to specify punching in JDF 1.3 and beyond.
<i>Resolution ?</i>	XYPair	Resolution of the output. If not specified, the default is taken from the resolution of the input ByteMap.
<i>RollCut ?</i>	double	Length of media to be cut off of a roll, in points.
<i>Sides = "OneSidedFront"</i> New in JDF 1.2	enumeration	Indicates whether the content layout is to be imaged on one or both sides of the media. <i>Sides</i> MUST NOT be used unless ImageSetterParams describes output to a proofer. See Table 7-216 for values <i>Sides</i> .
<i>SourceWorkStyle ?</i> New in JDF 1.2	enumeration	When proofing in a "RIP once, output many" (ROOM) workflow, <i>SourceWorkStyle</i> specifies the direction in which the bytemaps have been prepared for press. The device is to use this information to calculate a transformation that results in a proof that is identical to the press sheet. Possible values are identical to ConventionalPrintingParams/@WorkStyle .
<i>TransferCurve ?</i>	Transfer-Function	Area coverage correction of the device.
Media ? New in JDF 1.1	refelement	Describes the media to be used. Different Media MAY be specified in different partition leaves to enable content driven Media selection.
FitPolicy ? New in JDF 1.2	refelement	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.

— Attribute: Sides

Table 7-216: Sides attribute – possible values

Value	Description
<i>OneSidedBackFlipX</i>	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 " <i>WorkAndTumble</i> " with a blank front side.
<i>OneSidedBackFlipY</i>	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 " <i>WorkAndTurn</i> " with a blank front side.
<i>OneSidedFront</i>	Page content is imaged on the front side of media. This is the only value that is valid for filmsetting and platesetting. The default.
<i>TwoSidedFlipX</i>	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 " <i>WorkAndTumble</i> ".
<i>TwoSidedFlipY</i>	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 " <i>WorkAndTurn</i> ".

7.2.97 Ink

Resource describing what kind of ink or other colorant (e.g., toner, varnish) is to be used during printing or varnishing. The default unit of measurement for **Ink** is *Unit* = “g” (gram).

Resource Properties

Resource class:	Consumable
Resource referenced by:	ConventionalPrintingParams
Example Partition:	<i>FountainNumber, Separation, SheetName, Side, SignatureName, WebName</i>
Input of processes:	ConventionalPrinting, DigitalPrinting
Output of processes:	—

Table 7-217: Ink resource

Name	Data Type	Description
<i>ColorName</i> ?	string	Link to a definition of the color specifics. The value of <i>ColorName</i> color SHOULD match the <i>Name</i> 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 <i>ColorName</i> attribute is used differently in other resources where it refers to a <i>NamedColor</i> as defined in "NamedColor" on page 697.
<i>Family</i> ?	NMTOKEN	Ink family. Possible values include: <i>HKS</i> <i>PANTONE</i> <i>Toyo</i> <i>ISO</i> – [ISO2846-1:1997] (used by SWOP) <i>InkJet</i> It is also possible to specify liquids that are similar to ink. Possible values of this type include: <i>Varnish</i> <i>Silicon</i> <i>Toner</i>
<i>InkName</i> ? Modified in JDF 1.1	string	The name of ink is dependent on its <i>Family</i> . For example, the <i>InkName</i> “138 C” is a member of the <i>Pantone Family</i> .
<i>SpecialInk</i> ?	NMTOKEN	Specific ink attributes. Possible values include: <i>Aqueous</i> <i>DullVarnish</i> <i>GlossVarnish</i> <i>Protective</i> <i>SatinVarnish</i> <i>Silicone</i> <i>UV</i> <i>Metallic</i>
<i>SpecificYield</i> ?	double	Weight per area at total coverage in g/m ² .

7.2.98 InkZoneCalculationParams

This resource specifies the parameters for the **InkZoneCalculation** process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>TileID, WebName</i>
Input of processes:	InkZoneCalculation
Output of processes:	—

Table 7-218: InkZoneCalculationParams resource

Name	Data Type	Description
<i>FountainPositions</i> ?	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 <i>FountainNumber</i> = "0"), the second to the second position (<i>FountainNumber</i> = "1"), etc.
<i>PrintableArea</i> ?	rectangle	Position and size of the printable area of the print cylinder in the coordinates of the Preview resource. The Partition <i>TileID</i> MUST 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 is to specify a rectangle that encompasses the complete image to be printed.
<i>ZoneHeight</i> ?	double	The width of one zone in the feed direction of the printing machine being used.
<i>ZoneWidth</i> ? 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.
<i>Zones</i> ? Modified in JDF 1.2	integer	The number of ink zones of the press.
<i>ZonesY</i> ?	integer	Number of ink zones in feed direction of the press.
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.

7.2.99 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
Resource referenced by:	—
Example Partition:	<i>FountainNumber, Separation, SheetName, Side, SignatureName, WebName</i>
Input of processes:	ConventionalPrinting
Output of processes:	InkZoneCalculation

Table 7-219: InkZoneProfile resource (Section 1 of 2)

Name	Data Type	Description
<i>ZoneHeight</i> ?	double	The width of one zone in the feed direction of the printing machine being used.

Table 7-219: InkZoneProfile resource (Section 2 of 2)

Name	Data Type	Description
<i>ZoneSettingsX</i>	DoubleList	Each entry of the <i>ZoneSettingsX</i> 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.
<i>ZoneSettingsY?</i>	DoubleList	Each entry of the <i>ZoneSettingsY</i> 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.
<i>ZoneWidth</i>	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.

7.2.100 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.

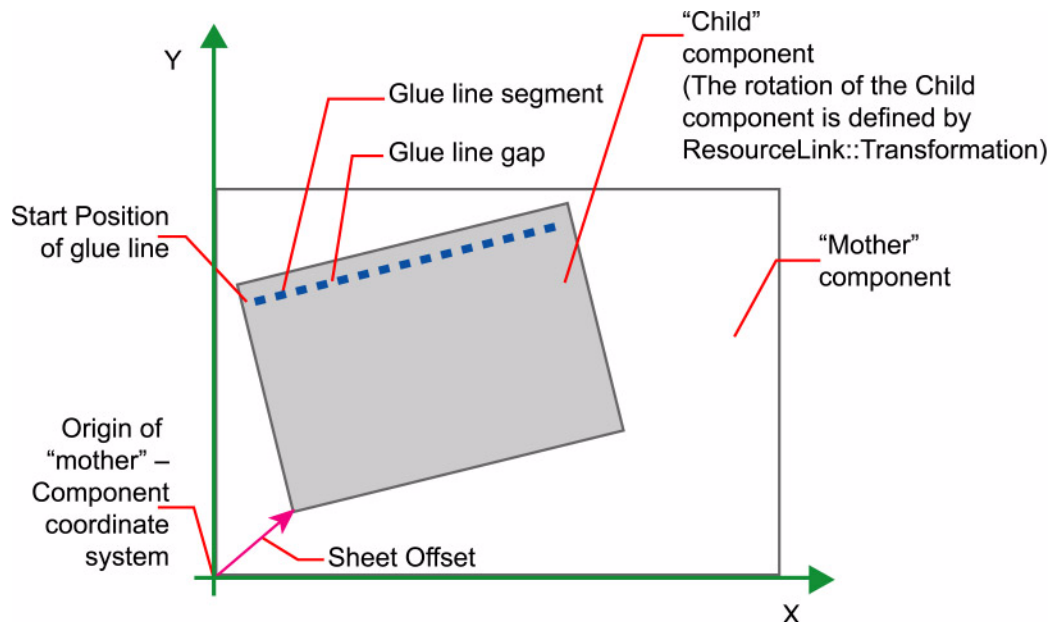


Figure 7-26: Parameters and coordinate system used for Inserting

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.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	Inserting
Output of processes:	—

Table 7-220: InsertingParams resource

Name	Data Type	Description
<i>FinishedPage</i> ? New in JDF 1.2	integer	Finished page number of the mother Component on which the child Component has to be placed. <i>FinishedPage</i> MUST NOT be specified unless <i>InsertLocation</i> = " <i>FinishedPage</i> ". Corresponds to <i>Folio</i> on InsertingIntent .
<i>InsertLocation</i> Modified in JDF 1.2	enumeration	Where to place the “child” sheet. Possible values are: <i>Back</i> <i>FinishedPage</i> – Place the child exactly onto the page specified in <i>FinishedPage</i> . New in JDF 1.2 <i>Front</i> <i>Overfold</i> – Place onto the overfold. Replaces <i>OverfoldLeft</i> and <i>OverfoldRight</i> . New in JDF 1.2 <i>OverfoldLeft</i> – Deprecated in JDF 1.2 <i>OverfoldRight</i> – Deprecated in JDF 1.2 Note: This was renamed from <i>Location</i> in JDF 1.2 due to a name clash with the <i>Location</i> partition key.
<i>Method</i> = "BlowIn"	enumeration	Inserting method. Possible values are: <i>BindIn</i> – Apply glue to fasten the insert. <i>BlowIn</i> – Loose insert.
<i>SheetOffset</i> ? Deprecated in JDF 1.1	XYPair	Offset between the sheet to be inserted and the “mother” sheet. <i>SheetOffset</i> is implied by the Transformation matrix in ResourceLink/@ <i>Transformation</i> of the child’s ComponentLink.
GlueLine *	refelement	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 7-221: Location of Inserts (Section 1 of 2)




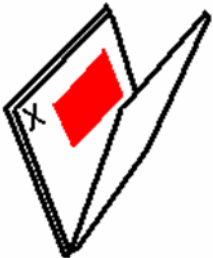
Front	Back	Overfold	FinishedPage
			

Table 7-221: Location of Inserts (Section 2 of 2)

Front	Back	Overfold	FinishedPage
Child on <i>Front</i> of mother component — is used for fixed inserts (e.g., gluing of inserts and so forth on signatures).	Child on <i>Back</i> 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. <i>Overfold</i> is used for loose inserts (e.g., inserts into newspapers)	Child on <i>FinishedPage</i> X of mother component — can be used for loose and fixed inserts.

7.2.101 InsertSheet

InsertSheet resources define device generated images and sheets which can be produced along with the job. **InsertSheet** elements include separator 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 **RunLists** 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**'s **InsertSheet** elements once, apply the **Layouts**' 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 has not been combined with **Imposition** or **LayoutPreparation**), only *Usage Header* or *Trailer* are valid.

Resource Properties

Resource class:	Parameter
Resource referenced by:	Disjointing, Layout, LayoutPreparationParams, RunList, Sheet
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-222: InsertSheet resource (Section 1 of 2)

Name	Data Type	Description
<p><i>IncludeInBundleItem</i> ? New in JDF 1.2</p>	enumeration	<p>Defines bundle items when this InsertSheet is not a subelement of RunList. If this InsertSheet is a subelement of a RunList, then <i>IncludeInBundleItem</i> MUST be ignored, and RunList/@EndOfBundleItem MUST be used instead. As an example, <i>IncludeInBundleItem</i> controls whether the InsertSheet is to be included in a bundle item for purposes of finishing the InsertSheet with other sheets. Possible values are:</p> <p><i>After</i> – This InsertSheet is to be included in the BundleItem that occurs after this InsertSheet. “<i>After</i>” is equivalent to “<i>None</i>” if no BundleItem is defined after this InsertSheet</p> <p><i>Before</i> – This InsertSheet is to be included in the BundleItem that occurs before this InsertSheet. “<i>Before</i>” is equivalent to “<i>None</i>” if no BundleItem is defined before this InsertSheet</p> <p><i>None</i> – This InsertSheet is not included in a BundleItem.</p> <p><i>New</i> – A new BundleItem is created. This InsertSheet will be in the new BundleItem by itself unless another InsertSheet with <i>IncludeInBundleItem</i> = “<i>Before</i>” occurs immediately after this InsertSheet.</p>
<p><i>IsWaste</i> ?</p>	boolean	<p>Specifies whether the InsertSheet is waste that is to be removed from the document before further processing. If “<i>true</i>”, the InsertSheet is to be discarded when finishing the document.</p>
<p><i>MarkList</i> ? New in JDF 1.1</p>	NMTOKENS	<p>List of marks that are to be marked on this InsertSheet. Ignored if a sheet is specified in this InsertSheet. Values include:</p> <p><i>CIELABMeasuringField</i> <i>ColorControlStrip</i> <i>ColorRegisterMark</i> <i>CutMark</i> <i>DensityMeasuringField</i> <i>IdentificationField</i> <i>JobField</i> <i>PaperPathRegisterMark</i> <i>RegisterMark</i> <i>ScavengerArea</i></p>

Table 7-222: InsertSheet resource (Section 2 of 2)

Name	Data Type	Description
SheetFormat ? New in JDF 1.1 Modified in JDF 1.2	NMTOKEN	Identifies that device-dependent information is to be included on the InsertSheet . Possible values include: <i>Blank</i> <i>Brief</i> <i>Duplicate</i> – Valid for <i>SheetUsage</i> = “ <i>Interleaved</i> ” or “ <i>InterleavedBefore</i> ”. Specifies that the interleaved sheet is to contain the same (duplicate) content as the previous (<i>Interleaved</i>) or following (<i>InterleavedBefore</i>) sheet. If there is content on both sides of the previous or following <u>sheet (duplex)</u> , then the InsertSheet has both sides duplicated. New in JDF 1.2 <i>Full</i> <i>Standard</i>
SheetType New in JDF 1.1	enumeration	Identifies the type of sheet. Possible values are: <i>AccountingSheet</i> – A sheet that reports accounting information for the job. <i>ErrorSheet</i> – A sheet that reports errors for the job. <i>FillSheet</i> – A sheet that fills ContentObject elements with no matching entry in the content RunList . <i>InsertSheet</i> – A sheet that is inserted to the job, (e.g., a preprinted cover). <i>JobSheet</i> – A sheet that delimits the job. <i>SeparatorSheet</i> – 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 is to be produced and inserted into the set of output pages. See Table 7-223 for values of <i>SheetUsage</i> .
Usage ? Deprecated in JDF 1.1	enumeration	Replaced by <i>SheetUsage</i> .
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. Replaced with Layout in JDF 1.3 and above.

— Attribute: SheetUsage

Table 7-223: SheetUsage attribute – possible values (Section 1 of 2)

Value	Description
<i>FillForceBack</i>	Valid for <i>SheetType</i> = "FillSheet". Contents of the RunList of the InsertSheet are used to fill the front surface of the current sheet before forcing the next page of the content RunList to the back surface of the current sheet if not already on a back surface.
<i>FillForceFront</i>	Valid for <i>SheetType</i> = "FillSheet". Contents of the RunList of the InsertSheet are used to fill the back surface of the current sheet before forcing the next page of the content RunList to the front surface of the next sheet if not already on a front surface.
<i>FillSheet</i>	Valid for <i>SheetType</i> = "FillSheet". Contents from the RunList of the InsertSheet are used to fill the current sheet.
<i>FillSignature</i>	Valid for <i>SheetType</i> = "FillSheet". Contents from the RunList of the InsertSheet are used to fill the current signature.
<i>FillSurface</i>	Valid for <i>SheetType</i> = "FillSheet". Contents from the RunList of the InsertSheet are used to fill the current surface.
<i>Header</i>	Valid for <i>SheetType</i> = "InsertSheet", "JobSheet" or "SeparatorSheet". The sheet is produced at the beginning of the job (for <i>JobSheet</i>), or at the beginning of each copy of each instance document (for <i>SeparatorSheet</i>), 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 <i>SheetType</i> .
<i>Interleaved</i>	Valid for <i>SheetType</i> = "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 <i>SheetType</i> = "SeparatorSheet".
<i>InterleavedBefore</i>	Valid for <i>SheetType</i> = "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 <i>SheetType</i> = "SeparatorSheet". New in JDF 1.2
<i>OnError</i>	Valid for <i>SheetType</i> = "ErrorSheet". The sheet is produced at the end of the job only when an error or warning occurs.
<i>Slip</i>	Valid for <i>SheetType</i> = "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 <i>SheetType</i> = "SeparatorSheet".
<i>SlipCopy</i>	Valid for <i>SheetType</i> = "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 <i>SheetType</i> = "SeparatorSheet".

Table 7-223: SheetUsage attribute – possible values (Section 2 of 2)

Value	Description
<i>Trailer</i>	Valid for <i>SheetType</i> = "AccountingSheet", "ErrorSheet", "InsertSheet", "JobSheet" and "SeparatorSheet". The sheet is produced at the end of the job (for <i>AccountingSheet</i> , <i>ErrorSheet</i> and <i>JobSheet</i>), or at the end of each copy of each instance document (for <i>SeparatorSheet</i>), 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 <i>SheetType</i> . ^a

- a. **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.

7.2.102 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
Example Partition:	—
Input of processes:	—
Output of processes:	—

7.2.103 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
Resource referenced by:	—
Example Partition:	<i>DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName</i>
Input of processes:	Interpreting
Output of processes:	—

Table 7-224: InterpretingParams resource (Section 1 of 3)

Name	Data Type	Description
<i>Center</i> = "false"	boolean	Indicates whether or not the finished page image is to be centered within the imageable area of the media. The <i>Center</i> is ignored if FitPolicy / @SizePolicy = "ClipToMaxPage" and clipping is specified.
FitToPage? Deprecated in JDF 1.1	boolean	Specifies whether the finished page contents is to be scaled to fit the media. In JDF 1.1 and beyond, use FitPolicy .

Table 7-224: InterpretingParams resource (Section 2 of 3)

Name	Data Type	Description
<i>MirrorAround</i> = "None"	enumeration	This attribute specifies the axis around which a RIP is to mirror an image. Note that this is mirroring in the RIP and not in the hardware of the output device. Possible values are: <i>None</i> – The default. <i>FeedDirection</i> – Image is mirrored around the feed-direction axis. <i>MediaWidth</i> – Image is mirrored around the media-width axis. <i>Both</i> – Image is mirrored around both possible axes.
<i>Polarity</i> = "Positive"	enumeration	The image MUST 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. Possible values are: <i>Positive</i> <i>Negative</i>
<i>Poster</i> ?	XYPair	Specifies whether the page contents is to be expanded such that each page covers X by Y pieces of media.
<i>PosterOverlap</i> ?	XYPair	This pair of real numbers identifies the amounts of overlap in points for the poster tiles across the horizontal and vertical axes, respectively.
<i>PrintQuality</i> = "Normal" New in JDF 1.1	enumeration	Generic switch for setting the quality of an otherwise inaccessible device. Possible values are: <i>High</i> – Highest quality available on the printer. <i>Normal</i> – The default quality provided by the printer. <i>Draft</i> – Lowest quality available on the printer.
<i>Scaling</i> ?	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 <i>FitToPage</i> = "true" or if <i>Poster</i> is present and has a value other than "1 1". Any scaling defined in FitPolicy MUST be applied after the scaling defined by this attribute.
<i>ScalingOrigin</i> ?	XYPair	A pair of real values that identify the point in the unscaled page that is to 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 <i>ScalingOrigin</i> defaults to "0 0"
FitPolicy ? New in JDF 1.1	refelement	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 <i>FitToPage</i> attribute. This FitPolicy element MUST be ignored in a combined process with LayoutPreparation .

Table 7-224: InterpretingParams resource (Section 3 of 3)

Name	Data Type	Description
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 *	refelement	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.

— Element: PDFInterpretingParams

[New in JDF 1.1](#)

Table 7-225: PDFInterpretingParams element (Section 1 of 2)

Name	Data Type	Description
<i>EmitPDFBG</i> = "true"	boolean	Indicates whether BlackGeneration functions are to be emitted.
<i>EmitPDFHalftones</i> = "true"	boolean	Indicates whether Halftones are to be emitted.
<i>EmitPDFTransfers</i> = "true"	boolean	Indicates whether Transfer functions are to be emitted
<i>EmitPDFUCR</i> = "true"	boolean	Indicates whether UnderColorRemoval functions are to be emitted.
<i>HonorPDFOverprint</i> = "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 preprocessed to simulate the effect of the overprint settings
<i>ICCColorAsDeviceColor</i> = "false"	boolean	Indicates whether colors specified by ICC color spaces are to be treated as device colorants.
<i>OCGDefault</i> = "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. One of: <i>Exclude</i> – All layers not explicitly listed are to be excluded. <i>FromPDF</i> – The guidelines in the PDF reference are to be used to determine whether to include each layer that is not explicitly listed. <i>Include</i> – All layers not explicitly listed are to be included.

Table 7-225: PDFInterpretingParams element (Section 2 of 2)

Name	Data Type	Description
<i>OCGIntent</i> ? New in JDF 1.3	NMTOKEN	If <i>OCGDefault</i> = "FromPDF", then the value of <i>OCGIntent</i> sets the intent for which OCGs are to be selected. Values include "Design" and "View", as described in [PDF1.6].
<i>OCGProcess</i> ? New in JDF 1.3	enumeration	If <i>OCGDefault</i> = "FromPDF", then the value of <i>OCGProcess</i> sets the purpose for which the Interpreting process is being performed. This, in turn, sets which value from a relevant optional content usage dictionary is to be used to determine whether each OCG is included in the InterpretedPDLDData: <i>Export</i> – PDF ExportState in the Export subdictionary. <i>Print</i> – PDF PrintState in the Print subdictionary <i>View</i> – PDF ViewState in the View subdictionary.
<i>OCGZoom</i> = "1.0" New in JDF 1.3	double	If <i>OCGDefault</i> = "FromPDF", then the value of <i>OCGZoom</i> 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 InterpretedPDLDData. A <i>OCGZoom</i> value of 1.0 is assumed to be a magnification of 100%.
<i>PrintPDFAnnotations</i> = "false"	boolean	Indicates whether the contents of annotations on PDF pages are to be included in the output. This only refers to annotations that are set to print in the PDF file.
<i>TransparencyRenderingQuality</i> ?	double	Possible values are 0 to 1. 0 represents the lowest allowable quality. 1 represents the highest desired quality.
<i>OCGControl</i> * New in JDF 1.3	element	Provides a list of the OCGs (layers) that are to be explicitly included or excluded in the InterpretedPDLDData. Any OCGs not listed in an <i>OCGControl</i> element will follow the rules set by <i>OCGDefault</i> .

— Element: **OCGControl**[New in JDF 1.3](#)

Table 7-226: OCGControl element

Name	Data Type	Description
<i>IncludeOCG</i> = "true"	boolean	Defines whether the optional content group(s) identified by <i>OCGName</i> are to be included in the InterpretedPDLDData. If "true", then the layer MUST be included. If "false", it MUST NOT. Note that the contents stream of excluded OCGs MUST still be interpreted so that changes to CTM, etc., are acted on. The objects drawn in excluded OCGs MUST NOT be rendered.
<i>OCGName</i>	string	The name of the optional content group(s) that MUST be included or excluded. Note that the <i>Name</i> attribute of an optional content group entry is encoded as a PDF text string, and <i>OCGName</i> is encoded with the Unicode variant identified in the JDF file header; names MUST be re-encoded as necessary for comparison. Using a value for <i>OCGName</i> that does not match any OCG in the referenced PDF file is an error (subject to <i>SettingsPolicy</i>), independent of the value of <i>IncludeOCG</i> .

7.2.103.1 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 MUST 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 MUST 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.

7.2.104 JacketingParams

[New in JDF 1.1](#)

Description of the setup of the jacketing machinery. Jacket height and width (1 and 3 in the figure below) are specified within the **Component** that describes the jacket.

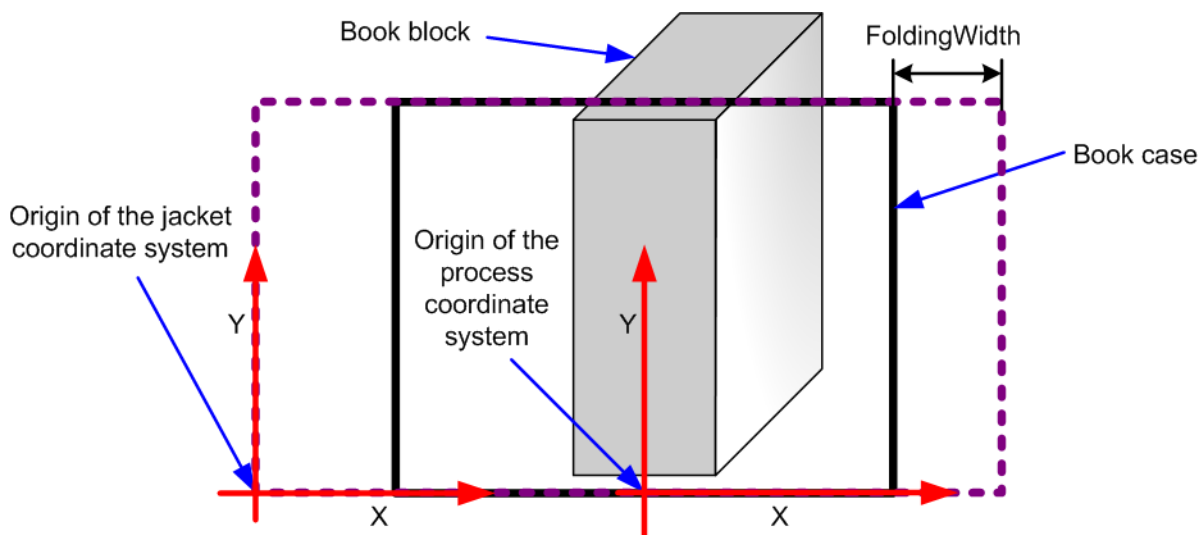
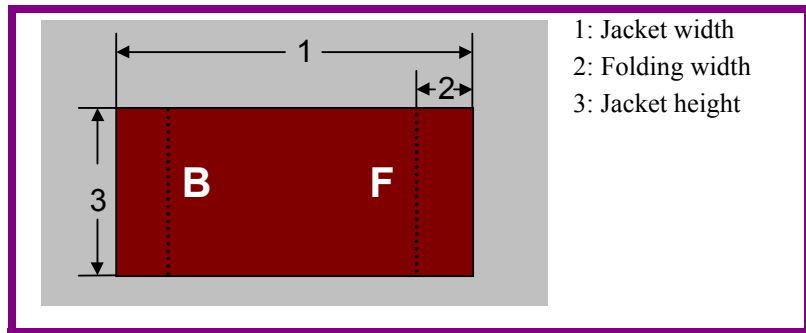


Figure 7-27: Parameters and coordinate system for jacketing

Resource Properties

Resource class: Parameter

Resource referenced by: —
 Example Partition: —
 Input of processes: *Jacketing*
 Output of processes: —

Table 7-227: JacketingParams resource

Name	Data Type	Description
<i>FoldingWidth</i>	double	Definition of the dimension of the folding width of the front cover fold (see “ <i>FoldingWidth</i> ” in the picture above). All other measurements are implied by the dimensions of the book.

7.2.105 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.

Resource Properties

Resource class: Parameter
 Resource referenced by: **Layout**
 Example Partition: —
 Input of processes: —
 Output of processes: —

Table 7-228: JobField resource (Section 1 of 3)

Name	Data Type	Description
<i>OperatorText?</i> Modified in JDF 1.2	string	Text from the operator. Note that this was erroneously described as text to the operator in JDF 1.1 and below.

Table 7-228: JobField resource (Section 2 of 3)

Name	Data Type	Description
<p><i>ShowList</i></p> <p>Modified in JDF 1.3</p>	NMTOKENS	<p>List of elements to display in the JobField. Values include:</p> <p><i>DeviceID</i> – ID of the device. This is a unique name within the workflow.</p> <p><i>EndTime</i> – Actual end time of the job.</p> <p><i>Error</i> – Errors that happened during the job.</p> <p><i>ErrorStats</i> – Statistics on errors that happened during execution.</p> <p><i>ExposedMediaName</i> – <i>DescriptiveName</i> of the exposed media, e.g. plate or proof that is being imaged.</p> <p><i>FriendlyName</i> – <i>FriendlyName</i> of the device.</p> <p><i>JobID</i> – <i>JobID</i> of the node that is executing.</p> <p><i>JobName</i> – <i>DescriptiveName</i> of the node that is executing.</p> <p><i>JobRecipientName</i> – Name of the recipient of the job.</p> <p><i>JobSubmitterName</i> – Name of the submitter of the job.</p> <p><i>LayPartCount2in1</i> – Number of partitions at level 2 within partition level 1 (the base) within the Layout resource. When using the RECOMMENDED <i>SignatureName</i> / <i>SheetName</i> partition keys, this equates to the number of signatures within the Layout. All other variants of <i>LayPartCount<n>in<m></i> can 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. New in JDF 1.3</p> <p><i>LayPartIndex2in1</i> – Index (1-based) of partition level 2 within partition level 1 (the base) within the Layout resource. When using the RECOMMENDED <i>SignatureName</i> / <i>SheetName</i> partition keys, this equates to the signature index within the Layout. All other variants of <i>LayPartIndex<n>in<m></i> can 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. New in JDF 1.3</p> <p><i>MediaBrand</i> – Brand of the media that is being printed.</p> <p><i>MediaType</i> – <i>DescriptiveName</i> of the media that is being printed.</p> <p><i>MoonPhase</i> – Phase of the moon at the <i>StartTime</i> of the job.</p> <p><i>Operator</i> – Name of the operator.</p> <p><i>OperatorText</i> – Text from the operator as defined in <i>OperatorText</i>.</p> <p><i>PrintQuality</i> – The quality of the printout. (High, Normal, Draft or device specific name)</p> <p><i>ProoferProfileName</i> – Name of the ICC profile for the proofing device.</p> <p><i>PressProfileName</i> – Name of the ICC profile for the final printing (used as intermediate space during proofing).</p> <p><i>Resolution</i> – Output resolution.</p> <p><i>ResolutionX</i> – Output resolution in X direction.</p> <p><i>ResolutionY</i> – Output resolution in Y direction.</p> <p><i>ScreeningFamily</i> – Name of the screening family of the output.</p> <p><i>StartTime</i> – Actual <i>StartTime</i> of the job.</p> <p><i>UserText</i> – User-defined text as defined in <i>UserText</i>.</p> <p><i>Warning</i> – Warnings that happened during the job. Warnings don't lose information in the resulting job, while errors do.</p> <p>In addition, the partition key names defined in Table 3-25, "Partitionable resource element," on page 86 are also supported.</p>

Table 7-228: JobField resource (Section 3 of 3)

Name	Data Type	Description
<i>UserText ?</i>	string	User-defined text to output with JobField.
DeviceMark ? Modified in JDF 1.3	refelement	DeviceMark defines the formatting parameters for the mark. If not specified, the settings defined in LayoutPreparationParams/DeviceMark are assumed.

7.2.106 LabelingParams

[New in JDF 1.1](#)

LabelingParams defines the details of the *Labeling* process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	Labeling
Output of processes:	—

Table 7-229: LabelingParams resource

Name	Data Type	Description
<i>Application ?</i>	NMTOKEN	Application method of the label. Values include: <i>Glue</i> – Glued onto the component. <i>Loose</i> – Loosely laid onto the component. <i>SelfAdhesive</i> – Self adhesive label. <i>Staple</i> – Stapled onto the component.
<i>CTM ?</i>	matrix	Position and orientation of the label lower-left-corner relative to the lower left corner of the component surface as defined by <i>Position</i> .
<i>Position ?</i>	enumeration	Position of the label on the bundle. One of: <i>Back</i> <i>Bottom</i> <i>Front</i> <i>Left</i> <i>Right</i> <i>Top</i>

7.2.107 LaminatingParams

[New in JDF 1.1](#)

This resource specifies the parameters needed for laminating.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>SheetName, Side</i>
Input of processes:	Laminating
Output of processes:	—

Table 7-230: LaminatingParams resource

Name	Data Type	Description
<i>AdhesiveType</i> ?	string	Type of adhesive used. Valid only when <i>LaminatingMethod</i> = <i>DispersionGlue</i> .
<i>GapList</i> ?	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 <i>LaminatingBox</i> is laminated.
<i>HardenerType</i> ?	string	Type of hardener used. Valid only when <i>LaminatingMethod</i> = <i>DispersionGlue</i> .
<i>LaminatingBox</i>	rectangle	Area on the Component to be laminated.
<i>LaminatingMethod</i> ?	enumeration	Laminating technology that is applied. One of: <i>CompoundFoil</i> <i>DispersionGlue</i> <i>Fusing</i> – New in JDF 1.3 <i>Unknown</i> – Deprecated in JDF 1.2
<i>NipWidth</i> ? 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.
<i>Temperature</i> ?	double	Temperature used in the lamination process, in ° Centigrade.

7.2.108 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

Resource referenced by: **Component**

Example Partition: *SignatureName, WebName, RibbonName, SheetName, Side, PartVersion*

Input of processes: ***ConventionalPrinting, DigitalPrinting, Imposition, InkZoneCalculation***

Output of processes: ***LayoutPreparation, Stripping***

Table 7-231: Layout resource (Section 1 of 3)

Name	Data Type	Description
<i>Automated</i> = "false"	boolean	If "true", the Imposition process is expected to perform automated page consumption. Automated page consumption is equivalent to the PrintLayout functionality provided in PJTF.
<i>LockOrigins</i> = "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 <i>LockOrigins</i> was copied from the deprecated Sheet resource.

Table 7-231: Layout resource (Section 2 of 3)

Name	Data Type	Description
MaxDocOrd = "1" New in JDF 1.1	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.
MaxOrd ?	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 MUST be calculated from the <i>Ord</i> values of the ContentObject elements in the Layout .
MaxSetOrd = "1" New in JDF 1.1	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.
Name ? New in JDF 1.1	string	Unique name of the Layout . The <i>Name</i> is used for external reference to a Layout .
SourceWorkStyle ? New in JDF 1.3	enumeration	Indicates which <i>WorkStyle</i> was used to create the Layout . This is only informative and can be useful when creating double sided proofs. Possible values are identical to those in ConventionalPrintingParams/@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". <i>SurfaceContentsBox</i> 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 <i>Media</i> is implied.
InsertSheet *	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.
LayerList ? New in JDF 1.1	element	List of LayerDetails elements.
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 MUST NOT be specified in one Layout . Note that at least one Media MUST 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.
PlacedObject * New in JDF 1.3	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. All PlacedObject elements MUST be specified in the partition leaves of the Layout . See "Position of PlacedObject elements in Layout " on page 492. Note: PlacedObject is not a container but an abstract type.

Table 7-231: Layout resource (Section 3 of 3)

Name	Data Type	Description
Signature * Deprecated in JDF 1.3	element	The Signature element has been replaced by a Layout partition, namely Layout [<i>@SignatureName</i>]. 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.

— Element: 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 7-232: LayerList element

Name	Data Type	Description
LayerDetails *	element	Details of the individual layers.

— Element: LayerDetails[New in JDF 1.1](#)

This element provides information about individual layers.

Table 7-233: LayerDetails element

Name	Data Type	Description
<i>Name ?</i>	string	Unique name of the layer.

— Element: Abstract PlacedObject

The marks that are to be placed on the designated surface of a **Layout** come in two varieties: **ContentObject** or **MarkObject** elements. All three inherit characteristics from the abstract **PlacedObject** which is described below.

Table 7-234: Abstract PlacedObject element (Section 1 of 3)

Name	Data Type	Description
<i>ClipBox ?</i>	rectangle	Clipping rectangle in the coordinates of the <i>SurfaceContentsBox</i> .
ClipPath? New in JDF 1.3	PDFPath	Clip path for the PlacedObject in the coordinates of the <i>SurfaceContentsBox</i> (lower left of <i>SurfaceContentsBox</i> is used as reference zero point, same as for <i>ClipBox</i>). The actual clip region is the union of <i>ClipBox</i> and <i>ClipPath</i> or the union of <i>ClipBox</i> and <i>SourceClipPath</i> . <i>ClipPath</i> and <i>SourceClipPath</i> MUST NOT be specified in the same PlacedObject . <i>ClipPath</i> SHOULD be specified when both <i>ClipPath</i> and <i>SourceClipPath</i> are known because <i>ClipPath</i> provides a more stable coordinate system (not sensitive to shifts caused by editing the page).

Table 7-234: Abstract PlacedObject element (Section 2 of 3)

Name	Data Type	Description
<i>CTM</i>	matrix	The coordinate transformation matrix (CTM — a Postscript term) of the object in the <i>SurfaceContentsBox</i> . For details, see "Equation for Surface Coordinate System Transformations" on page 27. 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 <i>SurfaceContentsBox</i> . For details, see "Source Coordinate Systems" on page 24. Note: <i>CTM</i> MUST be recalculated if the object is replaced afterwards with a new object with different dimensions.
<i>HalfTonePhaseOrigin</i> = "0 0"	XYPair	Location of the origin for screening of this ContentObject. Specified in the coordinate systems of <i>SurfaceContentsBox</i> .
<i>LayerID</i> ? New in JDF 1.1	integer	If a layout supports layering (e.g., for versioning), <i>LayerID</i> can be used to identify the layer that a ContentObject belongs to, (e.g., the language layer version). The details of the layers are specified in the Layout/LayerList/@LayerDetails key.
<i>OrdID</i> ? New in JDF 1.1	integer	If a Layout supports layering (e.g., for versioning), <i>OrdID</i> can be used to identify ContentObject elements that belong to the same final page. These will have a matching <i>OrdID</i> .
<i>SourceClipPath</i> ? Modified in JDF 1.3	PDFPath	Clip path for the PlacedObject in the source coordinate system. <i>SourceClipPath</i> 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.) MUST be applied prior to applying <i>SourceClipPath</i> . <i>ClipPath</i> and <i>SourceClipPath</i> MUST NOT be specified in the same PlacedObject. See Section 2.5.1.1, Source Coordinate Systems for definitions of source coordinate systems.
<i>TrimCTM</i> ? New in JDF 1.1	matrix	The transformation matrix of the object's trim box in the <i>SurfaceContentsBox</i> . Note that imposition programs that execute the Layout MUST recalculate the <i>CTM</i> in case the object is replaced with a new object with different dimensions, otherwise the position of the content inside the trim box will shift. This recalculation is based on <i>TrimCTM</i> , <i>TrimSize</i> and trim box.

Table 7-234: Abstract PlacedObject element (Section 3 of 3)

Name	Data Type	Description
TrimSize ? New in JDF 1.2	XYPair	<p>The size of the object's trim box as viewed in the <i>SurfaceContentsBox</i> (<i>TrimCTM</i> scaling and rotation applied). <i>TrimSize</i> is needed when replacing the object by a new object with a different dimension.</p> <p>Note: Recalculation of <i>PlacedObject/@CTM</i> is only necessary when the Imposition process 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, <i>PlacedObject/@CTM</i> recalculations SHOULD always be done according to <i>PlacedObject/@TrimCTM</i> and <i>PlacedObject/@TrimSize</i>. 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 (= <i>LayoutElement/@SourceTrimBox</i> when real trim box of the object is known).</p> <p>Usage of both <i>PlacedObject's TrimCTM</i> and <i>TrimSize</i> attributes will allow page replacements on any type of imposition Layout.</p>
Type ? Deprecated in JDF 1.1	enumeration	<p>Describes the kind of <i>PlacedObject</i>. Possible values are:</p> <p><i>Content</i></p> <p><i>Mark</i></p>

— Element: 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 7-235: 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. <i>Layout/@MaxDocOrd</i> equals <i>DocOrd</i> 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).
Ord ? Modified in JDF 1.1	integer	A non-negative, zero-based reference to an index in the content RunList . The index is incremented for every page of the RunList with <i>IsPage</i> = "true". The <i>Ord</i> value of the first page of a RunList has the value "0".
OrdExpression ?	string	Function to calculate an <i>Ord</i> value dynamically, using a value of <i>s</i> for signature number and <i>n</i> for total number of pages in the instance document. The <i>Ord</i> or <i>DocOrd</i> and <i>OrdExpression</i> are mutually exclusive in one <i>PlacedObject</i> .
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. <i>Layout/@MaxSetOrd</i> = <i>SetOrd</i> 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).

7.2.108.1 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 are 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.

7.2.108.1.1 Partition Key restrictions:

If *SignatureName SheetName* or *Surface* are specified in *PartIDKeys*, the order MUST be specified as *SignatureName SheetName Surface*.

Only a **Layout** with exactly *PartIDKeys* = "*SignatureName SheetName Surface*" 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.

7.2.108.1.2 Position of PlacedObject elements in Layout

In order to avoid ambiguities in the layering order, **MarkObject** elements and **ContentObject** elements MUST only be specified in the leaves of partitioned resources.

The following INVALID example is correct according to "Subelements in Partitioned Resources" on page 80. 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 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">
    <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">
      <RegisterMark Center="0.0 0.0" MarkType="Cross" MarkUsage="PaperPath" />
    </MarkObject>
    <Layout SheetName="Sheet00">
      <Layout Side="Front">
        <MarkObject Ord="2">
          <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>
```

This VALID example contains the same **PlacedObject** elements as the previous example but they are correctly specified in the leaves of the partitioned **Layout**.

```
<Layout PartIDKeys="SignatureName SheetName Side">
  <Layout SignatureName="Sig00">
    <Layout SheetName="Sheet00">
      <Layout Side="Front">
        <MarkObject Ord="2">
          <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>
```

```

</Layout>
<Layout SheetName="Sheet01">
  <Layout Side="Front">
    <MarkObject Ord="1">
      <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>

```

7.2.108.2 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 is to 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.

7.2.108.3 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**

7.2.108.4 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*.

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 7-236: 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)

Simple Multi-File separated RunList

This example specifies two pages contained in Presep.pdf and following that, pages 1, 3 and 5 of each preprepared 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 7-237: 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

7.2.108.5 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 MUST be expressed by explicitly including the * operator, (i.e., use “2*s”, not “2 s”). Remainders are discarded.

OrdExpression Examples (*Saddlestitched 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.

```
Front:
OrdExpression = "2*s" 0 2 4 6
OrdExpression = "4*((n+3)/4) - (s*2) - 1" 1513119
Back:
OrdExpression = "2*s+1" 1 3 5 7
OrdExpression = "4*((n+3)/4) - (s*2) - 2" 1412108
```

DocOrd Usage Examples (*Two-sided business cards 4/sheet*)

The following describes the Ord + DocOrd usage for a 4-up step + repeat business card

```
MaxDocOrd = 4
```

```
Front:
Ord = 0 DocOrd = 0
Ord = 0 DocOrd = 1
Ord = 0 DocOrd = 2
Ord = 0 DocOrd = 3
Back:
Ord = 1 DocOrd = 0
Ord = 1 DocOrd = 1
Ord = 1 DocOrd = 2
Ord = 1 DocOrd = 3
```

— Element: MarkObject

MarkObject elements describe containers for page marks on a surface. The PDL for the marks SHOULD exist prior to imposing and 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 7-238: MarkObject element

Name	Data Type	Description
<i>LayoutElementPageNum</i> = "0" New in JDF 1.1	integer	Page number to use from the PDL file described by <i>LayoutElement</i> .
<i>Ord</i> ? Modified in JDF 1.1A	integer	A non-negative reference to an index in the RunList (Marks). The index is incremented for every page of the RunList with <i>IsPage</i> = "true". The first page of a RunList has the value 0. At most one of <i>LayoutElement</i> , <i>Ord</i> or JobField MUST be specified in JDF 1.3 and above.
CIELABMeasuringField *	refelement	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.
DensityMeasuringField * 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 1.1 Deprecated in JDF 1.3	refelement	If none of <i>Ord</i> or <i>LayoutElement</i> are specified, it is assumed that the device can independently generate the mark. DeviceMark defines a set of formatting parameters for the mark. In JDF 1.3 and above, JobField/DeviceMark specifies the formatting parameters of the JobField and all other dynamically generated marks are positioned with <i>CTM</i> .
<i>DynamicField</i> *	element	Definition of text replacement for a <i>MarkObject</i> . <i>MarkObject/DynamicField</i> specifies text replacement within an existing PDL mark.
IdentificationField *	refelement	Specific information about this kind of mark object. See below for information regarding dynamically generated marks.
JobField * New in JDF 1.1	refelement	JobField specifies a dynamically generated mark, also known as a slug line. At most one of <i>LayoutElement</i> , <i>Ord</i> or JobField SHOULD be specified.
<i>LayoutElement</i> ?	refelement	PDL description of the mark. The <i>LayoutElement</i> and <i>Ord</i> are mutually exclusive within one <i>MarkObject</i> . At most one of <i>LayoutElement</i> , <i>Ord</i> or JobField MUST be specified in JDF 1.3 and above.
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.

7.2.108.6 Dynamic marks

JobField, **LayoutElement** 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 one *Ord* or **LayoutElement** is specified, the PDL of the mark is provided by the **RunList** (Marks) or **LayoutElement** 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* MUST 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 neither *Ord* nor **LayoutElement** is present, but **JobField** is present, an Imposition device SHOULD dynamically generate a slug line based on information in **JobField**.
- If none of *Ord*, **LayoutElement** 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.

— Element: DynamicField

Table 7-239: DynamicField element

Name	Data Type	Description
<i>Format</i>	string	Format string in C printf format that defines the replacement. For more information, see "Generating strings with Format and Template" on page 743
<i>InputField</i> ? Deprecated in JDF 1.1	string	String that MUST be replaced by the DynamicInput element in the Contents RunList referenced by <i>Ord</i> or <i>OrdExpression</i> .
<i>Ord</i> ?	integer	Reference to an index in the Contents RunList that contains DynamicInput elements. At most one of <i>Ord</i> or <i>OrdExpression</i> MUST be specified.
<i>OrdExpression</i> ?	string	Expression to calculate the reference to an index in the Contents RunList that contains DynamicInput fields. For details, see the definition of <i>OrdExpression</i> in the description of the PlacedObject element. At most one of <i>Ord</i> or <i>OrdExpression</i> MUST be specified.
<i>ReplaceField</i> ?	string	String that MUST be replaced by the instantiated text expression as defined by the <i>Format</i> and <i>Template</i> attributes in the file referenced by MarkObject/Ord , MarkObject/OrdExpression or MarkObject/LayoutElement . If <i>ReplaceField</i> is not specified, the Device that processes the DynamicField MUST format the DynamicField .
<i>Template</i>	string	Template to define a sequence of variables consumed by <i>Format</i> . For more information, see "Generating strings with Format and Template" on page 743. In addition, the <i>Name</i> attribute of DynamicInput elements of a RunList defines further variables.
DeviceMark ? New in JDF 1.1	refelement	DeviceMark defines the formatting parameters for the mark. If not specified, the DeviceMark settings defined in LayoutPreparationParams or in the Layout tree are assumed.

7.2.108.7 DynamicField Subelement Properties

DynamicField provides a description of dynamic text replacements for a **MarkObject** element. This element is to 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.

Example usage of a DynamicField Element:

```
<RunList Class="Parameter" ID="L3" PartIDKeys="Run" Status="Available">
  <DynamicInput Name="i1">Joe</DynamicInput>
  <DynamicInput Name="i2">John</DynamicInput>
  <LayoutElement ElementType="Graphic">
    <FileSpec URL="File:///Variable.pdf"/>
  </LayoutElement>
</RunList>
<Layout Class="Parameter" ID="Link0003" Status="Available">
```

```

<!--The MarkObject in the Layout hierarchy: -->
<MarkObject CTM="1 0 0 1 0 0">
  <LayoutElement ElementType="Graphic">
    <FileSpec URL="File:///MyReplace.pdf"/>
  </LayoutElement>
  <DynamicField Format="Replacement Text for %s and %s go in here at %s on %s" Ord="0"
ReplaceField="__xxx__" Template="i1,i2,Time,Date"/>
  <DynamicField Format="More Replacement Text for %s go in here" Ord="0"
ReplaceField="__yyy__" Template="SignatureName"/>
</MarkObject>
</Layout>

```

In the example above, 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 *MarkedObject* and Variable.pdf is placed at the position defined by the *CTM* of the *PlacedObject*.

Structure of Signature Subelement

[Deprecated in JDF 1.3](#)

The table defining the deprecated *Signature* subelement has been moved to "Layout Deprecated Subelement" on page 845. All attributes that were defined in *Signature* have been moved into **Layout**.

7.2.109 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:	RunList , Layout/PlacedObject
Example Partition:	<i>PageNumber</i>
Input of processes:	DBDocTemplateLayout , DBTemplateMerging , LayoutElementProduction
Output of processes:	DBDocTemplateLayout , LayoutElementProduction

Table 7-240: LayoutElement resource (Section 1 of 3)

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 <i>ElementType = "Page"</i> that results from the LayoutElementProduction process. The default case is that there is no clip path. <i>ClipPath</i> , <i>SourceClipBox</i> , <i>PlacedObject/@SourceClipPath</i> and <i>PlacedObject/@ClipBox</i> if supplied, MUST be concatenated.
ElementType ? Modified in JDF 1.3	enumeration	Describes the content type for this LayoutElement . See Table 7-241 for values of <i>ElementType</i> .
HasBleeds ? Modified in JDF 1.2	boolean	If " <i>true</i> ", the file has bleeds. If not specified, the set of values of PageList/PageData/@HasBleeds selected by <i>PageListIndex</i> is applied.
IgnorePDLCopies = "false" New in JDF 1.1	boolean	If " <i>true</i> ", any PDL defined copy count MUST be ignored.

Table 7-240: LayoutElement resource (Section 2 of 3)

Name	Data Type	Description
<i>IgnorePDLImposition</i> = "true" New in JDF 1.1	boolean	If "true", any PDL defined imposition definition MUST be ignored. Examples are PDF with embedded PJTF or PPML with a PRINT_LAYOUT. If <i>IgnorePDLImposition</i> = "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.
<i>IsBlank</i> ? 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 <i>PageListIndex</i> is applied. Note that in JDF 1.2 the description erroneously stated that <i>IsBlank</i> = "false" specifies a blank page.
<i>IsPrintable</i> ? Modified in JDF 1.2	boolean	If "true", the file is a PDL file and can be printed. Possible file types include PCL, PDF or PostScript files. Application files such as MS Word have <i>IsPrintable</i> = "false". If not specified, the set of values of PageList/PageData/@IsPrintable selected by <i>PageListIndex</i> is applied.
<i>IsTrapped</i> ? 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 <i>PageListIndex</i> is applied.
<i>PageListIndex</i> ? New in JDF 1.2	Integer-RangeList	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 MUST be specified if <i>PageListIndex</i> is specified.
<i>SourceBleedBox</i> ? 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 <i>PageListIndex</i> is applied.
<i>SourceClipBox</i> ? 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 <i>PageListIndex</i> is applied.
<i>SourceTrimBox</i> ? 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 <i>PageListIndex</i> is applied.
<i>Template</i> ? Modified in JDF 1.2	boolean	<i>Template</i> is "false" when this layout element is self-contained. This attribute is "true" if the LayoutElement represents a template that MUST be completed with information from a database. If not specified, the value of PageList/PageData/@Template is applied.

Table 7-240: **LayoutElement** resource (Section 3 of 3)

Name	Data Type	Description
ColorPool ? New in JDF 1.2	refelement	Definition of the color details.
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 .
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 is applied.

— Attribute: **ElementType**Table 7-241: **ElementType** attribute – possible values (Section 1 of 2)

Value	Description
<i>Auxiliary</i>	Any type of file that is needed to complete a layout but not explicitly displayed, (e.g., ICC profiles or fonts).
<i>Barcode</i>	A barcode. New in JDF 1.3
<i>Composed</i>	Combination of elements that define an element that is not bound to a document page.
<i>Document</i>	An ordered set of one or more pages.
<i>Graphic</i>	Line art.
<i>IdentificationField</i>	A general identification field excluding bar codes. New in JDF 1.3
<i>Image</i>	Bitmap image.
<i>MultiDocument</i>	An ordered set of one or more Documents including document breaks, (e.g., PPML, PPML/VDX, MIME Multipart/Related).
<i>MultiSet</i>	An ordered set of one or more Document sets including document set breaks, (e.g., PPML, PPML/VDX).
<i>Page</i>	Representation of one document page.
<i>Reservation</i>	Empty element. Content for this area of the page might be provided by a subsequent process.
<i>Surface</i>	Representation of an imposed surface.
<i>Text</i>	Formatted or unformatted text.
<i>Tile</i>	Representation of the contents of one tile.

Table 7-241: **ElementType** attribute – possible values (Section 2 of 2)

Value	Description
<i>Unknown</i>	Deprecated in JDF 1.2

— Element: Dependencies[New in JDF 1.2](#)

This element provides a container for dependent references of the **LayoutElement**.

Table 7-242: **Dependencies** element

Name	Data Type	Description
LayoutElement *	refelement	Description of dependent elements, (e.g., fonts, images, etc.).

7.2.110 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. In JDF 1.3 it only contains information for automated production of barcodes. The description of positioning of the graphics is work in progress and therefore not included in this version.

Resource Properties

Resource class:	Parameter
Resource referenced by:	RunList , Layout/PlacedObject
Example Partition:	—
Input of processes:	LayoutElementProduction
Output of processes:	—

Table 7-243: **LayoutElementProductionParams** resource

Name	Data Type	Description
LayoutElementPart *	element	Description of the specific parameters for generating a LayoutElement .

— Element: LayoutElementPart

LayoutElementPart is a generic placeholder for specifying details of **LayoutElementProduction**. In JDF 1.3 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 7-244: **LayoutElementPart** element

Name	Data Type	Description
BarcodeProductionParams *	element	Description of the specific parameters for barcode production.

— Element: BarcodeProductionParams

BarcodeProductionParams describes of the specific parameters for barcode production.

Table 7-245: **BarcodeProductionParams** element

Name	Data Type	Description
BarcodeReproParams ?	refelement	Description of the formatting and reproduction parameters for barcode production.
IdentificationField	refelement	Description of the barcode metadata.

7.2.111 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 specified, the *MinGutter* and *GutterPolicy* attributes of **FitPolicy**, which affect the adjustment of gutter widths, MUST 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
Resource referenced by:	—
Example Partition:	<i>DocIndex, DocRunIndex, RunIndex, SetIndex, SheetName</i>
Input of processes:	LayoutPreparation
Output of processes:	—

Table 7-246: LayoutPreparationParams resource (Section 1 of 8)

Name	Data Type	Description
<i>BindingEdge</i> ? 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. One of: <i>Left</i> <i>Right</i> <i>Top</i> <i>Bottom</i> <i>None</i>
<i>BackMarkList</i> ?	NMTOKENS	List of marks that are to be marked on each back surface. The appearance of the marks are defined by the process implementation. For a list of predefined values, see <i>FrontMarkList</i> .
<i>CreepValue</i> ?	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 7-246: LayoutPreparationParams resource (Section 2 of 8)

Name	Data Type	Description
<i>FinishingOrder</i> = "GatherFold"	enumeration	<p>Specifies the order of operations for finishing a bound booklet created from multiple imposed sheets.</p> <p>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 <i>PageDistributionScheme</i> and <i>FoldCatalog</i> attributes.</p> <p>Possible values are:</p> <p><i>FoldGather</i> – The sheets of a document are first folded according to the value of the <i>FoldCatalog</i> attribute and then gathered on a pile. Usually applies to finishing of perfect-bound documents.</p> <p><i>FoldCollect</i> – The sheets of a document are first folded, according to the value of the <i>FoldCatalog</i> attribute, and then collected on a saddle. Usually applies to finishing of both perfect-bound and saddle-stitched booklets.</p> <p><i>Gather</i> – The sheets of a document are gathered on a pile. No folding is assumed.</p> <p><i>GatherFold</i> – The sheets of a document are first gathered on a pile then folded according to the value of the <i>FoldCatalog</i> attribute. Usually applies to finishing of both perfect-bound and saddle-stitched booklets.</p>
<i>FoldCatalog</i> ?	string	<p>Description of the type of fold that will be applied to all printed sheets according to the folding catalog in the format "Fx-y" as shown in Figure 7-20 and Figure 7-21.</p> <p>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 <i>PageDistributionScheme</i> and <i>FinishingOrder</i> attributes.</p> <p>If not present, no folding other than the folding that is implied by <i>PageDistributionScheme</i> = "Saddle" is assumed.</p>
<i>FoldCatalogOrientation</i> = "Rotate0" New in JDF 1.3	Orientation	<p>This attribute specifies the orientation of how the identified fold catalog entry MUST 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).</p>
<i>FrontMarkList</i> ?	NMTOKENS	<p>List of marks that are to be marked on each front surface. The appearance of the marks are defined by the process implementation. See Table 7-247 for values of <i>FrontMarkList</i>.</p>

Table 7-246: LayoutPreparationParams resource (Section 3 of 8)

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 MUST 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 MUST be interpreted as the starting gutter width of the outermost sheet, otherwise it MUST 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 MUST be set to this value.
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 MUST be between zero and two (2) less than the second value of NumberUp. If not specified, then horizontal gutters MUST 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 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 MUST be set to this value.

Table 7-246: LayoutPreparationParams resource (Section 4 of 8)

Name	Data Type	Description
<i>NumberUp</i> ?	XYPair	<p>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).</p> <p>Compatibility Warning. In JDF 1.1 rows and columns were erroneously switched in the description.</p> <p>The relative positioning of the page cells within the multi-up grid are defined by the explicit or implied values of the <i>Gutter</i>, <i>HorizontalCreep</i>, <i>VerticalCreep</i> and <i>CreepValue</i> attributes.</p> <p>The distribution of content pages from the content RunList into the page cells is defined by the explicit or implied values of the <i>PageDistributionScheme</i>, <i>PresentationDirection</i>, <i>Sides</i>, <i>FinishingOrder</i> and <i>FoldCatalog</i> attributes and the implicit number of sheets comprising the bound component.</p>
<i>PageDistributionScheme</i> = "Sequential"	NMTOKEN	<p>Specifies how finished pages are to be distributed onto a multi-up grid of finished PageCell elements defined by the values of the <i>NumberUp</i> attribute. Possible values include:</p> <p><i>Saddle</i> – 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 is to be applied only to odd-numbered vertical gutters where any even-numbered gutters is to automatically creep in the opposite direction.</p> <p><i>Perfect</i> – 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.</p> <p><i>Sequential</i> – The finished pages are distributed onto the multi-up layout according to the value of the <i>PresentationDirection</i> attribute. Note that page distribution ordering for both <i>Saddle</i> and <i>Perfect</i> also 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 <i>FoldCatalog</i> and <i>FinishingOrder</i> attributes.</p> <p>Note: The <i>NumberUp</i> attribute MUST always specify a multi-up layout appropriate for a given finished page distribution ordering and <i>FoldCatalog</i>. Setting this attribute does not imply the multi-up grid dimensions are appropriate for the selected page distribution scheme.</p> <p>Note: In all cases, the order of finished pages as represented by the content RunList MUST be either in reader order or in an order appropriate for multi-up saddle stitching. Refer to the <i>PageOrder</i> attribute.</p>

Table 7-246: LayoutPreparationParams resource (Section 5 of 8)

Name	Data Type	Description															
<i>PageOrder</i> = "Reader"	NMTOKEN	<p>The assumed ordering of the finished pages in the RunList.</p> <p><i>Booklet</i> – The finished pages are ordered in the RunList and MUST be processed exactly in the order as specified by <i>PresentationDirection</i>. <i>NumberUp</i> MUST still be set to the appropriate value and is not implied by specifying <i>PageOrder</i> = "Booklet". <i>PageOrder</i> = "Booklet" MUST NOT be used in conjunction with <i>FoldCatalog</i>.</p> <p><i>Reader</i> – The finished pages are in reader order in the RunList.</p>															
<i>PresentationDirection</i> ?	enumeration	<p>Indicates the order in which finished pages will be distributed into the page cells of the <i>NumberUp</i> layout. If <i>PageDistributionScheme</i> = "Saddle", <i>PresentationDirection</i> applies to sets of two adjacent pages. This allows positioning of multiple page pairs for SaddleStitching onto one sheet. Possible values are:</p> <p><i>FoldCatalog</i> – Finished pages are imaged so that the result is compatible with a finished product produced from the folding catalog as specified in <i>FoldCatalog</i>.</p> <p><i>XYZ</i> – 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.</p> <ul style="list-style-type: none"> • <i>X</i> – Specifies flowing left to right across a sheet surface. • <i>x</i> – Specifies flowing right to left across a sheet surface. • <i>Y</i> – Specifies flowing bottom to top vertically across a sheet surface. • <i>y</i> – Specifies flowing top to bottom vertically across a sheet surface. • <i>Z</i> – Specifies flowing bottom of stack to top of it through the stack. • <i>z</i> – Specifies flowing top of stack to bottom of it through the stack. <p>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</p> <p>The following table specifies how cells are ordered on a simplex 4-up layout for a 2-sheet stack depending on <i>PresentationDirection</i>. 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.</p> <table border="1" data-bbox="704 1682 1422 1789"> <thead> <tr> <th><i>Xyz</i></th> <th><i>xyz</i></th> <th><i>XYZ</i></th> <th><i>Zxy</i></th> <th><i>yxZ</i></th> </tr> </thead> <tbody> <tr> <td>1 2 5 6</td> <td>2 1 6 5</td> <td>7 8 3 4</td> <td>4 2 3 1</td> <td>7 5 3 1</td> </tr> <tr> <td>3 4 7 8</td> <td>4 3 8 7</td> <td>5 6 1 2</td> <td>8 6 7 5</td> <td>8 6 4 2</td> </tr> </tbody> </table>	<i>Xyz</i>	<i>xyz</i>	<i>XYZ</i>	<i>Zxy</i>	<i>yxZ</i>	1 2 5 6	2 1 6 5	7 8 3 4	4 2 3 1	7 5 3 1	3 4 7 8	4 3 8 7	5 6 1 2	8 6 7 5	8 6 4 2
<i>Xyz</i>	<i>xyz</i>	<i>XYZ</i>	<i>Zxy</i>	<i>yxZ</i>													
1 2 5 6	2 1 6 5	7 8 3 4	4 2 3 1	7 5 3 1													
3 4 7 8	4 3 8 7	5 6 1 2	8 6 7 5	8 6 4 2													

Table 7-246: LayoutPreparationParams resource (Section 6 of 8)

Name	Data Type	Description
<i>Rotate</i> = "Rotate0"	enumeration	<p>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. One of:</p> <p><i>Rotate0</i></p> <p><i>Rotate90</i> – 90° counterclockwise rotation.</p> <p><i>Rotate180</i> – 180° rotation.</p> <p><i>Rotate270</i> – 90° clockwise rotation.</p> <p>For details of orthogonal rotations, refer to Table 2-4, “Matrices and Orientation values for describing the orientation of a Component,” on page 26. If a <i>RotatePolicy</i> value other than “NoRotate” is specified in, the value specified in <i>Rotate</i> can be modified accordingly.</p> <p>Note: A rotation of the grid also rotates the gutters, (i.e., it is applied after all other parameters have been evaluated and applied).</p>
<i>Sides</i> = "OneSidedFront"	enumeration	<p>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, <i>Sides</i> applies to the entire unfolded sheet.</p> <p>When a different value for the <i>Sides</i> attribute is encountered, it MUST force a new sheet. However, when the same value for the <i>Sides</i> attribute is restated for consecutive pages, it is the same as if that restatement was not present. Possible values are:</p> <p><i>OneSidedBackFlipX</i> – 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.</p> <p><i>OneSidedBackFlipY</i> – 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.</p> <p><i>OneSidedFront</i> – Page content is imaged on the front side of media.</p> <p><i>TwoSidedFlipX</i> – 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”.</p> <p><i>TwoSidedFlipY</i> – 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”.</p>
<i>StackDepth</i> ?	integer	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.

Table 7-246: LayoutPreparationParams resource (Section 7 of 8)

Name	Data Type	Description																							
StepDocs ? Modified in JDF 1.2	XYPair	<p>A list of two integers that specifies 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 <i>NumberUp</i> MUST be an integer multiple of <i>StepRepeat</i> * <i>StepDocs</i>. 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 <i>PresentationDirection</i> = "XYZ", <i>NumberUp</i> = "4 4" and <i>StepRepeat</i> = "2 2" and <i>StepDocs</i> =:</p>																							
		<table border="1"> <tr> <td>"2 1" (2 documents in X, 1 in Y)</td> <td>"1 2" (1 document in X, 2 in Y)</td> </tr> <tr> <td>A1 A1 B1 B1</td> <td>A1 A1 A2 A2</td> </tr> <tr> <td>A1 A1 B1 B1</td> <td>A1 A1 A2 A2</td> </tr> <tr> <td>A2 A2 B2 B2</td> <td>B1 B1 B2 B2</td> </tr> <tr> <td>A2 A2 B2 B2</td> <td>B1 B1 B2 B2</td> </tr> </table>	"2 1" (2 documents in X, 1 in Y)	"1 2" (1 document in X, 2 in Y)	A1 A1 B1 B1	A1 A1 A2 A2	A1 A1 B1 B1	A1 A1 A2 A2	A2 A2 B2 B2	B1 B1 B2 B2	A2 A2 B2 B2	B1 B1 B2 B2													
"2 1" (2 documents in X, 1 in Y)	"1 2" (1 document in X, 2 in Y)																								
A1 A1 B1 B1	A1 A1 A2 A2																								
A1 A1 B1 B1	A1 A1 A2 A2																								
A2 A2 B2 B2	B1 B1 B2 B2																								
A2 A2 B2 B2	B1 B1 B2 B2																								
StepRepeat ?	IntegerList	<p>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 <i>NumberUp</i> MUST be an integer multiple of <i>StepRepeat</i> * <i>StepDocs</i>. 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 <i>PresentationDirection</i> = "XYZ", <i>NumberUp</i> = "4 4" and <i>StepRepeat</i> =:</p>																							
		<table border="1"> <tr> <td>"2 2 1"</td> <td>"-2 2 1"</td> <td>"-2 -2 1"</td> <td>"2 -2 1"</td> <td>"1 4 1"</td> </tr> <tr> <td>1 1 2 2</td> <td>1 2 1 2</td> <td>1 2 1 2</td> <td>1 1 2 2</td> <td>1 2 3 4</td> </tr> <tr> <td>1 1 2 2</td> <td>1 2 1 2</td> <td>3 4 3 4</td> <td>3 3 4 4</td> <td>1 2 3 4</td> </tr> <tr> <td>3 3 4 4</td> <td>3 4 3 4</td> <td>1 2 1 2</td> <td>1 1 2 2</td> <td>1 2 3 4</td> </tr> <tr> <td>3 3 4 4</td> <td>3 4 3 4</td> <td>3 4 3 4</td> <td>3 3 4 4</td> <td>1 2 3 4</td> </tr> </table>	"2 2 1"	"-2 2 1"	"-2 -2 1"	"2 -2 1"	"1 4 1"	1 1 2 2	1 2 1 2	1 2 1 2	1 1 2 2	1 2 3 4	1 1 2 2	1 2 1 2	3 4 3 4	3 3 4 4	1 2 3 4	3 3 4 4	3 4 3 4	1 2 1 2	1 1 2 2	1 2 3 4	3 3 4 4	3 4 3 4	3 4 3 4
"2 2 1"	"-2 2 1"	"-2 -2 1"	"2 -2 1"	"1 4 1"																					
1 1 2 2	1 2 1 2	1 2 1 2	1 1 2 2	1 2 3 4																					
1 1 2 2	1 2 1 2	3 4 3 4	3 3 4 4	1 2 3 4																					
3 3 4 4	3 4 3 4	1 2 1 2	1 1 2 2	1 2 3 4																					
3 3 4 4	3 4 3 4	3 4 3 4	3 3 4 4	1 2 3 4																					
SurfaceContentsBox ? Modified in JDF 1.1A	rectangle	<p>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". <i>SurfaceContentsBox</i> 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 the Media is implied.</p>																							

Table 7-246: LayoutPreparationParams resource (Section 8 of 8)

Name	Data Type	Description
<i>VerticalCreep</i> ?	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 <i>NumberUp</i> . The value for an entry in this list MUST be between zero and two (2) less then the first value of <i>NumberUp</i> . An index value outside of this range is ignored. If not specified then vertical gutters MUST 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 ?	refelement	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 ?	refelement	Details how to fit the grid of <i>PageCell</i> elements onto the <i>SurfaceContentsBox</i> .
<i>ImageShift</i> ?	element	Details how to place the grid of <i>PageCell</i> elements into the <i>SurfaceContentsBox</i> . <i>ImageShift</i> MUST be applied before any transformations of the grid of <i>PageCell</i> elements as specified by <i>Rotate</i> 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 <i>ImageShift</i> will generally be required to allow for space when <i>CreepValue</i> is positive.
InsertSheet *	refelement	Additional sheets to be inserted before, after or within a job.
JobField *	refelement	Specific information about this kind of mark object.
Media ?	refelement	Specific information about the media.
<i>PageCell</i> ? Modified in JDF 1.1A	element	<i>PageCell</i> elements describe how page contents will be imaged onto individual page cells. At most one <i>PageCell</i> MUST be specified and it is applied to all page cells on both surfaces of a sheet.

— Attribute: FrontMarkList**Table 7-247: FrontMarkList attribute – possible values (Section 1 of 2)**

Value	Description	Value	Description
<i>CIELABMeasuringField</i>		<i>IdentificationField</i>	

Table 7-247: FrontMarkList attribute – possible values (Section 2 of 2)

Value	Description	Value	Description
<i>ColorControlStrip</i>		<i>JobField</i>	
<i>ColorRegisterMark</i>		<i>PaperPathRegisterMark</i>	
<i>CutMark</i>		<i>RegisterMark</i>	
<i>DensityMeasuringField</i>		<i>ScavengerArea</i>	

— Element: PageCell**Table 7-248: PageCell element (Section 1 of 2)**

Name	Data Type	Description
<i>Border ?</i> Modified in JDF 1.1A	double	<p>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 <i>TrimSize</i>. A value of "0" specifies no border.</p> <p>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 <i>TrimSize</i> attribute. The border marks MUST NOT overwrite the page contents of the trimmed page. Note that when the page cells are distributed evenly over the area of the <i>SurfaceContentsBox</i>, the page cells position and/or size can be adjusted to accommodate the border.</p> <p>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 <i>TrimSize</i> attribute. The border marks MAY overwrite the page contents of the trimmed page.</p> <p>The rectangle defined by the inside edge of the border defines a <i>ClipBox</i> beyond which no content will be imaged.</p>
<i>ClipBox ?</i>	rectangle	<p>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 <i>TrimSize</i> attribute. Page content data imaged outside of the region defined by this rectangle MUST be clipped. If <i>ClipBox</i> is larger than <i>TrimSize</i>, 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 <i>TrimSize</i>.</p>
<i>MarkList ?</i>	NMTOKENS	<p>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:</p> <p><i>CIELABMeasuringField</i> <i>ColorControlStrip</i> <i>ColorRegisterMark</i> <i>CutMark</i> <i>DensityMeasuringField</i> <i>IdentificationField</i> <i>JobField</i> <i>PaperPathRegisterMark</i> <i>RegisterMark</i> <i>ScavengerArea</i></p>

Table 7-248: PageCell element (Section 2 of 2)

Name	Data Type	Description
<i>Rotate</i> = "Rotate0"	enumeration	Orthogonal rotation to be applied to the contents in each page cell. One of: <i>Rotate0</i> <i>Rotate90</i> – 90° counterclockwise rotation. <i>Rotate180</i> – 180° rotation. <i>Rotate270</i> – 90° clockwise rotation. For details of orthogonal rotation, refer to Table 2-4, “Matrices and Orientation values for describing the orientation of a Component,” on page 26. If a <i>RotatePolicy</i> value other than "NoRotate" is specified, the value specified in <i>Rotate</i> can 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, <i>TrimSize</i> is calculated by subtracting the gutters from the LayoutPreparationParams/@SurfaceContentsBox and dividing by the appropriate <i>NumberUp</i> value.
Color ?	refelement	Color of the border.
DeviceMark ?	refelement	Details how device dependent marks are to be generated. Defaults to the value of DeviceMark in the parent LayoutPreparationParams .
FitPolicy ?	refelement	Details how page content is fit into the page cells. If the dimensions of the page contents vary, <i>FitPolicy</i> is applied to the contents of each cell individually.
ImageShift ?	element	Element which describes how content is to be placed into the page cells. X and Y are specified in the coordinate system of the PageCell .

— Element: 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 7-249: ImageShift element (Section 1 of 2)

Name	Data Type	Description
<i>PositionX ?</i>	enumeration	Indicates how content is to be positioned horizontally. The <i>ShiftBack</i> and <i>ShiftFront</i> are applied after <i>PositionX</i> and <i>PositionY</i> . Values are: <i>Center</i> – Center the content horizontally without regard to limitations of the receiving container. <i>Left</i> – Position the left edge of the content so that it is coincident with the left edge of the receiving container. <i>Right</i> – Position the right edge of the content so that it is coincident with the right edge of the receiving container. <i>Spine</i> – Position the content so that it is coincident with the vertical binding edge of the receiving container. New in JDF 1.2 <i>None</i> – Place the content wherever the print data specify. Deprecated in JDF 1.3

Table 7-249: ImageShift element (Section 2 of 2)

Name	Data Type	Description
<i>PositionY?</i>	enumeration	Indicates how content is to be positioned vertically. The <i>ShiftBack</i> and <i>ShiftFront</i> are applied after <i>PositionX</i> and <i>PositionY</i> . Values are: <i>Bottom</i> – Position the bottom edge of the content so that it is coincident with the bottom edge of the receiving container. <i>Center</i> – Center the content horizontally without regard to limitations of the receiving container. <i>Top</i> – Position the top edge of the content so that it is coincident with the top edge of the receiving container. <i>Spine</i> – Position the content so that it is coincident with the horizontal binding edge of the receiving container. New in JDF 1.2 <i>None</i> – Place the content wherever the print data specify. Deprecated in JDF 1.3
<i>ShiftBack?</i>	XYPair	The amount in X and Y direction by which the content is to be shifted on the back side of the receiving container. If not specified, <i>ShiftBack</i> MUST be calculated from <i>ShiftFront</i> so that the content remains aligned.
<i>ShiftFront</i> ="0 0"	XYPair	The amount in X and Y direction by which the content is to be shifted on the front side of the receiving container.

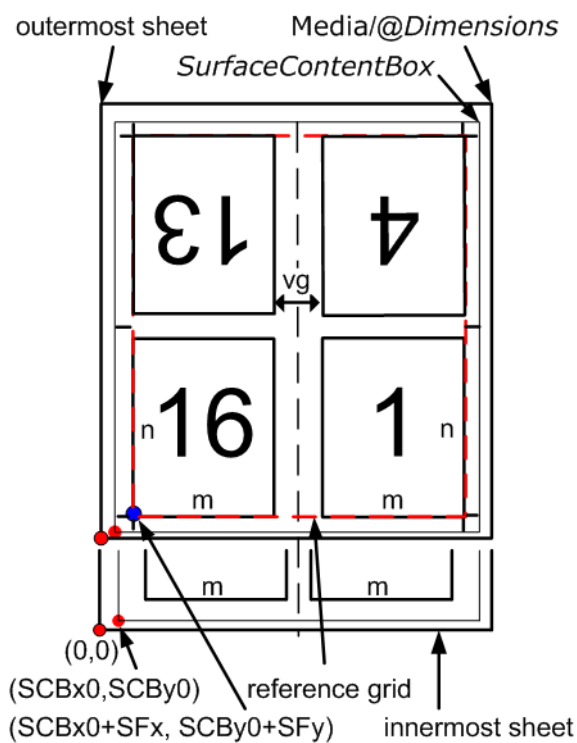


Figure 7-28: Diagram of a 4-up cross-folded saddle-stitch imposition with vertical gutter creep

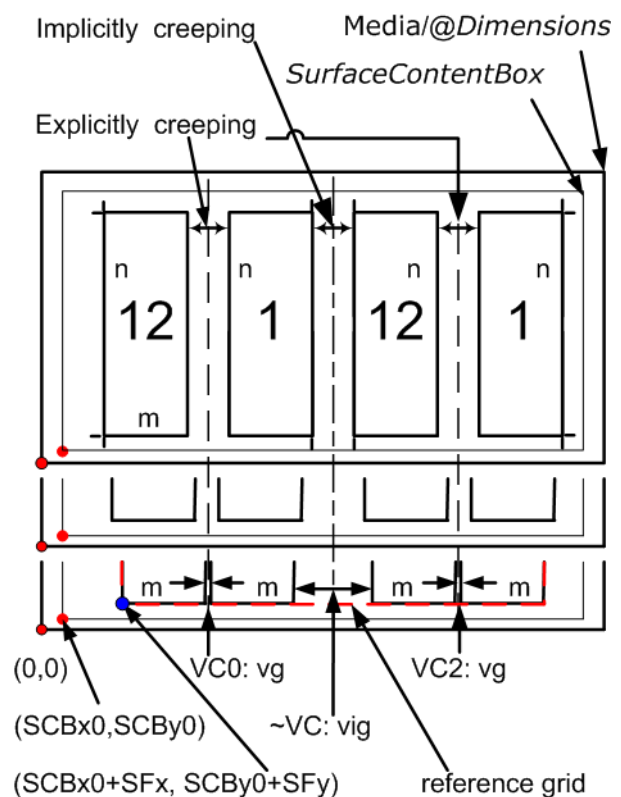


Figure 7-29: Diagram of a step-and-repeat 2-up saddle-stitch imposition with vertical spine gutter creep

The following terms are used in Figure 7-28

- **reference grid** in Figure 7-28 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 defined in the JDF below.

Figure 7-28 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.

```
<JDF ID="ID" Type="ProcessGroup" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" Version="1.3">
<ResourcePool>
<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="SCBx0 SCBy0 SCBx1 SCBy1"
VerticalCreep="0" GutterMinimumLimit="hml vml" CreepValue="0 -vc"
Gutter="hg vg" FinishingOrder="FoldCollect" FrontMarkList="CutMark"
BindingEdge="Left">
<PageCell TrimSize="m n">
<ImageShift PositionX="Spine" PositionY="Center" />
</PageCell>
<ImageShift PositionY="Bottom" PositionX="Left" ShiftFront="SFx SFy"/>
</LayoutPreparationParams>
</ResourcePool>
</JDF>
```

The following terms are used in Figure 7-29

- **reference grid** in Figure 7-29 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 defined in the JDF below.

Figure 7-29 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.

```
<LayoutPreparationParams Class="Parameter" ID="LPP_1" NumberUp="4 1"
PageDistributionScheme="Saddle" FoldCatalog="F4-1"
<!-- Note: folding pattern F4-1 applies to each of the two 2x1 signatures -->
FoldCatalogOrientation="Flip0" Sides="TwoSidedFlipY" StepRepeat="2 1 1"
<!-- Note: step and repeat by two in X direction logically divides grid into two 2x1
signatures-->
SurfaceContentsBox="SCBx0 SCBx1 SCBy0 SCBy1" VerticalCreep="0 2"
<!-- Note: first (VC0) and third (VC2) vertical gutters are explicitly creeping
and the rest (~VC) are implicitly creeping-->
ImplicitGutter="0 vig" ImplicitGutterMinimumLimit="0 vig1" CreepValue="0 +vc"
<!-- Note: Positive vertical creep value indicates initial gutter widths of inner
most sheet>
Gutter="0 vg" FinishingOrder="GatherFold" FrontMarkList="CutMark">
<!--Note: cut marks are located relative to largest page cell grid trim box-->
<PageCell TrimSize="m n">
<ImageShift PositionX="Spine" PositionY="Bottom"/>
</PageCell>
<ImageShift PositionY="Bottom" PositionX="Left" ShiftFront="SFx SFy"/>
</LayoutPreparationParams>
```

7.2.112 LongitudinalRibbonOperationParams

[Deprecated in JDF 1.1](#). See "LongitudinalRibbonOperationParams" on page 845 for details of this deprecated resource.

7.2.113 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
Resource referenced by:	—
Example Partition:	—
Input of processes:	ManualLabor
Output of processes:	—

Table 7-250: ManualLaborParams resource

Name	Data Type	Description
<i>LaborType</i> Modified in JDF 1.3	NMTOKEN	<p>List of types of manual labor that are performed. Values include:</p> <p><i>CreateCoatingForm</i> – create a form to apply coatings during or after printing</p> <p><i>EditArt</i> – Unspecific Art editing (for work on specific files LayoutElementProduction is to be used)</p> <p><i>EditMarks</i> – Marks editing</p> <p><i>EditTraps</i> – Traps editing</p> <p><i>ManageJob</i> – General work on the job.</p> <p><i>PhoneCallToCustomer</i> – Phone calls to ask/inform the Customer.</p> <p>Note that the data type was erroneously specified as NMTOKENS prior to JDF 1.3.</p>

7.2.114 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:	ExposedMedia, DigitalPrintingParams, InsertSheet, InterpretingParams, LayoutPreparationParams, RenderingParams, Sheet, StrippingParams, Tile
Example Partition:	<i>Location, SheetName, Side, SignatureName, TileID, WebName</i>
Input of processes:	Bending, ConventionalPrinting, ContactCopying, Cutting, DigitalPrinting, ImageSetting, Proofing
Output of processes:	—

Table 7-251: Media resource (Section 1 of 6)

Name	Data Type	Description
<i>BackCoatings?</i>	enumeration	Enumeration options are identical to <i>FrontCoatings</i> (see below), but applied to the back surface of the media. When not specified, defaults to the value of <i>FrontCoatings</i> .

Table 7-251: Media resource (Section 2 of 6)

Name	Data Type	Description
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, <i>BackGlossValue</i> defaults to the value of <i>FrontGlossValue</i> .
Brightness ?	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 paper or board in percent to the nearest 0.5% reflectance factor. If one value is specified, Brightness applies to the front and back. If two values are specified the first value applies to the front and the second applies to the back. See also <i>CIEWhiteness</i> .
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 <i>ColorName</i> color SHOULD match the <i>Name</i> attribute of a <i>Color</i> defined in a ColorPool resource that is linked to the process using this Media resource. In JDF 1.2 and beyond, use <i>MediaColorName</i> and <i>MediaColorNameDetails</i> .
CoreWeight ? New in JDF 1.3	double	Weight of the core of a roll, in grams [g]
Dimension ? Modified in JDF 1.1	XYPair	The X and Y dimensions of the chosen medium, measured in points. The X, Y values of <i>Dimension</i> 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 <i>Roll</i> media, the X coordinate specifies the reel width and the Y coordinate specifies the length of the web in points. If a <i>Dimension</i> coordinate is unknown, the value MUST 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).
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.
FluteDirection ? New in JDF 1.3	enumeration	Direction of the fluting. Possible values are: <i>LongEdge</i> – Along the longer axis as defined by <i>Dimension</i> . <i>ShortEdge</i> – Along the shorter axis as defined by <i>Dimension</i> . <i>XDirection</i> – Along the X-axis of the Media coordinate system <i>YDirection</i> – Along the Y-axis of the Media coordinate system

Table 7-251: Media resource (Section 3 of 6)

Name	Data Type	Description
FrontCoatings ? Modified in JDF 1.3	enumeration	What preprocess coating has been applied to the front surface of the media. Possible values are: <i>None</i> – No coating. <i>Coated</i> – A coating of a system-specified type. New in JDF 1.2 <i>Glossy</i> <i>HighGloss</i> <i>InkJet</i> – A coating intended for use with inkjet technology. New in JDF 1.2 <i>Matte</i> <i>Polymer</i> – Coating for a photo polymer process New in JDF 1.3 <i>Silver</i> – Coating for a silver halide process New in JDF 1.3 <i>Satin</i> <i>Semigloss</i>
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 ?	integer	The intended <i>Grade</i> of the media on a scale of 1 through 5. The <i>Grade</i> is ignored if <i>MediaType</i> is not "Paper". <i>Grade</i> of paper material is defined in accordance with the paper "types" set forth in [ISO12647-2:2004]. Offset printing paper types are defined with the following integer values: 1 – Gloss-coated paper. 2 – Matt-coated paper. 3 – Gloss-coated, web paper. 4 – Uncoated, white paper. 5 – Uncoated, yellowish paper. Note: [ISO12647-2:2004] paper <i>type</i> attribute values do NOT align with U.S. GRACOL paper <i>grade</i> attribute values, (e.g., [ISO12647-2:2004] type 1 does not equal U.S. GRACOL grade 1).
GrainDirection ? New in JDF 1.1 Modified in JDF 1.3	enumeration	Direction of the grain in the coordinate system defined by <i>Dimension</i> . Possible values are: <i>LongEdge</i> – Along the longer axis as defined by <i>Dimension</i> . <i>ShortEdge</i> – Along the shorter axis as defined by <i>Dimension</i> . <i>XDirection</i> – Along the X-axis of the Media coordinate system. New in JDF 1.3 <i>YDirection</i> – 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 <i>HoleType</i> , <i>Hole</i> or <i>HoleLine</i> , which includes the number of holes.

Table 7-251: Media resource (Section 4 of 6)

Name	Data Type	Description
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 "JDF/CIP4 Hole Pattern Catalog" on page 761. Allowed values are: <i>None</i> – No holes. <i>Explicit</i> – Holes are defined in an array of <i>Hole</i> or <i>HoleLine</i> . other values defined in "JDF/CIP4 Hole Pattern Catalog" on page 761.
ImagableSide ?	enumeration	Side of the chosen medium that are to be marked. Possible values are: <i>Front</i> <i>Back</i> <i>Both</i> <i>Neither</i>
InsideLoss ? New in JDF 1.3	double	The inside loss of corrugated board material in microns [µm]. Note: <i>InsideLoss</i> + <i>OutsideGain</i> need not be exactly equal to thickness.
LabColorValue ? New in JDF 1.2	LabColor	<i>LabColorValue</i> is the CIELAB color value of the media, computed as specified in [TAPPI T527].
MediaColorName ? Modified in JDF 1.1	NamedColor	A name for the color. Allowed values are defined in Section A.3.3.2, <i>NamedColor</i> . If more specific, specialized or site-defined media color names are needed, use <i>MediaColorNameDetails</i> .
MediaColorNameDetails ? New in JDF 1.2	string	A more specific, specialized or site-defined name for the media color. If <i>MediaColorNameDetails</i> is supplied, <i>MediaColorName</i> MUST also be supplied.
MediaSetCount ?	integer	When the input media is grouped in sets, identifies the number of pieces of media in each set. For example, if the <i>MediaTypeDetails</i> is " <i>PreCutTabs</i> ", a <i>MediaSetCount</i> of "5" would indicate that each set includes five tab sheets.
MediaType ? Modified in JDF 1.3	enumeration	Describes the medium being employed. Possible values are: <i>CorrugatedBoard</i> New in JDF 1.3 <i>Disc</i> – CD or DVD disc to be printed on. <i>EndBoard</i> – End board used in the Bundling process. <i>EmbossingFoil</i> <i>Film</i> <i>Foil</i> <i>GravureCylinder</i> – Gravure cylinder. New in JDF 1.3 <i>ImagingCylinder</i> – reusable direct imaging cylinder in a press. New in JDF 1.3 <i>LaminatingFoil</i> <i>Other</i> – Not one of the defined values. <i>Paper</i> <i>Plate</i> <i>SelfAdhesive</i> – New in JDF 1.3 <i>ShrinkFoil</i> <i>Transparency</i> <i>Unknown</i> – Deprecated in JDF 1.2

Table 7-251: Media resource (Section 5 of 6)

Name	Data Type	Description
<i>MediaTypeDetails</i> ? Modified in JDF 1.3	NMTOKEN	Additional details of the chosen medium. If <i>MediaTypeDetails</i> is specified, <i>MediaType</i> MUST be specified. See Table 7-252, “MediaTypeDetails attribute – possible values,” on page 520.
<i>MediaUnit</i> = “Sheet” Modified in JDF 1.2	enumeration	Describes the format of the media as it is delivered to the device. Possible values are: <i>Continuous</i> – Continuously connected sheets which can be fan folded. New in JDF 1.2 <i>Roll</i> <i>Sheet</i> – Individual cut sheets.
<i>Opacity</i> ? Modified in JDF 1.2	enumeration	The opacity of the media. See <i>OpacityLevel</i> to specify the degree of opacity for any of these values. Possible values are: <i>Opaque</i> – The media is opaque. With two-sided printing the printing on the other side does not show through under normal incident light. <i>Translucent</i> – The media is translucent to a system specified amount. For example, translucent media can be used for back lit viewing. New in JDF 1.2 <i>Transparent</i> – The media is transparent.
<i>OpacityLevel</i> ? 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.
<i>OuterCoreDiameter</i> ? New in JDF 1.3	double	Specifies the outer diameter of the core of a roll, in points.
<i>OutsideGain</i> ? New in JDF 1.3	double	The outside gain of corrugated board material in microns [μm].
<i>PlateTechnology</i> ? New in JDF 1.3	enumeration	Exposure technology of the plates. <i>InkJet</i> – Exposure with inkjet technology. Note that <i>FrontCoatings</i> = “Inkjet” specifies inkjet specific coating of paper or transparency Media, not of plates. <i>Thermal</i> – Thermal exposure <i>UV</i> – Ultraviolet exposure <i>Visible</i> – Visible light exposure
<i>Polarity</i> ?	enumeration	Polarity of the chosen medium. Possible values are: <i>Positive</i> <i>Negative</i>
<i>PrePrinted</i> = “false”	boolean	Indicates whether the media is preprinted.
<i>Recycled</i> ? 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 <i>RecycledPercentage</i> .
<i>RecycledPercentage</i> ? New in JDF 1.2	double	The percentage, between 0 and 100, of recycled material that the media is to contain.
<i>RollDiameter</i> ?	double	Specifies diameter of a roll, in points.

Table 7-251: Media resource (Section 6 of 6)

Name	Data Type	Description
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.
StockType ? New in JDF 1.1	NMTOKEN	Strings describing the available stock. Examples include: <i>Bristol</i> <i>Cover</i> <i>Bond</i> <i>Newsprint</i> <i>Index</i> <i>Offset</i> – This includes book stock. <i>Tag</i> <i>Text</i>
Texture ? New in JDF 1.1 Modified in JDF 1.2	NMTOKEN	The intended texture of the media. Examples include: <i>Antique</i> – Rougher than vellum surface. <i>Calendared</i> – Extra smooth or polished, uncoated paper. <i>Linen</i> – Texture of coarse woven cloth. <i>Smooth</i> <i>Stipple</i> – Fine pebble finish. <i>Uncalendared</i> – Rough, unpolished and uncoated papers. New in JDF 1.2 <i>Vellum</i> – Slightly rough surface.
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. The <i>UserMediaType</i> has been merged into <i>MediaTypeDetails</i> in JDF 1.1.
Weight ?	double	Weight of the chosen medium, measured in grams per square meter [g/m^2]. See "North American and Japanese Media Weight Explained" on page 721 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]
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 <i>LabColorValue</i> .
MediaLayers ? New in JDF 1.3	element	Subelement describing the layer structure of media such as corrugated or self adhesive materials.

— Attribute: MediaTypeDetails

Table 7-252: MediaTypeDetails attribute – possible values

Value	Description
<i>Aluminum</i>	Conventional or CtP press plate. Modified in JDF 1.3
<i>Cardboard</i>	
<i>CD</i>	CD disc to be printed on. New in JDF 1.3
<i>ContinuousLong</i>	Continuously connected sheets of an opaque material connected along the long edge.
<i>ContinuousShort</i>	Continuously connected sheets of an opaque material connected along the short edge.
<i>CtPVisiblePhotoPolymer</i>	Visible light CtP plate with photo polymer process. Deprecated in JDF 1.3
<i>CtPVisibleSilver</i>	Visible light CtP plate with silver halide process. Deprecated in JDF 1.3
<i>CtPThermal</i>	Thermal CtP plate. Deprecated in JDF 1.3
<i>DoubleWall</i>	Double wall corrugated board New in JDF 1.3
<i>DVD</i>	DVD disc to be printed on. New in JDF 1.3
<i>DryFilm</i>	
<i>Envelope</i>	Envelopes that can be used for conventional mailing purposes.
<i>EnvelopePlain</i>	Envelopes that are not preprinted and have no windows.
<i>EnvelopeWindow</i>	Envelopes that have windows for addressing purposes.
<i>Flute</i>	Flute layer of a corrugated board
<i>FullCutTabs</i>	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.
<i>ImageSetterPaper</i>	Contact paper as replacement for film.
<i>Labels</i>	Label stock, (e.g., a sheet of peel-off labels).
<i>Letterhead</i>	Separately cut sheets of an opaque material including a letterhead.
<i>MultiLayer</i>	Form medium composed of multiple layers which are preattached to one another, (e.g., for use with impact printers).
<i>MultiPartForm</i>	Form medium composed of multiple layers not preattached to one another; each sheet might be drawn separately from an input source.
<i>Photographic</i>	Separately cut sheets of an opaque material to produce photographic quality images.
<i>PlateUV</i>	Press plate for the UV process. Deprecated in JDF 1.3
<i>Polyester</i>	Conventional or CtP press plate. Modified in JDF 1.3
<i>PreCutTabs</i>	Media with tabs that are cut so that more than one tab is visible extending out beyond the edge of non-tabbed media.
<i>SingleFace</i>	Single face corrugated board New in JDF 1.3
<i>SingleWall</i>	Single wall corrugated board New in JDF 1.3
<i>Stationery</i>	Separately cut sheets of an opaque material, includes generic paper.
<i>TabStock</i>	Media with tabs, either precut or full-cut.–
<i>Tractor</i>	Tractor feed with holes.
<i>TripleWall</i>	Triple wall corrugated board New in JDF 1.3
<i>WetFilm</i>	Conventional photographic film.

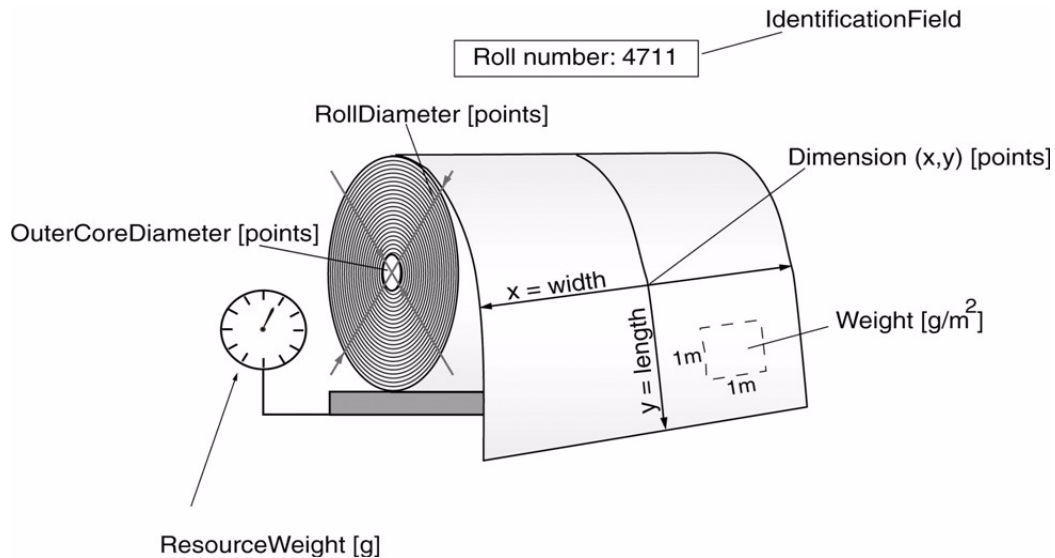


Figure 7-30: a paper-roll with some roll-specific information

— Element: **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.

Table 7-253: **MediaLayers** element

Name	Data Type	Description
GlueLine *	refelement	GlueLine element describing a glue layer of a layered Media resource. Each GlueLine element MUST have GlueLine/@AreaGlue = "true" .
Media *	refelement	Media elements describing a layer of a layered Media resource.

7.2.114.1 Corrugated Media:

Corrugated material consists of multiple sheets of paper (called liners) with fluted material in between. For background information on Corrugated Media, see <http://cpc.corrugated.org/Basics>. 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 <http://cpc.corrugated.org/Basics/BasicAllAbout.aspx>

InsideLoss, OutsideGain

InsideLoss, OutsideGain: 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.)

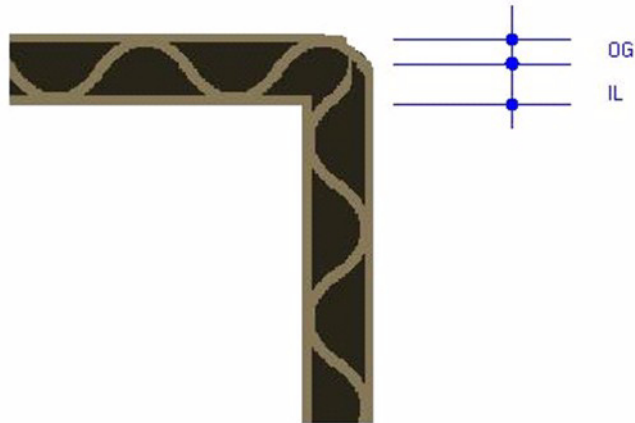


Figure 7-31: InsideLoss, OutsideGain

Example of Corrugated Media:

```
<Media Class="Consumable" ID="123456" 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="Clay"/>
    <!-- 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>
```

7.2.114.2 Self adhesive Media

Self adhesive media is described as **MediaLayer** elements with nested **Media** and **GlueLine** elements.

Example of self adhesive Media:

```
<Media Class="Consumable" ID="123456" 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="Permanent"
      GlueBrand="Uhu"/>
    <!-- Back -->
    <Media DescriptiveName="Blue Glassine 50" MediaType="Paper" Weight="50"/>
  </MediaLayers>
</Media>
```

7.2.115 MediaSource

[Deprecated in JDF 1.1](#) See "MediaSource" on page 846 for details of this deprecated resource.

7.2.116 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**'s 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
Resource referenced by:	—
Example Partition:	—
Input of processes:	Any
Output of processes:	—

Table 7-254: MiscConsumable resource

Name	Data Type	Description
<i>ConsumableType</i> ?	NMTOKEN	Identifies the type of MiscConsumable (machine-readable). A human-readable (possibly localized) description of the consumable SHOULD also be supplied in <i>DescriptiveName</i> . Possible values include: <i>Developer</i> – Starter <i>FuserOil</i> – Silicon Oil <i>Glue</i> <i>Paperclips</i> <i>Staples</i> <i>WasteContainer</i> – Waste Toner Bottle. <i>Wire</i> – bulk wire used for forming staples or other binding.

7.2.117 MISDetails

[New in JDF 1.2](#)

MISDetails is a container for MIS related information. It is referenced by Audit elements and JMF messages.

Resource Properties

Resource class:	ResourceElement
Resource referenced by:	—
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-255: MISDetails element

Name	Data Type	Description
<i>CostType</i> ?	enumeration	Whether or not this MISDetails is chargeable to the customer or not. One of: <i>Chargeable</i> <i>NonChargeable</i>
<i>DeviceOperationMode</i> ?	enumeration	<i>DeviceOperationMode</i> 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. <i>Productive</i> – The device is used to produce good product. Any times recorded in this mode are to be allocated against the job. <i>NonProductive</i> – 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. <i>Maintenance</i> – The device is used without the intention to produce good product, e.g., to perform (preventative) maintenance.
<i>WorkType</i> ?	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). One of: <i>Original</i> – Standard work that was originally planned for the job. <i>Alteration</i> – Work done to accommodate change made to the job at the request of the customer. <i>Rework</i> – Work done due to unforeseen problem with original work (bad plate, resource damaged, etc.)
<i>WorkTypeDetails</i> ?	string	Definition of the details of the work type for this MISDetails , (i.e., why the work was done). Values include: <i>CustomerRequest</i> – The customer requested change(s) requiring the work. <i>InternalChange</i> – Change was made for production efficiency or other internal reason. <i>ResourceDamaged</i> – A resource needs to be created again to account for a damaged resource (damaged plate, etc.). <i>EquipmentMalfunction</i> – Equipment used to produce the resource malfunctioned; resource needs to be created again. <i>UserError</i> – Incorrect operation of equipment or incorrect creation of resource requires creating the resource again.

7.2.118 NodeInfo

[Modified in JDF 1.3](#)

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.

Resource Properties

Resource class: Parameter

Resource referenced by: —

Example Partition: —
 Input of processes: *all*
 Output of processes: —

Table 7-256: NodeInfo resource (Section 1 of 2)

Name	Data Type	Description
<i>CleanupDuration</i> ?	duration	Estimated duration of the clean-up phase of the process.
<i>DueLevel</i> ?	enumeration	Description of the severity of a missed deadline. Possible values are: <i>Unknown</i> – Consequences of missing the deadline are not known. Deprecated in JDF 1.2 <i>Trivial</i> – Missing the deadline has minor or no consequences. <i>Penalty</i> – Missing the deadline incurs a penalty. <i>JobCancelled</i> – The job is cancelled if the deadline is missed.
<i>End</i> ?	dateTime	Date and time at which the process is scheduled to end.
<i>FirstEnd</i> ?	dateTime	Earliest date and time at which the process is to end.
<i>FirstStart</i> ?	dateTime	Earliest date and time at which the process is to begin.
<i>IPPVersion</i> ? 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.
<i>JobPriority</i> = "50" New in JDF 1.1 Modified in JDF 1.3	integer	The scheduling priority for the job where 100 is the highest and 0 is the lowest. Amongst the jobs that can be printed, all higher priority jobs are to be printed before any lower priority ones. If one or more of the deadline oriented attributes (e.g., <i>FirstStart</i> or <i>LastEnd</i>) is specified, such attribute(s) MUST be honored before considering <i>JobPriority</i> .
<i>LastEnd</i> ?	dateTime	Latest date and time at which the process is to end. This is the deadline to which <i>DueLevel</i> refers.
<i>LastStart</i> ?	dateTime	Latest date and time at which the process is to begin.
<i>NaturalLang</i> ? New in JDF 1.1	language	Language selected for communicating attributes. If not specified, the operating system language is assumed.
<i>NodeStatus</i> ? New in JDF 1.3	enumeration	Identifies the status of an individual part of the node. If not specified, defaults to <i>JDF/@Status</i> . Possible values are identical to those defined in <i>JDF/@Status</i> of Table 3-4, "JDF node," on page 41.
<i>NodeStatusDetails</i> ? New in JDF 1.3	string	Description of the status that provides details beyond the enumerative values given by <i>NodeStatus</i> . If not specified, defaults to <i>JDF/@StatusDetails</i> . For a list of supported values, See "StatusDetails Supported Strings" on page 703.
<i>MergeTarget</i> ? Deprecated in JDF 1.1	boolean	If <i>MergeTarget</i> = <i>true</i> and this node has been spawned, it MUST be merged with its direct ancestor by the controller that executes this node. The path of the ancestor is specified in the last <i>Ancestor</i> element located in the <i>AncestorPool</i> of this node. It is an error to specify both <i>MergeTarget</i> and <i>TargetRoute</i> in one node. Note: <i>MergeTarget</i> 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.

Table 7-256: NodeInfo resource (Section 2 of 2)

Name	Data Type	Description
<i>Route</i> ?	URL	The URL of the controller or device that is to execute this node. If <i>Route</i> is not specified, the routing controller MUST determine a potential target controller or device independently. For details, see Section 4.2, Process Routing. Note that the receiving Device MUST NOT use <i>Route</i> to determine whether to execute the node. Rather a Device MUST use a Device input resource (if specified) to determine whether to execute the node.
<i>rRefs</i> ? Deprecated in JDF 1.2	IDREFS	Array of <i>IDs</i> of any elements that are specified as ResourceRef elements. In version 1.1, <i>rRefs</i> contained the IDREF of an Employee. In JDF 1.2 and beyond, it is up to the implementation to maintain references.
<i>SetupDuration</i> ?	duration	Estimated duration of the setup phase of the process.
<i>Start</i> ?	dateTime	Date and time of the planned process start.
<i>TargetRoute</i> ?	URL	The URL where the JDF is to be sent after completion. If <i>TargetRoute</i> is not specified, it defaults to the input <i>Route</i> 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 is to be sent to the processor default output URL. JMF/QueueSubmissionParams/@ReturnURL takes precedence over NodeInfo/@TargetRoute of the JDF that is processed.
<i>TotalDuration</i> ?	duration	Estimated total duration of the process, including setup and cleanup.
<i>BusinessInfo</i> ?	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.
JMF *	element	Represents JMF query messages that set up a persistent channel, as described in Section 5.3.3, 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 Signals).
MISDetails ? New in JDF 1.2	element	Definition how the costs for the execution of this node are to be charged.
NotificationFilter *	element	Defines the set of Notification elements that are to be logged in the AuditPool. This provides a logging method for devices that do not support JMF messaging. For details of the NotificationFilter element, see Section 5.7.1, Events.

7.2.119 NumberingParams

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 MUST be defined per numbering machine.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—

Input of processes: **Numbering**

Output of processes: —

Table 7-257: NumberingParams resource

Name	Data Type	Description
NumberingParam *	element	Set of parameters for one numbering machine

— Element: **NumberingParam**

Table 7-258: NumberingParam element

Name	Data Type	Description
<i>Orientation</i>	double	Rotation of the numbering machine in degrees. If <i>Orientation</i> = "0", the top of the numbers is along the leading edge.
<i>StartValue</i> ?	string	First value of the numbering machine.
<i>Step</i> = "1"	integer	Number that specifies the difference between two subsequent numbers of the numbering machine.
<i>XPosition</i>	double	Position of the numbering machine along the printer axis.
<i>YPosition</i>	DoubleList	List of stamp positions, in points, starting from the leading edge.

7.2.120 ObjectResolution

ObjectResolution defines a resolution depending on *SourceObjects* data types.

Resource Properties

Resource class: Parameter

Resource referenced by: **InterpretingParams, RenderingParams, TrappingDetails**

Example Partition: —

Input of processes: —

Output of processes: —

Table 7-259: ObjectResolution resource

Name	Data Type	Description
<i>AntiAliasing</i> ? New in JDF 1.2	NMTOKEN	Indicates the anti-aliasing algorithm that the Device MUST 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: <i>AntiAlias</i> – Anti-aliasing MUST be applied. The algorithm is system specified. <i>None</i> – Anti-aliasing MUST NOT be applied.
<i>Resolution</i>	XYPair	Horizontal and vertical output resolution in DPI.
<i>SourceObjects</i> = "All"	enumerations	Identifies the class(es) of incoming graphical objects to render at the specified resolution. Possible values are: <i>All</i> <i>ImagePhotographic</i> – Contone images. <i>ImageScreenShot</i> – Images largely comprised of rasterized vector art. <i>LineArt</i> – Vector objects other than text. <i>SmoothShades</i> – Gradients and blends. <i>Text</i>

7.2.121 OrderingParams

Attributes of the **Ordering** process, which results in an acquisition.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	Ordering
Output of processes:	—

Table 7-260: OrderingParams resource

Name	Data Type	Description
<i>Amount</i>	double	Amount of the ordered resource.
<i>Unit</i>	string	Unit of measurement for <i>Amount</i> .
Comment	telem	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.
Contact * New in JDF 1.1	refelement	Address and further information of the Contact responsible for this order.

7.2.122 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 6.7.5, Packaging Processes. See "PackingParams" on page 847 for details of this deprecated resource.

7.2.123 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:	ExposedMedia,LayoutElement,RunList
Example Partition:	<i>PartVersion</i>
Input of processes:	—
Output of processes:	—

Table 7-261: PageList resource (Section 1 of 2)

Name	Data Type	Description
<i>AssemblyID</i> ? Deprecated in JDF 1.3	string	ID of the Assembly or AssemblySection that this finished page belongs to.
<i>AssemblyIDs</i> ? New in JDF 1.3	NMTOKENS	IDs of the Assembly elements or AssemblySection elements that this finished page belongs to.
<i>HasBleeds</i> ?	boolean	If " <i>true</i> ", the file has bleeds.

Table 7-261: PageList resource (Section 2 of 2)

Name	Data Type	Description
<i>IsBlank</i> ?	boolean	If " <i>true</i> ", the PageList has no content marks and is blank. Note that in JDF 1.2, the description erroneously stated that <i>IsBlank</i> = " <i>false</i> " specifies a blank page.
<i>IsPrintable</i> ?	boolean	If " <i>true</i> ", 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 <i>IsPrintable</i> = " <i>false</i> ".
<i>IsTrapped</i> ?	boolean	If " <i>true</i> ", the file has been trapped.
<i>JobID</i> ?	string	ID of the job that this finished page belongs to.
<i>PageLabelPrefix</i> ?	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.
<i>PageLabelSuffix</i> ?	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.
<i>SourceBleedBox</i> ?	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).
<i>SourceClipBox</i> ?	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.)
<i>SourceTrimBox</i> ?	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.)
<i>Template</i> = " <i>false</i> "	boolean	Template is " <i>false</i> " when this page is self-contained. This attribute is " <i>true</i> " if the PageList represents a template that MUST be completed with information from a database.
Assembly ? New in JDF 1.3	refelement	Assembly that is referred to by <i>AssemblyIDs</i> or contains the AssemblySection that is referred to by <i>AssemblyIDs</i> .
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 their index in the PageList . The PageData elements SHOULD, therefore, not be removed or inserted in a position other than the end of the list.
ScreeningParams ?	refelement	Specification of the screening properties.
SeparationSpec *	element	List of separation names defined in the PageList .

— Element: PageData

PageData defines the additional metadata of individual finished pages such as pagination details. **PageData** elements are referred to by index of the **PageData** in the **PageList**.

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 7-262: PageData element (Section 1 of 2)

Name	Data Type	Description
<i>AssemblyID</i> ? Deprecated in JDF 1.3	string	ID of the Assembly or AssemblySection that this finished page belongs to. If not specified, defaults to the value of PageList/@AssemblyID .
<i>AssemblyIDs</i> ? New in JDF 1.3	NMTOKENS	IDs of the Assembly elements or AssemblySection elements that this finished page belongs to. If not specified, defaults to the value of PageList/@AssemblyIDs .
<i>CatalogID</i> ?	string	Identification of the resource, (e.g., in a catalog environment). If not specified, defaults to the value of PageList/@CatalogID .
<i>CatalogDetails</i> ?	string	Additional details of a resource in a catalog environment. If not specified, defaults to the value of PageList/@CatalogDetails .
<i>FoldOutPages</i> ?	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 . If not specified, the PageData does not describe a part of a foldout.
<i>HasBleeds</i> ?	boolean	If " <i>true</i> ", the file has bleeds. If not specified, defaults to the value of PageList/@HasBleeds .
<i>IsBlank</i> ?	boolean	If " <i>true</i> ", the PageData has no content marks and is blank. If not specified, defaults to the value of PageList/@IsBlank . Note that in JDF 1.2 the description erroneously stated that <i>IsBlank</i> = " <i>false</i> " specifies a blank page.
<i>IsPrintable</i> ?	boolean	If " <i>true</i> ", the file is a PDL file and can be printed. Possible file types include PCL, PDF or PostScript files. Application files such as MS Word have <i>IsPrintable</i> = " <i>false</i> ". If not specified, defaults to the value of PageList/@IsPrintable .
<i>IsTrapped</i> ?	boolean	If " <i>true</i> ", the file has been trapped. If not specified, defaults to the value of PageList/@IsTrapped .
<i>JobID</i> ?	string	ID of the job that this finished page belongs to. If not specified, defaults to the value of PageList/@JobID .
<i>PageFormat</i> ? New in JDF 1.3	NMTOKEN	Defines the format of the page in a production workflow. For example in a newspaper workflow values include: <i>Broadsheet</i> – One single page that will be mounted on a broadsheet plate (one page goes on one (broadsheet) plate). <i>Tabloid</i> – One single page that will be paired with a second tabloid page. Later, the page pair will be mounted on a broadsheet plate. <i>Newspaper4up</i> – Four pages will be mounted on one plate. <i>Newspaper8up</i> – Eight pages will be mounted on one plate.
<i>PageLabel</i> ?	string	Complete identification of the finished page including <i>PageLabelPrefix</i> and <i>PageLabelSuffix</i> 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.

Table 7-262: PageData element (Section 2 of 2)

Name	Data Type	Description
<i>PageLabelPrefix</i> ?	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. If not specified, defaults to the value of PageList/@PageLabelPrefix .
<i>PageLabelSuffix</i> ?	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. If not specified, defaults to the value of PageList/@PageLabelSuffix .
<i>PageStatus</i> ? New in JDF 1.3	NMTOKENS	Status of a single PageData element. For a list of predefined values, see "MessageEvents and Milestone Values" on page 712.
<i>ProductID</i> ?	string	An ID of the page as defined in the MIS system. If not specified, defaults to the value of PageList/@ProductID .
<i>SourceBleedBox</i> ?	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, defaults to the value of PageList/@SourceBleedBox .
<i>SourceClipBox</i> ?	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, defaults to the value of PageList/@SourceClipBox .
<i>SourceTrimBox</i> ?	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, defaults to the value of PageList/@SourceTrimBox .
<i>Template</i> ?	boolean	Template is "false" when this page is self-contained. This attribute is "true" if the PageList represents a template that MUST be completed with information from a database. If not specified, it defaults to the value of PageList/@Template .
ElementColorParams ?	refelement	Color details of the PageData element. If not specified, defaults to the value of PageList/ElementColorParams .
ImageCompressionParams ?	refelement	Specification of the image compression properties. If not specified, defaults to the value of 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. If not specified, defaults to the value of PageList/ScreeningParams .
SeparationSpec *	element	List of separation names defined in the element. If none is specified, defaults to the values of PageList/SeparationSpec .

— Element: PageElement

[New in JDF 1.3](#)

PageElement defines the additional metadata of individual elements within a page.

Table 7-263: PageElement element (Section 1 of 2)

Name	Data Type	Description
<i>ContentListIndex</i> ?	integer	Index into a ContentList/ContentData element. If neither <i>ContentListIndex</i> nor PageElement are specified, this PageElement is a reservation.

Table 7-263: PageElement element (Section 2 of 2)

Name	Data Type	Description
<i>ElementPages ?</i>	Integer-RangeList	List of Pages that this PageElement traverses, e.g., fold out pages or multi-page ads.
<i>ContentType ?</i>	NMTOKEN	Type of content that is placed in this PageElement. For a list of predefined values, see ContentList/ContentData/@ContentType .
<i>RelativeBox ?</i>	Rectangle	Position of the PageElement in the coordinate system of the parent PageElement or PageList .
PageElement *	element	Further sub-page elements that comprise this PageElement.

7.2.124 Pallet

[New in JDF 1.1](#)

A **Pallet** represents the pallet used in packing goods.

Resource Properties

Resource class:	Consumable
Resource referenced by:	—
Example Partition:	—
Input of processes:	Palletizing
Output of processes:	—

Table 7-264: Pallet resource

Name	Data Type	Description
<i>PalletType</i>	NMTOKEN	Type of pallet used. Examples include: <i>2Way</i> – Two-way entry. <i>4Way</i> – Four-way entry. <i>Euro</i> – Standard 1*1 m Euro pallet.
<i>Size ?</i>	XYPair	Describes the length and width of the pallet, in points, (e.g., 3500 3500). If not specified, the size is defined by <i>PalletType</i> .

7.2.125 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
Resource referenced by:	—
Example Partition:	—
Input of processes:	Palletizing
Output of processes:	—

Table 7-265: PalletizingParams resource

Name	Data Type	Description
<i>Pattern ?</i>	string	Name of the palletizing pattern. Used to store a predefined pattern that defines the layers and positioning of individual component on the pallet.
<i>MaxHeight ?</i>	double	Maximum height of a loaded pallet in points.
<i>MaxWeight ?</i>	double	Maximum weight of a loaded pallet in grams.

7.2.126 PDFToPSConversionParams

This resource specifies a set of configurable options that can be used by processes that generate PostScript files. Prior to JDF 1.3, **PDFToPSConversionParams** was used only for converting PDF files to PostScript. The name “**PDFToPSConversionParams**” 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 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:	—
Example Partition:	<i>DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName</i>
Input of processes:	<i>PDFToPSConversion</i>
Output of processes:	—

Table 7-266: PDFToPSConversionParams resource (Section 1 of 4)

Name	Data Type	Description
<i>BinaryOK = "true"</i>	boolean	If “ <i>true</i> ”, binary data are to be included in the PostScript stream.
<i>BoundingBox ?</i>	rectangle	It is used for BoundingBox DSC comment in <i>CenterCropBox</i> calculations and for PostScript’s setpagedevice .
<i>CenterCropBox = "true"</i>	boolean	If “ <i>true</i> ”, the CropBox from the source document is centered on the page when the CropBox is smaller than MediaBox .
<i>GeneratePageStreams = "false"</i>	boolean	If “ <i>true</i> ”, the process emits individual streams of data for each page in the RunList .
<i>IgnoreAnnotForms = "false"</i>	boolean	If “ <i>true</i> ”, ignores annotations that contain a PDF XObject form. (PDF source only).
<i>IgnoreBG ? = "true"</i> 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.
<i>IgnoreColorSeps = "false"</i>	boolean	If “ <i>true</i> ”, ignores images for Level-1 separations.

Table 7-266: PDFToPSConversionParams resource (Section 2 of 4)

Name	Data Type	Description
IgnoreDeviceExtGState? Deprecated in JDF 1.1	boolean	If <i>true</i> , ignores all device-dependent extended graphic state parameters. This overrides <i>IgnoreHalftones</i> . 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.
<i>IgnoreDSC = "true"</i>	boolean	If <i>true</i> , ignores DSC (Document Structuring Conventions).
<i>IgnoreExternStreamRef = "false"</i>	boolean	If a PDF image resource uses an external stream and <i>IgnoreExternStreamRef = "true"</i> , ignores code that points to the external file. (PDF source only). Note that <i>IgnoreExternStreamRef</i> was misspelled as <i>IgnoreExternSreamRef</i> prior to JDF 1.3.
<i>IgnoreHalftones = "false"</i>	boolean	If <i>true</i> , ignores any halftone screening in the source file.
<i>IgnoreOverprint = "true"</i> New in JDF 1.1	boolean	Ignores OP parameters in a source PDF ExtGState dictionary setoverprint in a source PostScript file, etc.
<i>IgnorePageRotation = "false"</i>	boolean	If <i>true</i> , ignores a “concatenation” provided at the beginning of each page that orients the page so that it is properly rotated. Used when emitting EPS.
<i>IgnoreRawData = "false"</i>	boolean	If <i>true</i> , no unnecessary filters are to be added when emitting image data.
<i>IgnoreSeparableImages Only = "false"</i>	boolean	If <i>true</i> , and if emitting EPS, ignores only CMYK and gray images.
<i>IgnoreShowPage = "false"</i>	boolean	If <i>true</i> , ignores save-and-restore showpage in PostScript files
<i>IgnoreTransfers = "true"</i> 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.
<i>IgnoreTTFontsFirst = "false"</i>	boolean	If <i>true</i> , ignores TrueType fonts before any other fonts.
<i>IgnoreUCR = "true"</i> New in JDF 1.1	boolean	Ignores UCR, UCR2 parameters in a source PDF ExtGState dictionary, setundercolorremoval in a source PostScript file, etc.
<i>IncludeBaseFonts = "IncludeNever"</i>	enumeration	Determines when to embed the base fonts. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i> The base fonts are <i>Symbol</i> and the plain, bold, italic and bold-italic faces of <i>Courier</i> , <i>Times</i> , and <i>Helvetica</i> .

Table 7-266: PDFToPSConversionParams resource (Section 3 of 4)

Name	Data Type	Description
<i>IncludeCIDFonts</i> = "IncludeOncePerDoc"	enumeration	Determines when to embed CID fonts. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeEmbeddedFonts</i> = "IncludeOncePerDoc"	enumeration	Determines when to embed fonts in the document that are embedded in the source file. This attribute overrides the <i>IncludeType1Fonts</i> , <i>IncludeTrueTypeFonts</i> and <i>IncludeCIDFonts</i> attributes. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeOtherResources</i> = "IncludeOncePerDoc"	enumeration	Determines when to include all other types of resources in the file. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeProcSets</i> = "IncludeOncePerDoc"	enumeration	Determines when to include ProcSets in the file. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeTrueTypeFonts</i> = "IncludeOncePerDoc"	enumeration	Determines when to embed TrueType fonts. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeType1Fonts</i> = "IncludeOncePerDoc"	enumeration	Determines when to embed Type-1 fonts. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeType3Fonts</i> = "IncludeOncePerPage"	enumeration	Determines when to embed Type-3 fonts. It MUST always be set to <i>IncludeOncePerPage</i> . It is included here to complete the precedence hierarchy.
<i>OutputType</i> = "PostScript"	enumeration	Describes the kind of output to be generated. Possible values are: <i>PostScript</i> <i>EPS</i>
<i>PSLevel</i> = "2"	integer	Number that indicates the PostScript level. Values include "1", "2" or "3".
<i>Scale</i> = "100"	double	Number that indicates the wide-scale factor of documents. Full size = "100".
<i>SetPageSize</i> = "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.
<i>SetupProcsets</i> = "true"	boolean	If "true", indicates that if ProcSets are included, the init/term code is also included.

Table 7-266: PDFToPSConversionParams resource (Section 4 of 4)

Name	Data Type	Description
<i>ShrinkToFit</i> = "false"	boolean	If "true", the page is scaled to fit the printer page size. This field overrides scale
<i>SuppressCenter</i> = "false"	boolean	If "true", suppresses automatic centering of page contents whose crop box is smaller than the page size.
<i>SuppressRotate</i> = "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 <i>SuppressRotate</i> = "true". On the other hand, if <i>SuppressRotate</i> = "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).
<i>TTasT42</i> = "false"	boolean	If including TrueType fonts, converts to Type-42 instead of Type-1 fonts when <i>TTasT42</i> = "true".
<i>UseFontAliasNames</i> = "false"	boolean	If "true", font alias names are used when printing with system fonts.

7.2.127 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
Resource referenced by:	—
Example Partition:	—
Input of processes:	PDLCreation
Output of processes:	—

Table 7-267: PDLCreationParams resource

Name	Data Type	Description
<i>MimeType</i>	string	This resource identifies the MIME type associated with this output file format. For example "application/pdf".
PDFToPSConversionParams ?	reference	Postscript specific parameter resource for the output. MUST NOT be specified unless <i>MimeType</i> = "application/postscript"
PSToPDFConversionParams ?	reference	PDF specific parameter resource for the output. It MUST NOT be specified unless <i>MimeType</i> = "application/pdf"

7.2.128 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
Example Partition:	—
Input of processes:	Interpreting
Output of processes:	—

Table 7-268: PDLResourceAlias resource

Name	Data Type	Description
<i>ResourceType</i>	string	The type of PDL resource that is referenced. The semantic of this attribute is defined by the PDL.
<i>SourceName</i> ?	string	The name of the resource in the file referenced by the FileSpec or by the device.
FileSpec ?	reference	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.

7.2.129 PerforatingParams

[New in JDF 1.1](#)

PerforatingParams define the parameters for perforating a sheet.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	Perforating
Output of processes:	—

Table 7-269: PerforatingParams resource

Name	Data Type	Description
Perforate *	element	Defines one or more Perforate lines.

— Element: Perforate

Perforate describes one perforated line.

Table 7-270: Perforate element (Section 1 of 2)

Name	Data Type	Description
<i>Depth</i> ? New in JDF 1.2	double	Depth of the perforation, in microns [μm].
<i>RelativeStartPosition</i> ? New in JDF 1.2	XYPair	Relative starting position of the tool. The <i>RelativeStartPosition</i> 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 <i>StartPosition</i> or <i>RelativeStartPosition</i> MUST be specified.

Table 7-270: Perforate element (Section 2 of 2)

Name	Data Type	Description
<i>RelativeWorkingPath</i> ? New in JDF 1.2	XYPair	Relative working path of the tool beginning at <i>RelativeStartPosition</i> . Since the tools can only work parallel to the edges, one coordinate MUST be zero. The <i>RelativeWorkingPath</i> 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 <i>WorkingPath</i> or <i>RelativeWorkingPath</i> MUST be specified.
<i>StartPosition</i> ? Modified in JDF 1.2	XYPair	Starting position of the tool. If both <i>StartPosition</i> and <i>RelativeStartPosition</i> are specified, <i>RelativeStartPosition</i> is ignored. At least one of <i>StartPosition</i> or <i>RelativeStartPosition</i> MUST be specified.
<i>TeethPerDimension</i> ?	double	Number of teeth in a given perforation extent in teeth/point. MicroPerforation is defined by specifying a large number of teeth (<i>TeethPerDimension</i> > 1000).
<i>WorkingDirection</i>	enumeration	Direction from which the tool is working. Possible values are: <i>Top</i> – From above. <i>Bottom</i> – From below.
<i>WorkingPath</i> ? Modified in JDF 1.2	XYPair	Working path of the tool beginning at <i>StartPosition</i> . Since the tools can only work parallel to the edges, one coordinate MUST be zero. If both <i>WorkingPath</i> and <i>RelativeWorkingPath</i> are specified, <i>RelativeWorkingPath</i> is ignored. At least one of <i>WorkingPath</i> or <i>RelativeWorkingPath</i> MUST be specified.

7.2.130 Person

This resource provides detailed information about a person. It also has the ability to specify different communication channels to this person. The structure of the resource 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.

Resource Properties

Resource class:	Parameter
Resource referenced by:	Contact, Employee
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-271: Person resource (Section 1 of 2)

Name	Data Type	Description
<i>AdditionalNames</i> ?	string	Additional names of the contact person (vCard: N:other).
<i>FamilyName</i> ?	string	The family name of the contact person (vCard: N:family).
<i>FirstName</i> ?	string	The first name of the contact person (vCard: N:given).
<i>JobTitle</i> ?	string	Job function of the person in the company or organization (vCard: title).
<i>NamePrefix</i> ?	string	Prefix of the name, can include title (vCard: N:prefix).
<i>NameSuffix</i> ?	string	Suffix of the name (vCard: N:suffix).

Table 7-271: Person resource (Section 2 of 2)

Name	Data Type	Description
Address ? New in JDF 1.2	refelement	Address of the person.
ComChannel *	refelement	Communication channels to the person.

7.2.131 PlaceholderResource

This resource is used to link *ProcessGroup* 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
Resource referenced by:	—
Example Partition:	—
Input of processes:	any <i>ProcessGroup</i> nodes
Output of processes:	any <i>ProcessGroup</i> nodes

Resource Structure

The resource has no additional structure.

7.2.132 PlasticCombBindingParams

This resource describes the details of the **PlasticCombBinding** process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	PlasticCombBinding
Output of processes:	—

Table 7-272: PlasticCombBindingParams resource

Name	Data Type	Description
<i>Brand</i> ?	string	The name of the comb manufacturer and the name of the specific item.
<i>Color</i> ?	Named-Color	Determines the color of the plastic comb.
<i>Diameter</i> ?	double	The comb diameter is determined by the height of the block of sheets to be bound.
<i>Thickness</i> ?	double	The material thickness of the comb.
<i>Type</i> ? 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 MUST be the same. The following values from the hole type catalog in "JDF/CIP4 Hole Pattern Catalog" on page 761 exist: See Table 7-273 for values of <i>Type</i> In JDF 1.2 and beyond, 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.

— Attribute: Type

Table 7-273: PlasticCombBindingParams/@Type attribute – possible values

Value	Description
<i>P12m-rect-02</i> – Distance = 12 mm; Holes = 7 mm x 3 mm	Deprecated in JDF 1.2
<i>P16_gi-rect-0t</i> – Distance = 14.28 mm; Holes = 8 mm x 3 mm	Deprecated in JDF 1.2
<i>Euro</i> – (Distance = 12 mm; Holes = 7 mm x 3 mm)	Deprecated in JDF 1.1.
<i>USA1</i> – (Distance = 14.28 mm; Holes = 8 mm x 3 mm)	Deprecated in JDF 1.1.

7.2.133 PlateCopyParams

[Deprecated in JDF 1.1](#) See "PlateCopyParams" on page 848 for details of this deprecated resource.

7.2.134 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 "PreflightAnalysis" on page 822.

7.2.135 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 "PreflightInventory" on page 824.

7.2.136 PreflightParams

[New in JDF 1.2](#)

The **PreflightParams** resource specifies the tests for the **Preflight** process to run. These tests are defined using "ActionPool" on page 617, which defines a list of reporting actions to have for given document object tests defined into a **Test**. (See "TestPool" on page 642.) This section makes use of elements and attributes defined in "Device Capability Definitions" on page 613. It is suggested that readers familiarize themselves with that section and "Concept of the Preflight Process" on page 659.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	Preflight
Output of processes:	—

Table 7-274: PreflightParams resource

Name	Data Type	Description
ActionPool +	element	A set of ActionPool elements. Multiple ActionPool elements are equivalent to one ActionPool that contains all Action elements of the individual ActionPool elements.
TestPool New in JDF 1.3	element	Container for zero or more Test elements that are referenced from Action elements in the ActionPool. Although TestPool is New in JDF 1.3 , it is REQUIRED 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 "ActionPool" on page 617, 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.

— Element: PreflightAction

Table 7-275: PreflightAction element

Name	Data Type	Description
<i>SetRef?</i>	IDREF	A reference to a preflight Test ID used to filter a set of objects before applying the tests referenced by preflight Action. When <i>SetRef</i> is not defined, the Test is applied to all the objects.
<i>SetSplitBy</i> = "RunList"	enumeration	This is used to group objects in different ways. Possible values include: <i>Page</i> – Tests are applied on objects page per page. <i>Document</i> – Tests are applied on objects document per document. <i>RunList</i> – All objects of all pages included in all documents are processed together. <i>SetSplitBy</i> is only used when <i>SetRef</i> 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, <i>SetSplitBy</i> is set to "Page". In such a case, the report's content (as long as the <i>PRItem</i> 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.

— Element: 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 7-276: BasicPreflightTest element (Section 1 of 2)

Name	Data Type	Description
<i>DevNS</i> = "http://www.CIP4.org/JDFSchema_1_1"	URI	Namespace of the test that is described by <i>Name</i> in this BasicPreflightTest element.
<i>ListType?</i>	enumeration	Specifies what type of list or object the basic preflight test describes. The allowed values are identical to those defined in the <i>ListType</i> attribute of "State" on page 623.
<i>MaxOccurs</i> = "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 <i>MaxOccurs</i> is not "1", the BasicPreflightTest element refers to a list or RangeList of values, (e.g., a NameEvaluation will allow a list of NMOKENS).
<i>MinOccurs</i> = "1"	integer	Minimum number of elements in the list described by this BasicPreflightTest. Default = "1", (i.e., it is an individual value). If <i>MinOccurs</i> is not "1", the BasicPreflightTest element refers to a list or RangeList of values, (e.g., a NameEvaluation will allow a list of NMOKENS).
<i>Name?</i>	NMOKEN	Local name of the preflight constraint that is evaluated by this BasicPreflightTest. Valid <i>Name</i> values for the JDF namespace are defined in "PreflightParams" on page 540; preflight tests are defined through the use of constraints.

Table 7-276: BasicPreflightTest element (Section 2 of 2)

Name	Data Type	Description
PreflightArgument ?	element	Additional arguments for the preflight test. For details see "PreflightParams" on page 540 for the definition of PreflightArgument and constraints upon which preflight tests are defined.

— Element: PreflightArgument

This subelement is used by BasicPreflightTest when additional data are needed to determine object property.

Table 7-277: PreflightArgument element

Name	Data Type	Description
BoxArgument ?	element	Used by the InsideBox and OutsideBox tests.
BoxToBoxDifference ?	element	Used by the BoxToBoxDifference test.

— Element: BoxArgument

Table 7-278: BoxArgument element

Name	Data Type	Description
<i>Box</i>	enumeration	The box type used to verify inclusion or exclusion. Refer to "Box Properties" on page 663 for a description of the valid types of boxes
<i>MirrorMargins</i> ?	enumeration	The <i>MirrorMargins</i> attribute allows the flip of the <i>Offset</i> value depending on the RunList index. When the index is even, the original <i>Offset</i> value is preserved. When the index is odd, the <i>Offset</i> value is flipped. With a value of "Vertical", you turn [l b r t] into [r b l t]. With a value of "Horizontal", you turn [l b r t] into [l t r b]. If not specified, the value of <i>Offset</i> is not changed.
<i>Offset</i> ?	rectangle	The offset to build real rectangle to which test is made.
<i>Overlap</i> = "false"	boolean	Explains if overlap is allowed to check inclusion or exclusion.

— Element: BoxToBoxDifference

Table 7-279: BoxToBoxDifference element

Name	Data Type	Description
<i>FromBox</i> ?	enumeration	The "From" box used for BoxToBoxDifference calculation. Refer to "Box Properties" on page 663 for a description of the valid types of boxes
<i>ToBox</i> ?	enumeration	The "To" box used for BoxToBoxDifference calculation. Refer to "Box Properties" on page 663 for a description of the valid types of boxes

7.2.137 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 "PreflightProfile" on page 825.

7.2.138 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 "Device Capability Definitions" on page 613. It is suggested that reader's familiarize themselves with that section and "Concept of the Preflight Process" on page 659.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	all processes
Output of processes:	Preflight

Table 7-280: PreflightReport resource

Name	Data Type	Description
PreflightParams	refelement	References the PreflightParams that was used to create this report.
PreflightReportRulePool	refelement	References the PreflightReportRulePool that was used to create this report.
RunList	refelement	References the RunList that was used to create this report.
PRItem *	element	Describes the Action elements that produced an error or a warning.
<i>ErrorState ?</i>	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. Possible values include: <i>TestNotSupported</i> <i>TestWrongPDL</i> If not specified, no errors occurred.
<i>ErrorCount</i>	integer	The count of errors that were encountered while preflighting the job.
<i>WarningCount</i>	integer	The count of warnings that were encountered while preflighting the job.

— Element: 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 7-281: PRItem element

Name	Data Type	Description
<i>ActionRef</i>	IDREF	References the PreflightParams /ActionPool/Action that triggered this PRItem.
<i>Occurrences</i>	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.
<i>PageSet ?</i>	IntegerRange-List	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.

— Element: PRError

The PRError structure is used to describe generic errors that occurred while evaluating an object property while executing a Test.

Table 7-282: PRError element

Name	Data Type	Description
<i>ErrorType</i>	enumeration	Value is one of "TestWrongPDL" or "TestNotSupported".
<i>Value</i>	NMTOKEN	The name of the object property that was being tested when the process error occurred.

— Element: PRGroup

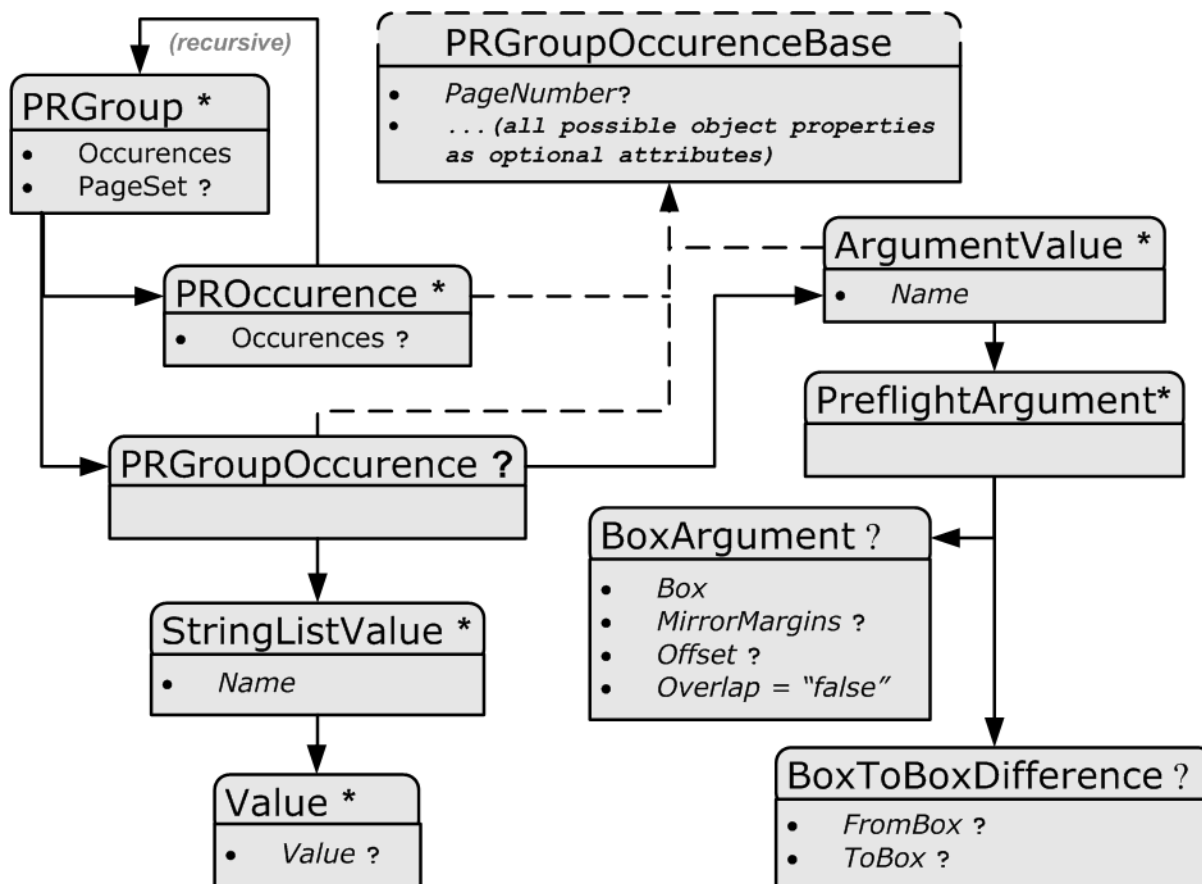


Figure 7-32: PRGroup – a diagram of its structure

The PRGroup structure is used to describe a group of document objects that share common properties and that failed the Action.

Table 7-283: PRGroup element (Section 1 of 2)

Name	Data Type	Description
<i>Occurrences</i>	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.
<i>PageSet ?</i>	IntegerRange-List	All run indices where there is an object of this group that gives an error on that page.

Table 7-283: PRGroup element (Section 2 of 2)

Name	Data Type	Description
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:

```

<PRItem Occurrences="4">
  <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>

```

— Element: Abstract PRGroupOccurrenceBase

PRGroupOccurrenceBase is an abstract class that serves as container for properties that were evaluated during the preflight process.

Table 7-284: Abstract PRGroupOccurrenceBase element

Name	Data Type	Description
<i>All possible object properties as OPTIONAL attributes.</i>	<i>As defined by the object property.</i>	An example is given below. See also section "Properties" on page 662 and following.
<i>PageNumber ?</i>	integer	Example of an integer attribute. The same format applies to boolean, Number, Name, NameList, enumeration, enumerations and string data types.

When the object does not support a certain property, the corresponding attribute in PROccurrence and PRGroupOccurrence MUST NOT be specified.

— Element: PRGroupOccurrence

PRGroupOccurrence specifies the shared properties of all PROccurrence elements in a PRGroup. PRGroupOccurrence inherits from PRGroupOccurrenceBase and adds the following elements:

Table 7-285: PRGroupOccurrence element

Name	Data Type	Description
StringListValue *	element	Describes the values of a StringList property.
ArgumentValue *	element	Describes the value of a property that is enhanced with additional arguments.

— Element: StringListValue

StringListValue specifies a type that returns a set of strings.

Table 7-286: StringListValue element

Name	Data Type	Description
<i>Name</i>	NMTOKEN	The name of the subject property.
<i>Value *</i>	element	Element of type StringEvaluation/Value. (See "StringEvaluation" on page 651.)

— Element: ArgumentValue

ArgumentValue specifies a value that is specified with additional arguments. This inherits from PRGroupOccurrenceBase and adds the following values:

Table 7-287: ArgumentValue element

Name	Data Type	Description
<i>Name</i>	NMTOKEN	The name of the subject property.
PreflightArgument	element	The argument that was used to evaluate this property. This is a PreflightArgument element. (See "PreflightArgument" on page 542.)

— Element: PROccurrence

PROccurrence describes an individual occurrence of a preflight action failure. This inherits from PRGroupOccurrence and adds the following values:

Table 7-288: PROccurrence element

Name	Data Type	Description
<i>Occurrences ?</i>	integer	Only used when the subject occurrence is a set-object. It describes the number of objects in the set.
PRGroup *	element	When this occurrence describes a set-object, the PRGroup elements describe the objects that are part of the set.

7.2.139 PreflightReportRulePool

[New in JDF 1.2](#)

The **PreflightReportRulePool** resource specifies how the **PreflightReport** is to log the errors that were found during the **Preflight** process. This section makes use of elements and attributes defined in "Device Capability Definitions" on page 613. It is suggested that reader's familiarize themselves with that section and "Concept of the Preflight Process" on page 659.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	Preflight
Output of processes:	—

Table 7-289: PreflightReportRulePool resource

Name	Data Type	Description
<i>MaxOccurrences ?</i>	integer	An upper bound to the maximum number of PROccurrence elements that are to be logged in the PreflightReport .
<i>ActionPools</i>	IDREFS	References the ActionPool whose reporting are defined by this rule.
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.

— Element: PRRule

The PRRule structure is used to define how the **PreflightReport** is to log the events that were found during the execution of one Action.

Table 7-290: PRRule element

Name	Data Type	Description
<i>ActionRefs +</i>	IDREFS	References the action for which the report behavior is defined in PRRuleAttr.
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).

— Element: PRRuleAttr

Table 7-291: PRRuleAttr element (Section 1 of 2)

Name	Data Type	Description
<i>GroupBy= "Tested"</i>	NMTOKENS	Group objects having the same N-pair of attributes listed here. For a complete description, see explanations below this table.

Table 7-291: PRRuleAttr element (Section 2 of 2)

Name	Data Type	Description
<i>ReportAttr</i> = "Tested Filename PageNumber"	NMTOKENS	When individual items are reported, these attributes are also reported. Attributes which are also being referred by <i>GroupBy</i> are ignored. See possible values and explanations below this table.
<i>LogErrors</i> ?	integer	When the Preflight process does not understand or cannot apply certain tests, that error MUST 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 "Concept of the Preflight Process" on page 659.)
<i>MaxGroups</i> ?	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 <i>MaxGroups</i> , that occurrence is no longer reported individually and no new group is created, although it is added to the <i>Occurrences</i> count and the <i>PageSet</i> .
<i>MaxPerGroup</i> ?	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 <i>MaxPerGroup</i> elements, that occurrence is no longer reported individually, although it is added to the <i>Occurrences</i> count and the <i>PageSet</i> .

The NMTOKENS for *GroupBy* and *ReportAttr* are one of:

- An object-specific attribute, (e.g., *ColorSpace*, *FontName*, etc.). At the time that we define the Test, we will almost automatically define these attributes.
- *Tested* – refers to all the attributes that are referred to in the Test(s) used by the Action(s) listed in the *ActionRefs*.
- *TestRelated* – refers to all the attributes referred in the Test(s) used by the Action(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(s) used by the Action(s) listed in *ActionRefs*.
- *BriefAppSpecific* – refers to a small list of attributes that the preflight agent (with preflight agent-specific logic) finds interesting for the Test(s) used by the Action(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(s) used by the Action(s) and on the preflight agent as demonstrated in the table below.

Table 7-292: Contingent Report Behavior (Section 1 of 2)

Preflight Agent	For ColorSpace Test	For FontEmbedded Test	Behavior
Preflight agent 1	<i>ColorSpace</i> <i>PageNumber</i>	<i>FontEmbedded</i> <i>PageNumber</i> <i>FontName</i>	<i>PageNumber</i> is always added. For color-related tests, <i>ColorSpace</i> is added. For font-related tests, <i>FontName</i> is added

Table 7-292: Contingent Report Behavior (Section 2 of 2)

Preflight Agent	For ColorSpace Test	For FontEmbedded Test	Behavior
Preflight agent 2	<i>ColorSpace</i> <i>PageNumber</i> <i>BoundingBox</i>	<i>FontEmbedded</i> <i>PageNumber</i> <i>BoundingBox</i> <i>FontSubset</i>	<i>PageNumber</i> and <i>BoundingBox</i> are always added. For color-related tests, <i>ColorSpace</i> is added. For font-related tests, <i>FontName</i> , <i>FontEmbedded</i> and <i>FontSubset</i> 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/>`.

7.2.140 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: —

Example Partition: *PreviewType*, *Separation*, *SheetName*, *Side*, *TileID*, *WebName*, *RibbonName*

Input of processes: *InkZoneCalculation*, *PreviewGeneration*, all other processes as thumbnails for user interface purposes

Output of processes: *PreviewGeneration*

Table 7-293: Preview resource (Section 1 of 2)

Name	Data Type	Description
<u>Compensation ?</u> <u>Modified in JDF 1.2</u>	enumeration	Compensation of the image to reflect the application of transfer curves to the image. Possible values are: <i>Unknown</i> – <u>Deprecated in JDF 1.2</u> <i>None</i> – No compensation. <i>Film</i> – Compensated until film exposure. <i>Plate</i> – Compensated until plate exposure. <i>Press</i> – Compensated until press.
<u>CTM ?</u> <u>New in JDF 1.1</u> <u>Modified in JDF 1.3</u>	matrix	Orientation of the Preview with respect to the Layout coordinate system. CTM is applied after any transformation defined within the referenced image file, (for example: the transformation defined in the CIP3PreviewImageMatrix of a PPF file). In case of PPF, <i>CTM</i> 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.

Table 7-293: Preview resource (Section 2 of 2)

Name	Data Type	Description
Directory ? New in JDF 1.1	URL	Defines a base URL for the files that represent this Preview . If <i>Directory</i> is specified, it MUST be an Absolute URI [RFC3986] that implicitly also specifies a Base URI which is used to resolve any relative URL of Preview . See "Resolving RunList/@Directory and FileSpec/@URL URI references" on page 745 and [FileURL] for examples.
PreviewFileType = "PNG" New in JDF 1.2	enumeration	The file type of the preview. Possible values are: <i>PNG</i> – The Portable Network Graphics format. <i>CIP3Multiple</i> – The format as defined in the CIP3 PPF specification. One or more previews per CIP3 file are supported. <i>CIP3Single</i> – The format as defined in the CIP3 PPF specification. Only one preview per CIP3 file is supported. The CIP3 formats were added in JDF 1.2 only for backwards compatibility since many systems only support CIP3 format. The CIP3 formats MUST NOT be used except in Preview resources that are used as input resources to InkZoneCalculation .
PreviewType ? Deprecated in JDF 1.2	enumeration	Type of the preview. Possible values are: <i>Separation</i> – Separated preview in medium resolution. <i>SeparationRaw</i> – Separated preview in medium resolution. <i>SeparatedThumbNail</i> – Very low resolution separated preview. <i>ThumbNail</i> – Very low resolution RGB preview. <i>Viewable</i> – RGB preview in medium resolution. In JDF 1.2 and beyond, <i>PreviewType</i> is still a partition key and MUST be used only as such — as a Preview attribute, <i>PreviewUsage</i> (below) replaces <i>PreviewType</i> .
PreviewUsage = "Separation" New in JDF 1.2	enumeration	The kind of the preview. Possible values are: <i>Separation</i> – Separated preview in medium resolution. Separation is generally used in InkZoneCalculation . <i>SeparationRaw</i> – Separated preview in medium resolution. This is identical to <i>Separation</i> except that no compensation has been applied. <i>SeparationRaw</i> is generally used for closed loop color control. <i>SeparatedThumbNail</i> – Very low resolution separated preview. <i>ThumbNail</i> – Very low resolution RGB preview. <i>Viewable</i> – RGB preview in medium resolution. <i>PreviewUsage</i> defines the semantics of the preview. If both <i>PreviewType</i> as a partition key and <i>PreviewUsage</i> are specified, they MUST match.
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 "Resolving RunList/@Directory and FileSpec/@URL URI references" on page 745 and "FileSpec Attributes and Container Subelement" on page 735 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 MUST match the <i>Separation</i> partition keys used in the JDF.

7.2.141 PreviewGenerationParams

Parameters specifying the size and the type of the preview.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>PreviewType, Separation, SheetName, Side, TileID, WebName, RibbonName</i>
Input of processes:	PreviewGeneration
Output of processes:	—

Table 7-294: PreviewGenerationParams resource (Section 1 of 2)

Name	Data Type	Description
<i>AspectRatio</i> = "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 <i>AspectRatio</i> only has an effect if <i>Size</i> is specified. One of: <i>CenterMax</i> – Keep the aspect ratio and preview <i>Size</i> , and center the image so that the preview has missing pixels at both sides of the larger dimension. <i>CenterMin</i> – Keep the aspect ratio and preview <i>Size</i> , and center the image so that the preview has blank pixels at both sides of the smaller dimension. <i>Crop</i> – Keep the aspect ratio, and modify the preview size so that the image fits into a bounding rectangle defined by <i>Size</i> . <i>Expand</i> – Keep the aspect ratio, and modify the preview size so that the smaller image dimension is defined by <i>Size</i> . <i>Ignore</i> – Fill the preview completely, keeping <i>Size</i> , even if this requires modifying the aspect ratio.
<i>Compensation</i> ? Modified in JDF 1.2	enumeration	Compensation of the image to reflect the application of transfer curves to the image. Possible values are: <i>None</i> – No compensation. <i>Film</i> – Compensated until film exposure. <i>Plate</i> – Compensated until plate exposure. <i>Press</i> – Compensated until press.
<i>PreviewFileType</i> = "PNG" New in JDF 1.2	enumeration	The file type of the preview to be generated. Possible values are: <i>PNG</i> – The Portable Network Graphics format. <i>CIP3Multiple</i> – The format as defined in the CIP3 PPF specification. One or more previews per CIP3 file are supported. <i>CIP3Single</i> – The format as defined in the CIP3 PPF specification. Only one preview per CIP3 file is supported. The CIP3 formats were added in JDF 1.2 only for backwards compatibility since many systems only support CIP3 format. The CIP3 formats MUST NOT be used except in Preview resources that are used as input resources to InkZoneCalculation .

Table 7-294: PreviewGenerationParams resource (Section 2 of 2)

Name	Data Type	Description
PreviewType ? Deprecated in JDF 1.1	enumeration	The kind of preview to be generated. Possible values are: <i>Separation</i> <i>Viewable</i> In JDF 1.1 and beyond, <i>PreviewType</i> is still a partition key and MUST be used only as such — as a Preview attribute, <i>PreviewUsage</i> (below) replaces <i>PreviewType</i> .
PreviewUsage = "Separation" New in JDF 1.1 Modified in JDF 1.2	enumeration	The kind of preview to be generated. Possible values are: <i>Separation</i> – Separated preview in medium resolution. <i>SeparationRaw</i> – Separated preview in medium resolution with no compensation. <i>SeparatedThumbNail</i> – Very low resolution separated preview. <i>ThumbNail</i> – Very low resolution RGB preview. <i>Viewable</i> – RGB preview in medium resolution. <i>PreviewUsage</i> defines the semantics of the preview. If both <i>PreviewType</i> as a partition key and <i>PreviewUsage</i> are specified, they MUST match.
Resolution ?	XYPair	Resolution of the preview, in dpi. If <i>PreviewUsage</i> = " <i>Separation</i> ", the default is "50.8 50.8".
Size ?	XYPair	Size of the preview, in pixels. If this attribute is present, the <i>Resolution</i> attribute evaluated according to the policy defined in <i>AspectRatio</i> . If <i>Size</i> is not specified, it MUST be calculated using the <i>Resolution</i> 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.

7.2.142 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
Resource referenced by:	—
Example Partition:	<i>SignatureName, SheetName, Side, Separation</i>
Input of processes:	ConventionalPrinting, DigitalPrinting
Output of processes:	—

Table 7-295: PrintCondition resource (Section 1 of 2)

Name	Data Type	Description
AimCurve ?	Transfer-Function	Describes the desired tone-value increase function. If not specified, it defaults to the media and printing machine-specific values

Table 7-295: PrintCondition resource (Section 2 of 2)

Name	Data Type	Description
<i>Density</i> ?	double	Density value of colorant (100% tint). Whereas Color/@NeutralDensity describes measurements of inks on substrate with wide-band filter functions, <i>Density</i> 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/PrintConditionColor/@Density .
<i>Name</i>	string	Name of the PrintCondition . Used to reference a PrintCondition from a Color/PrintConditionColor element.
ColorMeasurementConditions ?	refelement	Describes measurement conditions for color measurement and density measurement. If not specified, it defaults to the value of Color/PrintConditionColor/ColorMeasurementConditions
Device ?	refelement	Specifies the Device or Device group that this PrintCondition applies to.
FileSpec (<i>TargetProfile</i>) ?	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/PrintConditionColor/FileSpec (<i>TargetProfile</i>).

Example:

```
<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>
```

7.2.143 PrintRollingParams[New in JDF 1.2](#)**Resource Properties**

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	PrintRolling
Output of processes:	—

Table 7-296: PrintRollingParams resource

Name	Data Type	Description
<i>Copies</i> ?	integer	Number of copies on the roll. <i>Copies</i> MUST NOT be specified if <i>MaxDiameter</i> is present.
<i>MaxDiameter</i> ?	double	Maximal allowed diameter of roll. <i>MaxDiameter</i> MUST NOT be specified if <i>Copies</i> is present.

7.2.144 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 references:	CylinderLayoutPreparationParams
Resource inheritance:	—
Example Partition:	<i>RibbonName, WebName</i>
Input of processes:	<i>WebInlineFinishing</i>
Output of processes:	—

Table 7-297: ProductionPath resource

Name	Data Type	Description
<i>ProductionPathID</i> ?	string	Unique identification of the entire production path. If not specified, <i>PrintingUnitWebPath</i> MUST be specified.
<i>FolderSuperstructureWebPath</i> ?	element	Describes the path through the folder super-structure. The web will generally be cut into ribbons in this area of the production path.
<i>PostPressComponentPath</i> *	element	Describes the path through the inline postpress equipment. Folded sheets (Component) will be processed in this area of the production path.
<i>PrintingUnitWebPath</i> ?	element	Describes the path through the printing units. If not specified, <i>ProductionPathID</i> MUST be specified.

— Element: FolderSuperstructureWebPath, PostPressComponentPath and PrintingUnitWebPath

These are placeholders that might be filled with additional information in future versions of JDF. In JDF 1.3, paths are identified by ID only.

Table 7-298: FolderSuperstructureWebPath, PostPressComponentPath and PrintingUnitWebPath element

Name	Data Type	Description
<i>ProductionPathID</i> ?	string	Unique identification of the part of the production path specified in this element.

Examples of the different web path description levels:

Example on path level:

```
<ProductionPath ProductionPathID="ID_2webproduction_64pages"/>
```

Example on part path level:

```
<ProductionPath PartIDKeys="WebName">
  <ProductionPath WebName="1">
    <PrintingUnitWebPath ProductionPathID="ID_PrintingUnitWebPath"/>
    <FolderSuperstructureWebPath ProductionPathID="abcd"/>
    <PostPressComponentPath ProductionPathID="xyz"/>
  </ProductionPath>
</ProductionPath>
```

7.2.145 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 "ProofingParams" on page 849.

7.2.146 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: —

Example Partition: *DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName*

Input of processes: ***PSToPDFConversion***

Output of processes: —

Table 7-299: PSToPDFConversionParams resource (Section 1 of 3)

Name	Data Type	Description
<i>AllowJBIG2Globals</i> = "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.
<i>ASCII85EncodePages</i> = "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.
<i>AutoRotatePages</i> ?	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". Possible values are: <i>None</i> – Turns <i>AutoRotatePages</i> off. <i>All</i> – Takes the predominant text orientation across all pages and rotates all pages the same way. <i>PageByPage</i> – Does the rotation on a page-by-page basis, rotating each page individually. Useful for documents that use both portrait and landscape orientations.
<i>Binding</i> = "Left"	enumeration	Determines how the printed pages would be bound. Specify "Left" for left binding or "Right" for right binding.
<i>CompressPages</i> ?	boolean	Enables compression of pages and other content streams like forms, patterns and Type 3 fonts. If "true", use Flate compression.

Table 7-299: PSToPDFConversionParams resource (Section 2 of 3)

Name	Data Type	Description
<i>DefaultRenderingIntent</i> ? Modified in JDF 1.2	enumeration	Selects the rendering intent for the current job. Possible values are: <i>Default</i> Deprecated in JDF 1.2 <i>Perceptual</i> <i>Saturation</i> <i>RelativeColorimetric</i> <i>AbsoluteColorimetric</i> See the <i>Portable Document Format Reference Manual</i> for more information on rendering intent.
<i>DetectBlend</i> = "true"	boolean	Enables or disables blend detection. If "true" and if <i>PDFVersion</i> is 1.3 or higher, then blends will be converted to smooth shadings.
<i>DoThumbnails</i> = "true"	boolean	If "true", thumbnails are created.
<i>EndPage</i> ? Deprecated in JDF 1.3	integer	Number that indicates the last page that is displayed when the PDF file is viewed. <i>EndPage</i> MUST be either "-1" or greater than or equal to <i>StartPage</i> . When combined with <i>StartPage</i> , <i>EndPage</i> selects a range of pages to be displayed. The entire file MAY be converted, but only <i>StartPage</i> to <i>EndPage</i> pages, inclusive, are opened and viewed in a PDF viewing application.
<i>ImageMemory</i> ? 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.
<i>InitialPageSize</i> ? 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).
<i>InitialResolution</i> ? 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.
<i>OverPrintMode</i> ?	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].
<i>Optimize</i> = "true"	boolean	If "true", the PS-to-PDF converter optimizes the PDF file. See [PDF1.6] for more information on optimization.
<i>PDFVersion</i> ?	double	Specifies the version number of the PDF file produced. Possible values include all legal version designators, (e.g., 1.2, 1.5).
<i>StartPage</i> ? Deprecated in JDF 1.3	integer	Sets the first page that is to be displayed when the PDF file is opened with a PDF viewing application. <i>StartPage</i> MUST be greater than or equal to 1. <i>EndPage</i> MUST be either "-1" or greater than or equal to <i>StartPage</i> .

Table 7-299: PSToPDFConversionParams resource (Section 3 of 3)

Name	Data Type	Description
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.

— Element: AdvancedParams**Table 7-300: AdvancedParams element (Section 1 of 3)**

Name	Data Type	Description
<i>AllowPSXObject</i> s = "true" New in JDF 1.2	boolean	If "true", allows PostScript XObjects .
<i>AllowTransparency</i> = "false" New in JDF 1.2	boolean	If "true", allows transparency in the PDF.
<i>AutoPositionEPSInfo</i> = "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)
<i>EmbedJobOptions</i> = "false" New in JDF 1.2	boolean	If "true", the PDF settings used to create the PDF are embedded in the PDF.
<i>EmitDSCWarnings</i> = "false"	boolean	If "true", warning messages about questionable or incorrect DSC comments appear during the processing of the source PostScript file. (PostScript source only)
<i>LockDistillerParams</i> = "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. compatibility warning: In JDF 1.1A and previous versions, the definition of <i>LockDistillerParams</i> was accidentally inverted and is now consistent with the PostScript setdistillerparams operator.
<i>ParseDSCComments</i> = "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)
<i>ParseDSCCommentForDocInfo</i> = "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.
<i>PassThroughJPEGImages</i> = "false" New in JDF 1.2	boolean	If "true", JPEG images are passed through without recompressing them.

Table 7-300: AdvancedParams element (Section 2 of 3)

Name	Data Type	Description
<i>PreserveCopyPage</i> = "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)
<i>PreserveEPSInfo</i> = "true"	boolean	If "true", preserves the EPS information in a PostScript source file and stores it in the resulting PDF file. (PostScript source only)
<i>PreserveHalftoneInfo</i> = "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.
<i>PreserveOverprintSettings</i> = "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.
<i>PreserveOPIComments</i> = "true"	boolean	If "true", encapsulates Open Prepress Interface (OPI) low resolution images as a form and preserves information for locating the high resolution images.
<i>TransferFunctionInfo</i> = "Preserve" New in JDF 1.1	enumeration	Determines how transfer functions are handled. Possible values are: <i>Preserve</i> – Transfer functions are passed into the PDF file. <i>Remove</i> – Transfer functions are ignored. They are neither applied to the color values nor passed into the PDF file. <i>Apply</i> – 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.
<i>UCRandBGInfo</i> = "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. Possible values are: <i>Preserve</i> – The arguments are passed into the PDF file. <i>Remove</i> – The arguments are ignored.

Table 7-300: AdvancedParams element (Section 3 of 3)

Name	Data Type	Description
<i>UsePrologue</i> = "false"	boolean	If "true", the process MUST 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)

— Element: 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 7-301: PDFXParams element (Section 1 of 2)

Name	Data Type	Description
<i>PDFX1aCheck</i> = "false"	boolean	If "true", checks compliance with the PDF/X-1a standard [ISO15930-1:2001].
<i>PDFX3Check</i> = "false"	boolean	If "true", checks compliance with the PDF/X-3 standard [ISO15930-3:2002].
<i>PDFXBleedBoxtoTrimBoxOffset</i> ?	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 MUST be greater than or equal to 0.0. PDF BleedBox will be completely outside PDF TrimBox .
<i>PDFXCompliantPDFOnly</i> = "false"	boolean	If "true", produces a PDF document only if PDF/X compliance tests are passed.
<i>PDFXOutputCondition</i> ?	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.
<i>PDFXOutputIntentProfile</i> ?	string	If the source document does not specify an output intent name, then this value is used. See Table 7-302 for values.
<i>PDFXNoTrimBoxError</i> = "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.
<i>PDFXRegistryName</i>	URL	Indicates a location at which more information regarding the registry that defines the OutputConditionIdentifier can be obtained.
<i>PDFXSetBleedBoxToMediaBox</i> = "true"	boolean	If "true" and the BleedBox entry is not specified in the page object of the source document, BleedBox is set to MediaBox .

Table 7-301: PDFXParams element (Section 2 of 2)

Name	Data Type	Description
<i>PDFXTrapped</i> ?	enumeration	If a source document does not specify a Trapped state, then the value provided here is used. The value <i>Unknown</i> is to 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. Possible values are: <i>Unknown</i> <i>false</i> <i>true</i> Note that <i>Unknown</i> is prohibited in PDF/X files.
<i>PDFXTrimBoxToMediaBoxOffset</i> ?	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 MUST be greater than or equal to 0.0. The TrimBox will be completely inside MediaBox .

— Attribute: PDFXOutputIntentProfile

Table 7-302: PDFXOutputIntentProfile attribute – possible values

Value	Description
<i>None</i>	Used when it is REQUIRED that the source document specifies an intent; allows compliance checking to fail
<i>Euroscale Coated v2</i>	
<i>Euroscale Uncoated v2</i>	
<i>Japan Color 2001 Coated</i>	
<i>Japan Color 2001 Uncoated</i>	
<i>Japan Standard v2</i>	
<i>Japan Web Coated (Ad)</i>	
<i>U.S. Sheetfed Coated v2</i>	
<i>U.S. Sheetfed Uncoated v2</i>	
<i>U.S. Web Coated (SWOP) v2</i>	
<i>U.S. Web Uncoated v2</i>	
<i>Photoshop 4 Default CMYK</i>	
<i>Photoshop 5 Default CMYK</i>	

— Element: ThinPDFParams

Table 7-303: ThinPDFParams element (Section 1 of 2)

Name	Data Type	Description
<i>FilePerPage</i> = <i>"false"</i>	boolean	If <i>"true"</i> , the process generates 1 PDF file per page.
<i>SidelineEPS</i> = <i>"false"</i> New in JDF 1.2	boolean	If <i>"true"</i> , 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)
<i>SidelineFonts</i> = <i>"false"</i>	boolean	If <i>"true"</i> , font data are stored in external files during PDF generation.

Table 7-303: ThinPDFParams element (Section 2 of 2)

Name	Data Type	Description
<i>SidelineImages</i> = "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.

7.2.147 QualityControlParams

[New in JDF 1.2](#)

This set of parameters identifies how the **QualityControl** process is to operate. The **QualityControlParams** defines the generic set of parameters for the quality control process. The specific measurement conditions are defined in specialized subelements such as **BindingQualityParams**.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	QualityControl
Output of processes:	—

Table 7-304: QualityControlParams resource

Name	Data Type	Description
<i>TimeInterval</i> ?	duration	Time interval between individual tests.
<i>SampleInterval</i> ?	integer	Interval in number of samples between tests.
<i>BindingQualityParams</i> ?	element	Specification of the definition parameters of one individual resource.

— Element: BindingQualityParams

Table 7-305: BindingQualityParams element

Name	Data Type	Description
<i>FlexValue</i> ?	double	Flex quality parameter measured in [N/cm].
<i>PullOutValue</i> ?	double	Pull out quality parameter measured in [N/cm].

7.2.148 QualityControlResult

[New in JDF 1.2](#)

This set of parameters returns results of a **QualityControl** process. The **QualityControlResult** defines the generic set of results from the quality control process. The specific measurements are returned in specialized subelements such as **BindingQualityParams**. Additional detailed quality control result types are anticipated in future versions of the JDF specification.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	—
Output of processes:	QualityControl

Table 7-306: QualityControlResult resource (Section 1 of 2)

Name	Data Type	Description
<i>Failed</i> ?	integer	Total number of failed measurements.

Table 7-306: QualityControlResult resource (Section 2 of 2)

Name	Data Type	Description
<i>Passed</i> ?	integer	Total number of passed measurements.
BindingQualityParams ?	element	Reference to the measurement setup definition.
FileSpec ?	refelement	Location of an external file that contains details of the quality control measurement.
QualityMeasurement *	element	One individual measurement result.

— Element: QualityMeasurement

QualityMeasurement elements describe an individual measurement.

Table 7-307: QualityMeasurement element

Name	Data Type	Description
<i>End</i> ?	dateTime	Date and time of the end of the measurement. If not specified, the value of <i>Start</i> is applied.
<i>Failed</i> ?	integer	Total number of failed measurements.
<i>Passed</i> ?	integer	Total number of passed measurements.
<i>Condition</i> ?	NMTOKEN	Condition of the tested Component . If the Component passed the test, but the test itself destroyed the Component , the value is to be set to " <i>destroyed</i> ".
<i>Start</i> ?	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.

— Element: BindingQualityMeasurement**Table 7-308: BindingQualityMeasurement element**

Name	Data Type	Description
<i>FlexValue</i> ?	double	Flex quality parameter given in [N/cm].
<i>PullOutValue</i> ?	double	Pull out quality parameter given in [N/cm].

7.2.149 RasterReadingParams

[New in JDF 1.3](#)

This set of parameters specifies the details for *RasterReading*.

Resource Properties

Resource class: Parameter

Resource referenced by: —

Example Partition: —

Input of processes: —

Output of processes: *RasterReading*

Table 7-309: RasterReadingParams resource (Section 1 of 2)

Name	Data Type	Description
<i>Center</i> = "false"	boolean	Indicates whether or not the finished page image is to be centered within the imageable area of the media. The <i>Center</i> is ignored if FitPolicy / @SizePolicy = " <i>ClipToMaxPage</i> " and clipping is requested.

Table 7-309: RasterReadingParams resource (Section 2 of 2)

Name	Data Type	Description
<i>MirrorAround</i> = "None"	enumeration	This attribute specifies the axis around which a raster reader is to mirror an image. Possible values are: <i>None</i> – The default. <i>FeedDirection</i> – Image is mirrored around the feed-direction axis. <i>MediaWidth</i> – Image is mirrored around the media-width axis. <i>Both</i> – Image is mirrored around both possible axes.
<i>Polarity</i> = "Positive"	enumeration	The image MUST be RIPPed 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. Possible values are: <i>Positive</i> <i>Negative</i>
<i>Poster</i> ?	XYPair	Specifies whether the page contents is to be expanded such that each page covers X by Y pieces of media.
<i>PosterOverlap</i> ?	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.
<i>Scaling</i> ?	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 <i>FitToPage</i> = "true" or if <i>Poster</i> is present and has a value other than "1 1". Any scaling defined in FitPolicy MUST be applied after the scaling defined by this attribute.
<i>ScalingOrigin</i> ?	XYPair	A pair of real values that identify the point in the unscaled page that is to 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 <i>ScalingOrigin</i> defaults to "0 0"
FitPolicy ? New in JDF 1.1	refelement	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 <i>FitToPage</i> attribute. This FitPolicy element MUST 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.

7.2.150 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: Surface

Example Partition: —
Input of processes: Any printing process
Output of processes: —

Table 7-310: RegisterMark resource

Name	Data Type	Description
<i>Center</i>	XYPair	Position of the center of the register mark in the coordinates of the MarkObject that contains this mark.
<i>MarkType ?</i>	NMTOKEN	Type of RegisterMark . Possible values include: <i>Arc</i> <i>Circle</i> <i>Cross</i>
<i>MarkUsage ?</i> New in JDF 1.1	enumerations	Specifies the usage of the RegisterMark . Allowed values are: <i>Color</i> – The mark is used for separation color registration. <i>PaperPath</i> – The mark is used for paper path synchronization.
<i>Rotation ?</i>	double	Rotation in degrees. Positive graduation figures indicate counter-clockwise rotation; negative figures indicate clockwise rotation.
<i>SeparationSpec *</i> Modified in JDF 1.2	refelement	Set of separations to which the register mark is bound.

7.2.151 RegisterRibbon

[New in JDF 1.1](#)

Description of register ribbons. For the register ribbon, the length **MUST** be specified. There are two parameters, as shown in Figure 7-33, “RegisterRibbon lengths and coordinate system for BlockPreparation,” on page 564:

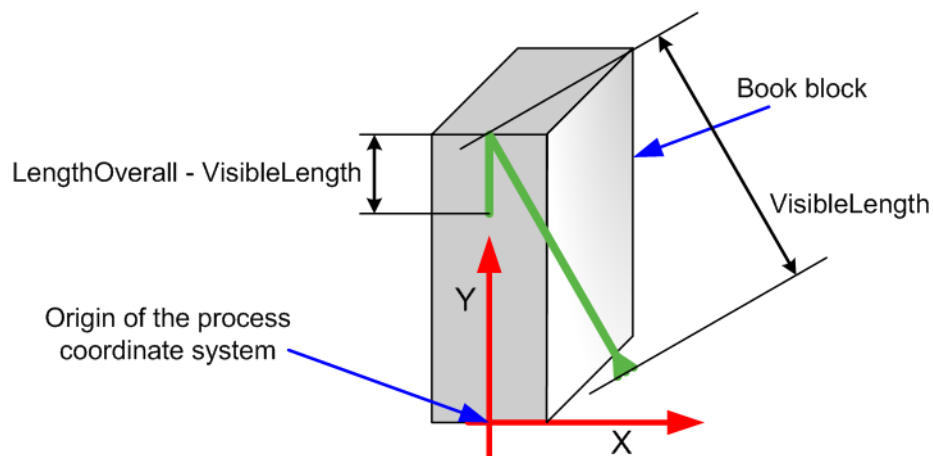


Figure 7-33: RegisterRibbon lengths and coordinate system for BlockPreparation

Resource Properties

Resource class: Consumable
Resource referenced by: **BlockPreparationParams**
Example Partition: —
Input of processes: —
Output of processes: —

Table 7-311: RegisterRibbon resource

Name	Data Type	Description
<i>LengthOverall</i>	double	Overall length of the register ribbon, (i.e., 1+2 in the picture above).
<i>Material ?</i>	string	Material of the register ribbon.
<i>RibbonColor ?</i>	NamedColor	Color of the ribbon.
<i>RibbonEnd ?</i>	NMTOKEN	End of the Ribbon. Values include: <i>Cut</i> <i>CutSealed</i> <i>Knot</i> <i>SealedOffset</i> – The ribbon is sealed a distance from the cut.
<i>VisibleLength</i>	double	Length of the register ribbon which will be seen when opening the book, (See picture above).

7.2.152 RenderingParams

This set of parameters identifies how the **Rendering** process is to operate. Specifically, these parameters define the expected output of the **ByteMap** resource that the **Rendering** process creates.

Resource Properties

Resource class: Parameter

Resource referenced by: —

Example Partition: *DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName*

Input of processes: **Rendering**

Output of processes: —

Table 7-312: RenderingParams resource (Section 1 of 2)

Name	Data Type	Description
<i>BandHeight ?</i>	integer	Height of output bands expressed in lines. For a frame device, the band height is simply the full height of the frame.
<i>BandOrdering ?</i>	enumeration	Indicates whether output buffers are generated in <i>BandMajor</i> or <i>ColorMajor</i> order. Possible values are: <i>BandMajor</i> – The position of the bands on the page is prioritized over the color. <i>ColorMajor</i> – All bands of a single color are played in order before progressing to the next plane. This is only possible with non-interleaved data.
<i>BandWidth ?</i>	integer	Width of output bands, in pixels.
<i>ColorantDepth ?</i>	integer	Number of bits per colorant. Determines whether the output is bitmaps or bytemaps.
<i>Interleaved ?</i>	boolean	If " <i>true</i> ", the resulting colorant values are interleaved and <i>BandOrdering</i> is ignored.
<i>AutomatedOverPrintParams ?</i>	refelement	Controls for overprint substitutions. Defaults to no automated overprint generation.

Table 7-312: RenderingParams resource (Section 2 of 2)

Name	Data Type	Description
ObjectResolution * Modified in JDF 1.2	refelement	Elements which define the resolutions to render the contents at. More than one element MAY be used to specify different resolutions for different <i>SourceObjects</i> types. If no ObjectResolution is specified, the value is implied from the input data.
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 is to obtain Media information from InterpretingParams/Media .

7.2.153 ResourceDefinitionParams

This set of parameters identifies how the **ResourceDefinition** process is to operate. Specifically, these parameters define how default parameters of applications and the input resource are to be combined.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	ResourceDefinition
Output of processes:	—

Table 7-313: ResourceDefinitionParams resource

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 <i>DefaultJDF</i> contains only a JDF resource element, not a complete JDF.
DefaultJDF ?	URL	Link to a JDF resource that defines preset values.
DefaultPriority = "DefaultJDF"	enumeration	Defines whether preset values of the application or of the resource specified in <i>DefaultJDF</i> have priority. Possible values are: <i>Application</i> —The application default settings are used to fill the resource. <i>DefaultJDF</i> —The settings specified in <i>DefaultJDF</i> are applied.
ResourceParam * New in JDF 1.1 Modified in JDF 1.3	element	Specification of the definition parameters of one individual resource.

— Element: ResourceParam

[New in JDF 1.1](#)

Table 7-314: ResourceParam element (Section 1 of 2)

Name	Data Type	Description
DefaultID ?	NMTOKEN	Resource/@ID of the default resource. If missing, it is assumed that the file specified by <i>DefaultJDF</i> contains only a JDF resource element, not a complete JDF.
DefaultJDF ?	URL	Link to a JDF resource that defines preset values. Defaults to the <i>DefaultJDF</i> specified in ResourceDefinitionParams .

Table 7-314: ResourceParam element (Section 2 of 2)

Name	Data Type	Description
<i>DefaultPriority</i> ?	enumeration	Defines whether preset values of the application or of the Resource specified in <i>DefaultJDF</i> have priority. Possible values are: <i>Application</i> <i>DefaultJDF</i> Defaults to the <i>DefaultPriority</i> specified in the parent ResourceDefinitionParams .

7.2.154 RingBindingParams

This resource describes the details of the **RingBinding** process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	RingBinding
Output of processes:	—

Table 7-315: RingBindingParams resource (Section 1 of 2)

Name	Data Type	Description
<i>BinderColor</i> ?	NamedColor	Color of the ring binder.
<i>BinderMaterial</i> ?	NMTOKEN	The following describe RingBinding binder materials used. Values include: <i>Cardboard</i> – Cardboard with no covering. <i>ClothCovered</i> – Cardboard with cloth covering. <i>PVC</i> – Solid PVC. <i>PVCCovered</i> – Cardboard with PVC covering.
<i>BinderName</i> ?	string	The name of the binder manufacturer and the name of the specific item.
<i>RingDiameter</i> ?	double	Diameter of the rings, in points.
<i>RingMechanic</i> ?	boolean	If “ <i>true</i> ”, a hand lever is available for opening.
<i>RingShape</i> ?	NMTOKEN	The possible values include the following RingBinding : <i>Round</i> <i>Oval</i> <i>D-shape</i> <i>SlantD</i>
<i>RingSystem</i> ?	enumeration	The following ring binding systems are used: <i>2HoleEuro</i> – In Europe <i>3HoleUS</i> – In North America <i>4HoleEuro</i> – In Europe In JDF 1.2 and beyond, use the value implied by HoleMakingParams/@HoleType .

Table 7-315: RingBindingParams resource (Section 2 of 2)

Name	Data Type	Description
<i>RivetsExposed</i> ?	boolean	The following RingBinding choice describes mounting of ring mechanism in binder case. If " <i>true</i> ", the heads of the rivets are visible on the exterior of the binder. If " <i>false</i> ", the binder covering material covers the rivet heads.
<i>SpineColor</i> ?	NamedColor	Color of the binders spine.
<i>SpineWidth</i> ?	double	The spine width is determined by the final height of the block of sheets to be bound.
<i>ViewBinder</i> ?	NMTOKEN	The possible values include the following RingBinding clear vinyl outer-wrap types on top of a colored base wrap: <i>Embedded</i> – Printed material is embedded by sealing between the colored and clear vinyl layers during the binder manufacturing. <i>Pocket</i> – 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 .

7.2.155 RollStand

[New in JDF 1.2](#)

Resource Properties

Resource class:	Handling
Resource referenced by:	—
Example Partition:	—
Input of processes:	PrintRolling
Output of processes:	—

Table 7-316: RollStand resource

Name	Data Type	Description
<i>MaxDiameter</i> ?	double	Maximal allowed diameter of the input component print roll.
<i>MaxWidth</i> ?	double	Maximal allowed width of the rolled input components.
Device ?	refelement	Further details of the RollStand .

7.2.156 RunList

RunList resources describe an ordered set of **LayoutElement** or **ByteMap** elements. Ordering and structure are defined using the generic partitioning mechanisms as described in Section 3.9.5, Description of Partitioned Resources.

RunList resources are used whenever an ordered set of page descriptions elements are specified. Depending on the process usage of a **RunList**, only certain *Types* of **LayoutElement** MAY be valid. For example, a pre-RIP imposition process requires **LayoutElement** elements of *Type* "*page*" or "*document*", whereas a post-RIP imposition process requires **ByteMap** elements. The usage is detailed in the descriptions of the processes that use the **RunList** resource. **RunList** resources allow structuring of multiple *Pages* into *Documents*. Multiple *Documents* that have a joint context MAY be grouped into *Sets*.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—

Example Partition: *PartVersion, Run, RunPage, RunSet, Separation, WebProduct*
Input of processes: RunLists are used as input resources by most processes that act on content data
Output of processes: RunLists are used as output resources by most processes that act on content data

Table 7-317: RunList resource (Section 1 of 4)

Name	Data Type	Description
<i>ComponentGranularity</i> = "Document" New in JDF 1.2	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 DigitalPrinting and Stitching node if <i>ComponentGranularity</i> = "Set". One of: <i>All</i> – The complete RunList , regardless of document or set breaks defines a new Component . <i>BundleItem</i> – An implicit PDL-defined document break or an explicit <i>EndOfBundleItem</i> defines a new Component . <i>Document</i> – An implicit PDL-defined document break or an explicit <i>EndOfDocument</i> defines a new Component . <i>Page</i> – Each page in the RunList defines a new Component . <i>Set</i> – Each set as defined by an implicit PDL-defined set break or an explicit <i>EndOfSet</i> defines a new Component .
<i>Directory</i> ?	URL	Defines a directory where the files that are associated with this RunList are to be copied to or from. If <i>Directory</i> is specified, it MUST be an Absolute URI [RFC3986] that implicitly also specifies a Base URI which is used to resolve any relative URL of RunList . See "Resolving RunList/@Directory and FileSpec/@URL URI references" on page 745 and [FileURL] for examples.
<i>DocCopies</i> = "1" New in JDF 1.1	integer	Number of instance document copies that this RunList represents. Specifying <i>DocCopies</i> is equivalent to repeating the sequence of RunList leaves between <i>EndOfDocument</i> = "true" for a total of <i>DocCopies</i> times. Note: It is illegal to specify <i>DocCopies</i> with different values of various leaves of a RunList representing the same instance document.
<i>DocNames</i> ?	Name-RangeList	A list of named documents in a multi-document file that supports named access to individual documents. The <i>DocNames</i> defaults to all documents. If <i>DocNames</i> occurs in the RunList , <i>Docs</i> is ignored if it is also present.
<i>Docs</i> ?	IntegerRangeList	Zero-based list of document indices in a multi-document file specified by the LayoutElement element.
<i>EndOfBundleItem</i> ? 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 <i>EndOfBundleItem</i> = "false", except for the last RunList partition, which always has an implied default value of <i>EndOfBundleItem</i> = "true". <i>EndOfBundleItem</i> MUST NOT be specified unless <i>ComponentGranularity</i> = "BundleItem".

Table 7-317: RunList resource (Section 2 of 4)

Name	Data Type	Description
<i>EndOfDocument</i> ?	boolean	If " <i>true</i> ", the last finished page in the RunList is the last page of an instance document. The precise handling of instance document changes is defined in the InsertSheet resource. If the RunList references a PDL that supports internal instance documents, <i>EndOfDocument</i> MAY be implied from the PDL. The implied default value of <i>EndOfDocument</i> = " <i>false</i> ", except for the last RunList partition leaf, which always has an implied default value of <i>EndOfDocument</i> = " <i>true</i> ".
<i>EndOfSet</i> ? New in JDF 1.1	boolean	If " <i>true</i> ", the last finished page in the RunList is the last page of a set of instance documents. The precise handling of instance document boundaries is defined in the InsertSheet resource. If the RunList references a PDL that supports internal sets, <i>EndOfSet</i> MAY be implied from the PDL. The implied default value of <i>EndOfSet</i> = " <i>false</i> ", except for the last RunList partition leaf, which always has an implied default value of <i>EndOfSet</i> = " <i>true</i> ".
<i>FirstPage</i> ?	integer	First finished page in the document that is described by this RunList . This attribute is generally used to describe preprepared files.
<i>IsPage</i> = " <i>true</i> "	boolean	If " <i>true</i> ", the individual RunList element defines one or more page slots, (e.g., for filling PlacedObjects). If " <i>false</i> ", the first parent partitioned RunList element with <i>IsPage</i> = " <i>true</i> " defines the page level. In general, <i>IsPage</i> = " <i>false</i> " for separations of a preprepared RunList .
<i>LogicalPage</i> ? Modified in JDF 1.1	integer	The logical page number of the first finished 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 finished page of the previous sibling RunList partition. If the RunList element is the first partition, <i>LogicalPage</i> defaults to "0". Note that is an error to specify <i>LogicalPage</i> to be less than the number of previously defined logical pages in the same partition, since this defines overlapping finished pages within the RunList partition.
<i>NDoc</i> ? New in JDF 1.1 Deprecated in JDF 1.2	integer	Total number of instance documents that are defined by the RunList . If <i>NDoc</i> 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 <i>Docs</i> is supported.
<i>NPage</i> ?	integer	Total number of pages (placed object slots or RunList elements with <i>IsPage</i> = " <i>true</i> ") that are defined by the RunList . If <i>NPage</i> is not specified, it defaults to all finished pages in the partitioned RunList elements that make up the RunList . If the RunList describes multiple instance documents or document sets, <i>NPage</i> refers to the total number of finished pages in all instance documents and sets. A RunList with <i>NPage</i> specified always refers to <i>NPage</i> pages, regardless of the number of pages of the referenced PDL. If <i>NPage</i> is not specified and no content is referenced, the RunList contains exactly one page.

Table 7-317: RunList resource (Section 3 of 4)

Name	Data Type	Description
<i>NSet</i> ? New in JDF 1.1 Deprecated in JDF 1.2	integer	Total number of instance document sets that are defined by the RunList . If <i>NSet</i> 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 <i>Sets</i> is supported.
<i>PageCopies</i> = "1" New in JDF 1.1	integer	Number of finished page copies that this RunList represents. Specifying <i>PageCopies</i> is equivalent to repeating the RunList leaves representing each page for a total of <i>PageCopies</i> times (e.g., a multiple represented by the value of <i>PageCopies</i> .) Note that pages specified by <i>PageCopies</i> are always assumed uncolated when calculating the index in the logical RunList , (e.g., <i>PageCopies</i> = "2" would result in a logical page sequence of 0 0 1 1 2 2, etc.).
<i>PageListIndex</i> ? New in JDF 1.2	IntegerRangeList	List of the indices of the <i>PageData</i> elements of the PageList specified in the LayoutElement referenced by this RunList . If not specified, the complete <i>PageListIndex</i> specified in the LayoutElement referenced by this RunList is applied.
<i>PageNames</i> ?	NameRangeList	A list of named pages in a multi-page file that supports named access to individual finished pages. The <i>PageNames</i> defaults to all pages. If <i>PageNames</i> is specified, then <i>FirstPage</i> , <i>NPage</i> , <i>SkipPage</i> and <i>Pages</i> MUST all be ignored if any is specified.
<i>Pages</i> ? Modified in JDF 1.1A	IntegerRangeList	Zero-based list of indices in the documents specified by the <i>LayoutElement</i> element and the <i>Docs</i> , <i>DocNames</i> , <i>Sets</i> and <i>SetNames</i> attributes. The <i>Pages</i> need not be in document order. If <i>Pages</i> is specified, <i>FirstPage</i> and <i>SkipPage</i> MUST be ignored. If none of <i>Pages</i> , <i>FirstPage</i> , <i>NPage</i> , <i>PageNames</i> or <i>SkipPage</i> is specified, all pages in the LayoutElement are selected.
<i>RunTag</i> ? New in JDF 1.1	NMTOKEN	Tag of a partition of a resource other than the RunList which is partitioned by <i>RunTags</i> . The partition matches if any of the entries in the <i>RunTags</i> list matches <i>RunTag</i> . Multiple entries in a RunList MAY have the same <i>RunTag</i> . If the RunList references a PDL that supports internal labels, <i>RunTag</i> MAY be implied from the PDL.
<i>SetCopies</i> = "1" New in JDF 1.1	integer	Number of instance document set copies that this RunList represents. Specifying <i>SetCopies</i> is equivalent to repeating the sequence of RunList leaves between <i>EndOfSet</i> = "true" for a total of <i>SetCopies</i> times. Note that it is illegal to specify <i>SetCopies</i> with different values of various leaves of a RunList representing the same instance document.
<i>SetNames</i> ? 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 <i>SetNames</i> defaults to all document sets specified by <i>Sets</i> . If <i>SetNames</i> occurs in the RunList , <i>Sets</i> is ignored if it is also present. <i>SetNames</i> is only valid if LayoutElement/@ElementType = "MultiSet".

Table 7-317: RunList resource (Section 4 of 4)

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 element. If not present, all document sets are selected. Sets is only valid if LayoutElement/@ElementType = "MultiSet" .
SkipPage ?	integer	Used when the RunList comprises every Nth page of the file. SkipPage indicates the number of finished pages to be skipped between each of the pages that comprise the RunList element. This is generally used to describe pre-separated 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 MUST 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 FileSpec is retained indefinitely. The FileSpec/Disposition takes precedence over RunList/Disposition .
DynamicInput *	element	Replacement text for a DynamicField element. This information defines the contents of a dynamic mark on the automated page layout (see Section 7.2.108.6, Dynamic marks). The mark MUST 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 <i>Name</i> attribute that, when linked to the <i>ReplaceField</i> attribute of the DynamicField element, defines the string that is to be replaced.
InsertSheet *	refelement	Describes how Sheets and Surfaces are to be completed and OPTIONAL media which MAY be inserted at the beginning or end of this RunList element.
InterpretedPDLData ? New in JDF 1.2	refelement	Represents the results of the PDL interpretation process. At most one of ByteMap , InterpretedPDLData or LayoutElement MUST 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, finished page or image. At most one of ByteMap , InterpretedPDLData or LayoutElement MUST be specified. If none of ByteMap , InterpretedPDLData or LayoutElement are specified, the RunList specifies empty content.
PageList ?	refelement	Specification of page metadata for pages described by this RunList .

— Element: DynamicInput

Table 7-318: DynamicInput element

Name	Data Type	Description
<i>Name ?</i>	string	Label that MUST match the <i>ReplaceField</i> attribute of the appropriate DynamicField element
—	text	Defines the text string that is to be inserted as a replacement for the text defined in <i>ReplaceField</i> of a DynamicField element.

Examples of Partitioning of a 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 **RunLists** appear in the XML document is significant. Note that the *Run* partitioning key has a string value, which MAY be non-numeric.

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>
```

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>
```

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>
```

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>

```

Simple Multi-File separated RunList

This example specifies all pages contained in Presep.pdf and following that, pages 1, 3 and 5 of each preprepared 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>
</RunList>

```

7.2.157 SaddleStitchingParams

[Deprecated in JDF 1.1](#) See "SaddleStitchingParams" on page 850 for details of this deprecated resource.

7.2.158 ScanParams

This resource provides the parameters for the **Scanning** process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>RunIndex</i>
Input of processes:	Scanning
Output of processes:	—

Table 7-319: ScanParams resource

Name	Data Type	Description
<i>BitDepth</i>	integer	Bit depth of a one-color separation.
<i>CompressionFilter</i> ?	enumeration	Specifies the compression filter to be used. Possible values include: <i>CCITTFaxEncode</i> – Used to select CCITT Group 3 or 4 facsimile encoding. <i>DCTEncode</i> – Used to select JPEG compression. <i>FlateEncode</i> – Used to select Zip compression. <i>WaveletEncode</i> – Used to select Wavelet compression. <i>JBIG2Encode</i> – Used to select JBIG2 monochrome compression.
<i>DCTQuality</i> ?	double	A value between 0 and 1 that indicates “how much” the process is to compress images. 0.0 means “do as loss-less compression as possible.” 1.0 means “do the maximum compression possible.”
<i>InputBox</i> ?	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.
<i>Magnification</i> = “1 1”	XYPair	Size of the output/size of the input for each dimension.
<i>MountID</i> ?	string	ID of the drum or other mounting device upon which the media is to be mounted.
<i>Mounting</i> ?	enumeration	Specifies how to mount originals. Possible values are: <i>Unfixed</i> – Original lies unfixed on the scanner tray/drum. <i>Fixed</i> – Original is fixed on the scanner tray/drum with transparent tape. <i>Wet</i> – Original is put in gel or oil and fixed on the scanner tray/drum. <i>Registered</i> – Original is fixed with registration holes. This value is used for copy dot scans.
<i>OutputColorSpace</i>	enumeration	Color space of the output images. Possible values are: <i>LAB</i> <i>RGB</i> <i>CMYK</i> <i>GrayScale</i>
<i>OutputResolution</i>	XYPair	X and Y resolution of the output bitmap, in dpi.
<i>OutputSize</i> ?	XYPair	X and Y dimension of the intended output image, in points.
<i>SplitDocuments</i> ?	integer	A number representing how many images are scanned before a new file is created.
FileSpec (<i>CorrectionProfile</i>) ?	refelement	A FileSpec resource pointing to an ICC profile that describes color corrections.
FileSpec (<i>TargetProfile</i>) ?	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).
FileSpec (<i>ScanProfile</i>) ?	refelement	A FileSpec resource pointing to an ICC profile that describes the scanner.

7.2.159 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:	Surface
Example Partition:	—
Input of processes:	Any printing process
Output of processes:	—

Table 7-320: ScavengerArea resource

Name	Data Type	Description
<i>Center</i>	XYPair	Position of the center of the scavenger area in the coordinates of the MarkObject that contains this mark.
<i>Rotation ?</i>	double	Rotation in degrees. Positive graduation figures indicate counter-clockwise rotation; negative figures indicate clockwise rotation.
<i>Size</i>	XYPair	Size of the scavenger area.
SeparationSpec * Modified in JDF 1.2	element	Set of separations to which the scavenger area is bound.

7.2.160 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:	ExposedMedia
Example Partition:	<i>Separation, SheetName, Side, SignatureName</i>
Input of processes:	Screening, ColorCorrection, ContoneCalibration
Output of processes:	—

Table 7-321: ScreeningParams resource

Name	Data Type	Description
<i>IgnoreSourceFile = "true"</i>	boolean	Specifies whether to ignore the screen settings (e.g., set-screen, 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.
<i>AbortJobWhenScreenMatchingFails ?</i> 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 <i>SettingsPolicy</i> in JDF 1.2 and beyond.
ScreenSelector * Modified in JDF 1.1	element	List of screen selectors. A screen selector is included for each separation, including a default specification.

— Element: ScreenSelector

Description of screening for a selection of source object types and separations.

Table 7-322: ScreenSelector element (Section 1 of 2)

Name	Data Type	Description
<i>Angle</i> ?	double	Specifies the first angle of the screen when AM screening is used, otherwise <i>Angle</i> is ignored. At most one of <i>Angle</i> or <i>AngleMap</i> MUST be specified. If neither <i>Angle</i> nor <i>AngleMap</i> are specified, the angle is determined by the default of the selected <i>ScreeningFamily</i> .
<i>AngleMap</i> ? 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 <i>AngleMap</i> = <i>Cyan</i> . In FM screening, <i>AngleMap</i> specifies the mapping of the separation specific screen functions, (e.g., threshold arrays). At most one of <i>Angle</i> or <i>AngleMap</i> MUST be specified. This mapping is not transitive, so, when <i>Separation</i> already specifies a color with a known default ^a , it specifies the angle of the separation defined by <i>AngleMap</i> prior to that separation being mapped. The following example specifies that <i>Black</i> is to be mapped to the <i>Cyan</i> default separation and <i>Cyan</i> to the <i>Black</i> default separation. The third line maps <i>Spot1</i> to <i>Magenta</i> . <pre><ScreenSelector AngleMap="Black" Separation="Cyan"/> <ScreenSelector AngleMap="Cyan" Separation="Black"/> <ScreenSelector AngleMap="Magenta" Separation="Spot1"/></pre>
<i>DotSize</i> ? New in JDF 1.1	double	Specifies the dot size of the screen, in microns [μm], when FM screening (<i>ScreeningType</i> = "FM" or "Adaptive") is used, otherwise <i>DotSize</i> is ignored.
<i>Frequency</i> ? 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 <i>Frequency</i> is ignored. With some screens, frequency can change as a function of gray level. In this case, the <i>Frequency</i> value is interpreted for a midtone (50%) gray level. If <i>Frequency</i> is not specified, the frequency is determined by the default of the selected <i>ScreeningFamily</i> .
<i>ScreeningFamily</i> ?	string	Vendor specific screening family name. Sample values removed in JDF 1.2
<i>ScreeningType</i> ? Modified in JDF 1.2	enumeration	General type of screening. Possible values are: <i>Adaptive</i> <i>AM</i> – Can be line or dot. (See <i>SpotFunction</i> .) <i>ErrorDiffusion</i> <i>FM</i> – Includes all stochastic screening types. <i>HybridAM-FM</i> <i>HybridAMline-dot</i>
<i>Separation</i> = "All"	string	The name of the separation. If <i>Separation</i> = "All", the <i>ScreenSelector</i> is to be applied to all separations that are not specified explicitly.
<i>SourceFrequency</i> ? Modified in JDF 1.2	DoubleRange	Specifies the line frequency of screens which is to 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 <i>ScreenSelector</i> .

Table 7-322: ScreenSelector element (Section 2 of 2)

Name	Data Type	Description
<i>SourceObjects</i> = "All"	enumerations	Identifies the class(es) of incoming graphical objects on which to use the selected screen. Possible values are: <i>All</i> <i>ImagePhotographic</i> – Contone images. <i>ImageScreenShot</i> – Images largely comprised of rasterized vector art. <i>Text</i> <i>LineArt</i> – Vector objects other than text. <i>SmoothShades</i> – Gradients and blends.
<i>SpotFunction</i> ?	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. Example values include: <i>Round</i> <i>Diamond</i> <i>Ellipse</i> <i>EllipseA</i> <i>InvertedEllipseA</i> <i>EllipseB</i> <i>EllipseC</i> <i>InvertedEllipseC</i> <i>Line</i> <i>LineX</i> <i>LineY</i> <i>Square</i> <i>Cross</i> <i>Rhomboid</i> <i>DoubleDot</i> <i>InvertedDoubleDot</i> <i>SimpleDot</i> <i>InvertedSimpleDot</i> <i>CosineDot</i> <i>Double</i> <i>InvertedDouble</i>

- a. In general this will be a CMYK process color, but it can also be another process color, (e.g., HexaChrome™).

7.2.161 SeparationControlParams

This resource provides the controls needed to separate composite color files.

Resource Properties

Resource class: Parameter

Resource referenced by: —

Example Partition: —
 Input of processes: **Separation**
 Output of processes: —

Table 7-323: SeparationControlParams resource

Name	Data Type	Description
AutomatedOverPrintParams ?	refelement	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.

7.2.162 SeparationSpec

This resource specifies a specific separation, and is usually used to define a list or sequence of separations.

Resource Properties

Resource class: ResourceElement
 Resource referenced by: **ColorantControl, LayoutElement, PageList, RegisterMark, TransferFunctionControl**

Example Partition: —
 Input of processes: —
 Output of processes: —

Table 7-324: SeparationSpec resource

Name	Data Type	Description
<i>Name</i>	string	Name of one specific separation.

7.2.163 ShapeCuttingParams

[New in JDF 1.1](#)

ShapeCuttingParams defines the details of the **ShapeCutting** process.

Resource Properties

Resource class: Parameter
 Resource referenced by: —
 Example Partition: —
 Input of processes: **ShapeCutting**
 Output of processes: —

Table 7-325: ShapeCuttingParams resource (Section 1 of 2)

Name	Data Type	Description
<i>DeliveryMode ?</i> New in JDF 1.3	enumeration	Following delivery modes of the cut shapes are possible: <i>FullSheet</i> – 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. <i>RemoveGripperMargin</i> – The output of the die-cutter are complete sheets. The blanks are kept in place with nicks. Front waste (gripper margin) has been removed. <i>SeparateBlanks</i> – The output of the die-cutter are blanks that have been removed from the sheets.

Table 7-325: ShapeCuttingParams resource (Section 2 of 2)

Name	Data Type	Description
SheetLay ? New in JDF 1.3	enumeration	Lay of input media. Reference edge of where paper is placed in the feeder. Possible values are: <i>Center</i> <i>Left</i> <i>Right</i>
DieLayout ? New in JDF 1.3	reference	An element containing the reference of an external file describing the cutting and other paths.
Shape *	element	Details of each individual cut shape

— Element: Shape**Table 7-326: Shape element**

Name	Data Type	Description
CutBox ?	rectangle	Specification of a rectangular window.
CutOut = "false"	boolean	If " <i>true</i> ", the inside of a specified shape will be removed. If " <i>false</i> ", 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"	enumeration	Type of cut or perforation used. Possible values are: <i>Cut</i> – Full cut. <i>Perforate</i> – Interrupted perforation that does not span the entire sheet
Material ?	string	Transparent material that fills a shape (e.g., an envelope window) that was cut out when <i>CutOut</i> = " <i>true</i> ".
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. Possible values are: <i>Rectangular</i> <i>Round</i> <i>Path</i>
StationName ? New in JDF 1.3	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).

7.2.164 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 "Layout" on page 487.

7.2.165 ShrinkingParams[New in JDF 1.1](#)

This resource provides the parameters for the **Shrinking** process in shrink wrapping.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>Shrinking</i>
Output of processes:	—

Table 7-327: ShrinkingParams resource

Name	Data Type	Description
<i>Duration ?</i>	duration	Shrinking time.
<i>ShrinkingMethod</i> = "ShrinkHot"	enumeration	Specifics of the shrinking method for shrink wrapping. <i>ShrinkCool</i> <i>ShrinkHot</i>
<i>Temperature ?</i>	double	Oven temperature in ° Centigrade.

7.2.166 SideSewingParams

[Deprecated in JDF 1.1](#) See "SideSewingParams" on page 852 for details of this deprecated resource.

7.2.167 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
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>SpinePreparation</i>
Output of processes:	—

Table 7-328: SpinePreparationParams resource (Section 1 of 2)

Name	Data Type	Description
<i>FlexValue ?</i> Deprecated in JDF 1.2	double	Flex quality parameter, in [N/cm]. In JDF 1.2 and beyond, <i>FlexValue</i> is defined in QualityControlParams/BindingQualityParams . See "QualityControlParams" on page 561 for details.
<i>MillingDepth ?</i> 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.
<i>NotchingDistance ?</i>	double	Notching distance, in points.
<i>NotchingDepth ?</i>	double	Notching depth relative to the leveled spine, in points. If not specified, there is no notching.
<i>Operations ?</i>	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. See Table 7-329 for values of <i>Operations</i> .
<i>PullOutValue ?</i> Deprecated in JDF 1.2	double	Pull out quality parameter, in [N/cm]. In JDF 1.2 and beyond, <i>PullOutValue</i> is defined in QualityControlParams/BindingQualityParams . See "QualityControlParams" on page 561 for details.

Table 7-328: SpinePreparationParams resource (Section 2 of 2)

Name	Data Type	Description
<i>StartPosition</i> = "0"	double	Starting position of milling tool along the Y-axis of the operation coordinate system.
<i>WorkingLength</i> ?	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.

— Attribute: Operations

Table 7-329: Operations attribute – possible values

Value	Description
<i>Brushing</i>	Brushes away dust from the spine to improve the binding quality.
<i>FiberRoughing</i>	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.
<i>Leveling</i>	After milling the spine, any uneven areas are leveled to achieve an even surface.
<i>Milling</i>	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.
<i>Notching</i>	This gives a clamping effect on the spine which is desirable for some products.
<i>Sanding</i>	Is used for voluminous book papers.
<i>Shredding</i>	Produces a relatively smooth surface. Further operations like <i>Notching</i> , <i>Leveling</i> , <i>FiberRoughing</i> , <i>Sanding</i> or <i>Brushing</i> are necessary.

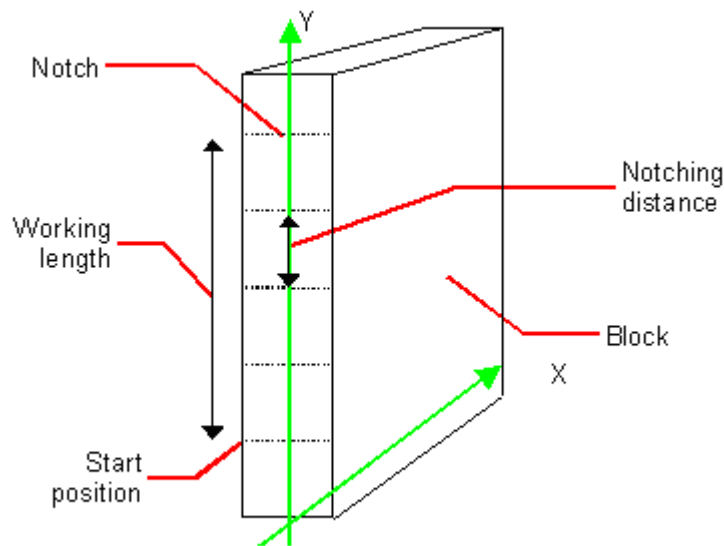


Figure 7-34: Parameters and coordinate systems for the SpinePreparation process

7.2.168 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

Resource referenced by: —
 Example Partition: —
 Input of processes: *SpineTaping*
 Output of processes: —

Table 7-330: SpineTapingParams resource

Name	Data Type	Description
<i>HorizontalExcess</i> ?	double	Taping spine excess on each side. The tape is assumed to be centered between left and right.
<i>StripBrand</i> ?	string	Strip brand.
<i>StripColor</i> ?	NamedColor	Color of the strip.
<i>StripLength</i> ?	double	Length of strip material along binding edge. If not defined, the default case is that the <i>StripLength</i> be equivalent to the length of the spine.
<i>StripMaterial</i> ?	enumeration	Strip material. Possible values are: <i>Calico</i> <i>Cardboard</i> <i>CrepePaper</i> <i>Gauze</i> <i>Paper</i> <i>PaperlinedMules</i> <i>Tape</i>
<i>TopExcess</i> = "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.

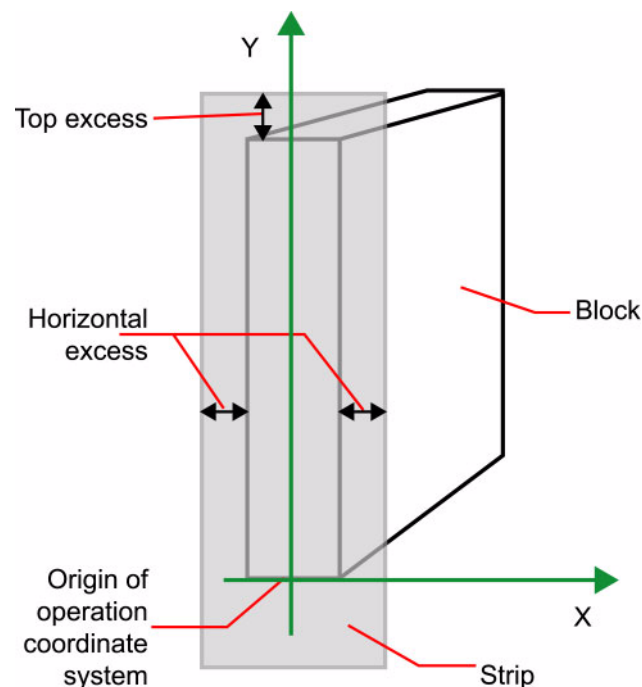


Figure 7-35: Parameters and coordinate system for the SpineTaping process

7.2.169 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 7-36).

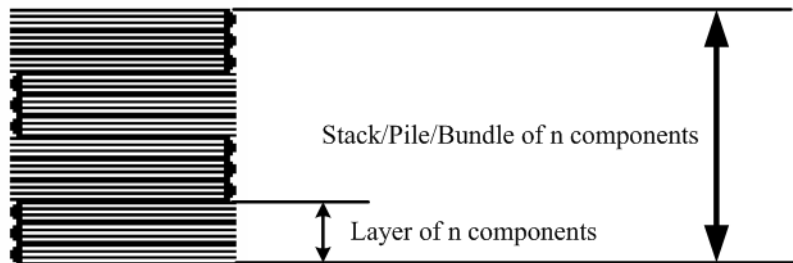


Figure 7-36: Stacking Layers

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 7-37).

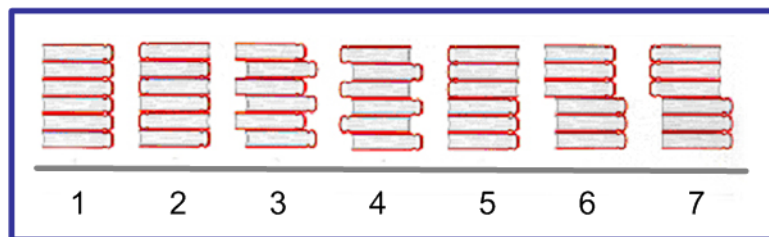


Figure 7-37: Pile Patterns

Table 7-331: Parameters in Stacking

File 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.

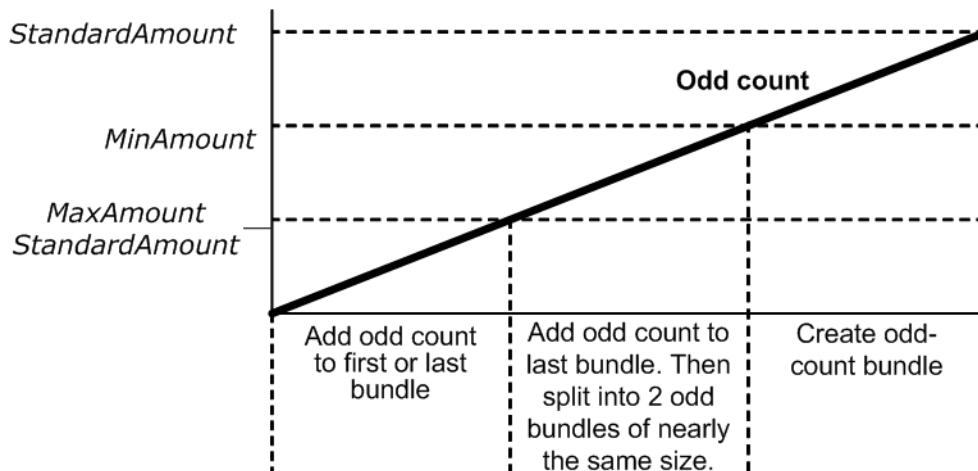


Figure 7-38: Odd count handling for a Bundle

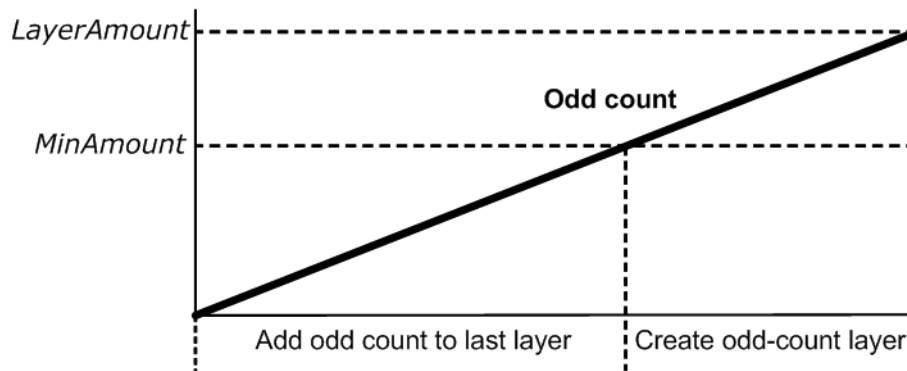


Figure 7-39: Odd count handling for a Layer

Resource Properties

Resource class: Parameter
 Resource referenced by: —
 Example Partition: —
 Input of processes: **Stacking**
 Output of processes: —

Table 7-332: StackingParams resource (Section 1 of 2)

Name	Data Type	Description
<i>Compensate</i> = "true"	boolean	180 degree rotation applied to successive layers to compensate for uneven stacking. If <i>LayerAmount</i> = <i>StandardAmount</i> , there is one layer, and effectively no compensation.
<i>LayerAmount</i> ? Modified in JDF 1.2	IntegerList	Ordered number of products in a layer. The first number is the first <i>LayerAmount</i> , 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 <i>StandardAmount</i> . When not known, the default case is that the value of <i>LayerAmount</i> be equivalent to the value of <i>StandardAmount</i> .
<i>MaxAmount</i> ?	integer	Maximum number of products in a stack, <i>MaxAmount</i> >= <i>StandardAmount</i> . When not known, the default case is that the value of <i>MaxAmount</i> be equivalent to the value of <i>StandardAmount</i> .

Table 7-332: StackingParams resource (Section 2 of 2)

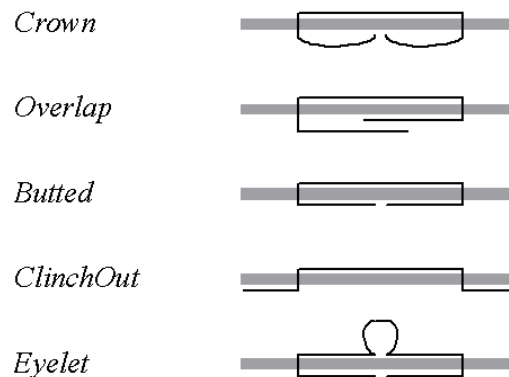
Name	Data Type	Description
<i>MinAmount</i> ?	integer	Minimum number of products in a stack or layer, $(MaxAmount - StandardAmount) \leq MinAmount < StandardAmount$ and $MinAmount < LayerAmount$. Where not known, the default case is to use a value equivalent to $MaxAmount - StandardAmount$.
<i>MaxWeight</i> ?	double	Maximum weight of a stack in grams.
<i>Offset</i> ? 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.
<i>StandardAmount</i> ? Modified in JDF 1.2	integer	Number of products in a standard stack.
<i>UnderLays</i> ? 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 <i>UnderLays</i> . 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 hardcover books.

7.2.170 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 7-40: Staple shapes**

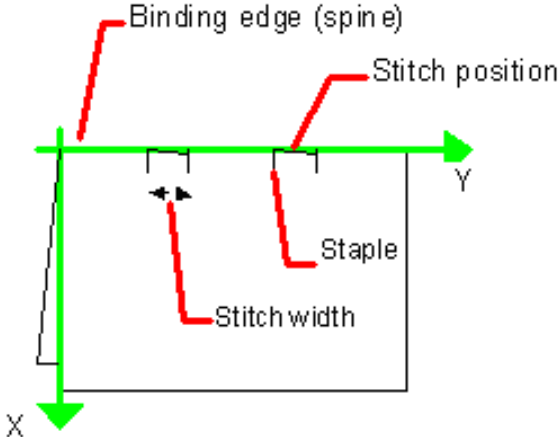


Figure 7-41: Parameters and coordinate system used for saddle stitching

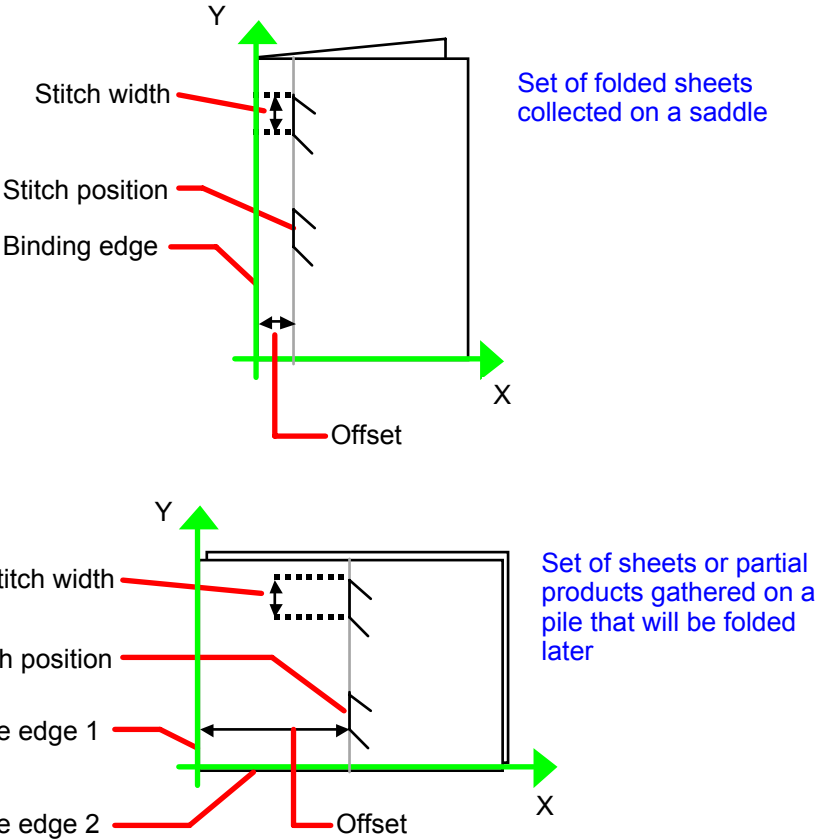


Figure 7-42: Parameters and coordinate system used for Stitching

Resource Properties

Resource class: Parameter

Resource referenced by: —
 Example Partition: *ProductionRun, SubRun, WebProduct*
 Input of processes: **Stitching**
 Output of processes: —

Table 7-333: **StitchingParams** resource (Section 1 of 2)

Name	Data Type	Description
<i>Angle</i> ?	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 <i>StitchType</i> = "Saddle", <i>Angle</i> MUST be ignored
<i>NumberOfStitches</i> ? 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.
<i>ReferenceEdge</i> ? 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 combined with other processes (e.g., DigitalPrinting) where, when combined, there is no input Component to be stitched. Possible values are: <i>Top</i> <i>Left</i> <i>Right</i> <i>Bottom</i> <i>ReferenceEdge</i> has been replaced in JDF 1.2 and beyond with an explicit <i>Transformation</i> or <i>Orientation</i> of the input Component . If both <i>Transformation/Orientation</i> and <i>ReferenceEdge</i> are specified, the result is the matrix product of both transformations. <i>Transformation/Orientation</i> MUST be applied first.
<i>Offset</i> ?	double	Distance between stitch and binding edge. If <i>StitchType</i> = "Saddle", <i>Offset</i> MUST be ignored. Note that it is possible to describe saddle stitching with an offset by defining <i>StitchType</i> = "Side" with a large <i>Offset</i> value.
<i>StapleShape</i> ?	enumeration	Specifies the shape of the staples to be used. Possible values are: <i>Crown</i> <i>Overlap</i> <i>Butted</i> <i>ClinchOut</i> <i>Eyelet</i> Representations of these values are displayed in Figure 7-40.
<i>StitchFromFront</i> ? Deprecated in JDF 1.2	boolean	If "true", Stitching is done from front to back. Otherwise it is done from back to front. The <i>StitchFromFront</i> has been replaced with an explicit <i>Transformation</i> or <i>Orientation</i> of the input Component .
<i>StitchPositions</i> ?	DoubleList	Array containing the stitch positions. The center of the stitch MUST be specified, and the number of entries MUST match the number given in <i>NumberOfStitches</i> .

Table 7-333: **StitchingParams** resource (Section 2 of 2)

Name	Data Type	Description
<i>StitchType</i> ? Modified in JDF 1.2	enumeration	Specifies the type of the Stitching operation. One of: <i>Corner</i> – Stitch in the corner that is at the clockwise end of the reference edge. For example, to stitch in the upper right corner set <code>ComponentLink/@Orientation = "Rotate90"</code> . <i>Saddle</i> – Stitch on the middle fold which is on the saddle. <i>Side</i> – Stitch along the reference edge.
<i>StitchWidth</i> ?	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.
<i>WireGauge</i> ?	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.
<i>WireBrand</i> ?	string	Brand of the wire to be used.

7.2.171 Strap

[New in JDF 1.1](#)

Resource Properties

Resource class:	Consumable
Resource referenced by:	—
Example Partition:	—
Input of processes:	Strapping
Output of processes:	—

Table 7-334: **Strap** resource

Name	Data Type	Description
<i>StrapColor</i> ?	NamedColor	Color of the string or strap.
<i>Material</i>	enumeration	Strap material. Possible values are: <i>AdhesiveTape</i> <i>Strap</i> <i>String</i>

7.2.172 StrappingParams

[New in JDF 1.1](#)

StrappingParams defines the details of **Strapping**.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	Strapping
Output of processes:	—

Table 7-335: StrappingParams resource

Name	Data Type	Description
<i>StrappingType</i>	enumeration	Strapping pattern. Allowed values are: <i>Single</i> – One strap. <i>Double</i> – Two parallel single straps. <i>Cross</i> – Two crossed straps. <i>DoubleCross</i> – Two cross straps that strap each side of a box.
<i>StrapPositions</i> ? New in JDF 1.3	NumberList	Positions of the Straps beginning from the origin of the coordinate system (bottom side) increasing from min to max in points. Each Strap is defined by a 3-tuple of which two values MUST 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 7-43 and Figure 7-44). 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.

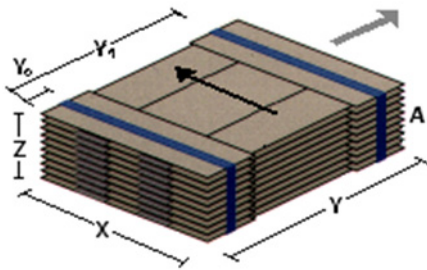


Figure 7-43: Strapped bundle

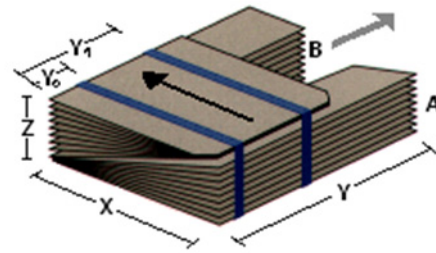


Figure 7-44: Strapped bundle with subbundles

7.2.173 StripBindingParams

[New in JDF 1.1](#)

This resource describes the details of the *StripBinding* process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>StripBinding</i>
Output of processes:	—

Table 7-336: StripBindingParams resource (Section 1 of 2)

Name	Data Type	Description
<i>Brand</i> ?	string	The name of the comb manufacturer and the name of the specific item.
<i>Distance</i> ? Deprecated in JDF 1.2	double	The distance between the pins and the distance between the holes of the prepunched sheets MUST be the same. In JDF 1.2 and beyond, use the value implied by HoleMakingParams/@HoleType .

Table 7-336: StripBindingParams resource (Section 2 of 2)

Name	Data Type	Description
<i>Length</i> ?	double	The length of the pin is determined by the height of the pile of sheets to be bound.
<i>StripColor</i> ?	NamedColor	Determines the color of the strip.
HoleMakingParams ? New in JDF 1.2	refelement	Details of the holes in StripBinding .

7.2.174 StrippingParams

[New in JDF 1.2](#)

The **StrippingParams** resource is a high-level description of how a **Component** is to 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 **BinderySignatures** 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 MUST 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

Resource referenced by: —

Example Partition: *SignatureName, SheetName, BinderySignatureName, PartVersion, SectionIndex, CellIndex*

Input of processes: **Stripping**

Output of processes: —

Table 7-337: StrippingParams resource (Section 1 of 2)

Name	Data Type	Description
<i>AssemblyID</i> ? Deprecated in JDF 1.3	string	Identification of the Assembly or AssemblySection to which the StrippingParams or partition belongs.
<i>AssemblyIDs</i> ? New in JDF 1.3	NMTOKENS	Identification of the Assembly elements or AssemblySection elements to which the StrippingParams or partition belongs.
<i>JobID</i> ?	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.
<i>SectionList</i> ?	IntegerList	List of numbered sections (of the AssemblySection elements with matching <i>JobID</i> and <i>AssemblyIDs</i>) 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 SignatureCells with <i>SectionIndex</i> = "0"; the section that matches the second entry is flowed into SignatureCells with <i>SectionIndex</i> = "1"; and so forth. <i>SectionList</i> MUST NOT be specified at the <i>CellIndex</i> partition level.

Table 7-337: StrippingParams resource (Section 2 of 2)

Name	Data Type	Description
<i>WorkStyle</i> ?	enumeration	The direction in which to turn the press sheet. Possible values are defined in ConventionalPrintingParams/@WorkStyle : <i>WorkStyle</i> MUST NOT be specified at partition levels lower than <i>SheetName</i> .
BinderySignature	refelement	Describes BinderySignature which is placed onto the sheets defined by StrippingParams . If multiple BinderySignature elements are placed on the same sheet, StrippingParams MUST be partitioned by <i>BinderySignatureName</i> . BinderySignature MUST NOT be specified at partition levels lower than <i>PartVersion</i> .
Device *	refelement	Devices that the MIS expects to execute this StrippingParams . This MAY include prepress devices, presses or finishing devices. Press devices MUST NOT be specified at partition levels lower than <i>SheetName</i> .
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 MUST NOT be specified at partition levels lower than <i>SheetName</i> .
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 <i>SignatureCells</i> , 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 <i>Orientation</i> attribute one can influence this default behavior. When the BinderySignature is defined by <i>FoldCatalog</i> or <i>Folds</i> , then, by default, the lay is placed on the left front side of the sheet. Using the <i>Orientation</i> attribute one can influence this default behavior. Position MUST NOT be specified at partition levels lower than <i>PartVersion</i> .
StripCellParams ?	element	Specification of the parameters of the cells in the layout.
StripMark * New in JDF 1.3	element	Indicates areas on the StrippingParams reserved for Marks. StripMark MUST NOT be specified at partition levels that are more granular than <i>SheetName</i> .

— Element: 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.

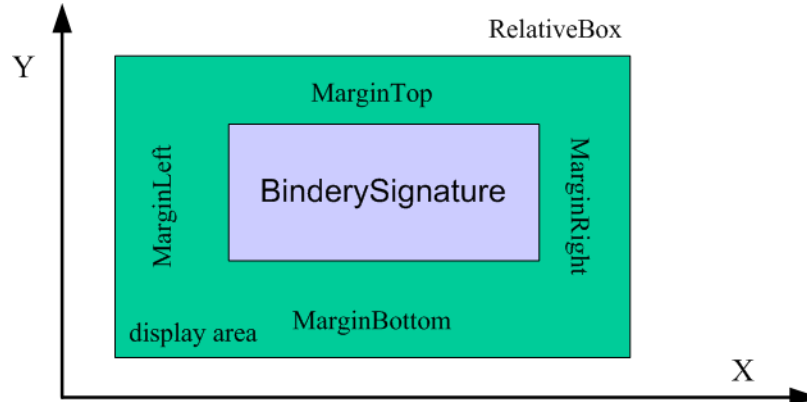


Figure 7-45: RelativeBox including margins

Table 7-338: Position element (Section 1 of 2)

Name	Data Type	Description
<i>AbsoluteBox</i> ? New in JDF 1.3	Rectangle	Absolute position 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 <i>Orientation</i> transformation. The display area includes the absolute margins defined by <i>MarginTop</i> , <i>MarginBottom</i> , <i>MarginLeft</i> and <i>MarginRight</i> . <i>AbsoluteBox</i> overrides <i>RelativeBox</i> if both are specified.
<i>BlockName</i> ? New in JDF 1.3	NMTOKEN	Identifies a CutBlock resulting from a <i>Cutting</i> process if the element specified by the Position is created by <i>Cutting</i> .
<i>MarginBottom</i> ?	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 .
<i>MarginTop</i> ?	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 .
<i>MarginLeft</i> ?	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 .
<i>MarginRight</i> ?	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 .
<i>Orientation</i> ?	Orientation	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,” on page 26.

Table 7-338: Position element (Section 2 of 2)

Name	Data Type	Description
<i>RelativeBox ?</i>	rectangle	Relative position of the display area of this BinderySignature on the front side of the StrippingParams . The BinderySignature is placed onto the display area after applying the <i>Orientation</i> transformation. The display area includes the absolute margins defined by <i>MarginTop</i> , <i>MarginBottom</i> , <i>MarginLeft</i> and <i>MarginRight</i> . <i>AbsoluteBox</i> overrides <i>RelativeBox</i> if both are specified. If neither <i>AbsoluteBox</i> nor <i>RelativeBox</i> are specified, the full relative media box "0 0 1.0 1.0" is applied.

— Element: StripCellParams

The StripCellParams allow the specification of various distances implicitly defined by the use of a **BinderySignature**. The picture 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.

Note: In practice, **StripCellParams** values will usually be greater than or equal to zero and have no default.

Table 7-339: StripCellParams element (Section 1 of 2)

Name	Data Type	Description
<i>BleedFace ?</i>	double	(F1) Value for the bleed at the face side.
<i>BleedSpine ?</i>	double	(S1) Value for the bleed at the spine side.
<i>BleedHead ?</i>	double	(H1) Value for the bleed at the head side.
<i>BleedFoot ?</i>	double	(T1) Value for the bleed at the foot side.
<i>TrimFace ?</i>	double	(F2) Value for the trim distance at the face side. When no Folding is done, this is the right margin. When <i>BinderySignatureType</i> = "Grid", the horizontal gutter between cells is <i>TrimFace</i> + <i>Spine</i> .
<i>Spine ?</i>	double	(S2) Amount of paper which is not cut-off from the spine. When no Folding is done, this is the left margin. When <i>BinderySignatureType</i> = "Grid", the horizontal gutter between cells is <i>TrimFace</i> + <i>Spine</i> .
<i>TrimHead ?</i>	double	(H2) Value for the trim distance at the head side. When no Folding is done, this is the top margin. When <i>BinderySignatureType</i> = "Grid", the vertical gutter between cells is <i>TrimHead</i> + <i>TrimFoot</i> .
<i>TrimFoot ?</i>	double	(T2) Value for the trim distance at the foot side. When no Folding is done, this is the bottom margin. When <i>BinderySignatureType</i> = "Grid", the vertical gutter between cells is <i>TrimHead</i> + <i>TrimFoot</i> .
<i>FrontOverfold ?</i>	double	(F3) Value for the overfold at the front side.
<i>BackOverfold ?</i>	double	(F3) Value for the overfold at the back side.
<i>MillingDepth ?</i>	double	(S3) Amount of paper cut-off from the spine.
<i>CutWidthHead ?</i>	double	(H3) Amount of paper lost by cutting at the head side.
<i>CutWidthFoot ?</i>	double	(T3) Amount of paper lost by cutting at the foot side.
<i>TrimSize ?</i>	XYPair	Defines the dimensions of the trim box.
<i>Creep ?</i>	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.

Table 7-339: StripCellParams element (Section 2 of 2)

Name	Data Type	Description
Mask ? New in JDF 1.3	enumeration	The definition of the clipping mask for the placed graphics. Possible values are: <i>None</i> – No mask <i>TrimBox</i> – The mask is derived from the TrimBox as defined by the SignatureCell and StripCellParams . <i>BleedBox</i> – The mask is derived from the BleedBox as defined by the SignatureCell and StripCellParams <i>SourceTrimBox</i> – The mask is derived from the TrimBox of the graphical element placed in the SignatureCell <i>SourceBleedBox</i> – The mask is derived from the BleedBox of the graphical element placed in the SignatureCell . <i>PDL</i> – The mask is derived from the PDL of the graphics. The attribute MaskSeparation determines which separation is to be used as the clipping mask for the graphics. <i>DieCut</i> – The mask is the cut line as defined in the DieLayout . <i>DieBleed</i> – 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 MUST be specified if and only if Mask = "PDL". Color/@ColorType of this separation MUST be <i>DieLine</i> .
Sides ?	enumeration	Indicates whether contents are to be printed on one or both sides of the media. Possible values are: <i>OneSided</i> – Page contents will only be imaged on one side of the media. <i>TwoSidedHeadToHead</i> – Impose pages upon the front and back sides of media sheets so that the head (top) of page contents back up to each other. <i>TwoSidedHeadToFoot</i> – 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.

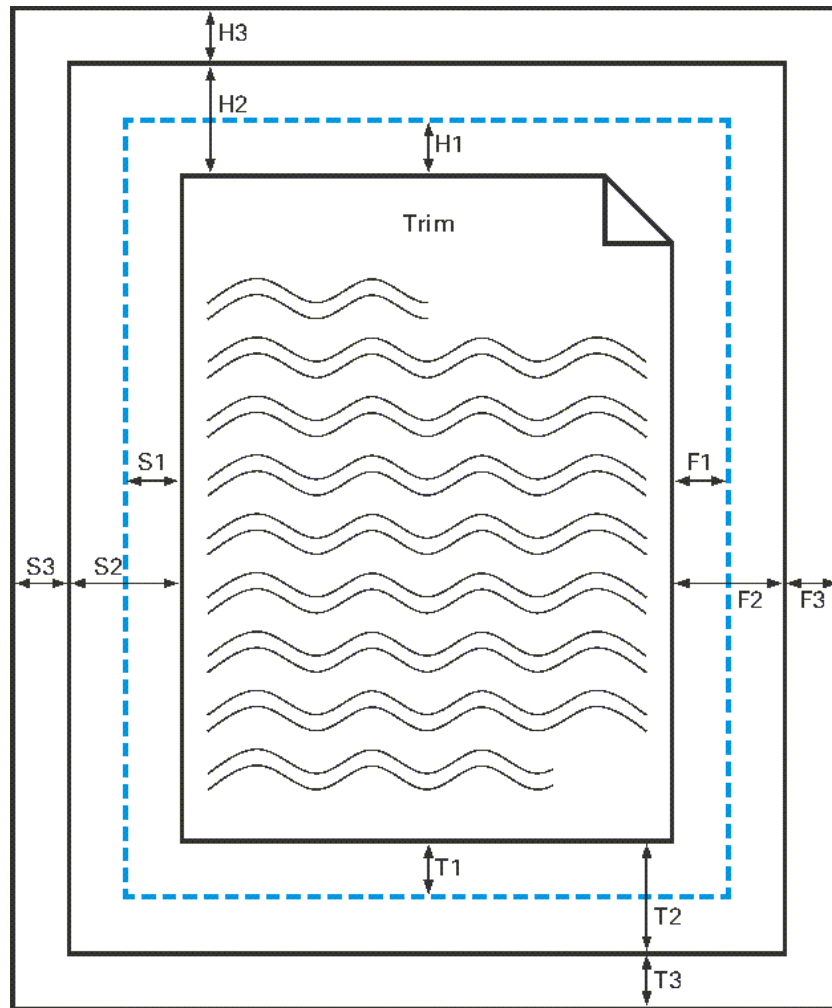


Figure 7-46: Definition of margins in StripCellParams

— Element: StripMark

[New in JDF 1.3](#)

The StripMark element specifies Marks to be placed on the sheet.

Table 7-340: StripMark element

Name	Data Type	Description
<i>MarkName</i> ?	NMTOKEN	Mark that is to be marked on the StrippingParams . See Table 7-341 for predefined values included in <i>MarkName</i> .
<i>MarkSide</i> ?	enumeration	Side and alignment of the marks. See Table 7-342 for allowed values of <i>MarkSide</i> .
<i>StripMarkDetails</i> ?	string	More detailed information about the StripMark.
Position ?	element	Specifies where to place the StripMark on the StrippingParams .
JobField ?	refelement	Specific Information about Marks of type <i>JobField</i> . JobField MUST NOT be specified unless <i>MarkName</i> = "JobField". This JobField MUST NOT contain a DeviceMark element. Positioning of the JobField is defined by the Position element.

— Attribute: **MarkName**Table 7-341: **MarkName** attribute – included values

Value	Description	Value	Description
<i>CIELABMeasuringField</i>		<i>IdentificationField</i>	
<i>ColorControlStrip</i>		<i>JobField</i>	
<i>ColorRegisterMark</i>		<i>PaperPathRegisterMark</i>	
<i>CutMark</i>		<i>RegisterMark</i>	
<i>DensityMeasuringField</i>		<i>ScavengerArea</i>	

— Attribute: **MarkSide**Table 7-342: **MarkSide** attribute – possible values

Value	Description
<i>Front</i>	The Mark is placed on the front side of the surface and Position is specified in the coordinate system of the front surface.
<i>Back</i>	The Mark is placed on the back side of the surface and Position is specified in the coordinate system of the back surface.
<i>TwoSidedBackToBack</i>	The position of the mark on the back is derived from the position of the mark on the front side and StrippingParams/@WorkStyle .
<i>TwoSidedIndependent</i>	The Mark is placed on both sides of the surface and the position is specified in the coordinate system of the respective surface.

7.2.175 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 "Layout" on page 487.

7.2.176 ThreadSealingParams[New in JDF 1.1](#)

This resource provides the parameters for the **ThreadSealing** process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	ThreadSealing
Output of processes:	—

Table 7-343: **ThreadSealingParams** resource (Section 1 of 2)

Name	Data Type	Description
<i>BlindStitch?</i>	boolean	A value of "true" specifies a blind stitch after the last stitch.
<i>ThreadMaterial?</i>	enumeration	Thread material. Possible values are: <i>Cotton</i> <i>Nylon</i> <i>Polyester</i>
<i>ThreadPositions?</i> Modified in JDF 1.2	DoubleList	Array containing the y-coordinate of the center positions of the thread.

Table 7-343: ThreadSealingParams resource (Section 2 of 2)

Name	Data Type	Description
<u>ThreadLength ?</u> <u>Modified in JDF 1.2</u>	double	Length of one thread.
<u>ThreadStitchWidth ?</u> <u>Modified in JDF 1.2</u>	double	Width of one stitch.
<u>SealingTemperature ?</u>	integer	Temperature needed for sealing thread and sheets together, in degrees centigrade.

7.2.177 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).

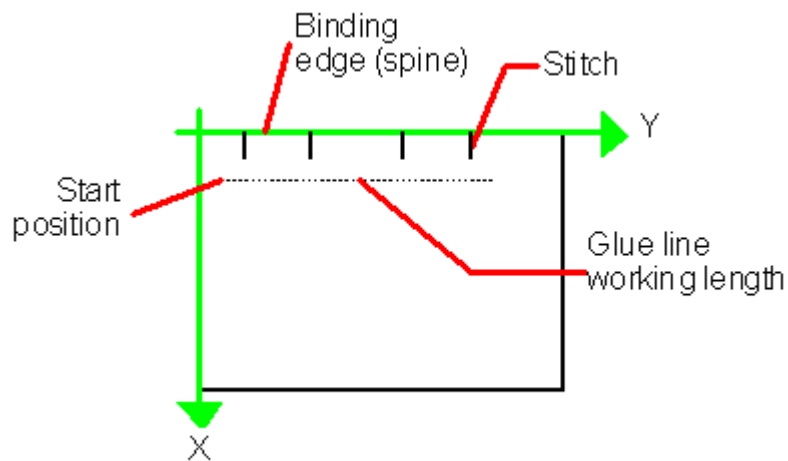


Figure 7-47: Parameters and coordinate system used for thread sewing

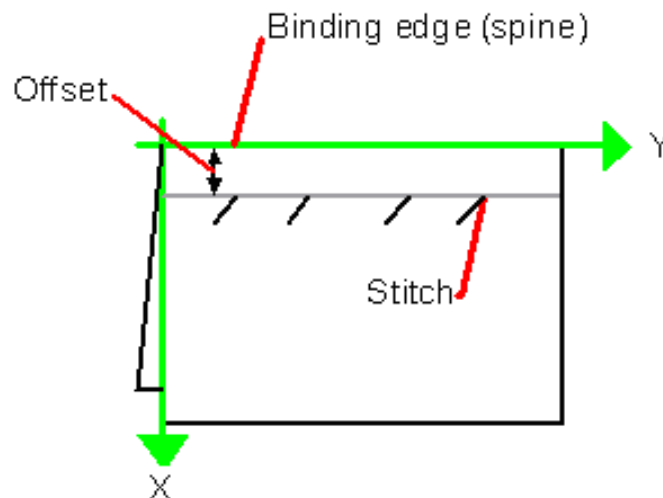


Figure 7-48: Parameters and coordinate system used for side sewing

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>ThreadSewing</i>
Output of processes:	—

Table 7-344: ThreadSewingParams resource

Name	Data Type	Description
<i>BlindStitch</i> = "false"	boolean	A value of "true" specifies a blind stitch after the last stitch.
<i>CastingMaterial</i> ?	enumeration	Casting material of the thread being used. Possible values are: <i>Cotton</i> <i>Nylon</i> <i>Polyester</i>
<i>CoreMaterial</i> ?	enumeration	Core material of the thread being used. This attribute MUST be used to define the thread material if there is no casting. Possible values are: <i>Cotton</i> <i>Nylon</i> <i>Polyester</i>
<i>GlueLineRefSheets</i> ? 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. <i>GlueLineRefSheets</i> MUST NOT be specified unless <i>GlueLine</i> is defined.
<i>Offset</i> ? New in JDF 1.1	double	Specifies the distance between the stitch and the binding edge. Used only for side stitching.
<i>NumberOfNeedles</i> ? Modified in JDF 1.2	integer	Specifies the number of needles to be used.
<i>NeedlePositions</i> ?	DoubleList	Array containing the y-coordinate of the needle positions. The number of entries MUST match the number specified in <i>NumberOfNeedles</i> .
<i>Sealing</i> ?	boolean	A value of "true" specifies thermo-sealing.
<i>SewingPattern</i> ?	enumeration	Sewing pattern. Possible values are: <i>Normal</i> <i>Staggered</i> <i>CombinedStaggered</i> <i>Side-Side</i> sewing.
<i>ThreadThickness</i> ?	double	Thread thickness.
<i>ThreadBrand</i> ?	string	Thread brand.
GlueLine *	element	Gluing parameters.

7.2.178 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 3.9.5, 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,” on page 87.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>TileID</i>
Input of processes:	Tiling
Output of processes:	—

Table 7-345: Tile resource

Name	Data Type	Description
<i>ClipBox</i>	rectangle	A rectangle that defines the bounding box of the surface contents which will be imaged on this Tile . The <i>ClipBox</i> is defined in the coordinate system of the surface.
<i>CTM</i>	matrix	A coordinate transformation matrix mapping the <i>ClipBox</i> 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.
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 Media in JDF 1.2

7.2.179 Tool

[New in JDF 1.1](#)

A **Tool** resource defines a generic tool that is customized for needed 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:	—
Example Partition:	—
Input of processes:	Embossing, ShapeCutting
Output of processes:	—

Table 7-346: Tool resource (Section 1 of 2)

Name	Data Type	Description
<i>ToolAmount</i> ? 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). Use ShapeCuttingParams/Shape elements to specify multiple cut forms in a cutting die in JDF 1.3 and above.
<i>ToolID</i> 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 7-346: Tool resource (Section 2 of 2)

Name	Data Type	Description
<i>ToolType ?</i> Modified in JDF 1.3	NMTOKEN	Type of the tool. Possible values include: <i>CentralStripper</i> – The center tool of the stripper toolset. Stripping means removing small parts of waste in between blanks. <i>CounterDie</i> – The lower tool of the die-cut pair with the counter (female) parts for the creases <i>CutDie</i> – The upper tool of the die-cut pair with the actual cutting and creasing knives. <i>EmbossingCalendar</i> <i>EmbossingStamp</i> <i>FrontWasteSeparator</i> – The tool to remove gripper margin from the sheet <i>LowerBlanker</i> – The lower tool of the blanker pair (blanking means separating blanks) <i>LowerStripper</i> – The lower tool of the stripper toolset <i>ToolSet</i> – The value "ToolSet" is used when the ToolID does not refer to single tools but to a collection of matching tools that have to be used in the process. E.g., when <i>ProductID</i> is a single stock item number in the MIS for a set consisting of a <i>CutDie</i> and a <i>CounterDie</i> . <i>UpperBlanker</i> – The upper tool of the blanker pair <i>UpperStripper</i> – The upper tool of the stripper toolset

7.2.180 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: http://www.cip4.org/documents/technical_info/cip3v3_0.pdf.

Resource Properties

Resource class:	Parameter
Resource referenced by:	Color, TransferCurvePool
Example Partition:	<i>RibbonName, SheetName, Side, WebName</i>
Input of processes:	—
Output of processes:	—

Table 7-347: TransferCurve resource

Name	Data Type	Description
<i>Curve</i>	TransferFunction	The density mapping curve for the separation defined by <i>Separation</i> .
<i>Separation ?</i>	string	The name of the separation. If <i>Separation</i> = "All", this curve is to be applied to all separations that are not explicitly defined.

7.2.181 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 element 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** element, the TransferCurveSet elements contain device dependent geometrical information, (e.g., *CTM* definitions).

Resource Properties

Resource class: Parameter
Resource referenced by: **TransferFunctionControl**
Example Partition: —
Input of processes: *ContoneCalibration, DigitalPrinting, ImageSetting, InkZoneCalculation, PreviewGeneration, Stripping*
Output of processes: *LayoutPreparation*

Table 7-348: TransferCurvePool resource

Name	Data Type	Description
TransferCurveSet *	element	The set of transfer curves.

— Element: TransferCurveSet

TransferCurveSet elements describe both the characteristic curve of transfer and the relation between the various process coordinate systems.

Table 7-349: TransferCurveSet element

Name	Data Type	Description
<i>CTM</i> ? New in JDF 1.1	matrix	Defines the transformation of the coordinate system in the device as defined by <i>Name</i> .
<i>Name</i> Modified in JDF 1.2	NMTOKEN	The name of the TransferCurveSet. Possible values include: <i>Film</i> — The transformation from the Layout system to the <i>Film</i> . In a CTP or DigitalPrinting environment, this defaults to the identity matrix and the identity TransferCurve. <i>Plate</i> — The transformation from the <i>Film</i> system to the <i>Plate</i> . In a DigitalPrinting environment, this defaults to the identity matrix and the identity TransferCurve . <i>Press</i> — The transformation from the <i>Plate</i> system to the <i>Press</i> . <i>Paper</i> — The transformation from the <i>Press</i> system to the <i>Paper</i> . <i>Proof</i> — The transformation from the Layout system to the <i>Proof</i> . New in JDF 1.2
TransferCurve * Modified in JDF 1.1	refelement	List of TransferCurve entries.

7.2.182 TransferFunctionControl

Resource Properties

Resource class: Parameter
Resource referenced by: **SeparationControlParams**
Example Partition: —
Input of processes: *ContoneCalibration*
Output of processes: —

Table 7-350: TransferFunctionControl resource

Name	Data Type	Description
<i>TransferFunctionSource</i> Modified in JDF 1.3	enumerations	Identifies the source of transfer curves which are to be applied during separation. <i>Custom</i> – Use the transfer curves provided in TransferCurvePool. <i>Device</i> – 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. <i>Document</i> – Use the transfer curves provided in the document. In JDF 1.3, the data type was modified from enumeration to enumerations. If multiple values are specified, the transfer functions that are specified by the individual enumeration values are concatenated.
TransferCurvePool ?	refelement	Provides a set of transfer curves to be used by the process.

7.2.183 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

Resource referenced by: —

Example Partition: *DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName*

Input of processes: **RIPing, Trapping**

Output of processes: —

Table 7-351: TrappingDetails resource (Section 1 of 2)

Name	Data Type	Description
<i>DefaultTrapping</i> = "false"	boolean	If "true", pages that have no defined TrapRegion elements are trapped using the set of TrappingParams. The bleed box is used for the trap zone. If "false", only pages that have TrapRegion elements are trapped.
<i>IgnoreFileParams</i> = "true"	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.
<i>Trapping</i> ? Deprecated in JDF 1.2	boolean	If "true", trapping is enabled. If "false", trapping is disabled. Use <i>NoOp</i> in JDF 1.2 and above.
<i>TrappingOrder</i> ?	element	Trapping processes will trap colorants as if they are laid down on the media in the order specified in <i>TrappingOrder</i> . The colorant order can affect which colors to spread, especially when opaque inks are used.
<i>TrappingType</i> ? 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.
TrappingParams ?	refelement	A TrappingParams resource that is used to define the default trapping parameters when <i>DefaultTrapping</i> = "true".

Table 7-351: TrappingDetails resource (Section 2 of 2)

Name	Data Type	Description
ObjectResolution * New in JDF 1.1	refelement	Elements which define the resolutions to trap the contents at. More than one element MAY be used to specify different resolutions for different <i>SourceObjects</i> types.
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.

— Element: TrappingOrder

Table 7-352: TrappingOrder element

Name	Data Type	Description
SeparationSpec * Modified in JDF 1.2	element	An array of colorant names.

7.2.184 TrappingParams

This resource provides a set of controls that are used to generate traps. 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: —

Output of processes: —

Table 7-353: TrappingParams resource (Section 1 of 4)

Name	Data Type	Description
<i>BlackColorLimit</i> ?	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.
<i>BlackDensityLimit</i> ?	double	A positive number that specifies the lowest neutral density of a colorant for trapping according to the black trapping rule.
<i>BlackWidth</i> ?	double	A positive number that specifies the trap width for trapping according to the black trapping rule. The <i>BlackWidth</i> is specified in <i>TrapWidth</i> units; a value of "1" means that the black trap width is one <i>TrapWidth</i> wide. The resulting black trap width is subject to the same device limits as <i>TrapWidth</i> .
<i>Enabled</i> ? Deprecated in JDF 1.2	boolean	If "true", trapping is enabled for zones that are defined with this parameter set. Use <i>NoOp</i> in JDF 1.2 and above.

Table 7-353: TrappingParams resource (Section 2 of 4)

Name	Data Type	Description
<i>HalftoneName ?</i>	string	A name that identifies a halftone object to be used when marking traps. The name is the value of the <i>ResourceName</i> attribute of some PDLResourceAlias resource. If absent, the halftone in effect just before traps are marked will be used, which MAY cause unexpected results.
<i>ImageInternalTrapping ?</i>	boolean	If <i>"true"</i> , the planes of color images are trapped against each other. If <i>"false"</i> , the planes of color images are not trapped against each other.
<i>ImageResolution ?</i>	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.
<i>ImageMaskTrapping ?</i>	boolean	Controls trapping when the <i>TrapZone</i> 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 <i>"1"</i> are marked, samples with a value of <i>"0"</i> are not marked. When <i>"false"</i> , 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 <i>"true"</i> , objects are trapped to the stencil mask, and to each other.
<i>ImageToImageTrapping ?</i>	boolean	If <i>"true"</i> , traps are generated along a boundary between images. If <i>"false"</i> , this kind of trapping is not implemented.
<i>ImageToObjectTrapping ?</i>	boolean	If <i>"true"</i> , images are trapped to other objects. If <i>"false"</i> , this kind of trapping is not implemented.
<i>ImageTrapPlacement ?</i>	enumeration	Controls the placement of traps for images. Possible values are: <i>Center</i> – Trap is centered on the edge between the image and the adjacent object. <i>Choke</i> – Trap is placed in the image. <i>Normal</i> – Trap is based on the colors of the areas. <i>Spread</i> – Trap is placed in the adjacent object.

Table 7-353: TrappingParams resource (Section 3 of 4)

Name	Data Type	Description
<i>ImageTrapWidth</i> ? 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 <i>ImageToImageTrapping</i> , <i>ImageToObjectTrapping</i> and/or <i>ImageInternalTrapping</i> are set to <i>true</i> . 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 <i>TrapWidth</i> , <i>BlackWidth</i> and <i>MinimumBlackWidth</i> . Values MUST be greater than or equal to zero. A value of 0.0 disables non-black image trapping. Defaults to <i>TrapWidth</i> .
<i>ImageTrapWidthY</i> ? 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 <i>ImageToImageTrapping</i> , <i>ImageToObjectTrapping</i> and/or <i>ImageInternalTrapping</i> are set to <i>true</i> . 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 <i>TrapWidth</i> , <i>BlackWidth</i> and <i>MinimumBlackWidth</i> . Values MUST be greater than or equal to zero. A value of 0.0 disables non-black image trapping. Defaults to <i>ImageTrapWidth</i> .
<i>MinimumBlackWidth</i> = "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.
<i>SlidingTrapLimit</i> ?	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 <i>SlidingTrapLimit</i> , then the trap slides. This applies to vignettes and non-vignettes. No slide occurs at "1".
<i>StepLimit</i> ? 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 <i>StepLimit</i> times the lower value ($\text{low} + \max(\text{StepLimit} * \text{low}, 0.05)$), 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 <i>ColorantZoneDetails</i> subelement for a colorant. The restriction that <i>StepLimit</i> be less than or equal to one (≤ 1) was removed in JDF 1.2.
<i>TrapColorScaling</i> ?	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 <i>ColorantZoneDetails</i> subelement for a colorant.

Table 7-353: TrappingParams resource (Section 4 of 4)

Name	Data Type	Description
<i>TrapEndStyle</i> = "Miter"	NMTOKEN	Instructs the trap engine how to form the end of a trap that touches another object. Possible values include: <i>Miter</i> <i>Overlap</i> Other values might be added later as a result of customer requests.
<i>TrapJoinStyle</i> = "Miter"	NMTOKEN	Specifies the style of the connection between the ends of two traps created by consecutive segments along a path. Possible values include: <i>Bevel</i> <i>Miter</i> <i>Round</i>
<i>TrapWidth</i> ? 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 <i>BlackWidth</i> .
<i>TrapWidthY</i> ? 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 <i>BlackWidth</i> . If not specified, defaults to the value of <i>TrapWidth</i> .
<i>ColorantZoneDetails</i> *	element	<i>ColorantZoneDetails</i> 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 <i>ColorantParams</i> array nor implied by the <i>ProcessColorModel</i> for the <i>ColorantControl</i> object in effect when these <i>TrappingParams</i> are applied, the entry is not used for trapping.

— Element: ColorantZoneDetails

Table 7-354: ColorantZoneDetails element

Name	Data Type	Description
<i>Colorant</i>	string	The colorant name that occurs in the <i>SeparationSpec/@Name</i> of the <i>ColorantParams</i> array of the <i>ColorantControl</i> object used by the process.
<i>StepLimit</i> ?	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 <i>StepLimit</i> 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 <i>StepLimit</i> attribute in the TrappingParams resource is used.
<i>TrapColorScaling</i> ?	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 <i>TrapColorScaling</i> attribute in the TrappingParams resource is used.

7.2.185 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
Example Partition:	—
Input of processes:	—
Output of processes:	—

Table 7-355: TrapRegion resource

Name	Data Type	Description
<i>TrapZone</i> ?	PDFPath	Each element within <i>TrapZone</i> is one subpath of a complex path. The <i>TrapZone</i> 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 <i>TrapZone</i> .
<i>Pages</i>	Integer-RangeList	Identifies a set of pages from the RunList to trap using the specified geometry and trapping style.
TrappingParams ?	refelement	The set of trapping parameters which will be used when trapping in this region.

7.2.186 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).

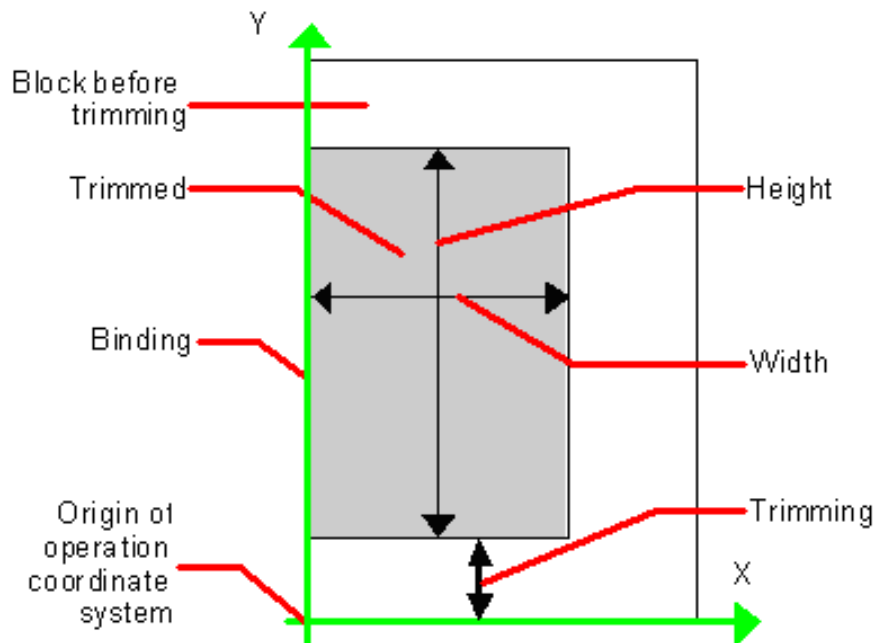


Figure 7-49: Parameters and coordinate system used for trimming

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>Trimming</i>
Output of processes:	—

Table 7-356: TrimmingParams resource

Name	Data Type	Description
<i>Height ?</i>	double	Height of the trimmed product.
<i>TrimCover</i> = " <i>Both</i> " New in JDF 1.3	enumeration	Specifies the covers to be trimmed. Covers containing flaps are generally not trimmed. <i>Back</i> – Trim back cover only <i>Both</i> – Trim front and back cover <i>Front</i> – Trim front cover only <i>Neither</i> – Do not trim cover.
<i>TrimmingOffset ?</i>	double	Amount to be cut at bottom side.
<i>TrimmingType ?</i> New in JDF 1.1 Deprecated in JDF 1.2	enumeration	Trimming operation to perform. Possible values are: <i>Detailed</i> – Cut the amount specified by <i>Height</i> , <i>Width</i> and <i>TrimmingOffset</i> . <i>SystemSpecified</i> – Cut the amount specified by the system.
<i>Width ?</i>	double	Width of the trimmed product.

7.2.187 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 resources, so that standard counting can be used. See "Resource Amount" on page 76. The section has details on tracking *Amount* and *ActualAmount*, Default units are "countable objects". See "Units" on page 12.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	Any
Output of processes:	—

Table 7-357: UsageCounter resource (Section 1 of 2)

Name	Data Type	Description
<i>CounterID</i>	string	The ID of this counter as defined by the counting Device.

Table 7-357: UsageCounter resource (Section 2 of 2)

Name	Data Type	Description
<i>CounterTypes ?</i>	NMTOKENS	<p>This attribute indicates the types of usage being counted by the UsageCounter. Values include:</p> <p>Media Sides values:</p> <p><i>Blank</i> – includes entirely blank counts</p> <p><i>Insert</i> – post fuser inserter</p> <p><i>OneSided</i> – includes one sided counts</p> <p><i>TwoSided</i> – includes two sided counts</p> <p>Media Size values:</p> <p><i>NormalSize</i> – includes normal size counts</p> <p><i>LargeSize</i> – includes large size counts</p> <p>Colorant values:</p> <p><i>Black</i> – includes black colorant only counts</p> <p><i>HighlightColor</i> – includes highlight colorant counts</p> <p><i>Color</i> – includes one or more non-black, non-highlight color colorants counts</p> <p>Usage values:</p> <p><i>User</i> – 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.</p> <p><i>Auxiliary</i> – includes all counts for work not requested by the user, e.g. banner, confirmation, slip, separator, error log.</p> <p><i>Waste</i> – includes waste.</p>
<i>Scope</i>	enumeration	<p>The scope of this usage counter. Values are:</p> <p><i>Lifetime</i> – count since machine last had a firmware reset. MUST NOT be specified when UsageCounter is used as a Resource in a JDF ticket.</p> <p><i>PowerOn</i> – count since the machine was powered on. MUST NOT be specified when UsageCounter is used as a Resource in a JDF ticket.</p> <p><i>Job</i> – count in the context of one JDF.</p>

7.2.188 VerificationParams

This resource provides the parameters of a **Verification** process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	Verification
Output of processes:	—

Table 7-358: VerificationParams resource

Name	Data Type	Description
<i>FieldRange</i> ?	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.
<i>InsertError</i> ?	string	Database insertion statement in <i>C printf</i> format defining how information read from the resource of the Verification process is to be stored in case of verification errors. The database is defined by the DBSelection resource of the Verification process. This field MUST be specified if a database is selected.
<i>InsertOK</i> ?	string	Database insertion statement in <i>C printf</i> format defining how information extracted from the IdentificationField is to be stored in case of verification success. The database is defined by the DBSelection resource of the verification node. This field MUST be specified if a database is selected.
<i>Tolerance</i> ?	double	Ratio of tolerated verification failures to the total number of tests. "0.0" = no failures allowed, "1.0" = all might fail.

Usage of FieldRange and Format Strings.

A database field name can be calculated from the characters of the **IdentificationField** using standard *C printf* notation and the *FieldRange* attribute. Each range that is defined in *FieldRange* is passed to *printf* as one string that is applied to the format. The order is maintained. Note that SQL was chosen for illustrative purposes only. The mechanism is defined for any database interface.

Example

IdentificationField string: 1234:John Doe

FieldRange: 5 ~ -1 0 ~ 3

InsertOK: Insert "true" into Va where Name = "%s" and ID = %s

Resulting string: Insert "true" into Va where Name = "John Doe" and ID = 1234

7.2.189 WebInlineFinishingParams

[New in JDF 1.3](#)

WebInlineFinishingParams specifies the parameters for web inline finishing equipment using the **WebInlineFinishing** process.

Resource Properties

Resource class: Parameter

Resource referenced by: —

Example Partition: *ProductionRun, SubRun, WebName, RibbonName, WebProduct*

Input of processes: **WebInlineFinishing**

Output of processes: —

Table 7-359: WebInlineFinishingParams resource

Name	Data Type	Description
<i>FolderProduction</i> *	element	Specifies the Folder setup for newspaper presses:

— Element: FolderProduction

Table 7-360: FolderProduction element

Name	Data Type	Description
<i>FolderModuleIndex</i>	integer	Identifies a particular folder module to be used. <i>FolderModuleIndex</i> MUST match Device/Module/@ModuleIndex .
<i>ProductionType</i> ="NonCollect"	enumeration	Indicates whether the product is collected or not. <i>Collect</i> <i>NonCollect</i>

7.2.190 WireCombBindingParams

This resource describes the details of the **WireCombBinding** process.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	WireCombBinding
Output of processes:	—

Table 7-361: WireCombBindingParams resource

Name	Data Type	Description
<i>Brand</i> ?	string	The name of the comb manufacturer (e.g., <i>Wire-O</i> ®) and the name of the specific item.
<i>Color</i> ?	NamedColor	Determines the color of the comb.
<i>Diameter</i> ?	double	The comb diameter is determined by the height of the block of sheets to be bound.
<i>Distance</i> ? Deprecated in JDF 1.2	double	The distance between the “teeth” and the distance between the holes of the prepunched sheets MUST be the same. In JDF 1.2 and beyond, use the value implied by HoleMakingParams/@HoleType .
<i>FlipBackCover</i> = "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.
<i>Material</i> ?	enumeration	The material used for forming the wire comb binding. Possible values are: <i>LaqueredSteel</i> <i>TinnedSteel</i> <i>ZincsSteel</i>
<i>Shape</i> = "Single"	enumeration	The shape of the wire comb binding. Possible values are: <i>Single</i> – Each “tooth” is made with one wire. <i>Twin</i> – The shape of each “tooth” is made with a double wire.
<i>Thickness</i> ?	double	The thickness of the comb material.
HoleMakingParams ? New in JDF 1.2	refelement	Details of the holes in WireCombBinding .

7.2.191 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
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>Wrapping</i>
Output of processes:	—

Table 7-362: WrappingParams resource

Name	Data Type	Description
<i>WrappingKind</i>	enumeration	<i>LooseWrap</i> – The wrap is loose around the component. <i>ShrinkWrap</i> – The wrap is shrunk around the component.

7.3 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 using capabilities, See “FileSpec” on page 430. For more details on preflight, See “Preflight” on page 235.

Capabilities descriptions that are saved in files MUST be formatted as a JMF/Signal/Response to the *KnownDevices* query.

7.3.1 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.



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.

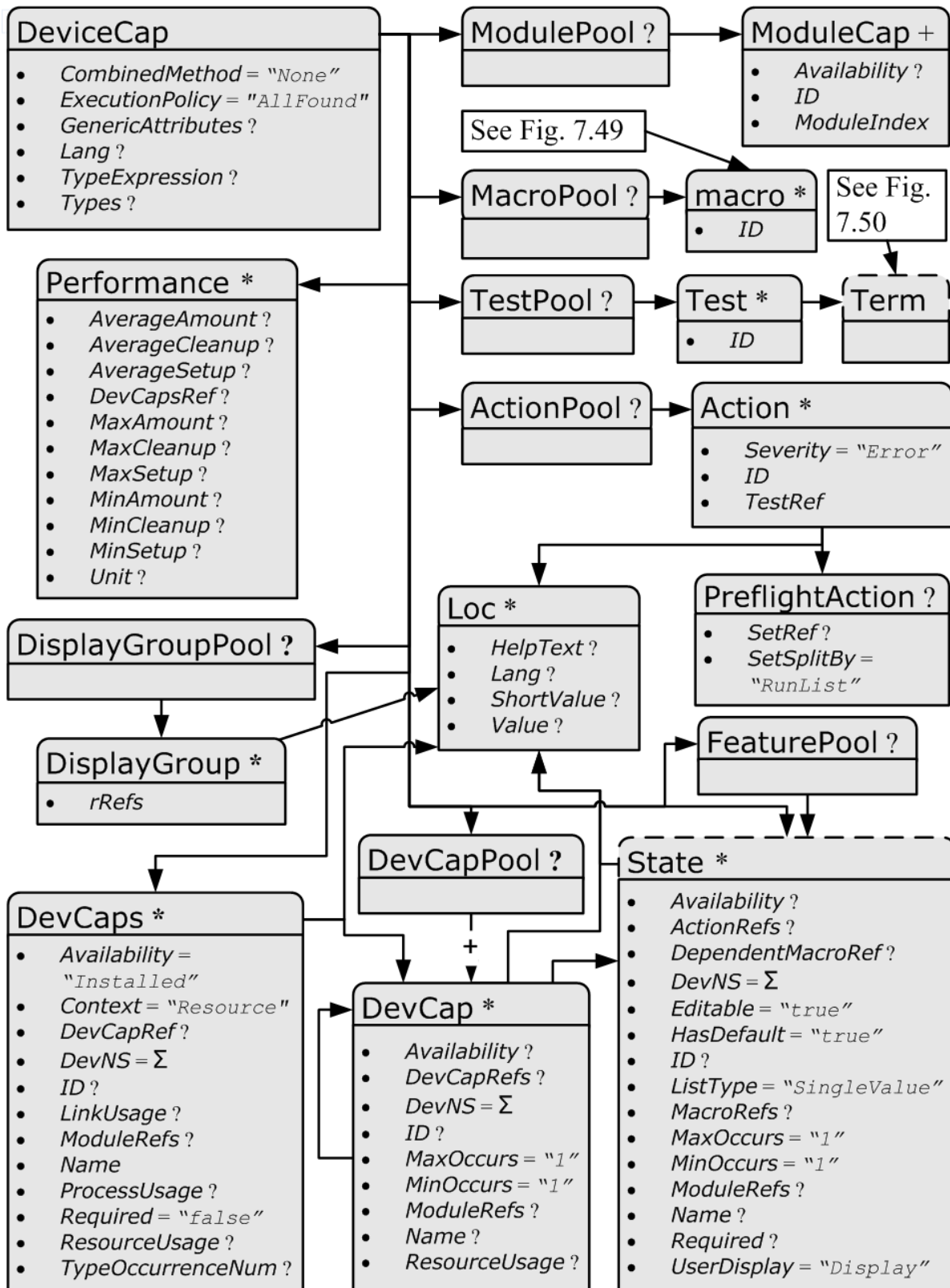


Figure 7-50: DeviceCap – a diagram of its structure

Table 7-363: DeviceCap element (Section 1 of 3)

Name	Data Type	Description
<i>CombinedMethod</i> = "None" Modified in JDF 1.3	enumerations	Specifies how the processes specified in <i>Types</i> are to be specified. Values are: <i>Combined</i> – The list of processes in <i>Types</i> MUST be specified as a <i>Combined</i> process. <i>CombinedProcessGroup</i> – The list of processes in <i>Types</i> MUST be specified either as a <i>Combined</i> process or as a <i>ProcessGroup</i> of individual processes. In JDF 1.3 and beyond, the pair of individual tokens: " <i>Combined ProcessGroup</i> " replace this single value. Deprecated in JDF 1.3 <i>GrayBox</i> – The list of processes in <i>Types</i> MUST be specified in a <i>ProcessGroup</i> with no nested JDF nodes, i.e., a gray box. New in JDF 1.3 <i>ProcessGroup</i> – The list of processes in <i>Types</i> MUST be specified as a <i>ProcessGroup</i> of individual processes. <i>None</i> – No support for <i>Combined</i> , <i>GrayBox</i> or <i>ProcessGroup</i> . Only one individual process type defined in <i>Types</i> is supported.
<i>ExecutionPolicy</i> = "AllFound" New in JDF 1.2	enumeration	Describes the policy for finding and executing JDF nodes as described in "Determining Executable Nodes" on page 123. <i>RootNode</i> – 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. <i>FirstFound</i> – 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. <i>AllFound</i> – 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.
<i>GenericAttributes</i> ?	NMTOKENS	List of all generic attributes that are supported and unrestricted by the device implementation. Descriptions of attributes that appear in <i>State</i> elements (see the following Section 7.3.7, State) overwrite the description in <i>GenericAttributes</i> .
<i>Lang</i> ? New in JDF 1.2	languages	Specifies the localization(s) provided with the capabilities. If not specified, no localizations are provided.
<i>OptionalCombinedTypes</i> ? Deprecated in JDF 1.2	NMTOKENS	List of optional JDF Node types. The entries of the list MUST be a subset of <i>Types</i> . For example, a RIP with optional in-RIP trapping would specify <i>OptionalCombinedTypes</i> = " <i>Trapping</i> " if <i>Types</i> = " <i>Trapping Interpreting Rendering</i> ". Replaced by <i>TypeExpression</i> in JDF 1.2 and beyond
<i>Type</i> ? Deprecated in JDF 1.2	NMTOKEN	JDF <i>Type</i> attribute of the supported process. Extension types MAY be specified by stating the namespace prefix in the value. In JDF 1.2 and beyond, a single value of type is also defined in the <i>Types</i> attribute.

Table 7-363: DeviceCap element (Section 2 of 3)

Name	Data Type	Description
TypeExpression ? New in JDF 1.2	regExp	Regular expression that defines the allowed values of the node's <i>Types</i> attribute. If not specified, defaults to the literal string defined in <i>Types</i> , (i.e., the ordered list of processes defined in <i>Types</i> MUST match exactly). In JDF 1.2 and above, one of <i>Types</i> or <i>TypeExpression</i> MUST be specified.
TypeOrder ? Deprecated in JDF 1.2	enumeration	Ordering restriction for <i>Combined</i> or <i>ProcessGroup</i> nodes. <i>Fixed</i> — The order of process types specified in the <i>Types</i> attribute is ordered, and each type can be specified only once, (e.g., <i>Cutting</i> , <i>Folding</i>). Order does matter. <i>Unordered</i> — The order of process types specified in the <i>Types</i> attribute is unordered, and each type can be specified only once, (e.g., <i>DigitalPrinting</i> , <i>Screening</i> , <i>Trapping</i>). Order does not matter. <i>Unrestricted</i> — The order of process types specified in the <i>Types</i> attribute is unordered, and each type can be specified in multiples, (e.g., <i>Cutting</i> , <i>Folding</i>). The device can do both processes, in any order multiple times. Replaced by <i>TypeExpression</i> in JDF 1.2 and beyond.
Types ? Modified in JDF 1.2	NMTOKENS	This attribute represents the list of supported JDF node <i>Type</i> values. If any of the node types are in a namespace other than JDF, the namespace prefix MUST be included in this node type name. The ordering is significant unless it is overridden by <i>@TypeExpression</i> . In JDF 1.2 and above, one of <i>Types</i> or <i>TypeExpression</i> MUST be specified.
ActionPool ? New in JDF 1.2	element	Container for zero or more 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 MUST 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., <i>CurrentValue</i> or <i>UserDisplay</i>) 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.

Table 7-363: DeviceCap element (Section 3 of 3)

Name	Data Type	Description
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 elements that define the parameter space that is covered by the device. One State element MUST be defined for each supported attribute of the JDF node that is not specified <i>GenericAttributes</i> or implied by <i>TypeExpression</i> or <i>Types</i> .

7.3.2 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 "Term" on page 643. 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 "TestPool" on page 642.

Table 7-364: ActionPool element

Name	Data Type	Description
Action *	element	A list of independent Action elements.

— Element: 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 **MUST** be taken.

Table 7-365: Action element

Name	Data Type	Description
<i>Severity</i> = "Error"	enumeration	Indicates how the severity of the failure is to be treated when the expression defined by <i>TestRef</i> is violated. One of: <i>Error</i> – The client is to display an error message and not allow the conflicting settings to persist. <i>Warning</i> – The client is to notify the user of the condition but allow the settings to persist if the user requests. <i>Information</i> – The client is to allow the settings to persist but inform the user of the issue.
<i>ID</i>	ID	Unique identifier of the Action element. This ID is used to refer to the Action element, (e.g., from a preflight report).
<i>TestRef</i>	IDREF	Reference to a Test element that is executed to evaluate this Action .
<i>Loc *</i>	element	Text to describe an error if the Test fails. (See "Loc" on page 621.)
PreflightAction ?	element	Provides additional constraints that are specific to the Preflight process. See "PreflightParams" on page 540.

7.3.3 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 7-366: DevCapPool element

Name	Data Type	Description
DevCap +	element	DevCap elements that can be referenced from multiple locations within the DeviceCap structure. DevCap/@ID MUST be specified for all DevCap elements in DevCapPool.

7.3.4 ModulePool

[New in JDF 1.3](#)

Table 7-367: 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.

— Element: 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 7-368: ModuleCap element

Name	Data Type	Description
<i>Availability ?</i>	enumeration	Specifies whether the feature described by this State element is available on the device. Possible values are: <i>Installed</i> – The feature is installed on the device and is available for use. <i>Module – Module</i> is not to be specified recursively in a ModuleCap . This value is only specified here to have a common enumeration set for all <i>Availability</i> attributes. <i>NotInstalled</i> – The feature has not been installed on the device. <i>NotLicensed</i> – The feature has been installed on the device but can not be used until licensed. <i>Disabled</i> – The feature is installed and licensed on the device, but has been disabled.
<i>ID</i>	ID	<i>ID</i> of the ModuleCap .
<i>ModuleID ?</i>	integer	ID of the module that this <i>ModuleCap</i> describes. Refers to Device/Module/ModuleID . If neither <i>ModuleID</i> nor <i>ModuleIndex</i> are specified, no further details of the Module are known.
<i>ModuleIndex ?</i>	integer	Index of the module that this <i>ModuleCap</i> describes. Refers to Device/Module/ModuleIndex . If neither <i>ModuleID</i> nor <i>ModuleIndex</i> are specified, no further details of the Module are known.

7.3.5 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 product intent resources as well as process definition resources.

Table 7-369: DevCaps element (Section 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. Possible values are: <i>Installed</i> – The feature is installed on the device and is available for use. <i>Module</i> – 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 <i>NotInstalled</i> – The feature has not been installed on the device. <i>NotLicensed</i> – The feature has been installed on the device but can not be used until licensed. <i>Disabled</i> – 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). One of: <i>Element</i> – The DevCaps context is describing a direct element, e.g., an AuditPool . <i>JMF</i> – The DevCaps context describes a JMF message. <i>Link</i> – The DevCaps context is describing a link to a resource. <i>Resource</i> – The DevCaps context is describing a resource.
DevCapRef ? New in JDF 1.3	IDREF	Reference to a reusable DevCap Element that is 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 MUST 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. One of: <i>Input</i> – The DevCaps describes an input resource. <i>Output</i> – The DevCaps describes an output resource. If not specified, this DevCaps applies to both usages.
ModuleRefs ? New in JDF 1.3	IDREFS	List of modules that are needed for this feature to be available. At least one entry MUST be specified if Availability = "Module". The list of Modules is specified in DeviceCap/ModulePool .

Table 7-369: DevCaps element (Section 2 of 3)

Name	Data Type	Description
Name Modified in JDF 1.3	NMTOKEN	Name of the element excluding the namespace prefix. If <i>Name</i> = "NodeInfo", it describes the structure of the NodeInfo information that is accepted by the device. When describing elements of a ResourceLink, <i>Name</i> MUST be the name of the referenced resource and <i>Context</i> = "Link". When DevCaps is specified as a subelement of MessageService, Name specifies the respective CommandTypeObj, QueryTypeObj or ResponseTypeObj of the JMF Message. Prior to JDF 1.3 The <i>ResourceUsage</i> and <i>ProcessUsage</i> of a resource are specified in this attribute. In JDF 1.3 and beyond, <i>Name</i> MUST always specify the actual Resource name.
ProcessUsage ? New in JDF 1.3	NMTOKEN	ResourceLink/@ProcessUsage of the link to the resource that is described by this DevCaps.
Required = "false" New in JDF 1.2	boolean	If "true", the element described by this DevCaps element MUST 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), <i>Required</i> MUST be "true".
ResourceUpdate ? Deprecated in JDF 1.3	NMTOKENS	Specifies the capability to handle partial updates defined in ResourceUpdate elements. Possible values include: <i>None</i> – ResourceUpdate is not supported. MUST NOT be combined with any other value. <i>JMFID</i> – JMF Resource messages that reference ResourceUpdate elements that have been previously loaded to the device are accepted. <i>PDLID</i> – 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.
TypeOccurrenceNum ? New in JDF 1.2	IntegerRange-List	Specifies which occurrence(s) of the name of this DevCaps element that is specified either within the DeviceCap Types attribute or by the TypeExpression attribute that this DevCaps element applies to. If not specified, elements belonging to all JDF nodes 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 MUST be a subset of Types in DeviceCap. Replaced by TypeOccurrenceNum in JDF 1.2 and beyond.

Table 7-369: DevCaps element (Section 3 of 3)

Name	Data Type	Description
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 sub-elements). Exactly one of <i>DevCapRef</i> or DevCap MUST 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 “Loc” on page 621.)

— Element: 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 7-370: Loc element

Name	Data Type	Description
<i>HelpText</i> ?	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.
<i>Lang</i> ?	language	The language code for this localization. If not specified, then it defaults to the value of the first language specified in the <i>Lang</i> attribute of the DeviceCap element. Note that each language in a list of localizations (i.e., LOC *) MUST be unique.
<i>ShortValue</i> ?	string	The short form of the localization. Defaults to the value of <i>Value</i> . This value would be used when a small fixed field is REQUIRED for the name of the field (a PDA for example).
<i>Value</i> ?	string	The localization of the value being localized. If not specified, then the value being localized is used as the <i>Value</i> , (e.g., the resource, ResourceLink, element, message, attribute name or attribute value).

7.3.6 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 7.3.7, State. The *Name* attribute of the State element MUST 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 7-371: DevCap element (Section 1 of 2)

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. Possible values are: <i>Installed</i> – The feature is installed on the device and is available for use. <i>Module</i> – The feature is provided by a module specified in <i>ModuleRefs</i> . If and only if all modules that are listed in <i>ModuleRefs</i> are available, the feature is available. New in JDF 1.3 <i>NotInstalled</i> – The feature has not been installed on the device. <i>NotLicensed</i> – The feature has been installed on the device but can not be used until licensed. <i>Disabled</i> – The feature is installed and licensed on the device but has been disabled. If not specified, the value specified in the parent DevCaps or DevCap element is applied.
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 <i>DevCapRefs</i> and DevCap elements exist, they specify the union of both.
<i>DevNS</i> = "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	<i>ID</i> of this DevCap. Used to reference a DevCap. DevCap/@ <i>ID</i> MUST be specified for all direct DevCap child elements in DevCapPool.
<i>MaxOccurs</i> = "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".
<i>MinOccurs</i> = "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 MUST be specified if <i>Availability</i> = "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/@ <i>Name</i> . <i>Name</i> MUST be specified for all direct DevCap child elements in DevCapPool or DevCap elements. Prior to JDF 1.3 <i>ResourceUsage</i> of a resource was specified in this attribute. In JDF 1.3 and beyond, <i>Name</i> MUST always specify the actual Resource name.
ResourceUsage ? New in JDF 1.3	NMTOKEN	Resource/@ <i>ResourceUsage</i> of the resource that is described by this DevCap.
DevCap *	element	Definition of the accepted parameter space for the messages or resources subelements. If Multiple DevCap elements with the same @ <i>Name</i> exist, they describe individual subelements with different properties. The properties MUST each be fulfilled by individual subelements of the element that is described by this DevCap. For instance, if two DevCap elements with the <i>MinOccurs</i> = "1" are specified, the JDF element MUST contain two elements with a node name = DevCap/@ <i>Name</i> .
Loc * New in JDF 1.2	element	The localization(s) of the element name. (See "Loc" on page 621.)

Table 7-371: DevCap element (Section 2 of 2)

Name	Data Type	Description
State *	element	Abstract State elements that define the parameter space that is covered by device. One State element MUST be defined for each supported attribute or Intent Span element of the element that this DevCap defines that is not specified DeviceCap/@GenericAttributes .

7.3.7 State

[New in JDF 1.1](#)

The following table 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.

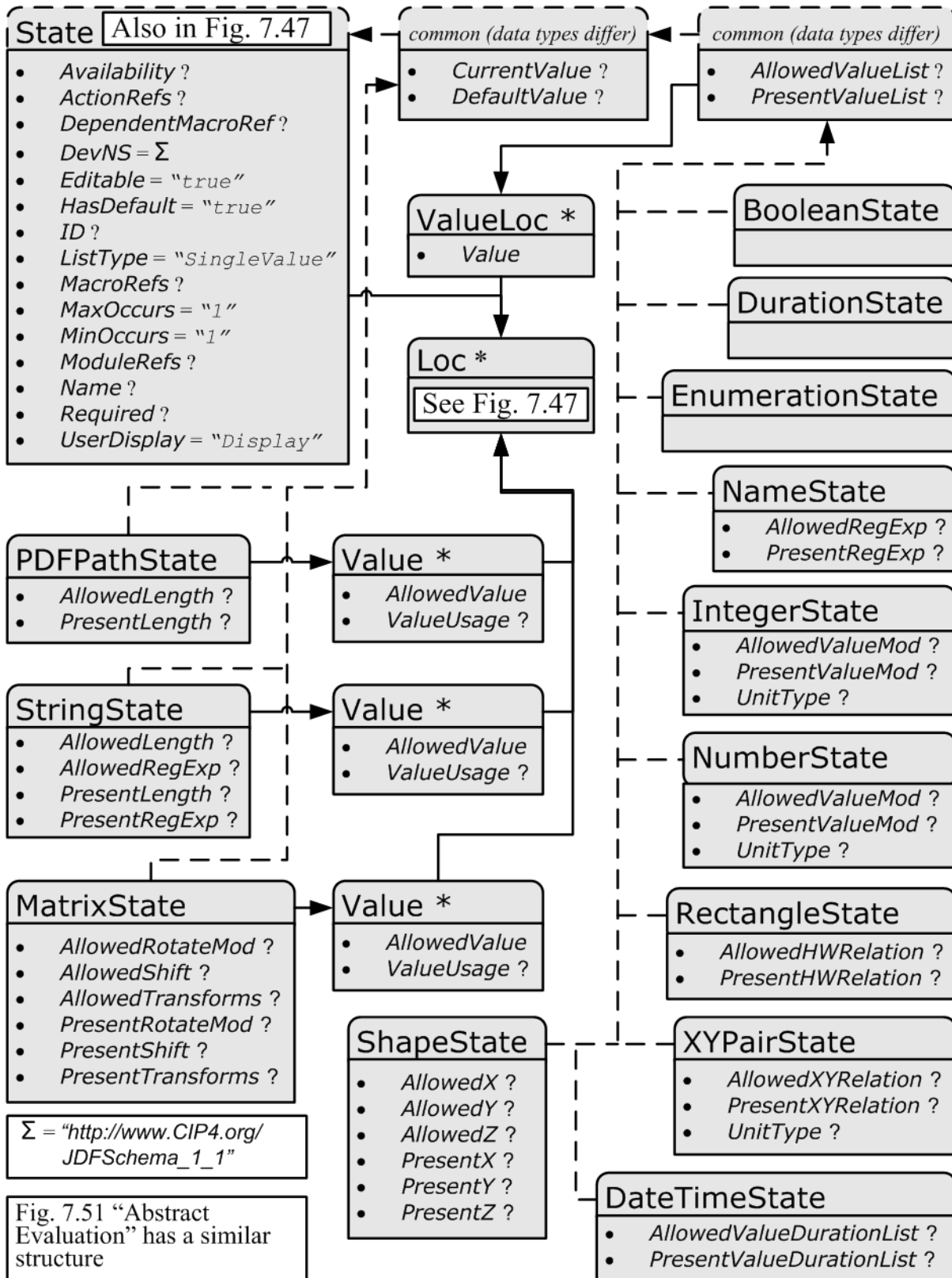


Figure 7-51: Abstract State element – a diagram of its structure

Table 7-372: Abstract State element (Section 1 of 2)

Name	Data Type	Description
Availability ? New in JDF 1.2	enumeration	Specifies whether the feature described by this State element is available on the device. Possible values are: <i>Installed</i> – The feature is installed on the device and is available for use. <i>Module</i> – The feature is provided by a module specified in <i>ModuleRefs</i> . If and only if all modules that are listed in <i>ModuleRefs</i> are available, the feature is available. New in JDF 1.3 <i>NotInstalled</i> – The feature has not been installed on the device. <i>NotLicensed</i> – The feature has been installed on the device but can not be used until licensed. <i>Disabled</i> – The feature is installed and licensed on the device, but has been disabled. If not specified, the value specified or implied by the parent element is applied.
ActionRefs ? New in JDF 1.2	IDREFS	Zero or more references to Action elements that operate on the parameter. All Action elements referenced MUST evaluate to <i>false</i> 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.
DependentMacroRef ? New in JDF 1.2	IDREF	A reference to a macro that conditionally modifies the <i>UserDisplay</i> attribute of this State element. If present, this referenced macro is to be executed when the <i>State/@UserDisplay</i> is <i>Dependent</i> and the user interface is being initialized. It is RECOMMENDED that the macro referenced by <i>DependentMacroRef</i> only change the value of <i>UserDisplay</i> or <i>Editable</i> attributes. For more information on macro definitions, see "MacroPool" on page 639.
<i>DevNS</i> = "http://www.CIP4.org/JDFSchema_1_1"	URI	Namespace of the attribute that is described by this State element.
Editable = "true" New in JDF 1.2	boolean	When <i>true</i> , the feature and its current value can be edited by the user. If <i>false</i> , the user interface MUST NOT allow user modification of the State 's current value.
<i>HasDefault</i> = "true"	boolean	A flag that describes whether the parameter has a default supplied by the device. If set, <i>DefaultValue</i> MUST be set.
ID ? New in JDF 1.2	ID	An identification value to allow external reference.
<i>ListType</i> = "SingleValue" New in JDF 1.2 Modified in JDF 1.3	enumeration	Specifies what type of list or object the State variable describes. See Table 7-373 for values of <i>ListType</i> .

Table 7-372: Abstract State element (Section 2 of 2)

Name	Data Type	Description
MacroRefs ? New in JDF 1.2	IDREFS	Zero or more references to macros that operate on the parameter. These macros set other State attribute values as appropriate. Any macros referenced in <i>MacroRefs</i> is to be evaluated whenever the attribute described by this State element is manipulated or changed to affect any necessary changes to other attributes. The Macros can change attributes such as the <i>CurrentValue</i> attribute of a State or its <i>UserDisplay</i> attribute. For more information on macro definitions, see "MacroPool" on page 639.
<i>MaxOccurs</i> = "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 <i>MaxOccurs</i> 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).
<i>MinOccurs</i> = "1" New in JDF 1.2	integer	Minimum number of elements in the list described by this State . If <i>MinOccurs</i> 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 MUST be specified if <i>Availability</i> = "Module". The list of Modules is specified in DeviceCap/ModulePool.
<i>Name</i> ?	NMTOKEN	Name of the attribute that is described by this State . If <i>Name</i> 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.
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 <i>DataType</i> = "XYPairState" and <i>Span</i> = "true". Replaced with <i>ListType</i> = "Span" in JDF 1.2 and beyond.
<i>UserDisplay</i> = "Display" New in JDF 1.2	enumeration	Indicates whether the feature is to be displayed in user interfaces. Possible values are: <i>Display</i> – The feature is to be displayed. <i>Hide</i> – The feature is not to be displayed. <i>Dependent</i> – The feature is to be conditionally displayed depending on the action specified by the macro referenced by <i>DependentMacroRef</i> . Note that this action is only taken when the user interface is first initialized.
<i>Loc</i> * New in JDF 1.2	element	The localization(s) of the <i>Name</i> of the attribute that is described by this State element. (See "Loc" on page 621.)

— Attribute: ListType

Table 7-373: ListType attribute – possible values

Value	Description
<i>CompleteList</i>	The State describes a list of individual values. Each value MUST occur exactly once.
<i>CompleteOrderedList</i>	The State describes an ordered list of individual values. Each value MUST occur only once and in the specified order.
<i>ContainedList</i>	The State describes a list of individual values. The State = " <i>true</i> " if at least one of the values occurs. This value is only expected to be used in BasicPreflightTest elements.
<i>List</i>	The State describes a list of individual values.
<i>OrderedList</i>	The State describes an ordered list of individual values.
<i>OrderedRangeList</i>	The State describes an ordered RangeList of individual values.
<i>Range</i>	The State describes an individual Range of values. New in JDF 1.3
<i>RangeList</i>	The State describes a RangeList of values.
<i>SingleValue</i>	The State describes an individual value.
<i>Span</i>	The State describes a Span element in a product intent resource.
<i>UniqueList</i>	The State describes a list of individual values. Each value can occur only once.
<i>UniqueRangeList</i>	The State describes a RangeList of values. Each explicit or implied value can occur only once.
<i>UniqueOrderedList</i>	The State describes an ordered list of individual values. Each value can occur only once.
<i>UniqueOrderedRangeList</i>	The State describes an ordered RangeList of individual values. Each explicit or implied value can occur only once.

The following types of **State** elements are defined:

Table 7-374: List of State elements (Section 1 of 2)

Name	Data Type	Description
BooleanState	element	Describes a set of boolean values.
DateTimeState New in JDF 1.2	element	Describes a set of dateTime values.
DurationState New in JDF 1.2	element	Describes a set of duration values.
EnumerationState	element	Describes a set of enumeration values.
IntegerState	element	Describes a numerical range of integer values.
MatrixState	element	Describes a range of matrices. Generally used to define valid orientations of Component resources.
NameState	element	Describes a set of NMTOKEN values.
NumberState	element	Describes a numerical range of values.
PDFPathState New in JDF 1.2	element	Describes a set of PDFPaths.
RectangleState New in JDF 1.2	element	Describes a set of 4 value rectangle values.
ShapeState	element	Describes a set of 3 value shape values.

Table 7-374: List of State elements (Section 2 of 2)

Name	Data Type	Description
StringState	element	Describes a set of string values.
XYPairState	element	Describes a set of XYPair values.

— Element: 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 7-375: BooleanState element

Name	Data Type	Description
AllowedValueList ? New in JDF 1.1A	enumerations	A list of all legal values. Allowed list values are the boolean " <i>true</i> " and " <i>false</i> ".
CurrentValue ?	boolean	Current value for the current running job set in the device.
DefaultValue ?	boolean	Default value if not specified in a submitted JDF. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = " <i>true</i> ".
PresentValueList ? New in JDF 1.1A	enumerations	A list of all supported values that can be chosen without operator intervention. Allowed list values are the boolean " <i>true</i> " and " <i>false</i> ". If not specified, the value of <i>AllowedValueList</i> is applied.
ValueLoc * New in JDF 1.2	element	Localization(s) of " <i>true</i> " and/or " <i>false</i> " values. (See Structure of the ValueLoc Subelement under "BooleanState" on page 628.)

— Element: 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 7-376: ValueLoc element

Name	Data Type	Description
<i>Value</i>	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), <i>Value</i> MUST be an instance of the appropriate data type.
<i>Loc *</i>	element	The localization(s) of the attribute value. (See "Loc" on page 621.)

— Element: DateTimeState[New in JDF 1.2](#)

This **State** subelement is used to describe ranges of date`Time` values. It inherits from the abstract **State** element described above.

Table 7-377: DateTimeState element (Section 1 of 2)

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.

Table 7-377: DateTimeState element (Section 2 of 2)

Name	Data Type	Description
<i>DefaultValue ?</i>	dateTime	Default value if not specified in a submitted JDF. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = "true".
<i>PresentValueDurationList ?</i>	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 <i>AllowedValueDurationList</i> is applied.
<i>PresentValueList ?</i>	DateTimeRangeList	Inclusive minimum and maximum allowed value that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueList</i> is applied.
ValueLoc *	element	Localization(s) of specific dates. (See Structure of the ValueLoc Subelement under "BooleanState" on page 628.)

— Element: 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 7-378: DurationState element

Name	Data Type	Description
<i>AllowedValueList ?</i>	DurationRangeList	A list of all supported values.
<i>CurrentValue ?</i>	duration	Current value for the current running job set in the device.
<i>DefaultValue ?</i>	duration	Default value if not specified in a submitted JDF. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = "true".
<i>PresentValueList ?</i>	DurationRangeList	Inclusive minimum and maximum allowed value that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueList</i> is applied.
ValueLoc *	element	Localization(s) of specific durations. (See Structure of the ValueLoc Subelement under "BooleanState" on page 628.)

— Element: 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 7-379: EnumerationState element (Section 1 of 2)

Name	Data Type	Description
<i>AllowedValueList ?</i>	enumerations	A list of all supported values. The values specified in <i>AllowedValueList</i> MUST be a subset of the enumeration specified by <i>Name</i> . 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 <i>AllowedValueList</i> , even when the entire range of schema-valid values is supported.

Table 7-379: EnumerationState element (Section 2 of 2)

Name	Data Type	Description
<i>CurrentValue</i> ?	enumeration	Current value for the current running job set in the device. <i>CurrentValue</i> MUST match the enumeration defined in the resource.
<i>DefaultValue</i> ?	enumeration	Default value if not specified in a submitted JDF. <i>DefaultValue</i> MUST match the enumeration defined in the resource, and MUST be specified if <i>HasDefault</i> = "true".
<i>PresentValueList</i> ?	enumerations	A list of values that can be chosen without operator intervention. <i>PresentValueList</i> MUST match the enumeration defined in the resource. If not specified, the value of <i>AllowedValueList</i> is applied.
<i>ValueLoc</i> * New in JDF 1.2	element	Localizations of the enumerations listed in <i>AllowedValueList</i> and <i>PresentValueList</i> . (See Structure of the ValueLoc Subelement under "BooleanState" on page 628.)

— Element: 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 7-380: IntegerState element (Section 1 of 2)

Name	Data Type	Description
<i>AllowedValueList</i> ? Modified in JDF 1.2	Integer-RangeList	A list of all supported values.
<i>AllowedValueMax</i> ? Deprecated in JDF 1.2	integer	Inclusive maximum allowed value. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
<i>AllowedValueMin</i> ? Deprecated in JDF 1.2	integer	Inclusive minimum allowed value. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
<i>AllowedValueMod</i> ? New in JDF 1.2	XYPair	X defines the Modulo and Y the offset of the allowed value. In other words, if <i>AllowedValueMod</i> = "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: $4 \text{ mod } 3 = 4 - 3 = 1$; $17 \text{ mod } 3 = 17 - 5 * 3 = 2$; and $3 \text{ mod } 3 = 3 - 3 = 0$.
<i>CurrentValue</i> ?	integer	Current value for the current running job set in the device.
<i>DefaultValue</i> ?	integer	Default value if not specified in a submitted JDF. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = "true".
<i>PresentValueList</i> ? Modified in JDF 1.2	Integer-RangeList	A list of values that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueList</i> is applied.
<i>PresentValueMax</i> ? Deprecated in JDF 1.2	integer	Inclusive maximum allowed value that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueMax</i> is applied. Replaced by <i>PresentValueList</i> in JDF 1.2 and beyond.
<i>PresentValueMin</i> ? Deprecated in JDF 1.2	integer	Inclusive minimum allowed value that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueMin</i> is applied. Replaced by <i>PresentValueList</i> in JDF 1.2 and beyond.
<i>PresentValueMod</i> ? New in JDF 1.2	XYPair	X defines the Modulo and Y the offset of the present value. In other words, if <i>AllowedValueMod</i> = "10 2", only the values ... -8,2,12,22 ... are allowed. If not specified, the value of <i>AllowedValueMod</i> is applied. If $((N\%X)-Y = 0)$ then N is a valid value.

Table 7-380: IntegerState element (Section 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. <i>UnitType</i> SHOULD be specified if the <i>IntegerState</i> represents a value that has units. User interfaces might not display correctly if <i>UnitType</i> is not specified for attributes with units. Possible values include: <i>Angle</i> – The attribute is defined in degrees. <i>AngularVelocity</i> – Rotations / minute. <i>Area</i> – Area in square meters (m ²). <i>Currency</i> – The local currency. <i>Length</i> – In points (1/72 inch). <i>LengthMu</i> – Length in microns (used for paper thickness). <i>LineScreen</i> – The lines per inch (lpi) for conventionally screened half-tone, screened grayscale and screened monotone bitmap images. <i>PaperWeight</i> – In grams per square meter (g/m ²). <i>Percentage</i> – A percentage value. <i>Pressure</i> – In Pascals. <i>Resolution</i> – The dots per inch (dpi) for print output and bitmap image (e.g., TIFF or BMP) file resolution. <i>ScreenResolution</i> – The pixels per inch (ppi) for screen display (e.g., softproof display and user interface display), scanner capture settings and digital camera settings. <i>SpotResolution</i> – 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. <i>Temperature</i> – Temperature in degrees Centigrade. <i>Velocity</i> – Defined as meters/hour. <i>Weight</i> – Weight in grams.
ValueLoc * New in JDF 1.2	element	Localization(s) of specific values. (See Structure of the ValueLoc Subelement under "BooleanState" on page 628.)

— Element: 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 physical resources, (e.g., in finishing devices).

Table 7-381: MatrixState element (Section 1 of 2)

Name	Data Type	Description
AllowedRotateMod ? New in JDF 1.2	double	Allowed Modulo of the allowed rotations and offset in degrees. Examples: <i>360</i> – No rotation <i>90</i> – Any orthogonal rotation. <i>0</i> – Any rotation is allowed.

Table 7-381: MatrixState element (Section 2 of 2)

Name	Data Type	Description
<i>AllowedShift</i> ? New in JDF 1.2	DoubleList	Minimum and maximum allowed shift of the matrix. If not specified, any shift is valid. If <i>AllowedTransforms</i> is specified, the implied shift defined in Table 2-4 on page 26 is subtracted from <i>AllowedShift</i> , thus all in-place rotations have an implied <i>AllowedShift</i> 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.
<i>AllowedTransforms</i> ? New in JDF 1.2	Orientations	List of valid orthogonal transformations of the matrix. Any of the eight predefined transforms for physical resources as defined in Table 2-4 on page 26.
<i>CurrentValue</i> ?	matrix	Current value for the current running job set in the device.
<i>DefaultValue</i> ?	matrix	Default value if not specified in a submitted JDF. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = "true".
<i>PresentRotateMod</i> ? New in JDF 1.2	double	Present Modulo of the allowed rotations and offset in degrees that can be chosen without operator intervention. Examples: 360 – No rotation is allowed. 90 – Any orthogonal rotation. 0 – Any rotation is allowed. If not specified, the value of <i>AllowedRotateMod</i> is applied.
<i>PresentShift</i> ? New in JDF 1.2	DoubleList	If <i>PresentTransforms</i> is specified, the implied shift defined in Table 2-4 on page 26 is subtracted from <i>PresentShift</i> , thus all in-place rotations have an implied <i>PresentShift</i> value of "0 0 0 0". If not specified, the value of <i>AllowedShift</i> is applied.
<i>PresentTransforms</i> ? New in JDF 1.2	Orientations	Any of the eight predefined transforms for physical resources as defined in Table 2-4 on page 26. If not specified, the value of <i>AllowedTransforms</i> is applied.
<i>Value</i> *	element	A list legal values.

— Element: MatrixState/Value**Table 7-382: MatrixState/Value element**

Name	Data Type	Description
<i>AllowedValue</i>	matrix	A legal value for a matrix variable.
<i>PresentValue</i> ? 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 <i>AllowedValue</i> is applied. In JDF 1.2 and beyond, use <i>ValueUsage</i> .
<i>ValueUsage</i> ? New in JDF 1.2	enumeration	Defines whether the value defined in <i>AllowedValue</i> means <i>Present</i> , <i>Allowed</i> or both. One of: <i>Present</i> – Present configuration is supported. <i>Allowed</i> – Allowed configuration is supported. If not specified, <i>Value</i> is valid for both <i>Present</i> and <i>Allowed</i> .
<i>Loc</i> * New in JDF 1.2	element	The localization(s) of the string defined in <i>AllowedValue</i> . (See "Loc" on page 621.)

— Element: 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 7-383: NameState element

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.
DefaultValue ?	NMTOKEN	Default value if not specified in a submitted JDF. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = "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 <i>AllowedRegExp</i> is applied.
PresentValueList ?	NMTOKENS	A list of values that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueList</i> is applied.
ValueLoc * New in JDF 1.2	element	Localization(s) of the NMTOKENS listed in <i>AllowedValueList</i> or <i>PresentValueList</i> or implied by <i>AllowedRegExp</i> or <i>PresentRegExp</i> . (See Structure of the ValueLoc Subelement under "BooleanState" on page 628.)

— Element: NumberState[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 7-384: NumberState element (Section 1 of 2)

Name	Data Type	Description
AllowedValueList ? Modified in JDF 1.2	DoubleRange-List	A list of supported values.
AllowedValueMax ? Deprecated in JDF 1.2	double	Inclusive maximum allowed value. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
AllowedValueMin ? Deprecated in JDF 1.2	double	Inclusive minimum allowed value. Replaced by <i>AllowedValueList</i> 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 <i>AllowedValueMod</i> = "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: $4 \bmod 3 = 4 - 3 = 1$; $17 \bmod 3 = 17 - 5 * 3 = 2$; and $3 \bmod 3 = 3 - 3 = 0$.
CurrentValue ?	double	Current value for the current running job set in the device.
DefaultValue ?	double	Default value if not specified in a submitted JDF. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = "true".
PresentValueList ? Modified in JDF 1.2	DoubleRange-List	A list of values that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueList</i> is applied.

Table 7-384: NumberState element (Section 2 of 2)

Name	Data Type	Description
PresentValueMax ? Deprecated in JDF 1.2	double	Inclusive maximum allowed value that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueMax</i> is applied. Replaced by <i>PresentValueList</i> 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 <i>AllowedValueMin</i> is applied. Replaced by <i>PresentValueList</i> 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 <i>AllowedValueMod</i> = "10 2", only the values ... -8,2,12,22 ... are allowed. If not specified, the value of <i>AllowedValueMod</i> is applied. If ((N%X)-Y = 0) then N is a valid value.
UnitType ? New in JDF 1.2	NMTOKEN	Specifies the unit type that this <i>State</i> element represents. Used to enable an application to localize the representation of the units. <i>UnitType</i> MUST be specified if the <i>NumberState</i> represents a value that has units. Possible values are defined in "IntegerState" on page 630 in the <i>UnitType</i> attribute definition. Some additional possible values that are typically used only with <i>Number</i> attributes are: <i>CMYKColor</i> – Four values representing a CMYK color. <i>LabColor</i> – Three values representing a Lab color. <i>sRGBColor</i> – Three values representing a sRGB color.
ValueLoc * New in JDF 1.2	element	Localization(s) of specific values. (See Structure of the ValueLoc Subelement under "BooleanState" on page 628.)

— Element: 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 7-385: PDFPathState element

Name	Data Type	Description
AllowedLength ?	Integer-Range	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. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = "true".
PresentLength ?	Integer-Range	Inclusive minimum and maximum length of valid PDF path in characters that can be chosen without operator intervention. If not specified, the value of <i>AllowedLength</i> is applied.
Value *	element	The localization(s) of the PDF path defined in <i>AllowedValue</i> .

— Element: PDFPathState/Value

Table 7-386: PDFPathState/Value element (Section 1 of 2)

Name	Data Type	Description
AllowedValue	PDFPath	A legal value for a matrix variable.

Table 7-386: PDFPathState/Value element (Section 2 of 2)

Name	Data Type	Description
<i>ValueUsage</i> ?	enumeration	Defines whether the value defined in <i>AllowedValue</i> means <i>Present</i> , <i>Allowed</i> or both. One of: <i>Present</i> – Present configuration is supported. <i>Allowed</i> – Allowed configuration is supported. If not specified, <i>Value</i> is valid for both <i>Present</i> and <i>Allowed</i> .
Loc *	element	The localization(s) of the string defined in <i>AllowedValue</i> . (See “Loc” on page 621.)

— Element: 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 7-387: RectangleState element

Name	Data Type	Description
<i>AllowedHWRelation</i> ?	XYRelation	Allowed relative value of width (X) vs. Height (Y).
<i>AllowedValueList</i> ?	RectangleRangeList	A list of ranges of allowed values that can be chosen.
<i>CurrentValue</i> ?	rectangle	Current value for the current running job set in the device.
<i>DefaultValue</i> ?	rectangle	Default value if not specified in a submitted JDF. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = “true”.
<i>PresentHWRelation</i> ?	XYRelation	Allowed relative value of width (X) vs. Height (Y). If not specified, the value of <i>AllowedHWRelation</i> is applied.
<i>PresentValueList</i> ?	RectangleRangeList	A list of ranges of values that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueList</i> is applied.
ValueLoc *	element	A list of supported values. The <i>ValueLoc/@Value</i> attribute MUST be a representation of a rectangle. This can also be used to localize (or provide names for) specific rectangles. (See Structure of the ValueLoc Subelement under “BooleanState” on page 628.)

— Element: 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 7-388: ShapeState element (Section 1 of 2)

Name	Data Type	Description
<i>AllowedValueList</i> ? Modified in JDF 1.2	ShapeRange-List	A list of values that can be chosen.
<i>AllowedValueMax</i> ? Deprecated in JDF 1.2	shape	Inclusive maximum allowed value. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
<i>AllowedValueMin</i> ? Deprecated in JDF 1.2	shape	Inclusive minimum allowed value. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.

Table 7-388: ShapeState element (Section 2 of 2)

Name	Data Type	Description
AllowedX ? New in JDF 1.2	Double-RangeList	Allowed X-axis of the <i>Shape</i> .
AllowedY ? New in JDF 1.2	Double-RangeList	Allowed Y-axis of the <i>Shape</i> .
AllowedZ ? New in JDF 1.2	Double-RangeList	Allowed Z-axis of the <i>Shape</i> .
CurrentValue ?	shape	Current value for the current running job set in the device.
DefaultValue ?	shape	Default value if not specified in a submitted JDF. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = "true".
PresentValueList ? Modified in JDF 1.2	Shape-RangeList	A list of values that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueList</i> 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 <i>AllowedValueMax</i> is applied. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
PresentValueMin ? Deprecated in JDF 1.2	shape	Inclusive minimum allowed value that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueMin</i> is applied. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
PresentX ? New in JDF 1.2	Double-RangeList	Present X-axis of the <i>Shape</i> that can be chosen without operator intervention. If not specified, the value of <i>AllowedX</i> is applied.
PresentY ? New in JDF 1.2	Double-RangeList	Present Y-axis of the <i>Shape</i> that can be chosen without operator intervention. If not specified, the value of <i>AllowedY</i> is applied.
PresentZ ? New in JDF 1.2	Double-RangeList	Present Z-axis of the <i>Shape</i> that can be chosen without operator intervention. If not specified, the value of <i>AllowedZ</i> is applied.
ValueLoc * New in JDF 1.2	element	A list of supported shapes. (See Structure of the ValueLoc Subelement under "BooleanState" on page 628.)

— Element: 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 7-389: StringState element (Section 1 of 2)

Name	Data Type	Description
AllowedLength ? New in JDF 1.2	Integer-Range	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. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = "true".
PresentLength ? New in JDF 1.2	Integer-Range	Inclusive minimum and maximum length of valid string in characters that can be chosen without operator intervention. If not specified, the value of <i>AllowedLength</i> is applied.

Table 7-389: StringState element (Section 2 of 2)

Name	Data Type	Description
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 <i>AllowedRegExp</i> is applied.
Value * Modified in JDF 1.2	element	A list legal values.

— Element: StringState/Value

[New in JDF 1.1](#)

Table 7-390: StringState/Value element

Name	Data Type	Description
<i>AllowedValue</i>	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 <i>AllowedValue</i> is applied. In JDF 1.2 and beyond, use <i>ValueUsage</i> .
ValueUsage ? New in JDF 1.2	enumeration	Defines whether the value defined in <i>AllowedValue</i> means <i>Present</i> , <i>Allowed</i> or both. One of: <i>Present</i> – Present configuration is supported. <i>Allowed</i> – Allowed configuration is supported. If not specified, <i>Value</i> is valid for both <i>Present</i> and <i>Allowed</i> .
Loc * New in JDF 1.2	element	The localization(s) of the string defined in <i>AllowedValue</i> . (See “Loc” on page 621.)

— Element: 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 7-391: XYPairState element (Section 1 of 2)

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 <i>AllowedValueList</i> in JDF 1.2 and beyond.
AllowedValueMin ? Deprecated in JDF 1.2	XYPair	Inclusive minimum allowed value. Replaced with <i>AllowedValueList</i> 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. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = “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 <i>AllowedValueList</i> is applied.

Table 7-391: XYPairState element (Section 2 of 2)

Name	Data Type	Description
PresentValueMax ? Deprecated in JDF 1.2	XYPair	Inclusive maximum allowed value that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueMax</i> is applied. Replaced with <i>PresentValueList</i> 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 <i>AllowedValueMin</i> is applied. Replaced with <i>PresentValueList</i> 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 <i>AllowedXYRelation</i> is applied.
UnitType ? New in JDF 1.2	NMTOKEN	Specifies the unit type that this <i>State</i> element represents. Used to enable an application to localize the representation of the units. <i>UnitType</i> MUST be specified if the <i>IntegerState</i> represents a value that has units. Possible values are defined in "IntegerState" on page 630 in the <i>UnitType</i> attribute definition.
ValueLoc * New in JDF 1.2	element	A list of supported shapes. (See Structure of the ValueLoc Subelement under "BooleanState" on page 628.)

7.3.8 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.

Example

```
<DeviceCap>
  <DisplayGroupPool>
    <DisplayGroup rRefs="bad">
      <Loc HelpText="Parameters for scanning configuration" Lang="en"
        Value="ScanningParameters"/>
    </DisplayGroup>
  </DisplayGroupPool>
</DeviceCap>
```

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.

Table 7-392: DisplayGroupPool element

Name	Data Type	Description
<i>DisplayGroup</i> *	element	Declares a set of references to <i>State</i> elements that are intended to be displayed as a group in user interfaces.

— Element: DisplayGroup

Each *DisplayGroup* element declares a group of features that are intended to be displayed together in user interfaces.

Table 7-393: DisplayGroup element

Name	Data Type	Description
<i>rRefs</i>	IDREFS	References to State elements. (See "State" on page 623 for details of the State element.)
Loc *	element	Localized strings describing the DisplayGroup.

7.3.9 FeaturePool

[New in JDF 1.2](#)

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 "MacroPool" on page 639.

These features can be mapped to JDF/@*NamedFeatures*. A feature from JDF/@*NamedFeatures* is selected by specifying an NMTOKEN pair that matches entries from FeaturePool/EnumerationState/@*Name* and FeaturePool/EnumerationState/@*AllowedValueList*

Example:

```
<DeviceCap>
  <FeaturePool>
    <EnumerationState AllowedValueList="Mono ColorTransparency Photo" ID="sm"
      HasDefault="false" MacroRefs="ScanModeMac" Name="ScanMode" UserDisplay="Display"/>
  </FeaturePool>
</DeviceCap>
```

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.

Table 7-394: FeaturePool element

Name	Data Type	Description
State *	element	Abstract State elements that define the accepted parameter space for the messages or resources subelements. These abstract subelements are identical in form to other State elements, but typically are only "macro" features that control other features through macros . For more information on macro definitions, see "MacroPool" on page 639. For details of the State element, see "State" on page 623.

7.3.10 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 7-395: MacroPool element

Name	Data Type	Description
macro *	element	A list of independent macros.

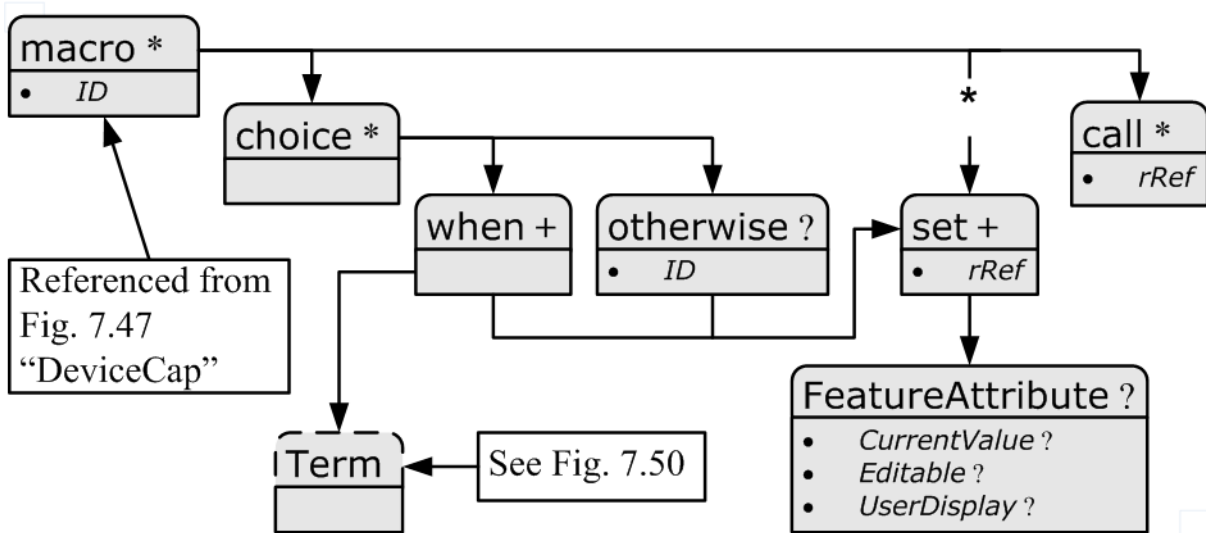


Figure 7-52: macro element – a diagram of its structure

— Element: 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 "Term" on page 643) and a **set** element. When the expression evaluates to "*true*", the action specified in the **set** element is to be performed. If no evaluation in any **when** element in a **choice** evaluates to "*true*", the action(s) specified in the **otherwise** element is to 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 **MUST** 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 7-396: macro element

Name	Data Type	Description
<i>ID</i>	ID	Unique identifier of a macro element. This <i>ID</i> 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 MUST be specified in macro .
set *	element	An element that sets one or more State attribute values. At least one of choice , set or call MUST 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 MUST be specified in macro .

— Element: 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 "Term" on page 643) and a **set** element. When the expression evaluates to "*true*", the action specified in the **set** element is to 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 is to be performed.

Table 7-397: 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 <i>"true"</i> .

— Element: otherwise

The **otherwise** subelement sets one or more feature values if none of the **when** expressions in a **choice** element evaluate to *"true"*.

Table 7-398: otherwise element

Name	Data Type	Description
set +	element	An element that sets one or more feature values.

— Element: 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 "Term" on page 643) and a **set** element. When the Term evaluates to *"true"*, the action specified in the set element is to be performed.

Table 7-399: 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.

— Element: set

The **set** subelement sets one or more **State** element attribute values.

Table 7-400: set element

Name	Data Type	Description
<i>rRef</i>	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).

— Element: 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 7-401: FeatureAttribute element (Section 1 of 2)

Name	Data Type	Description
<i>CurrentValue ?</i>	string	The value to change the <i>CurrentValue</i> attribute of the State element to. Note that the mapping of the string to the actual data type of the State element MUST be performed by the application processing the capabilities.
<i>Editable ?</i>	boolean	When <i>"true"</i> , the feature and its current value can be edited by the user. If <i>"false"</i> , the user interface MUST NOT allow user modification of the current value of the State element.

Table 7-401: FeatureAttribute element (Section 2 of 2)

Name	Data Type	Description
<i>UserDisplay ?</i>	enumeration	Indicates under which conditions the feature is to be displayed in user interfaces. Possible values are the same as the <i>UserDisplay</i> attribute of the <i>State</i> element.

— Element: call

The *call* subelement is used to call other macro elements, effectively using them as macro “templates”.

Table 7-402: call element

Name	Data Type	Description
<i>rRef</i>	IDREF	Reference to a macro.

7.3.11 Performance

[New in JDF 1.1](#)

The *Performance* element describes speed as the capability to consume or produce a JDF Resource.

Table 7-403: Performance element

Name	Data Type	Description
<i>AverageAmount ?</i>	double	Average amount produced/consumed per hour assuming an average job.
<i>AverageCleanup ?</i>	duration	Average time needed to clean the device after a job.
<i>AverageSetup ?</i>	duration	Average time needed to setup the device before a job.
<i>DevCapsRef ?</i> New in JDF 1.2	IDREF	Reference to the <i>DevCaps</i> element that describes the resource whose performance is specified by this <i>Performance</i> element.
<i>MaxAmount ?</i>	double	Maximum amount produced/consumed per hour, assuming an ideal job. The default value of “0” translates to the value of <i>AverageAmount</i> .
<i>MaxCleanup ?</i>	duration	Maximum time needed to clean the device after a job, assuming a worst case job. Defaults to <i>AverageCleanup</i> .
<i>MaxSetup ?</i>	duration	Maximum time needed to setup the device before a job, assuming a worst case job. Defaults to <i>AverageSetup</i> .
<i>MinAmount ?</i>	double	Minimum amount produced/consumed per hour, assuming a worst case job. Defaults to <i>AverageAmount</i> .
<i>MinCleanup ?</i>	duration	Minimum time needed to clean the device after a job, assuming an ideal job. Defaults to <i>AverageCleanup</i> .
<i>MinSetup ?</i>	duration	Minimum time needed to setup the device before a job, assuming an ideal job. Defaults to <i>AverageSetup</i> .
<i>Name ?</i> Deprecated in JDF 1.2	NMTOKEN	Name of the input resource type that is processed by the device, (e.g., Media, Ink, RunList). Replaced with <i>DevCapsRef</i> in JDF 1.2 and beyond
<i>Unit ?</i>	NMTOKEN	Unit of measure of resource consumption per hour. Defaults to the resource’s generic units as defined in Table 1-7, “Units used in JDF,” on page 13.

7.3.12 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 7-404: TestPool element

Name	Data Type	Description
Test *	element	A list of independent Tests.

— Element: 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 Tests and macros by representing a commonly used Boolean expression.

Table 7-405: 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).

7.3.13 Term

The abstract 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. The Term elements are composed of Boolean combinations of three types of elements:

- Boolean expressions (i.e., nesting) comprising of “and”, “or”, “not” and “xor”. (See “Boolean Operators” on page 645.)
- Evaluation elements, which evaluate a JDF State attribute value to create a simple true or false Boolean expression, (e.g., “Is the value of BitDepth equal to 8?”). (See “Evaluation” on page 646.)
- TestRef — A reference to a constraint Test element. This referenced constraint is then used as a nested Boolean expression. (See “TestRef” on page 652.)

Table 7-406: List of abstract Term elements (Section 1 of 2)

Name	Defined in Section	Description
and	See “Boolean Operators” on page 645.	Boolean AND operator.
not	See “Boolean Operators” on page 645.	Boolean negation.
or	See “Boolean Operators” on page 645.	Boolean OR operator.
xor	See “Boolean Operators” on page 645.	Boolean exclusive or (XOR) operator.
BooleanEvaluation	See “Evaluation” on page 646.	Describes operations on a set of Boolean values.
DateTimeEvaluation	See “Evaluation” on page 646. See “Evaluation” on page 646.	Describes operations on a set of dateTime values.
DurationEvaluation	See “Evaluation” on page 646.	Describes operations on a set of duration values.
EnumerationEvaluation	See “Evaluation” on page 646.	Describes operations on a set of enumeration values.
IntegerEvaluation	See “Evaluation” on page 646.	Describes operations on a numerical range of integer values.
IsPresentEvaluation	See “Evaluation” on page 646.	Checks for the existence of a tag, element or feature.

Table 7-406: List of abstract Term elements (Section 2 of 2)

Name	Defined in Section	Description
MatrixEvaluation	See “Evaluation” on page 646.	Describes operations on a range of matrices. Generally used to define valid orientations of Component resources.
NameEvaluation	See “Evaluation” on page 646.	Describes operations on a set of NMTO-KEN values
NumberEvaluation	See “Evaluation” on page 646.	Describes operations on a numerical range of values.
PDFPathEvaluation	See “Evaluation” on page 646.	Describes operations on PDFPath.
RectangleEvaluation	See “Evaluation” on page 646.	Describes operations on a set of four-value rectangle values.
ShapeEvaluation	See “Evaluation” on page 646.	Describes operations on a set of three-value shape values.
StringEvaluation	See “Evaluation” on page 646.	Describes operations on a set of string values.
XYPairEvaluation	See “Evaluation” on page 646.	Describes operations on a set of XYPair values.
TestRef	See “TestRef” on page 652.	Reference to a constraint Test to be evaluated as a nested Boolean expression inside a larger expression.

Example:

```

<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>

```

Note: **Term** is an abstract element, so it will never appear in a JDF document. In the “*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.

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.

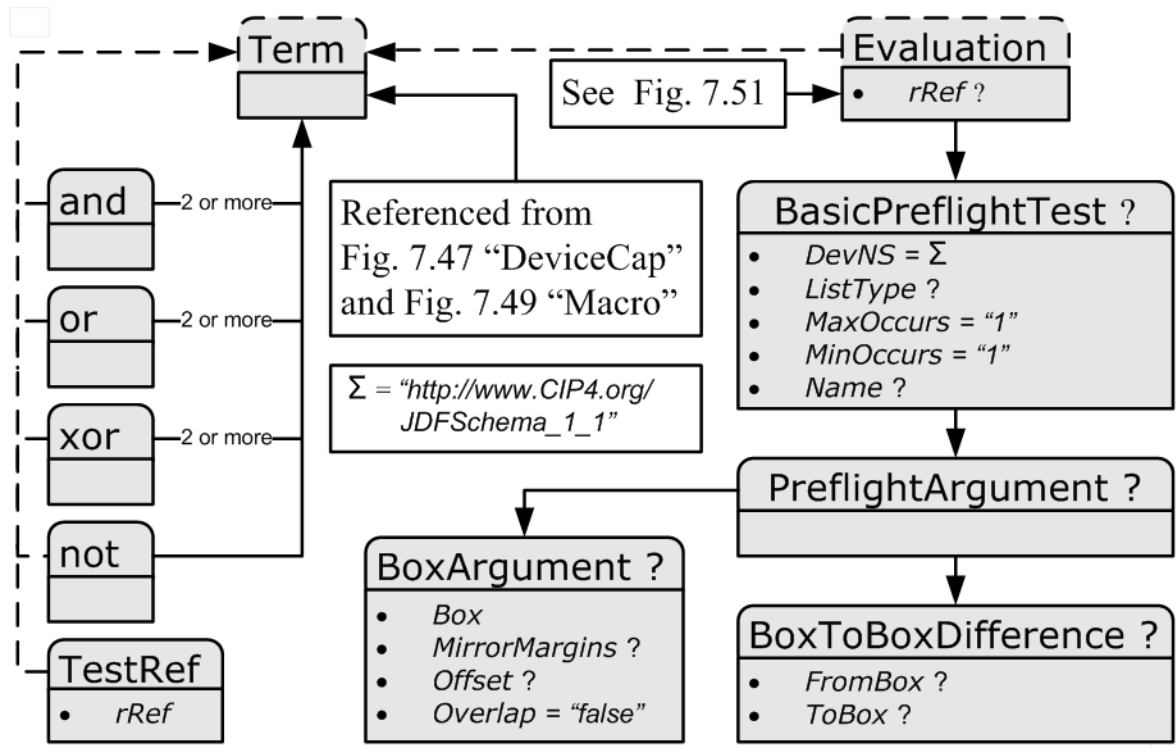


Figure 7-53: Abstract Term element – a diagram of its structure

7.3.13.1 Boolean Operators

The Boolean operators that are defined in this section are instances of Terms, and, thus, they MAY be nested. They are used both in device capabilities and preflighting context.

— Element: 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 7-407: and element

Name	Data Type	Description
Term	element	Any element derived from an abstract Term.
Term +	element	Any element derived from an abstract Term.

— Element: 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 7-408: or element

Name	Data Type	Description
Term	element	Any element derived from an abstract Term.
Term +	element	Any element derived from an abstract Term.

— Element: 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** MUST evaluate to *“true”* for the **xor** to evaluate to *“true”*. Note that this is different from the mathematical behavior of “xor”.

Table 7-409: xor element

Name	Data Type	Description
Term	element	Any element derived from an abstract Term .
Term +	element	Any element derived from an abstract Term .

— Element: not

The **not** subelement inverts the Boolean state of a **Term**.

Table 7-410: not element

Name	Data Type	Description
Term	element	Any element derived from an abstract Term .

7.3.13.2 Evaluation

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.

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 7-411: Mapping of Evaluation elements 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	all	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.

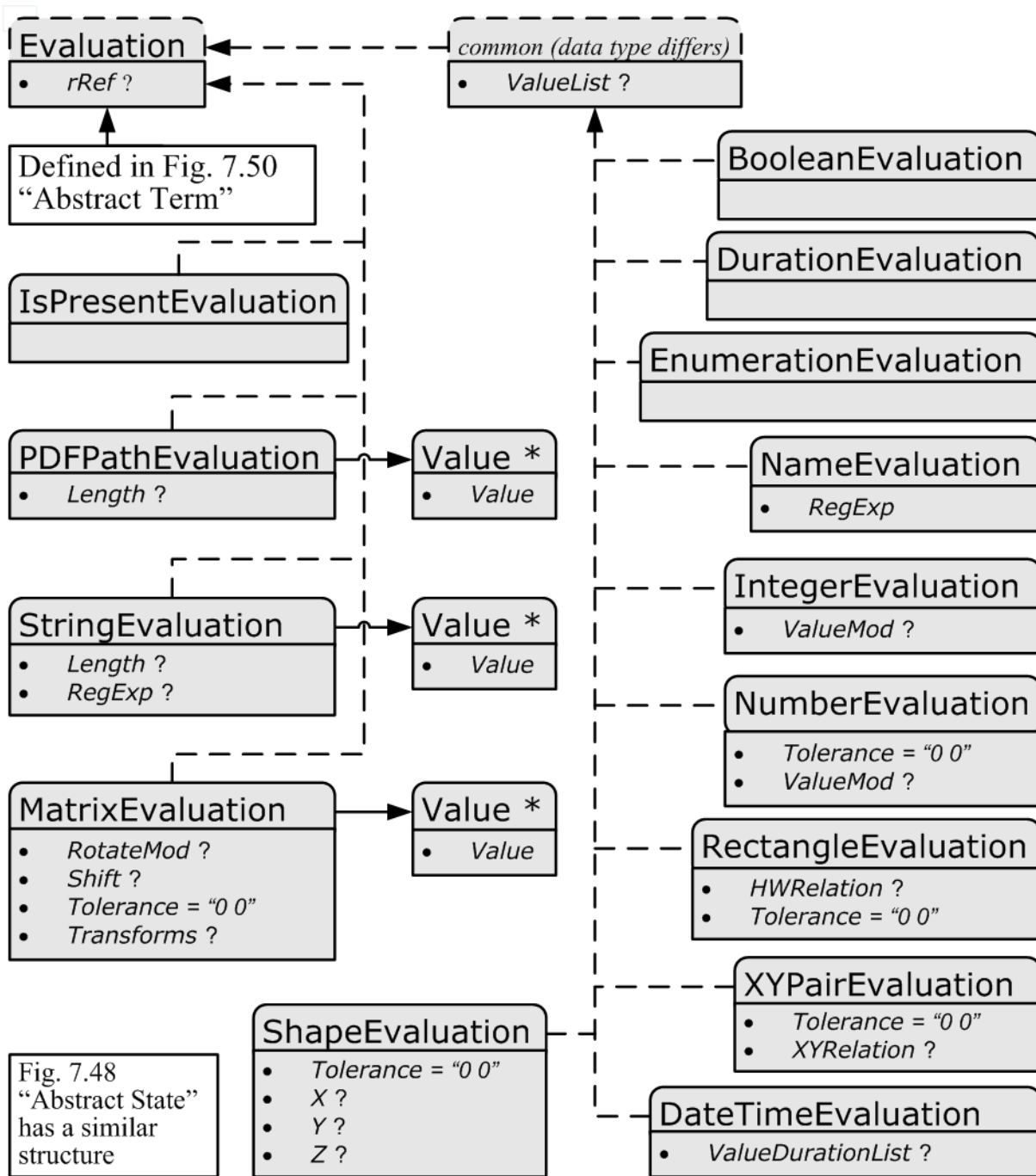


Figure 7-54: Abstract Evaluation element – a diagram of its structure

— Element: Abstract Evaluation

The following table describes the common, data type-independent parameters of all Evaluation elements.

Table 7-412: Abstract Evaluation element

Name	Data Type	Description
<i>rRef</i> ?	IDREF	A reference to State or Module elements when used in the context of device capability descriptions. Exactly one of BasicPreflightTest or <i>rRef</i> MUST be specified.
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 MUST reference the appropriate State element using <i>rRef</i> . For details of the BasicPreflightTest , see "PreflightParams" on page 540. Exactly one of BasicPreflightTest or <i>rRef</i> MUST be specified.

— Element: 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 7-413: BooleanEvaluation element

Name	Data Type	Description
<i>ValueList</i> ?	enumerations	A list of all supported values. Allowed list values are the Boolean values "true" and "false".

— Element: DateTimeEvaluation

The **DateTimeEvaluation** element declares a Boolean value for comparison in an expression to a **DateTimeState** element in constraints.

Table 7-414: DateTimeEvaluation element

Name	Data Type	Description
<i>ValueDurationList</i> ?	DurationRangeList	List of inclusive minimum and maximum allowed values relative to the current system time.
<i>ValueList</i> ?	DateTimeRangeList	A list of all supported values.

— Element: DurationEvaluation

The **DurationEvaluation** 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 7-415: DurationEvaluation element

Name	Data Type	Description
<i>ValueList</i> ?	DurationRangeList	A list of all supported values.

— Element: EnumerationEvaluation

The **EnumerationEvaluation** element declares an enumeration value for comparison in an expression to an **EnumerationState** element in constraints.

Table 7-416: EnumerationEvaluation element

Name	Data Type	Description
<i>ValueList</i> ?	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 <i>ValueList</i> , even when the entire range of schema-valid values is supported.

— Element: IntegerEvaluation

The IntegerEvaluation element declares an Integer value for comparison in an expression to a IntegerState element in constraints.

Table 7-417: IntegerEvaluation element

Name	Data Type	Description
<i>ValueList</i> ?	IntegerRangeList	A list of all supported values.
<i>ValueMod</i> ?	XYPair	X defines the Modulo and Y the offset of the allowed value. In other words, if <i>AllowedValueMod</i> = "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: $4 \bmod 3 = 4 - 3 = 1$; $17 \bmod 3 = 17 - 5 * 3 = 2$; and $3 \bmod 3 = 3 - 3 = 0$.

— Element: 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 parameters. IsPresentEvaluation/@rRef MAY reference a DevCap element in order to test for the existence of an element.

— Element: MatrixEvaluation

The MatrixEvaluation element declares a matrix value for comparison in an expression to a MatrixState element in constraints.

Table 7-418: MatrixEvaluation element (Section 1 of 2)

Name	Data Type	Description
<i>RotateMod</i> ?	double	Allowed Modulo of the allowed rotations and offset in degrees. Examples: <i>360</i> – No rotation is allowed. <i>90</i> – Any orthogonal rotation. <i>0</i> – Interpreted to mean that any rotation is allowed. Note: Although this seems counter-intuitive and contrary to the convention set in JDF coordinate systems, the application of <i>RotateMod</i> in practice will involve subtracting values by the value of the <i>RotateMod</i> . Hence, any number is reduced by "0" and is unaffected by the subtraction.
<i>Shift</i> ?	DoubleList	If <i>Transforms</i> is specified, the implied shift defined in Table 2-4, "Matrices and Orientation values for describing the orientation of a Component," on page 26 is subtracted from <i>Shift</i> , thus all in-place rotations have an implied Shift value of "0 0 0 0".
<i>Tolerance</i> = "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.

Table 7-418: MatrixEvaluation element (Section 2 of 2)

Name	Data Type	Description
<i>Transforms</i> ?	Orientations	Any of the eight predefined transforms for physical resources as defined in Table 2-4, “Matrices and Orientation values for describing the orientation of a Component,” on page 26.
<i>Value</i> *	element	A list supported values. The <i>Value/@Value</i> attribute MUST be a representation of a matrix.

— Element: MatrixEvaluation/Value**Table 7-419: MatrixEvaluation/Value element**

Name	Data Type	Description
<i>Value</i>	matrix	A supported value for a matrix variable.

— Element: NameEvaluation

The NameEvaluation element declares a NMTOKEN value for comparison in an expression to a NameState element in constraints.

Table 7-420: NameEvaluation element

Name	Data Type	Description
<i>RegExp</i>	regExp	Regular expression that limits the allowed values.
<i>ValueList</i> ?	NMTOKENS	A list of supported values.

— Element: NumberEvaluation

The NumberEvaluation element declares a number value for comparison in an expression to a NumberState element in constraints.

Table 7-421: NumberEvaluation element

Name	Data Type	Description
<i>Tolerance</i> = "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.
<i>ValueList</i> ?	DoubleRangeList	A list of supported values.
<i>ValueMod</i> ?	XYPair	X defines the Modulo and Y the offset of the allowed value. In other words, if <i>AllowedValueMod</i> = "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: $4 \bmod 3 = 4 - 3 = 1$; $17 \bmod 3 = 17 - 5 * 3 = 2$; and $3 \bmod 3 = 3 - 3 = 0$.

— Element: PDFPathEvaluation

The PDFPathEvaluation element declares a PDF path value for comparison in an expression to a PDFPathState element in constraints.

Table 7-422: PDFPathEvaluation element

Name	Data Type	Description
<i>Length</i> ?	IntegerRange	Inclusive minimum and maximum length of valid PDF path in characters.
<i>Value</i> *	element	PDF path values for comparison in an expression to a PDFPathState element.

— Element: **PDFPathEvaluation/Value**

Table 7-423: PDFPathEvaluation/Value element

Name	Data Type	Description
<i>Value</i>	PDFPath	A supported value for a PDF path attribute.

— Element: **RectangleEvaluation**

The **RectangleEvaluation** element declares a Boolean value for comparison in an expression to a **RectangleState** element in constraints.

Table 7-424: RectangleEvaluation element

Name	Data Type	Description
<i>HWRRelation</i> ?	XYRelation	Allowed relative value of width (X) versus height (Y).
<i>Tolerance</i> = "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.
<i>ValueList</i> ?	RectangleRangeList	A list of ranges of allowed values that can be chosen.

— Element: **ShapeEvaluation**

The **ShapeEvaluation** element declares a shape value for comparison in an expression to a **State** element in constraints.

Table 7-425: ShapeEvaluation element

Name	Data Type	Description
<i>Tolerance</i> = "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.
<i>ValueList</i> ?	ShapeRangeList	A list of ranges of values that can be chosen.
<i>X</i> ?	DoubleRangeList	Allowed X-axis of the Shape .
<i>Y</i> ?	DoubleRangeList	Allowed Y-axis of the Shape .
<i>Z</i> ?	DoubleRangeList	Allowed Z-axis of the Shape .

— Element: **StringEvaluation**

The **StringEvaluation** element declares a string value for comparison in an expression to a **StringState** element in constraints.

Table 7-426: StringEvaluation element

Name	Data Type	Description
<i>Length</i> ?	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").
<i>RegExp</i> ?	regExp	Regular expression that limits the allowed values.
<i>Value</i> *	element	A string value for comparison in an expression to a StringEvaluation element.

— Element: StringEvaluation/Value

Table 7-427: StringEvaluation/Value element

Name	Data Type	Description
<i>Value</i>	string	A supported value for a string attribute.

— Element: XYPairEvaluation

The XYPairEvaluation element declares a XYPair value for comparison in an expression to a XYPairState element in constraints.

Table 7-428: XYPairEvaluation element

Name	Data Type	Description
<i>Tolerance</i> = "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.
<i>ValueList</i> ?	XYPairRangeList	A list of values that can be chosen.
<i>XYRelation</i> ?	XYRelation	Relative value of X vs. Y.

7.3.13.3 TestRef

The TestRef element refers to another constraint that is to be evaluated as part of the parent constraint.

Table 7-429: TestRef element

Name	Data Type	Description
<i>rRef</i>	IDREF	Reference to a Test to be evaluated as a nested Boolean expression inside a larger expression.

7.3.14 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.

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).

Device Query:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_2" Version="1.3"
  TimeStamp="2005-04-05T16:45:43+02:00" SenderID="Controller">
  <Query ID="DeviceQuery" Type="KnownDevices">
    <DeviceFilter DeviceDetails="Capability" Localization="fre"/>
  </Query>
</JMF>
```

Device Response:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_2" SenderID="Scanner" TimeStamp="2005-06-
05T16:45:43+02:00" Version="1.3">
  <Response ID="xyz" Type="KnownDevices" refID="DeviceQuery">
    <DeviceList>
      <DeviceInfo>
        <Device Class="Implementation" DeviceID="Joe the Drum" ID="IDXYZ"
          KnownLocalizations="En Fre" ModelName="Bongo" Status="Available">
```

```

<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="DCTEncode">
      <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="DCTEncode">
      <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>

```

Device Description of a Scanner #2

This second example illustrates the use of constraints, macros and DisplayGroups in a capability response. For the sake of simplicity, the only localizations returned are for the constraints.

Device Query:

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Controller"
    TimeStamp="2005-04-05T16:45:43+02:00" Version="1.3">
    <Query ID="DeviceQuery" Type="KnownDevices">
        <DeviceFilter DeviceDetails="Capability" Localization="en"/>
    </Query>
</JMF>

```

Device Response:

```

<JMF SenderID="Scanner" TimeStamp="2004-10-17T14:30:47Z" xmlns="http://www.CIP4.org/
JDFSchema_1_1" Version="1.3" DescriptiveName="Example from JDF 1.2 Spec Document">
    <Response ID="xyz" Type="KnownDevices" refID="DeviceQuery" ReturnCode="0"
        Acknowledged="false">
    <DeviceList>
        <DeviceInfo DeviceStatus="Idle">
            <Device DeviceID="Joe the Drum" ID="IDXYZ" 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_2" 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="islbit">
            <IntegerEvaluation ValueList="1" rRef="btd" />
        </Test>
        <Test ID="BD-bw">
            <and>
                <TestRef rRef="islbit" />
                <EnumerationEvaluation ValueList="FlateEncode DCTEncode" rRef="cmp" />
            </and>
        </Test>
        <Test ID="ctcmp">
            <and>
                <not>
                    <TestRef rRef="islbit" />
                </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_2" Availability="Installed" Name="NodeInfo" ResourceUpdate="None">
  <DevCap MinOccurs="1" Name="NodeInfo" DevNS="http://www.CIP4.org/
JDFSchema_1_2" MaxOccurs="1">
    <StringState UserDisplay="Display" DevNS="http://www.CIP4.org/
JDFSchema_1_2" Editable="true" MinOccurs="1" MaxOccurs="1" Name="TargetRoute"
HasDefault="true" ListType="SingleValue" />
    <IntegerState Name="JobPriority" DevNS="http://www.CIP4.org/
JDFSchema_1_2" 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_2">
  <DevCap MinOccurs="1" Name="ExposedMedia" DevNS="http://www.CIP4.org/
JDFSchema_1_2" MaxOccurs="1">
    <DevCap MinOccurs="1" Name="Media" DevNS="http://www.CIP4.org/
JDFSchema_1_2" MaxOccurs="1">

```

```

        <NameState MinOccurs="1" DefaultValue="Sheet" UserDisplay="Display"
        Editable="true" ListType="SingleValue" HasDefault="true" Name="MediaUnit" DevNS="http://
        /www.CIP4.org/JDFSchema_1_2" MaxOccurs="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_2"
        MaxOccurs="1" />
    </DevCap>
    </DevCap>
    </DevCaps>
    <DevCaps Required="false" Context="Resource" DevNS="http://www.CIP4.org/
    JDFSchema_1_2" Availability="Installed" Name="ScanParams" ResourceUpdate="None">
        <DevCap MinOccurs="1" Name="ScanParams" DevNS="http://www.CIP4.org/
        JDFSchema_1_2" 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" DevNS="http://www.CIP4.org/
            JDFSchema_1_2" MaxOccurs="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" DevNS="http://www.CIP4.org/JDFSchema_1_2" MaxOccurs="1" />
            <NumberState MinOccurs="1" UserDisplay="Display" Editable="true"
            ID="mag" ListType="SingleValue" HasDefault="true" AllowedValueMax="100.0"
            AllowedValueMin="0.01" DevNS="http://www.CIP4.org/JDFSchema_1_2" MaxOccurs="1"
            Name="Magnification" />
            <EnumerationState ActionRefs="cd" MinOccurs="1"
            AllowedValueList="GrayScale CMYK RGB LAB" UserDisplay="Display" Editable="true"
            ID="colorspace" ListType="SingleValue" HasDefault="true" Name="OutputColorSpace"
            DevNS="http://www.CIP4.org/JDFSchema_1_2" MaxOccurs="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" DevNS="http://www.CIP4.org/JDFSchema_1_2"
            MaxOccurs="1" />
        </DevCap>
    </DevCaps>
    </DeviceCap>
    </Device>
    </DeviceInfo>
    </DeviceList>
    </Response>
</JMF>

```

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" Version="1.3">
    <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>
    </ResourcePool>
    <ResourceLinkPool>
        <ScanParamsLink Usage="Input" rRef="Link0007"/>
        <ExposedMediaLink Usage="Input" rRef="Link0008"/>
    </ResourceLinkPool>

```

```
</JDF>
```

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" Version="1.3">
  <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>
  </ResourcePool>
  <ResourceLinkPool>
    <ScanParamsLink Usage="Input" rRef="Link0012"/>
    <ExposedMediaLink Usage="Input" rRef="Link0013"/>
  </ResourceLinkPool>
</JDF>
```

7.4 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 "DeviceCap" on page 613.

In order to define one **Test**, you can combine one or more basic tests using the Boolean logic as defined in "Device Capability Definitions" on page 613. 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.

7.4.1 Object Classes

The following is the list of the real objects that can be preflighted in a document:

Table 7-430: Object Classes for a document

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).
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.

In the following table, you can see the list of object classes with the properties set that they implement.

Table 7-431: Properties Implemented by Class

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					
PDL Object														X

7.4.1.1 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.

This example checks if the *TrappedKey* is defined in a PDF document.

```
<Test ID="PT01">
  <IsPresentEvaluation>
    <BasicPreflightTest Name="TrappedKey"/>
  </IsPresentEvaluation>
</Test>
```

This example checks if the value of the *TrappedKey* = “Unknown” in a PDF document.

```
<Test ID="PT02">
  <EnumerationEvaluation ValueList="Unknown">
    <BasicPreflightTest Name="TrappedKey"/>
  </EnumerationEvaluation>
```

</Test>

Table 7-432: Mapping between property types (in the preflight spec) and evaluations

Property Type	Evaluation	Expected usage for BasicPreflightTest ListType
presence	IsPresentEvaluation	-
boolean	BooleanEvaluation	SingleValue.
BooleanList	BooleanEvaluation	Any of <i>ListType</i> 's value that refers to a list.
DateTime	DateTimeEvaluation	SingleValue.
DateTimeList	DateTimeEvaluation	Any of <i>ListType</i> 's value that refers to a list.
enumeration	NameEvaluation	SingleValue.
enumerations	NameEvaluation	Any of <i>ListType</i> 's value that refers to a list.
integer	IntegerEvaluation	SingleValue.
IntegerList	IntegerEvaluation	Any of <i>ListType</i> 's value that refers to a list.
Name	NameEvaluation	SingleValue.
NameList	NameEvaluation	Any of <i>ListType</i> 's value that refers to a list.
double	NumberEvaluation	SingleValue.
DoubleList	NumberEvaluation	Any of <i>ListType</i> 's value that refers to a list.
rectangle	RectangleEvaluation	SingleValue.
RectangleList	RectangleEvaluation	Any of <i>ListType</i> 's value that refers to a list.
string	StringEvaluation	SingleValue.
StringList	StringEvaluation	Any of <i>ListType</i> 's value that refers to a list.
XYPair	XYPairEvaluation	SingleValue.
XYPairList	XYPairEvaluation	Any of <i>ListType</i> 's value that refers to a list.

7.4.1.2 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.

7.4.2 Properties

In the following pages, a full list of properties (and attribute definitions) is defined, that can be found, extracted, and evaluated from a document. The properties are grouped by classes (see Table 7-430, “Object Classes for a document,” on page 660.)

7.4.2.1 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 7-433: Annotation properties

Name	Type	Description	Set	Tag	Documents
<i>AnnotationPrintFlag</i>	boolean	Is “ <i>true</i> ” when it will be printed on the final document.	—	—	PDF
<i>AnnotationType</i>	NMTOKEN	The type of annotations. See Table 7-434 for values of <i>AnnotationType</i> .	—	—	PDF
<i>TrapnetAnnotationPDFX</i>	NMTOKENS	The PDF/X versions to which the <i>TrapNet</i> annotation complies, (e.g., “ <i>PDF/X-1a:2003</i> ”).	—	—	PDF

— Attribute: AnnotationType

Table 7-434: AnnotationType attribute – possible values

Value	Description	Value	Description
<i>Circle</i>		<i>Sound</i>	
<i>FileAttachment</i>		<i>Square</i>	
<i>FreeText</i>		<i>Squiggly</i>	
<i>Highlight</i>		<i>Stamp</i>	
<i>Ink</i>		<i>StrikeOut</i>	
<i>Link</i>		<i>Text</i>	
<i>Line</i>		<i>TrapNet</i>	
<i>Movie</i>		<i>Underline</i>	
<i>Popup</i>		<i>Widget</i>	
<i>PrinterMark</i>			

— Attribute:

7.4.2.2 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. The allowed types of boxes are listed in the following table. Note that *BOX* is an attribute of the *BoxArgument* and the following box types are possible values of the *BOX* attribute.

Table 7-435: Box properties and Box attribute – possible values

Box Type	Description
<i>ArtBox</i>	Defines the extent of the page's meaningful content (including potential white space) as intended by the page's creator.
<i>BleedBox</i>	Defines the region to which the contents of the page is to 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.
<i>CropBox</i>	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.
<i>MarginsBox</i>	Defines the trim box minus the margins.
<i>MediaBox</i>	Defines the boundaries of the physical medium on which the page is to 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.
<i>SlugBox</i>	Defines an area where document related information and objects that will not be on the final document could be printed.
<i>TrimBox</i>	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.

The following are other property attributes related to boxes:

Table 7-436: Box properties – other ones

Name	Type	Description	Set	Tag	Documents
<i>BoundingBox</i>	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	—	—
<i>DifferentBoxSize</i>	enumerations	This is the list of boxes, which are different on one page from the same boxes on another page. Refer to Table 7-435 on page 663 for a list of valid types of boxes.	Only	—	—
<i>InsideBox</i>	boolean	Is "true" when an object is inside a given box. <i>InsideBox</i> MUST be qualified by <i>BoxArgument</i> subelement.	—	—	—
<i>OutsideBox</i>	boolean	Is "true" when an object is outside a given box. <i>OutsideBox</i> MUST be qualified by <i>BoxArgument</i> subelement.	—	—	—

Example:

The following is an example of *Test* using *InsideBox* and a *BoxArgument* subelement:

```
<Test ID="PT01">
  <BooleanEvaluation ValueList="true">
    <BasicPreflightTest Name="InsideBox">
      <PreflightArgument>
        <BoxArgument Box="TrimBox" Overlap="true"/>
      </PreflightArgument>
    </BasicPreflightTest>
  </BooleanEvaluation>
</Test>
```

7.4.2.3 Class Properties

Each object can define the name of the class of objects it belongs to:

Table 7-437: Class properties

Name	Type	Description	Set	Tag	Documents
<i>ClassName</i>	NMTOKEN	The name of the class to which the object belongs. See Table 7-438 for values of <i>ClassName</i> .	—	—	—
<i>PropertyList</i>	enumerations	The list of properties the object has. See Table 7-439 for values of <i>PropertyList</i>	—	—	—

— Attribute: ClassName**Table 7-438: ClassName attribute – possible values**

Value	Description	Value	Description
<i>Annotation</i>		<i>MaskUsingVector</i>	
<i>Document</i>		<i>Page</i>	
<i>Font</i>		<i>PageBox</i>	
<i>Image</i>		<i>PDL</i>	
<i>ImageMask</i>		<i>Shading</i>	
<i>MaskUsingImage</i>		<i>Text</i>	
<i>MaskUsingText</i>		<i>Vector</i>	

— Attribute: PropertyList**Table 7-439: PropertyList attribute – possible values**

Value	Description	Value	Description
<i>Annotation</i>		<i>Logical</i>	
<i>Box</i>		<i>Page</i>	
<i>Class</i>		<i>PageBox</i>	
<i>Colorant</i>		<i>PDLObject</i>	
<i>Document</i>		<i>Reference</i>	
<i>Fill</i>		<i>Shading</i>	
<i>Font</i>		<i>Stroke</i>	
<i>Graphic</i>		<i>Text</i>	
<i>Image</i>		<i>Vector</i>	

7.4.2.4 Colorant Properties

Every visible object or group of objects will imply a given number of separations.

Table 7-440: Colorant properties

Name	Type	Description	Set	Tag	Documents
<i>AliasSeparations</i>	boolean	Is " <i>true</i> " when some of the separations have different names but the same color values.	Yes	—	—
<i>AmbiguousSeparations</i>	boolean	Is " <i>true</i> " when some of the separations have the same name but different color values.	Yes	—	—
<i>InkCoverage</i>	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	—	—
<i>SeparationList</i>	string	List of all separations necessary to print one object or a group of objects.	Yes	—	—

7.4.2.5 Document Properties

This is the list of properties (attributes) that define parts of a document.

Table 7-441: Document properties (Section 1 of 4)

Name	Type	Description	Set	Tag	Documents
<i>Author</i>	string	A string describing the author of the document.	—	Yes	—

Table 7-441: Document properties (Section 2 of 4)

Name	Type	Description	Set	Tag	Documents
<i>Binding</i>	enumeration	The binding of the document, whose value can be either: <i>Left</i> <i>Right</i>	—	Yes	PDF
<i>CreationDate</i>	dateTime	The date when the document was created according to the file system.	—	—	—
<i>CreationDateInDocument</i>	dateTime	The date when the document was created according to data inside the document.	—	Yes	—
<i>CreationID</i>	NMTOKEN	An NMTOKEN which can uniquely identify a document when created. In case of a PDF, it matches exactly the first element of ID array.	—	Yes	—
<i>Creator</i>	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	—
<i>DocumentCompression</i>	enumerations	A list of all compression types used in the document (including image compression referenced by <i>CompressionTypes</i> in Image properties). See <i>CompressionTypes</i> for possible values.	—	—	—
<i>DocumentCorruption</i>	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. Possible values include: <i>InvalidOffsets</i> – 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)	—	—	—
<i>DocumentEncoding</i>	enumeration	The document encoding which can be either: <i>ASCII</i> <i>Binary</i>	—	—	PS, PDF
<i>DocumentIsGoodCompression</i>	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.	—	—	—
<i>EncryptedDocument</i>	boolean	Is "true" if document is encrypted.	—	—	—

Table 7-441: Document properties (Section 3 of 4)

Name	Type	Description	Set	Tag	Documents
<i>EncryptionFilter</i>	NMTOKEN	The Filter name of encryption for a PDF file.	—	Yes	PDF
<i>EncryptionLength</i>	integer	The length of the encryption key of a PDF file in bits.	—	Yes	PDF
<i>EncryptionRestrictions</i>	NMTOKENS	The actions that are forbidden by the encryption. Possible values are: <i>Assembly</i> – Inserting or removing pages. <i>Copying</i> – Extracting part of the content. <i>DisabledAccess</i> – Allowing copying specifically for providing access to the disabled. <i>EditingAnnotations</i> <i>EditingContent</i> <i>FillingIn</i> – Filling in forms. <i>HighResPrinting</i> <i>Printing</i>	—	—	PDF
<i>EncryptionSubFilter</i>	NMTOKEN	The SubFilter name of encryption for a PDF file.	—	Yes	PDF
<i>EncryptionV</i>	integer	The V integer of encryption for a PDF file.	—	Yes	PDF
<i>FileName</i>	string	The file name, including file extension, in the file system. This is not the full path.	—	—	—
<i>FileSize</i>	integer	The file size expressed in bytes.	—	—	—
<i>Keywords</i>	string	A string made of keywords describing the document.	—	Yes	—
<i>Linearized</i>	boolean	Is "true" if the document is linearized, (i.e., prepared for web download).	—	—	PDF
<i>ModificationDate</i>	dateTime	The date when the document was last modified according to the file system.	—	—	—
<i>ModificationDateInDocument</i>	dateTime	The date when the document was last modified according to data inside the document.	—	Yes	—
<i>ModificationID</i>	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	—
<i>NumberOfPages</i>	integer	The number of finished pages contained in the document.	—	—	—

Table 7-441: Document properties (Section 4 of 4)

Name	Type	Description	Set	Tag	Documents
<i>OutputIntentColorSpace</i> = "None"	NMTOKEN	The color space belonging to the output intent of the document. Possible values include: <i>None</i> – The default value to be used if this property is not present. <i>CMYK</i> <i>Gray</i> <i>RGB</i>	—	Yes	PDF
<i>OutputIntentStandard</i>	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.	—	—	—
<i>PagesHaveSameOrientation</i>	boolean	Is "true" when all pages have the same orientation.	—	—	—
<i>PDFXVersion</i>	NMTOKEN	The PDF/X version key present in the document.	—	Yes	PDF
<i>PDLType</i>	NMTOKEN	The type of document using MIME-type. <i>PDLType</i> is "application/pdf" for instance. See Table H-1 on page 729 and Table H-2 on page 732 for examples.	—	—	—
<i>PDLVersion</i>	string	The version of document according to the <i>PDLType</i> . See Table H-1 on page 729 and Table H-2 on page 732 for examples.	—	—	—
<i>Producer</i>	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	—
<i>SeparationFlag</i>	boolean	Is "true" if the document is made of separations or is not composite.	—	—	PS, PDF
<i>Subject</i>	string	A string describing the subject of the document.	—	Yes	—
<i>Title</i>	string	A string describing the title of the document.	—	Yes	—
<i>TrappedKey</i>	enumeration	A value explaining the use of trapping on the document. The values can be "true", "false" or "Unknown". It matches exactly the <i>TrappedKey</i> information of PDF.	—	Yes	—

7.4.2.6 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 7-442: Fill properties

Name	Type	Description	Set	Tag	Documents
<i>FillColorName</i>	string	The name of the color of the fill of the vector object.	—	—	—
<i>FillColorType</i>	enumeration	This is an enumeration of known colors to draw fill. See Table 7-443 for values of <i>FillColorType</i> .	—	—	—
<i>HasFillColor</i>	boolean	Is <i>"true"</i> if the vector object is drawn with a fill color.	—	—	—

— Attribute: FillColorType**Table 7-443: FillColorType attribute – possible values**

Value	Description
<i>CMYGray</i>	Will print with the same percentage 0-100% exclusive on Cyan, Magenta and Yellow separations.
<i>CMYBlack</i>	Will print with 100% on Cyan, Magenta and Yellow separations and less than 100% on the Black separation.
<i>Other</i>	Any other combinations of separations.
<i>PureBlack</i>	Will print as 100% on the black separation with 0% on the other separation(s).
<i>PureGray</i>	Will print as 1-99% on the black separation with 0% on the other separation(s).
<i>RegistrationBlack</i>	Will print as 100% on all the separations.
<i>RegistrationGray</i>	Will print as 0-100% exclusive on all the separations (assuming all the separations use the same value).
<i>RichBlack</i>	Will print as 100% on the black separation with more than 0% on one or more of the other separations.
<i>White</i>	Will print as 0% on all the separations.

7.4.2.7 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 7-444: Font properties (Section 1 of 2)

Name	Type	Description	Set	Tag	Documents
<i>EmbeddingRestrictionFlag</i>	boolean	Is <i>"true"</i> if a font cannot be embedded.	—	—	—
<i>FontCorrupted</i>	boolean	Is <i>"true"</i> if a font is corrupted or invalid. The implementation of this check MAY vary from one application to another.	—	—	—
<i>FontCreator</i>	string	The font creator.	—	—	—
<i>FontEmbedded</i>	boolean	Is <i>"true"</i> if a font is embedded into the document.	—	—	—
<i>FontIsStandardLatin</i>	boolean	Is <i>"true"</i> when all characters belong to the standard Latin character set.	—	—	—
<i>FontName</i>	string	The font name.	—	—	—

Table 7-444: Font properties (Section 2 of 2)

Name	Type	Description	Set	Tag	Documents
<i>FontNotUsed</i>	boolean	Is <i>"true"</i> if a font is not used to draw characters from the document.	—	—	—
<i>FontSubset</i>	boolean	Is <i>"true"</i> if a font is only a subset of a main font.	—	—	PS, PDF
<i>FontType = "Other"</i>	enumeration	This is the type of the font. Possible values referencing standardized types of fonts are specified in Table 7-445:	—	—	—
<i>FontVendor</i>	string	The font vendor.	—	—	—
<i>IsFontScreenOnly</i>	boolean	Is <i>"true"</i> if a font referenced in the document contains only screen description.	—	—	Authoring
<i>PSFontName</i>	NMTOKEN	The PostScript font name.	—	—	PS, PDF

— Attribute: FontType**Table 7-445: FontType attribute – possible values**

Value	Description	Value	Description
<i>CIDFontType0</i>		<i>Type1</i>	
<i>CIDFontType1</i>		<i>Type1CMultipleMaster</i>	
<i>CIDFontType2</i>		<i>Type2C</i>	
<i>CIDFontType3</i>		<i>Type3</i>	
<i>CIDFontType4</i>		<i>PDFType3</i>	
<i>OpenType</i>		<i>Type42</i>	Embedded TrueType into a PostScript font.
<i>TrueType</i>		<i>Unknown</i>	Type of font that can not be resolved for any reason, (i.e., missing font, etc.).
<i>Type0</i>	PostScript Type0 without the CID	<i>Other</i>	To be used when the property is not any of the values listed above.

7.4.2.8 Graphic Properties

This is a list of property attributes that specifically apply to objects that can be displayed or printed.

Table 7-446: Graphic properties (Section 1 of 3)

Name	Type	Description	Set	Tag	Documents
<i>AlphaIsShape</i>	boolean	The <i>AlphaIsShape</i> of a PS or PDF object.	—	—	PS, PDF
<i>AlternateColorSpace</i>	enumeration	The alternate color space of the object is one of the given. Values are identical to those defined in <i>ColorSpace</i> , below.	—	Yes	PS, PDF
<i>BelongsToAnnotation</i>	boolean	Is <i>"true"</i> when this object belongs to an annotation.	—	—	—

Table 7-446: Graphic properties (Section 2 of 3)

Name	Type	Description	Set	Tag	Documents
<i>BlackGeneration</i>	enumeration	The <i>BlackGeneration</i> function of a PS or PDF object. Values include: <i>Identity</i> – Defines identity function. <i>Custom</i> – Used when the function is described.	—	Yes	PS, PDF
<i>BlendMode</i>	NMTOKEN	The <i>BlendMode</i> of a PS or PDF object.	—	—	PS, PDF
<i>ColorSpace</i>	enumeration	The color space of the object is one of the following values: <i>CalGray</i> <i>CalRGB</i> <i>CIEBasedA</i> <i>CIEBasedABC</i> <i>CIEBasedDEF</i> <i>DeviceCMYK</i> <i>DeviceGray</i> <i>DeviceN</i> <i>DeviceRGB</i> <i>ICCBased</i> <i>Lab</i> <i>Separation</i>	—	—	PS, PDF
<i>EmbeddedPS</i>	boolean	Is "true" if a PDF object uses PostScript to be drawn.	—	—	PDF
<i>Flatness</i>	double	A number giving the value of PS or PDF <i>Flatness</i> .	—	Yes	PS, PDF
<i>HasSoftMask</i>	boolean	Is "true" when the object is using a soft-mask using pixel values.	—	—	—
<i>HalfTone</i>	NMTOKEN	The value of the Halftone used in a document: "Named", "1", "5", "6", "10", "16".	—	Yes	PS, PDF
<i>HalfTonePhase</i>	XYPair	The value of the <i>HalfTonePhase</i> associated with the object.	—	Yes	PS, PDF
<i>HasColorLUT</i>	boolean	Is "true" when an object is using indexed colors in a table to describe color.	—	—	—
<i>NumberOfColorsInLUT</i>	integer	The number of colors in the color table used to display an indexed image.	—	—	—
<i>OverPrintFlag</i>	boolean	Is "true" when one object has been set to overprint.	—	—	—
<i>OverPrintMode</i>	integer	An integer giving the PostScript or PDF value for overprint mode.	—	—	PS, PDF
<i>RenderingIntent</i>	NMTOKEN	The rendering intent of a PS or PDF object.	—	Yes	PS, PDF
<i>Smoothness</i>	double	A number giving the value of PS or PDF <i>Smoothness</i> .	—	Yes	PS, PDF

Table 7-446: Graphic properties (Section 3 of 3)

Name	Type	Description	Set	Tag	Documents
<i>TransferFunction</i>	enumeration	The transfer function of a PS or PDF object. Values include: <i>Custom</i> – Used when the function is described. <i>Identity</i> – Defines identity function.	—	Yes	PS, PDF
<i>TransparencyFlag</i>	boolean	Is " <i>true</i> " when the object has transparency. A transparency that is null has the " <i>false</i> " value.	—	—	—
<i>UnderColorRemoval</i>	enumeration	The <i>UnderColorRemoval</i> function of a PS or PDF object. Values include: <i>Custom</i> – Used when the function is described. <i>Identity</i> – Defines identity function.	Yes	Yes	PS, PDF

7.4.2.9 Image Properties

This group of property attributes is very specific to images displayed using pixels:

Table 7-447: Image properties (Section 1 of 2)

Name	Type	Description	Set	Tag	Documents
<i>AlternateImages</i>	NMTO-KENS	When to draw some of the alternate images that correspond with the given image. The PDF specification defines "Print" as a possible value, but any other application-specific value could be used.	—	Yes	PDF
<i>BitsPerSample</i>	integer	The number of bits used to represent color on every separation.	—	—	—
<i>CompressionRatio</i>	double	For all compression types to which it makes sense, the tests apply to the quality expressed as percentage of compression.	—	—	—
<i>CompressionTypes</i>	enumerations	The type of method used to compress or encode the image. Values include: <i>ASCII85</i> <i>ASCIIHex</i> <i>CCITT</i> <i>JBIG2</i> <i>JPEG</i> <i>JPEG2000</i> <i>LZW</i> <i>None</i> <i>RunLength</i> <i>ZIP</i> Where <i>JPEG</i> , <i>JPEG2000</i> and/or <i>JBIG2</i> are specified, they can be concatenated and only <i>JPEG</i> need be used.	—	—	—

Table 7-447: Image properties (Section 2 of 2)

Name	Type	Description	Set	Tag	Documents
<i>EffectiveResolution</i>	XYPair	The horizontal and vertical resolutions of the scaled image, in dots per inch.	—	—	—
<i>EstimatedJPEGQuality</i>	integer	For <i>JPEG</i> 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.	—	—	—
<i>ImageFlipped</i>	enumeration	The way the image is flipped. Possible values include: <i>None</i> <i>Horizontal</i> <i>Vertical</i>	—	—	—
<i>ImageMaskType</i>	enumeration	The type of masks used by image. The allowed values include: <i>NoMask</i> – Used when the image does not use specific mask. <i>BitmapMask</i> – Used when the image is masked using a bitmap image <i>ColorKeyMask</i> – Used when some colors are masked out to display the image (such like video chroma-key).	—	—	—
<i>ImageRotation</i>	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° rotation is valid, (e.g., one full rotation + 180° rotation).	—	—	—
<i>ImageScalingRatio</i>	double	The ratio between X and Y scaling of an image.	—	—	—
<i>ImageSkew</i>	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.	—	—	—
<i>OriginalResolution</i>	XYPair	The horizontal and vertical resolutions of the image before scaling.	—	—	—
<i>PixelHeight</i>	integer	Image height in pixels.	—	—	—
<i>PixelWidth</i>	integer	Image width in pixels.	—	—	—

The JPEG quality algorithm is based on a technique used by the IJG library (<http://www.iijg.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

```

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.

7.4.2.10 Logical Properties

The logical properties are mainly used with **Set** to count the number of objects.

Table 7-448: Logical properties

Name	Type	Description	Set	Tag	Documents
<i>Count</i>	Integer	The number of objects contained in the referenced set of objects.	Yes	—	—

7.4.2.11 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 7-449: PageBox properties

Name	Type	Description	Set	Tag	Documents
<i>PageBoxType</i>	enumeration	Refer to "Box Properties" on page 663 for a list of the valid types of boxes. When not known, the default is to leave <i>PageBoxType</i> empty.	—	—	—

7.4.2.12 Pages Properties

This is the list of elements and attributes related to the page object in a document.

Table 7-450: Pages properties (Section 1 of 2)

Name	Type	Description	Set	Tag	Documents
<i>BlankPage</i>	boolean	Is " <i>true</i> " when the trim box and the bleed box area, when defined, do not output any marks.	—	—	—
<i>BlendColorSpace</i>	enumeration	The page blend color space. For the list of possible values, see <i>ColorSpace</i> in Graphics Properties above.	—	Yes	PDF
<i>PageHasUnknownObjects</i>	boolean	Page contains unknown objects but the PDL was set to ignore these errors. Examples are the use of BX/EX in PDF.	—	—	—
<i>PageNumber</i>	integer	The page index in the RunList .	—	—	—

Table 7-450: Pages properties (Section 2 of 2)

Name	Type	Description	Set	Tag	Documents
<i>ReversePageNumber</i>	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.	—	—	—
<i>BoxToBoxDifference</i>	element	The rectangle from calculating the differences between two rectangles: <i>FromBox</i> and <i>ToBox</i> . The calculation is made using the following formula: <i>FromBox</i> (left)– <i>ToBox</i> (left), <i>FromBox</i> (bottom)– <i>ToBox</i> (bottom), <i>ToBox</i> (right)– <i>FromBox</i> (right), <i>ToBox</i> (top)– <i>FromBox</i> (top). To define the two boxes used, options are given in <i>BoxToBoxDifference</i> argument. See Table 7-279, "BoxToBoxDifference element," on page 542	—	—	—

Note that *BoxToBoxDifference* element is always a subelement of a **PreflightArgument**.

Example:

```
<Test ID="PT01">
  <RectangleEvaluation ValueList="0 0 10 10">
    <BasicPreflightTest Name="BoxToBoxDifference">
      <PreflightArgument>
        <BoxToBoxDifference FromBox="TrimBox" ToBox="BleedBox"/>
      </PreflightArgument>
    </BasicPreflightTest>
  </RectangleEvaluation>
</Test>
```

7.4.2.13 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 7-451: PDLObject properties

Name	Type	Description	Set	Tag	Documents
<i>PDLObjectType</i>	NMTOKEN	The type of specific PDL object. Possible values include: <i>AcroForm</i> – The PDF AcroForm. <i>Actions</i> – The PDF Actions. <i>Bookmarks</i> – The PDF Bookmarks. <i>JavaScript</i> – The PDF JavaScript. <i>Thread</i> – The PDF Thread. <i>Thumbnails</i> – The PDF Thumbnails.	—	—	PDF

7.4.2.14 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 "Font Properties" on page 669).

Table 7-452: Reference properties

Name	Type	Description	Set	Tag	Documents
<i>ExternalReferenceMissing</i>	boolean	Is " <i>true</i> " when the target of an external reference is missing.	—	—	—
<i>HasExternalReference</i>	boolean	Is " <i>true</i> " when some of the page graphical contents have a link on files.	—	—	—
<i>HasOPI</i>	boolean	Is " <i>true</i> " if there is OPI information associated with the object.	—	—	PS, PDF
<i>OPIMissing</i>	boolean	Is " <i>true</i> " when the target of OPI comments associated with the object is missing.	—	—	PS, PDF
<i>OPIType</i>	NMTOKEN	The OPI type of OPI comments associated with the object. Sometimes in PS, the comments are not OPI comments. Other values include: <i>OPIComments</i> <i>OtherComments</i>	—	—	PS, PDF
<i>OPIVersion</i>	NMTOKENS	The OPI versions of OPI comments associated with the object.	—	—	PS, PDF

7.4.2.15 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 7-453: Shading properties

Name	Type	Description	Set	Tag	Documents
<i>ShadingType</i>	enumeration	The type of shading. Values are: <i>Smooth</i> <i>Vector</i>	—	—	—

7.4.2.16 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 7-454: Stroke properties (Section 1 of 2)

Name	Type	Description	Set	Tag	Documents
<i>HasStrokeColor</i>	boolean	Is " <i>true</i> " if the vector object is drawn with a stroke color.	—	—	—
<i>StrokeAlternateColorSpace</i>	enumeration	The alternate color space of the stroke of one object. For the list of possible values, see <i>ColorSpace</i> in Graphics Properties above.	—	Yes	PS, PDF

Table 7-454: Stroke properties (Section 2 of 2)

Name	Type	Description	Set	Tag	Documents
<i>StrokeColorName</i>	string	The name of the color of the stroke of the vector object.	—	—	—
<i>StrokeColorSpace</i>	enumeration	The color space of the stroke of one object. For the list of possible values, see <i>ColorSpace</i> in Graphics Properties above.	—	—	PS, PDF
<i>StrokeColorType</i>	enumeration	This is an enumeration of known colors used to draw stroke. Refer to <i>FillColorType</i> for the list of possible values.	—	—	—
<i>StrokeOverprintFlag</i>	boolean	Is <i>"true"</i> when the stroke of one object has been set to overprint.	—	—	—
<i>StrokeShadingType</i>	enumeration	The type of shading used in the stroke. Values are: <i>Smooth</i> <i>Vector</i>	—	—	—
<i>StrokeThickness</i>	double	The thickness of the stroke of the vector object.	—	—	—

7.4.2.17 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 7-455: Text properties (Section 1 of 2)

Name	Type	Description	Set	Tag	Documents
<i>CharacterProblem</i>	enumeration	Problem encountered to render character. The possible values are: <i>Corrupted</i> – Used when a character was found but could not be rendered. <i>IncorrectEncoding</i> – Used when encoding information is missing, incomplete or otherwise incorrect. <i>Missing</i> – Use when the character could not be found in font. <i>Others</i> – Used in all other cases.	—	—	—
<i>MissingPrinterFont</i>	boolean	Is <i>"true"</i> if a referenced font has no printer information.	Yes	—	—
<i>MissingScreenFont</i>	boolean	Is <i>"true"</i> if a referenced font has no screen information.	—	—	—
<i>TextSize</i>	double	The size in points of the character.	—	—	—

Table 7-455: Text properties (Section 2 of 2)

Name	Type	Description	Set	Tag	Documents
<i>UseArtificialTextEffect</i>	enumerations	The artificial text effects list used to draw a character. Values include: <i>Bold</i> <i>Italic</i> <i>Outline</i> <i>Shadow</i> <i>Underline</i> The authoring applications can apply the text effect directly, whereas in PS or PDF, the effect will be calculated.	—	—	—

7.4.2.18 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 7-456: Vector properties

Name	Type	Description	Set	Tag	Documents
<i>NumberOfPathPoints</i>	integer	The number of points used to create a vector path.	—	—	—

Chapter 8 Building a System Around JDF

8.1 Implementation Considerations and Guidelines

JDF parsing. JDF devices **MUST** implement JDF parsing. At a minimum, a device **MUST** 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 **MUST** be able to consume the inputs and produce the outputs for each process type it is able to execute. See “Determining Executable Nodes” on page 123.

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.

8.2 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. In version JDF 1.2 and above, the restrictions on transport layer have been loosened.

8.2.1 File-Based Protocol (JDF + JMF)

The file-based protocol is a solution for JDF job tickets and JMF messages. A file-based protocol can be based on hot folders. A Device that implements hot holders **MUST** 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.

8.2.1.1 JMF Transport Using The File Protocol

[New in JDF 1.2](#)

In order to allow JMF messaging based on a file protocol, a set of additional conventions **MUST** be defined. There are some important differences between http and file-based protocols that **MUST** be taken into account:

- HTTP provides a URL to which the sender sends the file, while with the file protocol, the sender provides the URL to the receiver that specifies a location that is accessible to the receiver.
- HTTP is a synchronous protocol that ensures an immediate response, whereas the file protocol is asynchronous. Therefore, an application **MUST** either poll for responses or react to operating system events that signal the existence of the response file.
- HTTP provides a method for detecting that an incoming request is complete. Access to the file from the reading and writing application **MUST** be synchronized, so the reader does not read an incomplete file that is still being written.
- When the receiving end of an HTTP connection is unavailable, the sender is immediately aware that it is unable to connect. In case of a file, the file will simply be orphaned and the sender **MUST** check whether the file has been retrieved by the receiver.
- With HTTP the sender pushes the request. With the file protocol if the sender writes the file, the sender **MUST** ensure that the path to the file does not clash with some other sender’s file, including any directories that the sender might have to create.
- HTTP connections are transient. Files **MUST** be removed by the receiver after reading them, depending on the supplied **FileSpec/Disposition**.
- The response to an HTTP command is received on the same connection, whereas the response to a file query **MUST** be placed into a new file. Therefore the expected location of the response file **MUST** be specified by the application that generates the query.

- An HTTP socket can accept multiple Acknowledge messages on the same socket in sequence. Multiple Acknowledges as files MUST follow a unique naming scheme in order to avoid overwriting existing Acknowledge files.

8.2.2 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.

8.2.2.1 Protocol Implementation Details

JDF Messaging does not specify a standard port.

Implementation of Messages

Only HTTP servers MUST be targeted by Query or Command messages. This is done with a standard HTTP Post request. The JMF is the body of the HTTP post message. The Response 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 is 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 Java applet that polls the server in regular intervals can be used.

8.2.3 HTTPS-Based Protocol – SSL with two-way authentication

[New in JDF 1.3](#)

8.2.3.1 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 necessary in an individual installation).

Support for JMF over HTTPS does not require the implementation of any additional JMF messages. Certificates are assumed to be transported over proprietary channels.

8.3 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 are 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 MUST be at least one JDF or JMF part. If a JMF part is included there MUST be only one. If the packaging has ordered parts

(Multipart/Related) the JMF part **MUST** be first. The JDF parts **MUST** 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) **MUST** be updated with the URL at which the referenced items are stored.

8.3.1 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.

8.3.2 MIME Types and File Extensions

The MIME type for JDF is not yet registered with IANA <http://www.iana.org/>. The registration process is ongoing and the MIME types will be registered as:

JDF — application/vnd.cip4-jdf+xml

JMF — application/vnd.cip4-jmf+xml

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.

8.3.2.1 MIME Headers

[New in JDF 1.2](#)

This section defines the normative extensions when using MIME to package JMF or JDF.

8.3.2.1.1 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 8-1: MIME Content-Types

MIME Type	Description
application/vnd.cip4-jdf+xml	A JDF File. The root XML element MUST be JDF.
application/vnd.cip4-jmf+xml	A JMF File. The root XML element MUST 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 MUST be JDF or JMF.

8.3.2.1.2 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 **MUST** be an Email address as long as it is defined using US-ASCII. Each value of *Content-ID* **MUST** 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.

8.3.2.1.3 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 MUST 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.

8.3.2.1.4 Content-Disposition Header

This field is OPTIONAL. See [RFC2231] It allows a filename to be specified for a body part. The *Disposition-Type* MUST 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;
```

8.3.2.2 Example 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--
```

8.3.2.3 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:

```

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 "&") MUST be hex-encoded using the %hh escape mechanism in the URL (see [RFC3986]). Therefore, matching the cid URL with the Content-ID MUST take account of the escaped equivalencies. Case-insensitive matching MUST be used.

8.3.2.4 Ordering of Body Parts in MIME Multipart/Related[New in JDF 1.2](#)

The first body part of the MIME Multipart message MUST 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: 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
...
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFCClient"
TimeStamp="2005-07-07T13:15:56+01:00" Version="1.3">
  <Command ID="C0001" Type="SubmitQueueEntry">
    <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" standalone="no" ?>
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" Activation="Active" ID="JDF_c"
JobID="Geef62b72-0f6e-4195-a412-aaa3123d200b" Status="Waiting" Type="Product"
Version="1.3">
  <ResourcePool>
    <RunList Class="Parameter" DocCopies="1" FirstPage="0" ID="RunList4" IsPage="true"
      NDoc="1" PageCopies="1" Status="Available">
      <LayoutElement ElementType="Document" HasBleeds="false" ID="LayoutElement_1"
        IgnorePDLCopies="true" IgnorePDLImposition="true" IsPrintable="true">
        <FileSpec AppOS="Windows" Compression="None" Disposition="Retain"
          ID="FileSpec_9" 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.

8.4 MIS Requirements

MIS systems MAY:

- Ignore Audit elements when they receive complete information about a process execution via JMF.
- Decompose JDF into an internal format such as database tables.

8.5 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 JMF 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 Encoding

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 *XList*, *XRange* and *XRangeList*, 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 Values

A.2.1 boolean

Has the value space for supporting the mathematical concept of binary-valued logic:

Encoding

boolean attributes 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

```
<Example Enable="true"/>
```

A.2.2 CMYKColor

XML attributes of type CMYKColor are used to specify CMYK colors.

Encoding

CMYKColor attributes are primitive data types and are encoded as a string 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:

```
<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

```
<Example StartDate="1999-05-31"/>
```

A.2.4 dateTime

Represents a specific instant of time. It MUST be a Coordinated Universal Time (UTC) or the time zone MUST be indicated by the offset to UTC. In other words, the time MUST 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 MUST be finished before ‘anytime’.

Encoding

It is represented as a union of the XML Schema type: *dateTime* and the infinity value tokens *INF* and *-INF*

Example

```
<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

```
<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 white-space or *dateTime* values.

Example:

```
<XXX RangeList="1999-05-31T18:20:00Z ~ 1999-05-31T18:20:00Z 1999-05-31T13:20:00-05:00 ~
INF"/>
```

A.2.7 double

double Corresponds 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

```
<Example NegativePi="-3.14"/>
```

A.2.8 DoubleList

[New in JDF 1.2](#)

XML attributes 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

```
<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:

```
<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:

```
<XXX list="-1 ~ -6 3.14 ~ 5.13 7 9 ~ 128 131 255 ~ INF"/>
```

A.2.11 duration

Represents a duration 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*

Example

```
<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) character and OPTIONAL additional whitespace. Note: It is now RECOMMENDED that the ‘~’ is surrounded by whitespace to aid validation and parsing.

Examples:

```
<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:

```
<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

```
<Example Month="2002-11"/>
```

A.2.15 hexBinary

Represents arbitrary hex encoded binary data.

Encoding

It is represented identically to the XML Schema type: *hexBinary*

Example

```
<Example Hex="0A1C"/>
```

A.2.16 ID

Represents the ID attribute from [XML]. It represents a name or string that contains no space characters and starts with a letter, ‘:’ or ‘_’. Each ID value MUST be unique within a JDF document and thus uniquely identify the elements that bear them.

Encoding

It is represented identically to the XML Schema type: *ID*

Example

```
<Example ID="R-16"/>
```

A.2.17 IDREF

IDREF Represents the IDREF attribute from [XML]. For a valid XML-document, an element with the ID value specified in IDREF MUST be present in the scope of the document.

Encoding

It is represented identically to the XML Schema type: *IDREF*

Example

```
<Example IDREF="R-16"/>
```

A.2.18 IDREFS

IDREFS Represents the IDREFS attribute from [XML]. More specifically, this is a whitespace-separated list of IDREF values.

Encoding

It is represented identically to the XML Schema type: *IDREFS*

Example

```
<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

```
<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

```
<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.

Examples:

```
<XXX range="-3 ~ -5"/>
<XXX range="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:

```
<XXX list="-1 ~ -6 3 ~ 5 7 9 ~ 128 131"/>
```

A.2.23 LabColor

XML attributes 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:

```
<Color ... Lab="51.9 12.6 -18.9">
```

A.2.24 language

Represents a natural language defined in [RFC1766].

Encoding

It is represented identically to the XML Schema type: *language*

Example

```
<Example Language="de"/> - German
<Example Language="de-CH"/> - Swiss German
<Example Language="en"/> - English
<Example Language="en-GB"/> - British English
```

A.2.25 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

```
<ContentObject CTM="1 0 0 1 3.14 21631.3" />
```

A.2.26 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, alphabetical ordering is implied.
- 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.

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

```
<XXX NameRange="Jack ~ Jill"/>
```

A.2.27 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:

```
<XXX list="A brian ~ fred x z"/>
```

A.2.28 NMTOKEN

Represents the NMTOKEN attribute type from [XML]. It represents a name or string that contains no space characters.

Encoding

It is represented identically to the XML Schema type: *NMTOKEN*

Example

```
<Example Alias="ABC_6"/>
```

A.2.29 NMTOKENS

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

```
<Example AliasList="ABC_6 ABCD_3 DEGF"/>
```

A.2.30 PDFPath

[Modified in JDF 1.3](#)

XML attributes 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:

```
<ElementWithPath path="0 0 m 10 10 l 20 20 l"/>
```

A.2.31 rectangle

XML attributes of type rectangle are used to describe rectangular locations on the page, sheet or other printable surface. A rectangle is represented as an array 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:

```
<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.2.32 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:

```
<XXX range="1 2 3 4 ~ 5 6 7 8"/>
<XXX range="-INF -INF 3 4 ~ 0 1 INF INF"/>
```

A.2.33 RectangleRange List

[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:

```
<XXX RectangleRangeList="1 2 3 4 ~ 5 6 7 8 9 10 11 12 13 14 15 16"/>
```

A.2.34 regExp

Represents a regular expression as defined in [\[XMLSchema\]](#).

Encoding

It is represented identically to the XML Schema type: *normalizeString*

Example

```
<Example expression="Foo({1|2}*)" />
```

A.2.35 shape

XML attributes 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:

```
<XXX Dimensions="10 20 40"/>
```

A.2.36 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 $x1 \leq x \leq x2$ and $y1 \leq y \leq y2$ and $z1 \leq z \leq z2$. 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 **MUST** be \geq 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:

```
<XXX Shaperange="1 2 3 ~ 4 5 6"/>
<XXX Shaperange="1 2 3 ~ 4 INF 6"/>
```

A.2.37 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:

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.2.38 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:

```
<Color sRGB="0.3 0.6 0.8" ... >
```

A.2.39 string

Represents character strings in XML.

Encoding

It is represented identically to the XML Schema type: *normalisedString NB*. This means that tabs, linefeeds and so on are not valid characters.

Example:

```
<Example Name="Test With Space"/>
```

A.2.40 TimeRange

[Deprecated in JDF 1.2](#)

A.2.41 TransferFunction

XML attributes 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 **MUST** be applied:

- 1 If $x \leq$ first stimuli, then the result is the y value of the first xy pair.
- 2 If $x \geq$ 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 **MUST** be explicitly specified and are **NOT** defaulted to “0 0” or “1 1”.

Example:

```
<someElementWithTransferCurve someCurve="0 0 .1 .2 .5 .6 .8 .9 1 1"/>
```

A.2.42 URI

[Modified in JDF 1.3](#)

Short for URI-reference. Represents 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

It is represented identically to the XML Schema type: *anyURI*

Example

```
<Example URI="http://www.w3.org/1999/XMLSchema"/>
```

A.2.43 URL

[Modified in JDF 1.3](#)

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

It is represented identically to the XML Schema type: *anyURI*

Example

```
<Example URL="file://hubble/grün.txt"/>
```

A.2.44 XYPair

XML attributes of type XYPair are used to describe sizes like *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:

```
<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.2.45 XYPairRange

XML attributes of type XYPairRange are used to describe a range of XYPair values. The range “ $x_1\ y_1 \sim x_2\ y_2$ ” describes the area $x_1 \leq x \leq x_2$ and $y_1 \leq y \leq y_2$. Thus the XYPair “2 3” is within “1 2 ~ 3 4”. Note that this implies that both values of the second entry MUST be \geq 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:

```
<XXX XYrange="1 2 ~ 3 4"/>
<XXX XYrange="-INF 2 ~ 3 INF"/>
```

A.2.46 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:

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.2.47 XPath

[New in JDF 1.2](#)

Represents an XPath expression. [XPath]

Encoding

It is represented identically to the XML Schema type: *token*

Example

```
<Example xpath= "JDF/AuditPool/Created/@TimeStamp" />
```

A.3 Enumerations and Lists

A.3.1 enumeration

Represents a closed set of values.

Encoding

It is represented by an enumerated list of values derived from the XML Schema type: *NMTOKEN*

Example

```
<Example Orientation="Flip90"/>
```

A.3.2 enumerations

Represents a list of values taken from a closed set. Values MAY be repeated within the list. If there are any implications to the order of the values this will be detailed in the appropriate items description, otherwise none is implied.

Encoding

It is represented by a whitespace-separated list of enumeration values derived from the XML Schema type: *NMTOKEN*

Example

```
<Example Orientations="Rotate90 Flip90"/>
```

A.3.3 Defined JDF enumeration Data Types

This section is a list of defined enumeration data types. These types are to be used wherever possible for enumerated values and lists of values.

A.3.3.1 JDFJMFVersion

Describes the schema version of a JDF or JMF instance.

Table A-1: JDFJMFVersion enumeration Values

Enumeration Value	Comment
1.1	JDF 1.1
1.2	JDF 1.2
1.3	JDF 1.3

A.3.3.2 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-2: Named Colors

Color name/ Enumeration Value	Comment	Color name/ Enumeration Value	Comment
<i>Black</i>	—	<i>MultiColor</i>	New in JDF 1.1
<i>Blue</i>	—	<i>Mustard</i>	New in JDF 1.1
<i>Brown</i>	—	<i>NoColor</i>	—
<i>Buff</i>	—	<i>Orange</i>	—
<i>Cyan</i>	New in JDF 1.2	<i>Pink</i>	—
<i>Gold</i>	—	<i>Red</i>	—
<i>Goldenrod</i>	—	<i>Silver</i>	—
<i>Gray</i>	—	<i>Turquoise</i>	—
<i>Green</i>	—	<i>Violet</i>	—
<i>Ivory</i>	—	<i>White</i>	—
<i>Magenta</i>	New in JDF 1.2	<i>Yellow</i>	—

A.3.3.3 Orientation

Orientation of a physical resource. For details see Table 2-4, “Matrices and Orientation values for describing the orientation of a Component,” on page 26.

Table A-3: Page Orientation

Enumeration Value	Comment
<i>Rotate0</i>	
<i>Rotate90</i>	
<i>Rotate180</i>	
<i>Rotate270</i>	
<i>Flip0</i>	
<i>Flip90</i>	
<i>Flip180</i>	
<i>Flip270</i>	

A.3.3.4 Side

Describes one side of imageable material.

Table A-4: Side enumeration Values

Enumeration Value	Comment
<i>Front</i>	
<i>Back</i>	

A.3.3.5 WorkStyle

Table A-5: WorkStyle enumeration Values (Section 1 of 2)

Enumeration Value	Comment
<i>Simplex</i>	No turning.

Table A-5: WorkStyle enumeration Values (Section 2 of 2)

Enumeration Value	Comment
<i>Perfecting</i>	Many sheetfed 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 <i>WorkStyle</i> is similar to <i>WorkAndTumble</i> , but <i>Perfecting</i> is an in-line operation during the press run. Therefore, an additional plate (set) is needed during this press run.
<i>WorkAndBack</i>	This <i>WorkStyle</i> 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 <i>WorkAndBack</i> .
<i>WorkAndTurn</i>	<i>WorkAndTurn</i> refers to the turning of the first-run sheet for subsequent perfecting. The front lays remain unchanged but the side lays MUST 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.
<i>WorkAndTumble</i>	The <i>WorkAndTumble</i> 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) is used again in the second press run, imaging the other sheet surface.
<i>WorkAndTwist</i>	Done between two press runs. The palette is twisted 180 degree 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) stay in the machine. This <i>WorkStyle</i> is used for saving plate or film material. It is no longer a common <i>WorkStyle</i> .

A.3.4 XYRelation

[New in JDF 1.2](#)

XML attributes of type XYRelation define the relationship between two ordered numbers.

Table A-6: XYRelation enumeration Values

Enumeration Value	Comment
<i>gt</i>	$X > Y$
<i>ge</i>	$X \geq Y$
<i>eq</i>	$X = Y$
<i>le</i>	$X \leq Y$
<i>lt</i>	$X < Y$
<i>ne</i>	$X \neq Y$

A.4 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.4.1 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 MUST 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 prevalidation of valid JDF-candidates but does not preclude all syntactically invalid files to be validated.

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, Combined and so on.) General definitions with the appropriate name are provided to enable consistent use of `xsi:type`.

Example

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="BackCover" Status="InProgress"
Type="DigitalPrinting" Version="1.3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http:// www.CIP4.org/JDFSchema_1_1 JDF.xsd"
xsi:type="DigitalPrinting">
  <ResourceLinkPool>
    <DeviceLink Usage="Input" rRef="Entire_Book"/>
    <RunListLink Usage="Input" rRef="Entire_Book"/>
  </ResourceLinkPool>
</JDF>
```

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="1.3" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance" xsi:type="jdf:DigitalPrinting">
  <jdf:ResourcePool>
    <jdf:Device Class="Implementation" DeviceID="Unknown Device" ID="Device_001"
Status="Available"/>
    <jdf:RunList Class="Parameter" ID="RunList_001" Status="Unavailable"/>
  </jdf:ResourcePool>
  <jdf:ResourceLinkPool>
    <jdf:DeviceLink Usage="Input" rRef="Device_001"/>
    <jdf:RunListLink Usage="Input" rRef="RunList_001"/>
  </jdf:ResourceLinkPool>
</jdf:JDF>
```

The JDF Schema defines types for JDF Process nodes and JMF Messages. It is RECOMMENDED that these types are used with `xsi:type`.



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.1.2 Using xsi:type with JMF Messages

[New in JDF 1.2](#)

JMF Messages are organized into families — Command, Acknowledge, etc. (See “JMF Message Families” on page 147.)— and each of these families has messages for each message *Type* — Events, KnownControllers, 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.

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="TestSender" TimeStamp="2003-11-07T12:15:56Z" Version="1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance">
  <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>
```

Note JMF messages also do not have to be in the default namespace as in the JDF Node example above.

Appendix C Supported String and NMTOKEN values

C.1 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*. Note that *JDF/@Status* = "Setup", "Cleanup" and "Stopped" can include the description of a device with no job assigned to it.

Table C-1: StatusDetails and Status mapping for generic devices (Section 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	Down	Breakdown of the device, repair needed.
Calibrating	Setup	Setup	The Device is calibrating, either manually or automatically.
ControlDeferred	__a	Stopped	The Machine is not accessible by the Device.
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. <i>Failure</i> has specialized subcategories: <i>PaperJam</i> , <i>DoubleFeed</i> , <i>BadFeed</i> , <i>BadTrim</i> , <i>ObliqueSheet</i> , <i>IncorrectComponent</i> , <i>IncorrectThickness</i> .
Good	InProgress	Running	Production of products in progress, good copy counter is on, waste copy counter is off
Idling	Stopped	Running	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.
JobCanceledByOperator New in JDF 1.3	Stopped	-	The job was canceled by the Device operator using AbortQueueEntry or means local to the Device.
JobCanceledByUser New in JDF 1.3	Aborted	-	The job was canceled by the owner of the job using AbortQueueEntry.

Table C-1: StatusDetails and Status mapping for generic devices (Section 2 of 3)

StatusDetails	JDF/@Status	DeviceStatus	Description
<i>JobCompletedSuccessfully</i> New in JDF 1.3	<i>Completed</i>	-	The job completed successfully.
<i>JobCompletedWithErrors</i> New in JDF 1.3	<i>Completed</i>	-	The job completed with errors (and possibly warnings too)
<i>JobCompletedWithWarnings</i> New in JDF 1.3	<i>Completed</i>	-	The job completed with warnings.
<i>JobHeldOnCreate</i> New in JDF 1.3	<i>Waiting</i>	-	The job was submitted to the queue with the <code>Queue/@Status = "Held"</code> , the job's <code>QueueSubmissionParams/@Held = "true"</code> , or <code>JDF/@Activation = "Held"</code> .
<i>JobHeld</i> New in JDF 1.3	<i>Waiting</i>	-	The Device held the job that had been waiting (by performing a <code>HoldQueueEntry</code> request on a <code>WaitingQueueEntry</code>).
<i>JobIncoming</i> New in JDF 1.3	<i>Waiting</i>	-	The Device is retrieving/accepting document data.
<i>JobResuming</i> New in JDF 1.3	<i>Waiting</i>	-	The Device is in the process of moving the job from a suspended condition to a candidate for processing (<code>ResumeQueueEntry</code>).
<i>JobScheduling</i> New in JDF 1.3	<i>Waiting</i>	-	The Device is scheduling the job for processing.
<i>JobStreaming</i> New in JDF 1.3	<i>InProgress</i>	-	Same as <i>JobIncoming</i> 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).
<i>JobSuspended</i> New in JDF 1.3	<i>Suspended</i>	-	The Device suspended the job that had been processing (e.g. by performing a <code>SuspendQueueEntry</code> request on a <code>RunningQueueEntry</code>) and other jobs can be processed by the Device.
<i>JobSuspending</i> New in JDF 1.3	<i>InProgress</i>	<i>Running</i>	The Device is in the process of moving the job from a processing condition to a suspended condition where other jobs can be processed.
<i>Maintenance</i>	<i>Stopped</i>	<i>Stopped</i>	Maintenance of the device. <i>Maintenance</i> has specialized subcategories: <i>BlanketChange</i> and <i>SleeveChange</i> .

Table C-1: StatusDetails and Status mapping for generic devices (Section 3 of 3)

StatusDetails	JDF/@Status	DeviceStatus	Description
<i>MissResources</i>	<i>Stopped</i>	<i>Stopped</i>	Production has been stopped because resources are missing or unavailable. Waits for new resources; subcategory of <i>Pause</i> .
<i>MovingToPaused</i> New in JDF 1.3	<i>InProgress</i>	<i>Running</i>	The Device has been paused, but the machine(s) are taking an appreciable time to stop.
<i>OutputAreaFull</i> New in JDF 1.3	<i>Stopped</i>	<i>Stopped</i>	One or more output areas are full, e.g., tray, stacker, collator.
<i>OutputTrayMissing</i> New in JDF 1.3	<i>Stopped</i>	<i>Stopped</i>	One or more output trays are not in the Device
<i>PaperJam</i>	<i>Stopped</i>	<i>Stopped</i>	Media jam in the device; subcategory of <i>Failure</i> .
<i>Pause</i>	<i>Stopped</i>	<i>Stopped</i>	Machine paused; restart is possible. <i>Pause</i> has specialized subcategories: <i>MissResources</i> and <i>WaitForApproval</i> .
<i>ProcessingToStopPoint</i> New in JDF 1.3	<i>InProgress</i>	<i>Running</i>	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.
<i>Repair</i>	<i>Stopped</i>	<i>Down</i>	The device is being repaired after a break down.
<i>ShutDown</i>	<i>Stopped</i>	<i>Down</i>	Machine stopped (can be switched off), restart requires a run up.
<i>SizeChange</i>	<i>Setup</i>	<i>Setup</i>	Changing setup for media size.
<i>WaitForApproval</i>	<i>Stopped</i>	<i>Stopped</i>	Production has been stopped because a necessary approval is still missing, subcategory of <i>Pause</i> .
<i>WarmingUp</i>	<i>Setup</i>	<i>Setup</i>	Device is warming up after power up or power saver mode wake-up.
<i>Waste</i>	<i>InProgress</i>	<i>Running</i>	Production of products in progress, good copy counter is off, waste copy counter is on.
<i>WasteFull</i>	<i>Stopped</i>	<i>Stopped</i>	The Device waste receptacle is full.

a. “—” means that the JDF/@Status is unknown if the device is not accessible.

Table C-2: Printing Device specific StatusDetails (Section 1 of 2)

StatusDetails	JDF/@Status	DeviceStatus	Description
<i>BlanketChange</i>	<i>Stopped</i>	<i>Stopped</i>	Changing of blankets; subcategory (e.g., a ‘specialization’) of <i>Maintenance</i> .

Table C-2: Printing Device specific StatusDetails (Section 2 of 2)

StatusDetails	JDF/@Status	DeviceStatus	Description
<i>BlanketWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of the blanket; subcategory of <i>WashUp</i> .
<i>CleaningInkFountain</i>	<i>Cleanup</i>	<i>Cleanup</i>	Cleaning of the ink fountain; subcategory of <i>WashUp</i> .
<i>CylinderWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of impression cylinders; subcategory of <i>WashUp</i> .
<i>DampeningRollerWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of the dampening roller; subcategory of <i>WashUp</i> .
<i>FormChange</i>	<i>Setup</i>	<i>Setup</i>	In conventional printing, changing of plates; in direct imaging printing, imaging or reimaging of plates.
<i>InkRollerWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of the inking roller; subcategory of <i>WashUp</i> .
<i>PlateWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of the plate; subcategory of <i>WashUp</i> .
<i>SleeveChange</i>	<i>Stopped</i>	<i>Stopped</i>	Changing of sleeves; subcategory for <i>Maintenance</i> .
<i>WashUp</i>	<i>Cleanup</i>	<i>Cleanup</i>	Machine is washed before, during or after production. <i>WashUp</i> has specialized subcategories: <i>BlanketWash</i> , <i>CleaningInkFountain</i> , <i>CylinderWash</i> , <i>DampeningRollerWash</i> , <i>InkRollerWash</i> , or <i>PlateWash</i> . <i>WashUp</i> is the default which is assumed if <i>StatusDetails</i> is not specified.

Table C-3: Postpress Device specific StatusDetails

StatusDetails	JDF/@Status	DeviceStatus	Description
<i>DoubleFeed</i> New in JDF 1.2	<i>Stopped</i>	<i>Stopped</i>	Double feeds on a feeder; subcategory of <i>Failure</i> .
<i>BadFeed</i> New in JDF 1.2	<i>Stopped</i>	<i>Stopped</i>	Bad feed on a feeder; subcategory of <i>Failure</i> .
<i>BadTrim</i> New in JDF 1.2	<i>Stopped</i>	<i>Stopped</i>	Bad trimmed components; subcategory of <i>Failure</i> .
<i>ObliqueSheet</i> New in JDF 1.2	<i>Stopped</i>	<i>Stopped</i>	Oblique sheets on components; subcategory of <i>Failure</i> . Oblique sheets are sheets or signatures which are not properly aligned within a pile (e.g., on a gathering or collecting chain).
<i>IncorrectComponent</i> New in JDF 1.2	<i>Stopped</i>	<i>Stopped</i>	Incorrect components on a feeder; subcategory of <i>Failure</i> .
<i>IncorrectThickness</i> New in JDF 1.2	<i>Stopped</i>	<i>Stopped</i>	Incorrect thickness of components; subcategory of <i>Failure</i> .

C.2 ModuleType Supported Strings

Both the `ModuleStatus` element (see Table 5-58, “ModuleStatus element,” on page 187) and the `ModulePhase` element (see Table 3-34, “ModulePhase audit element,” on page 106) contain a `ModuleType` attribute that defines individual modules within a machine. The following table defines individual `ModuleType` values.

Table C-4: ModuleType definition for conventional printing devices

ModuleType	Description
<i>Feeder</i>	Feeder module, feeds the device with paper.
<i>PrintModule</i>	Unit for printing a color. Describes one cylinder and one side.
<i>CoatingModule</i>	Unit for coatings, for example, full coating of varnish.
<i>Drier</i>	Module for drying the previously printed color or varnish.
<i>PerfectingModule</i>	Unit for perfecting, reversing device.
<i>ExtensionModule</i>	Unit for extending the distance between modules, for example to increase the distance between the last printing module and the delivery module.
<i>Delivery</i>	Delivery module, unit for gathering the printed sheets.
<i>Imaging</i>	Imaging Module in a direct to plate machine.
<i>Numbering</i>	Numbering unit.

Table C-5: ModuleType definition Gathering / Collecting

ModuleType	Description
<i>Feeder</i> New in JDF 1.2	Feeder module, feeds the device with paper.
<i>Chain</i> New in JDF 1.2	Transport chain or conveyer to transport gathered / collected product.
<i>PaperPath</i> New in JDF 1.2	Paper path module, path that paper follows through the machine.

Table C-6: ModuleType definition for DigitalPrinting

ModuleType	Description
<i>FarmPrinter</i> New in JDF 1.3	Individual Printer in a printer farm of printers.
<i>Fuser</i> New in JDF 1.2	Fuser module — fuses the toner onto the media.

Table C-7: Module Type definition for web printing devices; Possible Modules of a PrintingUnit-WebPath (Section 1 of 2)

ModuleType	Description
<i>ChillUnit</i> New in JDF 1.3	Chill unit that chills down the heated printed paper.
<i>ImprintUnit</i> New in JDF 1.3	Printing unit that allows changing plates during production run, doing imprints.
<i>PrintUnit</i> New in JDF 1.3	A <i>PrintUnit</i> consists of multiple <i>PrintModule</i> units.

Table C-7: Module Type definition for web printing devices; Possible Modules of a PrintingUnit-WebPath (Section 2 of 2)

ModuleType	Description
<i>RemoisteningModule</i> New in JDF 1.3	Module that can be used for high gloss varnish, remoistened glue, rub-off ink or encapsulated fragrances. The <i>RemoisteningModule</i> is located between last printing unit and dryer.
<i>Rollstand</i> New in JDF 1.3	The <i>Rollstand</i> feeds the web into the process-unit chain.
<i>UVCoater</i> New in JDF 1.3	The <i>UVCoater</i> module applies UV-varnish with subsequent drying in a UV-dryer.

Table C-8: Module Type definition for web printing devices; Possible Modules of a FolderSuperstructureWebPath (Section 1 of 2)

ModuleType	Description
<i>CrossCutter</i> New in JDF 1.3	Cuts the web / ribbon n-times into sheets and transports the sheets to inline postpress-equipment
<i>Delivery</i> New in JDF 1.3	Delivers the printed and/or folded sheets out of the folder
<i>Folder</i> 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)
<i>Former</i> New in JDF 1.3	Module for gathering ribbons and in most instances doing the first fold of the ribbons (quarter fold).
<i>GluingAndSofteningModule</i> New in JDF 1.3	Consists multiple heads, spread out in the press for gluing or/and softening of ribbons or folded sheets
<i>MoebiusDeinfiniteizer</i> New in JDF 1.3	Used to resolve the infinite loops caused by printing on interleaving surfaces of Möbius banded webs.
<i>PerforatingModule</i> New in JDF 1.3	Module for doing cross, longitudinal or diagonal perforations and die cuts on a web. Module is placed between <i>ChillUnit</i> and <i>Folder</i> .
<i>PlanoModule</i> New in JDF 1.3	The <i>PlanoModule</i> cuts the web / ribbon into sheets and stacks the sheets to a pile
<i>PloughFoldModule</i> New in JDF 1.3	The <i>PloughFoldModule</i> does a quarter fold to ribbons or webs, mostly found in front of a <i>Folder</i> module
<i>Rewinder</i> New in JDF 1.3	Rewinds the printed web to a roll.
<i>RibbonCompensator</i> New in JDF 1.3	Controls the web / ribbons in running direction regarding the cross cut
<i>Slitter</i> New in JDF 1.3	Module for cutting in machine direction
<i>Stitcher</i> New in JDF 1.3	Stitches folded sheets together
<i>Superstructure</i> 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.
<i>TurnerBar</i> New in JDF 1.3	Turns the front side of a web to the back side and vice versa.

Table C-8: Module Type definition for web printing devices; Possible Modules of a FolderSuper-structureWebPath (Section 2 of 2)

ModuleType	Description
<i>TurnerBarUnit</i> New in JDF 1.3	Turns the front side of a web to the back side and vice versa in a separate unit.

Table C-9: Module Type definition for web printing devices; Possible Modules of a PostPressComponentPath

ModuleType	Description
<i>BundlingModule</i> New in JDF 1.3	The <i>BundlingModule</i> is used for bundling components
<i>LabelingModule</i> New in JDF 1.3	The <i>LabelingModule</i> is used for labelling a bundle.
<i>PalletizingModule</i> New in JDF 1.3	The <i>PalletizingModule</i> collects the Bundles on a pallet. See “Palletizing” on page 260.
<i>PrintRoll</i> New in JDF 1.3	The <i>PrintRoll</i> is used for rolling components. See “PrintRolling” on page 261.
<i>Stacker</i> New in JDF 1.3	Stacks the component to a pile. See “Stacking” on page 263.
<i>Trimmer</i> New in JDF 1.3	Trims the component to its final size. See “Trimming” on page 265.

C.3 NotificationDetails

The Notification element is used for messaging and logging of events. It is defined in Section 3.10.1.2, Notification. Notifications are grouped into five classes: *event*, *information*, *warning*, *error* and *fatal*. For notification classes see Section 4.6.1, Classification of Notifications. 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. It is defined in Section 3.10.1.2.1, NotificationDetails. The structure of various predefined NotificationDetails-types and their descriptions are listed in the following sections.

C.3.1 Predefined NotificationDetails

This section defines elements that are derived from the abstract NotificationDetails element. The Table C-10 specifies value of NotificationDetails/@Type for each element with the same name.

Table C-10: Type attribute

Name	Description
<i>Barcode</i>	for Barcode element
<i>FCNKey</i>	for FCNKey element
<i>SystemTimeSet</i>	for SystemTimeSet element
<i>CounterReset</i>	for CounterReset element
<i>Error</i>	for Error element
<i>Event</i>	for Event element
<i>Milestone</i>	for Milestone element

C.3.1.1 Barcode

A bar code has been scanned.

Table C-11: Barcode element

Name	Data Type	Description
<i>Code</i>	string	Contains the scanned bar code.

C.3.1.2 FCNKey

A function key has been activated at a console.

Table C-12: FCNKey element

Name	Data Type	Description
<i>Key</i>	integer	Contains the number of that function key.

C.3.1.3 SystemTimeSet

The system time of a device/controller/agent has been set, (e.g., readjusted, changed to daylight saving time, etc.).

Table C-13: SystemTimeSet element

Name	Data Type	Description
<i>NewTime</i>	dateTime	Contains the new time.
<i>OldTime ?</i>	dateTime	Contains the old time.

C.3.1.4 CounterReset

The production counter of a device has been reset.

Table C-14: CounterReset element

Name	Data Type	Description
<i>CounterID ?</i>	string	Identification of the counter that has been set.
<i>LastCount ?</i>	integer	Last counter value before reset.

C.3.1.5 Error

This element provides additional information for common errors.

Table C-15: Error element

Name	Data Type	Description
<i>ErrorID ?</i> Modified in JDF 1.3	string	Internal Error ID of the application that declares the error.
<i>Resend ?</i> New in JDF 1.3	enumeration	Expected resending policy to fix the error. One of: <i>Required</i> – A corrected version of the offending JMF MUST be resent. <i>Prohibited</i> – A corrected version of the offending JMF MUST NOT be resent.
<i>ReturnCode ?</i> New in JDF 1.2	integer	JDF defined return code for an error. See “Supported Error Codes in JMF and Notification elements” on page 717.
<i>ErrorData *</i> New in JDF 1.3	element	Additional details of the error.

This element provides additional information for locating errors.

Table C-16: ErrorData element

Name	Data Type	Description
<i>ErrorType</i>	enumeration	Details of the error of the attribute or element specified in <i>Path</i> . Allowed values are: <i>Invalid</i> : the attribute or element has an invalid value <i>Missing</i> : the attribute or element is missing. <i>Unsupported</i> : the attribute or element is not known by the receiver.
<i>ErrorURL</i> ?	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.
<i>FixExpression</i> ?	regExp	Expression that defines the acceptable valid values for the attribute de-fined by <i>Path</i> . <i>FixExpression</i> MUST NOT be specified if <i>Path</i> specifies an element.
<i>Path</i> ?	XPath	XPath location of the erroneous attribute or element in the offending JMF or referenced file. If <i>ErrorURL</i> is specified, <i>Path</i> refers to the XML that is referenced by <i>ErrorURL</i> , otherwise it refers to the JMF that caused the error. <i>Path</i> MUST NOT be specified if <i>ErrorURL</i> references a format other than XML.

C.3.1.6 Event

[New in JDF 1.2](#)

This element provides additional information for common events.

Table C-17: Event element

Name	Data Type	Description
<i>EventID</i>	string	Internal Event ID of the application that emits the event.
<i>EventValue</i> ?	string	Additional user defined value related to this event.

C.3.1.7 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 “Status” on page 182.) and available/consumed resources (See “Resource” on page 172.), 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. A milestone has been reached during production of the Job.

Table C-18: Milestone element

Name	Data Type	Description
<i>MilestoneType</i>	NMTOKEN	Type of Milestone. For a list of predefined Values see: "MessageEvents and Milestone Values" on page 712:
<i>TypeAmount</i> ?	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 of a Milestone JMF:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="WorkflowController"
TimeStamp="2005-07-25T12:32:48+02:00" Version="1.3">
  <Signal ID="S1" Type="Notification">
    <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>
```

```

</Notification>
</Signal>
</JMF>

```

C.4 MessageEvents and Milestone Values

The following table defines a list of values that are valid for *MessageEvents*, *PageList/PageData/@PageStatus* and *Milestone/@MilestoneType*. The column “*JDF Process*” specifies the *Category* or *Type* of the node that the *CustomerMessage* MAY reside in. The column “*JDF Intent Resource*” specifies the **Resource** of a Product Intent node that *CustomerMessage* relates to. “*PageStatus*” specifies whether the value MAY be used as *PageList/PageData/@PageStatus*. “*Milestone*” specifies whether the value MAY be used as *Milestone/@MilestoneType*. Note that Milestones MAY refer to events involving multiple objects, although the *Milestone/@MilestoneType* is specified as a singular.

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 C-19: MessageEvents and MilestoneType Values (Section 1 of 3)

MessageEvents and MilestoneType Values	JDF Process	JDF Intent Resource	Milestone	Page-Status	Description
<i>Accepted</i>	DigDel	ArtDelI	—	Yes	The receiver acknowledged that the files are accessible for their destination.
<i>BindingCompleted</i>	—	All	Yes	Yes	All binding nodes including packing of the job have been completed. Postpress nodes are defined according to Section 6.6, Postpress Processes.
<i>BindingInProgress</i>	—	All	Yes	Yes	At least one of the binding nodes of the job is in progress status.
<i>Delivered</i>	DigDel	ArtDelI	Yes	Yes	The files were delivered to the destination.
<i>DeviceStopped</i>	All	—	—	—	The device that executes the node has been stopped.
<i>DigitalArtArrived</i>	—	All	Yes	Yes	Digital content has been received.
<i>JobCompletedSuccessfully</i>	All	All	Yes	Yes	Job completed successfully.
<i>JobCompletedWithErrors</i>	All	All	Yes	Yes	Job completed with errors.
<i>JobCompletedWithWarnings</i>	All	All	Yes	Yes	Job completed with warnings.
<i>JobInProgress</i>	All	All	Yes	Yes	Job is in progress.
<i>PageApproved</i>	—	ProofingI	Yes	Yes	Planned page proofs have been approved.
<i>PageCompleted</i>	—	All	Yes	Yes	Pages are ready (no further page processing or page proofing required).

Table C-19: MessageEvents and MilestoneType Values (Section 2 of 3)

MessageEvents and MilestoneType Values	JDF Process	JDF Intent Resource	Milestone	Page-Status	Description
<i>PageDeleted</i>	—	All	—	Yes	Specifies that this, originally planned, page was deleted. For instance, in the past, the page status was ' <i>PagePreliminary</i> '. Due to a reduction of the total number of pages, this specific page may have been deleted.
<i>PagePlanned</i>	—	All	Yes	Yes	Specifies that this page is ready for further processing. Its planning process is finished.
<i>PagePreliminary</i>					It is planned to produce this page, but its planning process is not finished yet.
<i>PageProofed</i>	—	ProofingI	Yes	Yes	Planned page proofs have been made
<i>PDLProduced</i>	All	All	Yes	Yes	Indicates that content data has been produced and is ready for production.
<i>ProofSent</i>	—	ProofingI	Yes	Yes	Planned proofs sent to customer.
<i>PostPressCompleted</i>	—	All	Yes	Yes	All Postpress nodes including packing of the job have been completed. Postpress nodes are defined according to Section 6.6, Postpress Processes.
<i>PostPressInProgress</i>	—	All	Yes	Yes	At least one of the Postpress nodes of the job is in progress status.
<i>PrePressCompleted</i>	—	All	Yes	Yes	All Prepress nodes of the job have been completed. Prepress nodes are defined according to Section 6.4, Prepress Processes. In conventional prepress, this is the case when all plates have been made.
<i>PrePressInProgress</i>	—	All	Yes	Yes	At least one of the Prepress nodes of the job is in progress status.
<i>PressCompleted</i>	—	All	Yes	Yes	All Press nodes of the job have been completed. Press nodes are defined according to Section 6.5, Press Processes.
<i>PressInProgress</i>	—	All	Yes	Yes	At least one of the Press nodes of the job is in progress status.
<i>ShippingCompleted</i>	Delivery	DeliveryI	Yes	Yes	Final product was delivered to the customer or distributors.

Table C-19: MessageEvents and MilestoneType Values (Section 3 of 3)

MessageEvents and MilestoneType Values	JDF Process	JDF Intent Resource	Miles tone	Page-Status	Description
<i>ShippingInProgress</i>	Delivery	DeliveryI	Yes	Yes	Final product is being shipped.
<i>SurfaceAssigned</i>	—	All	Yes	Yes	Surfaces have their corresponding pages assigned (e.g., could be proofed)
<i>SurfaceProofed</i>	—	ProofingI	Yes	Yes	Planned imposition proofs have been made
<i>SurfaceApproved</i>	—	ProofingI	Yes	Yes	Planned imposition proofs have been approved.
<i>SurfaceCompleted</i>	—	All	Yes	Yes	Planned surfaces are ready (i.e., plates could be made)

Example:

```
<CustomerInfo CustomerJobName="Job title ..."
  <CustomerMessage Language="FR" MessageEvents="JobCompletedSuccessfully
    JobCompletedWithErrors" ShowList="StartTime EndTime Error"
    UserText="This is the Mighty Mouse brochure job that should be approved by
      Walt Disney">
    <ComChannel ChannelType="Email" Locator="admin@mycompany.com"/>
  </CustomerMessage>
</CustomerInfo>
```

This is an instruction to generate an Email message in French to admin@mycompany.com when the job defined in the JDF node that includes CustomerInfo/CustomerMessage, is completed. Two Email messages are requested — when job is completed either successfully or with errors. The Email message SHOULD include start and end time of the job processing and error information if it exists. It SHOULD also include the text “This is the Mighty Mouse brochure job that should be approved by Walt Disney”.

C.5 Input Tray and Output Bin Names

[New in JDF 1.2](#)

Location/@*LocationName* MAY also 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 “Locations of Physical Resources” on page 94 for more details on the use of location.)

Table C-20: Locations within Printers (Section 1 of 2)

Name	I/O	Description
<i>AnyLargeFormat</i>	IO	The location that holds larger format media with one dimension larger than 11 inches. The media dimensions MUST be specified. <i>AnyLargeFormat</i> is defined for a PPD.
<i>AnySmallFormat</i>	IO	The location that holds smaller format media. The media dimensions MUST be specified. <i>AnySmallFormat</i> is defined for a PPD.
<i>AutoSelect</i>	IO	The location that the device selects based on the Media specification.
<i>Bottom</i>	IO*	The bin that, when facing the device, can best be identified as ‘bottom’.
<i>Booklet</i>	O	The bin where the Device places booklets.

Table C-20: Locations within Printers (Section 2 of 2)

Name	I/O	Description
<i>BypassTray</i>	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.
<i>BypassTray-N</i>	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', ...
<i>Center</i>	—	The bin that, when facing the device, can best be identified as 'center.' Deprecated in JDF 1.2 — use Middle instead.
<i>Continuous</i>	IO	The location to handle continuous media, (i.e., continuously connected sheets.)
<i>Disc</i>	IO	The location to handle CD or DVD discs to be printed on.
<i>Disc-N</i>	IO	The location to handle CD or DVD discs to be printed on. N = '1', '2', ...
<i>Envelope</i>	IO	The location that is to contain envelopes.
<i>Envelope-N</i>	IO	The location that is to contain envelopes. N = '1', '2', ...
<i>FaceDown</i>	O	The bin that can best be identified as 'face down' with respect to the device.
<i>FaceUp</i>	O	The bin that can best be identified as 'face up' with respect to the device.
<i>FitMedia</i>	O	Requests the device to select a bin based on the size of the media.
<i>Front</i>	IO*	The location that, when facing the device, can best be identified as 'front.'
<i>InsertTray</i>	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.)
<i>InsertTray-N</i>	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.)
<i>LargeCapacity</i>	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.
<i>LargeCapacity-N</i>	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.
<i>Left</i>	IO*	The bin that, when facing the device, can best be identified as 'left.'
<i>MyMailbox</i>	O	The job will be output to the bin that is best identified as 'MyMailbox'
<i>Mailbox-N</i>	O	The job will be output to the bin that is best identified as 'Mailbox-1', 'Mailbox-2'...etc.
<i>Middle</i>	IO*	The bin that, when facing the device, can best be identified as 'middle'.
<i>Rear</i>	IO*	The bin that, when facing the device, can best be identified as 'rear'.
<i>Right</i>	IO*	The bin that, when facing the device, can best be identified as 'right'.
<i>Roll</i>	IO	The location to handle roll fed media.
<i>Roll-N</i>	IO	The Nth location to handle the Nth roll fed media.
<i>Side</i>	IO*	The bin that, when facing the device, can best be identified as 'side'.
<i>Stacker-N</i>	O	The job will be output to the bin that is best identified as 'Stacker-1', 'Stacker-2' ... etc.
<i>Top</i>	IO*	The bin that, when facing the device, can best be identified as 'top'.
<i>Tray</i>	IO	The location for a single tray device.
<i>Tray-N</i>	IO	The job will be output to the tray that is best identified as 'Tray-1', 'Tray-2' ... etc.

Following is a table that lists some common location names that are analogous to a location name in the above table. The location names listed in the table above SHOULD be used when possible.

Name	Location Name to use instead
Back	<i>Rear</i>
Cassette	<i>Tray-N</i>
Lower	<i>Bottom</i>
Main	<i>LargeCapacity</i>
Upper	<i>Top</i>

Appendix D Supported Error Codes in JMF and Notification elements

The following list defines the standard *ReturnCode* for messaging. The ID numbers are decimal. 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 D-1: Return codes for JMF (Section 1 of 2)

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/command not implemented
6	Invalid parameters
7	Insufficient parameters
8	Device not available (controller exists but not the device or queue)
9	Message incomplete.
10	Message Service is busy. New in JDF 1.3
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	<i>JobID</i> not known by controller
104	<i>JobPartID</i> 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
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	Queue request failed because the Queue is <i>Closed</i> or <i>Blocked</i> and does not accept new entries. New in JDF 1.1
113	Queue entry is already in the resulting status. New in JDF 1.2
114	Queue entry is already completed or aborted and therefore does not accept changes.
115	Queue entry is not running. New in JDF 1.2
116	Queue entry already exists. Used when a <i>QueueEntry</i> with identical <i>JobID</i> , <i>JobPartID</i> and <i>Part</i> already exists. New in JDF 1.3 .

Table D-1: Return codes for JMF (Section 2 of 2)

ReturnCode	Description
120	Cannot access referenced URL. URI Reference cannot be resolved. Used when a referenced entity, e.g., a JDF in a <code>SubmitQueueEntry</code> cannot be found. New in JDF 1.3
121	Unknown <i>DeviceID</i> . No Device is known with the <i>DeviceID</i> specified. New in JDF 1.3
130	Ganging is not supported. A gang job has been submitted to a queue that does not support ganging. New in JDF 1.3
131	<i>GangName</i> not known. A job has been submitted with an unknown <i>GangName</i> . New in JDF 1.3
200 – ...	Job and pipe specific errors
200	Invalid resource parameters
201	Insufficient resource parameters
202	PipeID unknown
203	Unlinked ResourceLink
204	Could not create new JDF node. New in JDF 1.3
300	Authentication denied. New in JDF 1.3
301	Secure channel not supported - I don't support secure channel for this message. New in JDF 1.3
302	Secure channel required - I require secure channel for this message. New in JDF 1.3
303	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) New in JDF 1.3

Appendix E Color Adjustment Attribute Description and Usage

[New in JDF 1.2](#)

This appendix describes several alternative usages of some attributes in the `ColorCorrectionOp` element (see `ColorCorrectionParams/ColorCorrectionOp` in "ColorCorrectionParams" on page 364.) 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 product 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.

E.1 Adjustment Using Direct Attributes

This section describes the following attributes that provide direct adjustments to various aspect of the color space:

Table E-1: Attributes for Color Space Adjustment

Attribute Name	Allowed Value Range
<i>AdjustCyanRed</i>	-100 to +100
<i>AdjustMagentaGreen</i>	-100 to +100
<i>AdjustYellowBlue</i>	-100 to +100
<i>AdjustContrast</i>	-100 to +100
<i>AdjustHue</i>	-100 to +100
<i>AdjustLightness</i>	-100 to +100
<i>AdjustSaturation</i>	-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. [$L += \text{AdjustLightness}$]
- *AdjustContrast* scales the L* channel about mid-scale (where $L = 50$).
[$L = 50 + (L - 50) * (\text{AdjustContrast} / 100 + 1)$]
- *AdjustSaturation* scales the a* and b* channels about zero. [$a^* = (\text{AdjustSaturation} / 100 + 1)$] and
[$b^* = (\text{AdjustSaturation} / 100 + 1)$]

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

$$\begin{aligned} \mathbf{a}^* & += 0.641 * \text{AdjustCyanRed} \\ & - 0.996 * \text{AdjustMagentaGreen} \\ & + 0.078 * \text{AdjustYellowBlue} \\ \mathbf{b}^* & += 0.767 * \text{AdjustCyanRed} \\ & + 0.092 * \text{AdjustMagentaGreen} \\ & - 0.997 * \text{AdjustYellowBlue} \end{aligned}$$

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.

- $a = a * \cos(\text{AdjustHue}) - b * \sin(\text{AdjustHue})$
- $b = a * \sin(\text{AdjustHue}) + b * \cos(\text{AdjustHue})$

E.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.

E.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 ColorCorrectionOp element with the *ResourceUsage* attribute set to "*AbstractProfile*" references an ICC profile to be used in this manner.

E.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.

Appendix F North American and Japanese Media Weight Explained

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.

F.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:

$$\text{Basis Weight (lb.)} \times (1406.5 / \text{Square inches in basic size}) = \text{grams per square meter}$$

For example, the grammage of 65 lb. cover stock when the cover is 20 x 26 can be calculated as follows:

$$65 \times (1406.5 / (20 \times 26)) = 65 \times 2.70 = 176 \text{ g/m}^2$$

The following table defines the basic sizes and the factor that *USWeight* is multiplied by to calculate *Weight* for various stock types.

Table F-1: Conversion Factor from USWeight (lbs) to Weight (g/m²)

Stock Type	Basis size in Inches	Weight / USWeight	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 F-2: Grammage Equivalents for Common (US) Basis Weights (Section 1 of 2)

Grammage (g/m ²)	Basis Weight	Grammage (g/m ²)	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	216	80# Cover
67	45# Bond	218	125# Tag

Table F-2: Grammage Equivalents for Common (US) Basis Weights (Section 2 of 2)

Grammage (g/m ²)	Basis Weight	Grammage (g/m ²)	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

F.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 at <http://www.edsebooks.com/paper/jpaper.html>. For more help with grammage and basis weight conversion, see also Basis Weight and Grammage Conversion Tables at <http://home.inter.net/eds/paper/grammage.html>.

Following is the Japanese/grammage conversion formula:

$$\text{Basis Weight (kg) / Basic Size (m}^2\text{)} = \text{grams per square meter}$$

For example, the grammage of 70 kg Shiroku-ban stock when the size is 0.788 x 1.091 can be calculated as follows:

$$70 / (0.788 \times 1.091) = 81.4 \text{ g/m}^2$$

In the table below, trade-sheet size is given in mm.

Table F-3: Japanese Media Weight (Section 1 of 2)

Paper Grade *	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)
Joushitsuishi	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 F-3: Japanese Media Weight (Section 2 of 2)

Paper Grade *	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)
Chuushitsushi	--	45	--	30	54.2
	--	55	--	36.5	66.3
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
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
Aatoposutoshi	180	--	125	--	209.3
	200	--	139	--	232.6
	220	--	153	--	255.0

* The following describes the five paper grades in the above table:

- **Joushitsu** ("top-quality paper") contains 100% chemical pulp;
- **Chuushitsu** ("medium-quality paper") contains a minimum of 70% chemical pulp;
- **Aatoshi** ("art paper") is machine coated paper, available in top quality and medium quality (Joushitsu and Chuushitsu);
- **Mashinkootoshi** ("machine coated paper"), also called Kootoshi, is machine coated paper given only a thin coat of clay;
- **Aatoposutoshi** ("art-post paper") is cover stock coated on one side.

Appendix G Media Sizes

The following table defines a set of named media sizes as defined by [PPD].

Implementation Remark

Since Media sizes may be real numbers, comparison of Media sizes SHOULD take into account certain rounding errors. For example, different Media sizes MAY be considered equal when all numbers are the same within a range of 1 point.

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 G-1: Media Sizes (Section 1 of 3)

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	
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	
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

Table G-1: Media Sizes (Section 2 of 3)

Media Size	Size in Points	Size in Millimeters	Size in Inches	Notes
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	113 x 162	4.49 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
EnvC6	323 x 459	113 x 162	4.49 x 6.38	I, E
EnvC65	324 x 648	113 x 229	4.5 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

Table G-1: Media Sizes (Section 3 of 3)

Media Size	Size in Points	Size in Millimeters	Size in Inches	Notes
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
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	
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	

Appendix H FileSpec Attribute Examples for MimeType and MimeVersion Attributes

[New in JDF 1.2](#)

This appendix lists examples values for the following attributes of the **FileSpec** resource: *MimeType* and *MimeVersion*. 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 "Document Properties" on page 665 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 *MimeVersion* 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 (ex. "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 "/" (ex. "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 H-1 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 *MimeVersion* values. A blank *MimeVersion* table entry indicates that there is no recognized version number for the *MimeType*. Table H-1 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.

Table H-1: MimeType (IANA Registered) and MimeVersion Combinations (Section 1 of 3)

MimeType	MimeVersion	File Extension	Description [iana-mt] indicates IANA registration
application/mac-binhex40	HQX/4.0	.hqx	Macintosh BinHex 4.0 7-bit encoding ^a [RFC1741]
application/msword	MSWORD/5.0	.doc	Microsoft Word
application/msword	MSWORD/6.0	.doc	Microsoft Word
application/msword	MSWORD/2000	.doc	Microsoft Word
application/msword	MSWORD/XP	.doc	Microsoft Word
application/pdf	PDF/1.0	.pdf	Adobe Portable Document Format
application/pdf	PDF/1.1	.pdf	Adobe Portable Document Format
application/pdf	PDF/1.2	.pdf	Adobe Portable Document Format
application/pdf	PDF/1.3	.pdf	Adobe Portable Document Format
application/pdf	PDF/1.4	.pdf	Adobe Portable Document Format
application/pdf	PDF/1.5	.pdf	Adobe Portable Document Format
application/pdf	PDF/1.6	.pdf	Adobe Portable Document Format [PDF1.6]
application/pdf	PDF/X-1a:2001	.pdf	Portable Document Format (PDF) PDF/X-1a [ISO15930-1:2001]

Table H-1: MimeType (IANA Registered) and MimeTypeVersion Combinations (Section 2 of 3)

MimeType	MimeType-Version	File Extension	Description [iana-mt] indicates IANA registration
application/pdf	PDF/X-2:2003	.pdf	Portable Document Format (PDF) PDF/X-2 [ISO15930-5:2003]
application/pdf	PDF/X-3:2002	.pdf	Portable Document Format (PDF) PDF/X-3 [ISO15930-3:2002]
application/pdf	PDF/X-1a:2003	.pdf	Portable Document Format (PDF) PDF/X-3 [ISO15930-4:2003]
application/pdf	PDF/X-3:2003	.pdf	Portable Document Format (PDF) PDF/X-3 [ISO15930-6:2003]
application/postscript	PS/1	.ps	Adobe PostScript™ See [RFC2045] and [RFC2046]
application/postscript	PS/2	.ps	Adobe PostScript™ See [RFC2045] and [RFC2046]
application/postscript	PS/3	.ps	Adobe PostScript™ See [RFC2045] and [RFC2046]
application/vnd.cip4-jdf+xml	JDF 1.0	.jdf	CIP4 Job Definition Format (JDF) version 1.0, April 2001
application/vnd.cip4-jdf+xml	JDF 1.1	.jdf	CIP4 Job Definition Format (JDF) version 1.1, May 2002 and 1.1A, August 2002.
application/vnd.cip4-jdf+xml	JDF 1.2	.jdf	CIP4 Job Definition Format (JDF) version 1.2
application/vnd.cip4-jdf+xml	JMF 1.0	.jmf	CIP4 Job Definition Format (JDF) version 1.0, April 2001 (See Job Messaging Format)
application/vnd.cip4-jdf+xml	JMF 1.1	.jmf	CIP4 Job Definition Format (JDF) version 1.1 and 1.1A (See Job Messaging Format)
application/vnd.cip4-jdf+xml	JMF 1.2	.jmf	CIP4 Job Definition Format (JDF) version 1.2 (See Job Messaging Format)
application/vnd.cip3-ppf	PPF/1.0	.ppf	CIP3 Print Production Format (PPF) version 1.0, 1995 [PPF]
application/vnd.cip3-ppf	PPF/3.0	.ppf	CIP3 Print Production Format (PPF) version 3.0, 1998 [PPF]
application/vnd.hp-PCL	PCL/3	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.hp-PCL	PCL/4	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.hp-PCL	PCL/5	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.hp-PCL	PCL/5e	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.hp-PCL	PCL/6	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.hp-PCL	PCL/X	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.podi-ppml+xml	PPML/1.5	.ppml	Personalized Print Markup Language [PPML]

Table H-1: MimeType (IANA Registered) and MimeTypeVersion Combinations (Section 3 of 3)

MimeType	MimeType-Version	File Extension	Description [iana-mt] indicates IANA registration
application/vnd.podi-ppml+xml	PPML/2.0	.ppml	Personalized Print Markup Language [PPML]
application/vnd.podi-ppml+xml	PPML/2.1	.ppml	Personalized Print Markup Language [PPML]
application/vnd.Quark.QuarkXPress	XPress/4.11	.qxd .qxt .qwd .qwt .qxl .qxb	QuarkXPress [Quark]
application/vnd.Quark.QuarkXPress	XPress/4.31	.qxd .qxt .qwd .qwt .qxl .qxb	QuarkXPress [Quark]
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 <i>Compression</i> value, including "None", "Compress", "Gzip" and "ZLIB".
image/jpeg		.jpeg .jpg	JPEG See [RFC2045] and [RFC2046]. Note: 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" <i>MimeType</i> value is used to identify these file types.
image/tiff	tiff/6.0	.tiff .tif	Tag Image File Format ^b [RFC3302]
multipart/related		.mjd	Multipart/Related with JDF as the first part [RFC2387]
multipart/related		.mjm	Multipart/Related with JMF as the first part [RFC2387]

- a. 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.
- b. 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.

Table H-2 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 H-2 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 H-2: MimeType (File Type) and MimeTypeVersion combinations (Section 1 of 3)

MimeType	MimeTypeVersion	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	DCS/2.0	.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	ICC-Profile/2.1.0	.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]
ICC Profile	ICC-Profile/2.2.0	.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]
ICC Profile	ICC-Profile/2.4.0	.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]
ICC Profile	ICC-Profile/4.0.0	.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]
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	TIFF-IT/FP:1998	.fp	TIFF/IT ^a [ISO12639:2004] — Full Page — baseline
TIFF/IT	TIFF-IT/CT:1998	.ct	TIFF/IT [ISO12639:2004] — Continuous Tone picture data — baseline
TIFF/IT	TIFF-IT/LW:1998	.lw	TIFF/IT [ISO12639:2004] — Continuous Line art — baseline

Table H-2: MimeType (File Type) and MimeTypeVersion combinations (Section 2 of 3)

MimeType	MimeTypeVersion	File Extension	Description [iana-mt] indicates IANA registration
TIFF/IT	TIFF-IT/HC:1998	.hc	TIFF/IT [ISO12639:2004] — High-resolution Continuous tone image data — baseline
TIFF/IT	TIFF-IT/MP:1998	.mp	TIFF/IT [ISO12639:2004] — monochrome picture image data — baseline
TIFF/IT	TIFF-IT/BP:1998	.bp	TIFF/IT [ISO12639:2004] — Binary Picture image data — baseline
TIFF/IT	TIFF-IT/BL:1998	.bl	TIFF/IT [ISO12639:2004] — Binary Line art image data — baseline
TIFF/IT	TIFF-IT/FP/P1:1998	.fp	TIFF/IT [ISO12639:2004] — Full Page — profile 1
TIFF/IT	TIFF-IT/CT/P1:1998	.ct	TIFF/IT [ISO12639:2004] — Continuous Tone picture data — profile 1
TIFF/IT	TIFF-IT/LW/P1:1998	.lw	TIFF/IT [ISO12639:2004] — Color Line art data — profile 1
TIFF/IT	TIFF-IT/HC/P1:1998	.hc	TIFF/IT [ISO12639:2004] — High-resolution Continuous tone image data — profile 1
TIFF/IT	TIFF-IT/MP/P1:1998	.mp	TIFF/IT [ISO12639:2004] — monochrome picture image data — profile 1
TIFF/IT	TIFF-IT/BP/P1:1998	.bp	TIFF/IT [ISO12639:2004] — Binary Picture image data — profile 1
TIFF/IT	TIFF-IT/BL/P1:1998	.bl	TIFF/IT [ISO12639:2004] — Binary Line art image data — profile 1
TIFF/IT	TIFF-IT/FP:2004 ^b	.fp	TIFF/IT [ISO12639:2004] — Full Page — baseline
TIFF/IT	TIFF-IT/CT:2004	.ct	TIFF/IT [ISO12639:2004] — Continuous Tone picture data — baseline
TIFF/IT	TIFF-IT/LW:2004	.lw	TIFF/IT [ISO12639:2004] — Color Line art data — baseline
TIFF/IT	TIFF-IT/HC:2004	.hc	TIFF/IT [ISO12639:2004] — High-resolution Continuous tone image data — baseline
TIFF/IT	TIFF-IT/MP:2004	.mp	TIFF/IT [ISO12639:2004] — monochrome picture image data — baseline
TIFF/IT	TIFF-IT/BP:2004	.bp	TIFF/IT [ISO12639:2004] — Binary Picture image data — baseline
TIFF/IT	TIFF-IT/BL:2004	.bl	TIFF/IT [ISO12639:2004] — Binary Line art image data — baseline
TIFF/IT	TIFF-IT/SD:2004	.sd	TIFF/IT [ISO12639:2004]
TIFF/IT	TIFF-IT/FP/P1:2004	.fp	TIFF/IT [ISO12639:2004] — Full Page — profile 1
TIFF/IT	TIFF-IT/CT/P1:2004	.ct	TIFF/IT [ISO12639:2004] — Continuous Tone picture data — profile 1
TIFF/IT	TIFF-IT/LW/P1:2004	.lw	TIFF/IT [ISO12639:2004] — Color Line art data — profile 1
TIFF/IT	TIFF-IT/HC/P1:2004	.hc	TIFF/IT [ISO12639:2004] — High-resolution Continuous tone image data — profile 1
TIFF/IT	TIFF-IT/MP/P1:2004	.mp	TIFF/IT [ISO12639:2004] — monochrome picture image data — profile 1

Table H-2: MimeType (File Type) and MimeTypeVersion combinations (Section 3 of 3)

MimeType	MimeTypeVersion	File Extension	Description [iana-mt] indicates IANA registration
TIFF/IT	TIFF-IT/BP/P1:2004	.bp	TIFF/IT [ISO12639:2004] — Binary Picture image data — profile 1
TIFF/IT	TIFF-IT/BL/P1:2004	.bl	TIFF/IT [ISO12639:2004] — Binary Line art image data — profile 1 ^c
TIFF/IT	TIFF-IT/FP/P2:2004	.fp	TIFF/IT [ISO12639:2004] — Full Page — profile 2
TIFF/IT	TIFF-IT/CT/P2:2004	.ct	TIFF/IT [ISO12639:2004] — Continuous Tone picture data — profile 2
TIFF/IT	TIFF-IT/LW/P2:2004	.lw	TIFF/IT [ISO12639:2004] — Color Line art data — profile 2
TIFF/IT	TIFF-IT/HC/P2:2004	.hc	TIFF/IT [ISO12639:2004] — High-resolution Continuous tone image data — profile 2
TIFF/IT	TIFF-IT/MP/P2:2004	.mp	TIFF/IT [ISO12639:2004] — monochrome picture image data — profile 2
TIFF/IT	TIFF-IT/BP/P2:2004	.bp	TIFF/IT [ISO12639:2004] — Binary Picture image data — profile 2
TIFF/IT	TIFF-IT/BL/P2:2004	.bl	TIFF/IT [ISO12639:2004] — Binary Line art image data — profile 2
TIFF/IT	TIFF-IT/SD/P2:2004	.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]

- a. The file format TIFF/IT MUST 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.
- b. 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].
- c. Note: There is no TIFF/IT P1 conformance level of SD in [ISO12639:2004].

Appendix I FileSpec Attributes and Container Subelement

[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 **FileSpec** attributes: *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**.

I.1 FileSpec attribute value examples

Table I-1 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 MUST 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 "FileSpec" on page 430).

Table I-1: Use Cases showing MimeType, URL and Compression attribute values (Section 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 I-1: Use Cases showing mimeType, URL and Compression attribute values (Section 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.

I.2 Corresponding XML examples

The above use case examples are represented in XML as follows:

1 Single a.pdf PDF file, no compression:

```
<FileSpec mimeType="application/pdf" URL="ftp://www.any.com/share/a.pdf"/>
```

2 Single a.pdf PDF file, with Gzip compression:

```
<FileSpec Compression="Gzip" mimeType="application/pdf" URL="a.pdf">
  <Container>
    <FileSpec mimeType="Gzip" URL="ftp://www.any.com/a.gz"/>
  </Container>
</FileSpec>
```

3 Single a.pdf PDF file, no compression, but Base64 encoded:

```
<FileSpec Compression="Base64" mimeType="application/pdf" URL="a.pdf">
  <Container>
    <FileSpec mimeType="Base64" URL="ftp://www.any.com/a.mme"/>
  </Container>
</FileSpec>
```

4 Single PDF file, no compression, but BinHex encoded:

```
<FileSpec Compression="BinHex" mimeType="application/pdf" URL="a.pdf">
  <Container>
    <FileSpec mimeType="application/mac-binhex40" URL="ftp://www.any.com/a.hqx"/>
  </Container>
</FileSpec>
```

5 Single a.pdf PDF file, in b.zip ZIP file containing one or more files:

```
<FileSpec Compression="ZLIB" mimeType="application/pdf" URL="a.pdf">
  <Container>
    <FileSpec mimeType="application/zip" URL="ftp://www.any.com/b.zip"/>
  </Container>
</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 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>
```

- 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 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>
```

- 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 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>
```

I.3 Additional examples showing partitioning of FileSpec

This section has additional examples of container files and various schemes of partitioning.

- 1 Package b.zip contains multiple pdf files a.pdf, b.pdf etc.

```
<FileSpec ID="ID_002" MimeType="application/zip" URL="ftp://www.any.com/b.zip"/>
  <FileSpec Compression="Deflate" ID="A_FILE" MimeType="application/pdf" URL="a.pdf">
    <Container>
      <FileSpecRef rRef="ID_002"/>
    </Container>
  </FileSpec>
  <FileSpec Compression="Deflate" ID="B_FILE" MimeType="application/pdf" URL="b.pdf">
    <Container>
      <FileSpecRef rRef="ID_002"/>
    </Container>
  </FileSpec>
```

- 2 Package b.zip contains two pdf files a.pdf, b.pdf and a tiff, c.tiff used by a partitioned resource

```
<FileSpec ID="ID_003" MimeType="application/zip" URL="ftp://www.any.com/b.zip"/>
```

```
<FileSpec Compression="Deflate" ID="ALL_FILES" 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>
```

3 Single a.pdf PDF file, in b.zip which is contained in c.tar file:

```
<FileSpec ID="ID_004_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>

<FileSpec ID="ID_004_ZIP" MimeType="application/zip" URL="b.zip">
  <Container>
    <FileSpecRef rRef="ID_004_TAR"/>
  </Container>
</FileSpec>

<FileSpec Compression="Deflate" ID="C_FILE" MimeType="application/pdf" URL="a.pdf">
  <Container>
    <FileSpecRef rRef="ID_004_ZIP"/>
  </Container>
</FileSpec>
```

4 Multiple files in several zip's contained in a tar file, various examples with and without partitioning,

So the file layout looks like:

```
b.tar
  c.zip
    a.pdf
    b.pdf
  d.zip
    a.pdf
    b.pdf
```

Scheme 1 — No Partitioning

```
<FileSpec ID="ID_005_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>

<FileSpec ID="ID_005_ZIP_C" MimeType="application/zip" URL="c.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>
<FileSpec ID="ID_005_ZIP_D" MimeType="application/zip" URL="d.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>

<FileSpec Compression="Deflate" ID="A_ENGLISH_FILE" MimeType="application/pdf"
URL="a.pdf">
  <Container FileSpecRef="ID_005_ZIP_C"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="B_ENGLISH_FILE" MimeType="application/pdf"
URL="b.pdf">
  <Container FileSpecRef="ID_005_ZIP_C"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="A_GERMAN_FILE" MimeType="application/pdf"
URL="a.pdf">
  <Container FileSpecRef="ID_005_ZIP_D"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="B_GERMAN_FILE" MimeType="application/pdf"
URL="b.pdf">
```

```

  <Container FileSpecRef="ID_005_ZIP_D"/>
</FileSpec>

```

Scheme 2 — Intermediate container partitioned

```

<FileSpec ID="ID_005_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>

<FileSpec ID="ID_005_ZIPS" MimeType="application/zip" PartIDKeys="PartVersion">
  <Container FileSpecRef="ID_005_TAR"/>
  <FileSpec ID="EnglishFiles" PartVersion="English" URL="c.zip"/>
  <FileSpec ID="GermanFiles" PartVersion="German" URL="d.zip"/>
</FileSpec>

<FileSpec Compression="Deflate" ID="A_ENGLISH_FILE" MimeType="application/pdf"
URL="a.pdf">
  <Container FileSpecRef="EnglishFiles"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="B_ENGLISH_FILE" MimeType="application/pdf"
URL="b.pdf">
  <Container FileSpecRef="EnglishFiles"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="A_GERMAN_FILE" MimeType="application/pdf"
URL="a.pdf">
  <Container FileSpecRef="GermanFiles"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="B_GERMAN_FILE" MimeType="application/pdf"
URL="b.pdf">
  <Container FileSpecRef="GermanFiles"/>
</FileSpec>

```

Scheme 3 — the pdf's partitioned

```

<FileSpec ID="ID_005_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>

<FileSpec ID="ID_005_ZIP_C" MimeType="application/zip" URL="c.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>
<FileSpec ID="ID_005_ZIP_D" MimeType="application/zip" URL="d.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>

<FileSpec Compression="Deflate" ID="ALL_FILES" PartIDKeys="PartVersion DocIndex">
  <FileSpec ID="ENGLISH_FILES" PartVersion="English">
    <Container FileSpecRef="ID_005_ZIP_C"/>
    <FileSpec DocIndex="1" ID="A_ENGLISH_FILE" MimeType="application/pdf" URL="a.pdf"/>
    <FileSpec DocIndex="2" ID="B_ENGLISH_FILE" MimeType="application/pdf" URL="b.pdf"/>
  </FileSpec>
  <FileSpec ID="GERMAN_FILES" PartVersion="German">
    <Container FileSpecRef="ID_005_ZIP_D"/>
    <FileSpec DocIndex="1" ID="A_GERMAN_FILE" MimeType="application/pdf" URL="a.pdf"/>
    <FileSpec DocIndex="2" ID="B_GERMAN_FILE" MimeType="application/pdf" URL="b.pdf"/>
  </FileSpec>
</FileSpec>

```

Scheme 3a — As above but the file layout is not reflected in the container structure, the files are intermingled

```

<FileSpec ID="ID_005_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>

<FileSpec ID="ID_005_ZIP_C" MimeType="application/zip" URL="c.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>
<FileSpec ID="ID_005_ZIP_D" MimeType="application/zip" URL="d.zip">

```

```

    <Container FileSpecRef="ID_005_TAR"/>
  </FileSpec>

<FileSpec Compression="Deflate" ID="ALL_FILES" MimeType="application/pdf"
PartIDKeys="PartVersion DocIndex">
  <FileSpec ID="ENGLISH_FILES" PartVersion="English">
    <FileSpec DocIndex="1" ID="A_ENGLISH_FILE" URL="a.pdf">
      <Container FileSpecRef="ID_005_ZIP_C"/>
    </FileSpec>
    <FileSpec DocIndex="2" ID="B_ENGLISH_FILE" URL="a.pdf">
      <Container FileSpecRef="ID_005_ZIP_D"/>
    </FileSpec>
  </FileSpec>
<FileSpec ID="GERMAN_FILES" PartVersion="German">
  <FileSpec DocIndex="1" ID="A_GERMAN_FILE" URL="b.pdf">
    <Container FileSpecRef="ID_005_ZIP_C"/>
  </FileSpec>
  <FileSpec DocIndex="2" ID="B_GERMAN_FILE" URL="b.pdf">
    <Container FileSpecRef="ID_005_ZIP_D"/>
  </FileSpec>
</FileSpec>
</FileSpec>

```

Scheme 4 — Both partitioned

```

<FileSpec ID="ID_005_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>

<FileSpec ID="ID_005_ZIPS" MimeType="application/zip" PartIDKeys="PartVersion">
  <Container FileSpecRef="ID_005_TAR"/>
  <FileSpec ID="EnglishFiles" PartVersion="English" URL="c.zip"/>
  <FileSpec ID="GermanFiles" PartVersion="German" URL="d.zip"/>
</FileSpec>

<FileSpec Compression="Deflate" ID="ALL_FILES" PartIDKeys="PartVersion DocIndex">
  <FileSpec ID="ENGLISH_FILES" PartVersion="English">
    <Container FileSpecRef="EnglishFiles"/>
    <FileSpec DocIndex="1" ID="A_ENGLISH_FILE" MimeType="application/pdf" URL="a.pdf"/>
    <FileSpec DocIndex="2" ID="B_ENGLISH_FILE" MimeType="application/pdf" URL="b.pdf"/>
  </FileSpec>
  <FileSpec ID="GERMAN_FILES" PartVersion="German">
    <Container FileSpecRef="GermanFiles"/>
    <FileSpec DocIndex="1" ID="A_GERMAN_FILE" MimeType="application/pdf" URL="a.pdf"/>
    <FileSpec DocIndex="2" ID="B_GERMAN_FILE" MimeType="application/pdf" URL="b.pdf"/>
  </FileSpec>
</FileSpec>

```

- 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:

```

b.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 ID="ID_005_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>

<FileSpec ID="ID_005_ZIP_C" MimeType="application/zip" URL="c.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>
<FileSpec ID="ID_005_ZIP_D" MimeType="application/zip" URL="d.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>

<FileSpec Compression="Deflate" FileFormat="%s.pdf" FileTemplate="all" ID="PDF_FILES"
MimeType="application/pdf">
  <Container FileSpecRef="ID_005_ZIP_C"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="Pictures" URL="Pictures/TIFS/a.pdf">
  <Container FileSpecRef="ID_005_ZIP_D"/>
</FileSpec>

```

I.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:

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="FileSpecProposal01" JobID="bookJob"
Status="Waiting" Type="Product" Version="1.3">
  <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>

```

I.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 I-2: AppOS and OSVersion Examples

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
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 J Generating strings with Format and Template

[New in JDF 1.3](#)

JDF specifies a set of *XXXFormat XXXTemplate* pairs that allow dynamic generation of strings based on the standard C printf() function. (See [K&R]). The following instances are specified:

- *Query/@AcknowledgeFormat* and *Query/@AcknowledgeTemplate*;
- *Command/@AcknowledgeFormat* and *Command/@AcknowledgeTemplate*;
- **FileSpec/@FileFormat** and **FileSpec/@FileTemplate**;
- **IdentificationField/@ValueFormat** and **IdentificationField/@ValueTemplate**
- **Layout/MarkObject/DynamicField/@Format** and **Layout/MarkObject/DynamicField/@Template**.

The function defined with 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 the following operators: +, -, *, /, %, (,) which are evaluated using standard C-operator precedence and the variables defined in the following table or any valid *PartIDKeys* value of a partitioned resource.:

Table J-1: Predefined variables used in @FileTemplate and @ValueTemplate (Section 1 of 2)

Name	Description
<i>AcknowledgeType</i>	Corresponds to the JMF <i>AcknowledgeType</i> in the Acknowledge message. See "Acknowledge" on page 152.
<i>all</i>	Selects all matching elements. Valid only when FileSpec is used as an input resource.
<i>CustomerID</i>	CustomerID.
<i>Date</i>	Current <i>Date</i> in [ISO8601:2004] format.
<i>element</i>	Integer iterator over all elements in a given page. Restarts at 0 for each page.
<i>Generated</i>	System generated file name.
<i>i</i>	Integer iterator over all files produced by this process. 0-based numbering.
<i>input</i>	Local file name of the input file. Valid only when FileSpec is used as an output resource.
<i>jobID</i>	Job ID string.
<i>jobName</i>	<i>DescriptiveName</i> of the Node that is being processed.
<i>jobPartID</i>	JobPartID string.
<i>LayPartCount2in1</i> 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 <i>SignatureName / SheetName</i> partition keys, this equates to the number of signatures within the Layout . All other variants of <i>LayPartCount<n>in<m></i> 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 .
<i>LayPartIndex2in1</i> 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 <i>SignatureName / SheetName</i> partition keys, this equates to the signature number within the Layout . All other variants of <i>LayPartIndex<n>in<m></i> 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 .
<i>page</i>	Integer iterator over the page number of a document. This value is equivalent to value <i>r</i> (below) for the case that each run contains exactly one page.
<i>r</i>	Integer iterator over all RunList partitions with a partition key of "Run" in an input RunList .

Table J-1: Predefined variables used in @FileTemplate and @ValueTemplate (Section 2 of 2)

Name	Description
<i>ri</i>	Integer iterator over all indices in an input "Run" of a RunList . This index is equivalent to looping over a RunIndex.
<i>sep</i>	Separation as defined in the separation PartIDKey(s) of a partitioned resource.
<i>SheetNum</i> New in JDF 1.3	Integer iterator over the sheet number of a document.
<i>surf</i>	Surface string, "Front" or "Back". Deprecated in JDF 1.3 Use The partition Key "Side" instead.
<i>SystemRoot</i>	Root of system directory file structure. This token provides an operating system, independent way to refer to the root.
<i>TileX</i>	X coordinate of a Tile.
<i>TileY</i>	Y coordinate of a Tile.
<i>Time</i>	Current <i>Time</i> in [ISO8601:2004] format.
<i>TotalPagesInDoc</i>	Total # of pages in a document. New in JDF 1.3

Example using FileTemplate and FileFormat:

```
<FileSpec FileFormat = "file://myserver.mydomain.com/next/%s/%4.i/m%4.i.pdf"
FileTemplate = "JobID,i/100,i%100"/>
```

With *JobID* = "j001" and a **RunList** defining 2023 created files, the above 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"
```

Appendix K Resolving RunList/@Directory and FileSpec/@URL URI 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** relements.

K.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 MUST 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* MUST 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** element 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 "FileSpec" on page 430.)

After determining the Base URI depending on the presence or absence of **RunList/@Directory** and **FileSpec/Container/FileSpec** element as described above, each *URL* attribute in a **RunList** relement (e.g., **LayoutElement/FileSpec/@URL** or **InsertSheet/Sheet/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).

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 clear 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.

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.

Appendix L 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 L-1: References (Section 1 of 13)

Term	Definition
[Adb-TN5044]	<p><i>Adobe Technical note 5044</i></p> <p>Date: 24 May 1996 Produced by: Adobe Systems Inc. Available at: http://partners.adobe.com/public/developer/en/ps/sdk/5044.ColorSep_Conv.pdf</p>
[AdsML]	<p><i>AdsML 1.0 Specification & Schema</i></p> <p>Date: 17 May 2004 Produced by: AdsML Technical Working Group Available at: http://www.adsml.org</p>
[CCIR601-2]	<p><i>CCIR Recommendation 601-2</i></p> <p><i>Encoding Parameters of Digital Television for Studios, 1990, Volume XI — Part 1, Broadcasting Service (Television), pp. 95-104.</i></p> <p>Date: 1990 Produced by: International Telecommunication Union Available at: International Telecommunication Union, General Secretariat — Sales Section, Place des Nations, CH-1211 Geneva 20 (Switzerland)</p>
[CGATS.12/1]	<p><i>CGATS.12/1</i></p> <p><i>Graphic technology — Prepress digital data exchange — Use of PDF for composite data — Part 1: Complete exchange (PDF/X-1).</i></p> <p>Date: 14 October 1999 Produced by: Committee for Graphic Arts Technologies Standards (NPES serves as the American National Standards Institute (ANSI) secretariat to CGATS.) Available at: The publication is available in hardcopy only and may be ordered via a form at http://www.npes.org/standards/Standards-Technical-OrderForm.pdf.</p>
[CGATS.20-2002]	<p><i>CGATS.20-2002</i></p> <p><i>Graphic technology - Variable data printing exchange using PPML and PDF (PPML/VDX).</i></p> <p>Date: 2002 Produced by: Committee for Graphic Arts Technologies Standards (NPES serves as the American National Standards Institute (ANSI) secretariat to CGATS.) Available at: The publication is available in hardcopy only and may be ordered via a form at http://www.npes.org/standards/Standards-Technical-OrderForm.pdf.</p>
[CIE 15:2004]	<p><i>CIE 15:2004</i></p> <p><i>Colorimetry, 3rd Edition.</i></p> <p>Date: 2004 Produced by: Commission Internationale de l'Eclairage International (CIE) Available at: http://www.cie.co.at</p>
[ColorPS]	<p><i>Color Separation Conventions for PostScript Language Programs</i></p> <p><i>Technical Note #5044</i></p> <p>Date: 24 May 1996 Produced by: Adobe Systems Inc. Available at: http://partners.adobe.com/asn/developer/pdfs/tn/5044.ColorSep_Conv.pdf</p>

Table L-1: References (Section 2 of 13)

Term	Definition
[DCS2.0]	<p><i>Document Color Separation (DCS), version 2.0</i></p> <p>Date: Revised May 1995 Produced by: Adobe Software Inc. Available at: http://www.npes.org/standards/Tools/DCS20Spec.pdf.</p>
[distparm]	<p><i>Tech note 5151</i> <i>Acrobat Distiller Parameters</i></p> <p>Date: August 2002 Produced by: Adobe Systems, Inc. Available at: http://partners.adobe.com/misc/search.html</p>
[ECMA]	<p><i>ECMA Code of Folding Carton Styles</i></p> <p>Date: - Produced by: European Carton Makers Association Available at: http://www.ecma.org/download/orderformpublications.pdf</p>
[FEFCO]	<p><i>FEFCO European Federation of Corrugated Board Manufacturers</i></p> <p>Date: - Produced by: European Federation of Corrugated Board Manufacturers Available at: http://www.fefco.org</p>
[FileURL]	<p><i>CIP4 Application Note — Use of the File URL in JDF</i></p> <p>Date: August 2003 Produced by: CIP4 Organization Available at: http://www.cip4.org</p>
[FIRST]	<p><i>Flexographic Image Reproduction Specifications & Tolerances (FIRST)</i> <i>Second Edition</i></p> <p>Date: November 1999 Produced by: Flexography Technical Association Available at: http://www.fta-ffta.org.</p>
[GRACoL]	<p><i>General Requirements for Applications in Commercial offset Lithography (GRACoL)</i> <i>Version 6.0</i></p> <p>Date: June 2002 Produced by: IDEAlliance (formerly Graphic Communications Association) Available at: http://www.gracol.com.</p>
[iana-mt]	<p><i>IANA Registry of MIME Media Types</i></p> <p>Available at: http://www.iana.org/assignments/media-types</p>
[iana-os]	<p><i>IANA Registry of Operating System Names</i></p> <p>Available at: http://www.iana.org/assignments/operating-system-names</p>
[ICC.1]	<p><i>Specification ICC.1:2001-12</i> <i>File Format for Color Profiles, Version 4.0.0</i></p> <p>Date: 2001 Produced by: International Color Consortium (ICC) Available at: http://www.color.org/ICC_Minor_Revision_for_Web.pdf</p>
[IEEE754]	<p><i>IEEE 754-1985</i> <i>Standard for Binary Floating-Point Arithmetic</i></p> <p>Date: 1985 Produced by: IEEE Available at: http://grouper.ieee.org/groups/754/</p>

Table L-1: References (Section 3 of 13)

Term	Definition
[IEEE1284]	<p><i>IEEE 1284-2000</i> <i>IEEE Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers</i></p> <p>Date: 2000 Produced by: IEEE Available at: http://standards.ieee.org/catalog/olis/busarch.html</p>
[IEEE-ISTO 5100.1-2001]	<p><i>IEEE-ISTO 5100.1-2001</i> <i>IPP/1.1: finishings attribute values extension</i></p> <p>Date: February 5, 2001 Produced by: IEEE-ISTO Available at: ftp://ftp.pwg.org/pub/pwg/standards/pwg5100.1.pdf, .doc, .rtf</p>
[IEEE-ISTO 5100.2-2001]	<p><i>IEEE-ISTO 5100.2-2001</i> <i>IPP/1.0 & 1.1: "Output-bin" attribute extensions</i></p> <p>Date: February 7, 2001 Produced by: IEEE-ISTO Available at: ftp://ftp.pwg.org/pub/pwg/standards/pwg5100.2.pdf, .doc, .rtf</p>
[IEEE-ISTO 5100.3-2001]	<p><i>IEEE-ISTO 5100.3-2001</i> <i>Production Printing Attributes - Set1</i></p> <p>Date: February 17, 2001 Produced by: IEEE-ISTO Available at: ftp://ftp.pwg.org/pub/pwg/standards/pwg5100.3.pdf, .doc, .rtf</p>
[IEEE-ISTO 5100.4-2001]	<p><i>IEEE-ISTO 5100.4-2001</i> <i>Override Attributes for Documents and Pages</i></p> <p>Date: February 7, 2001 Produced by: IEEE-ISTO Available at: ftp://ftp.pwg.org/pub/pwg/standards/pwg5100.4.pdf, .doc, .rtf</p>
[ifra]	<p><i>IfraTrack Specification</i> <i>Ifra Special Report 6.21.2, Version 2.0</i></p> <p>Date: June 1998 Produced by: Ifra Available at: http://www.ifra.com/</p>
[ISO5-3:1995]	<p><i>ISO 5-3:1995</i> <i>Photography -- Density measurements -- Part 3: Spectral conditions.</i></p> <p>Date: 1995 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO5-4:1995]	<p><i>ISO 5-4:1995</i> <i>Photography -- Density measurements -- Part 4: Geometric conditions for reflection density.</i></p> <p>Date: 1995 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO216:1975]	<p><i>ISO 216:1975</i> <i>Writing paper and certain classes of printed matter -- Trimmed sizes -- A and B series.</i></p> <p>Date: 1975 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>

Table L-1: References (Section 4 of 13)

Term	Definition
[ISO269:1985]	<p><i>ISO 269:1985</i> <i>Correspondence envelopes -- Designation and sizes</i> Date: 1985 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO2470:1999]	<p><i>ISO 2470:1999</i> <i>Paper, board and pulps -- Measurement of diffuse blue reflectance factor (ISO brightness.</i> Date: 1999 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO2471:1998]	<p><i>ISO 2471:1998</i> <i>Paper and board—Determination of opacity (paper backing)—Diffuse reflectance method.</i> Date: 1998 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO2846-1:1997]	<p><i>ISO 2846-1:1997</i> <i>Graphic technology - Colour and transparency of ink sets for four-colour-printing - Part 1: Sheet-fed and heat-set web offset lithographic printing.</i> Date: 1997 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO3166-1:1997]	<p><i>ISO 3166-1:1997</i> <i>Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes.</i> Date: 1997 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO4217:2001]	<p><i>ISO 4217:2001</i> <i>Codes for the representation of currencies and funds</i> Date: 2001 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO8254-1:1999]	<p><i>ISO 8254-1:1999</i> <i>Paper and board -- Measurement of specular gloss -- Part 1: 75 degree gloss with a converging beam, TAPPI method</i> Date: 1999 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO8601:2004]	<p><i>ISO 8601:2004</i> <i>Data elements and interchange formats - Information interchange - Representation of dates and times.</i> Date: 2004 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>

Table L-1: References (Section 5 of 13)

Term	Definition
[ISO12639:2004]	<p>ISO 12639:2004</p> <p><i>Graphic technology - Prepress digital data exchange — Tag image file format for image technology (TIFF/IT)</i></p> <p>Date: 2004 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO12647-2:2004]	<p>ISO 12647-2:2004</p> <p><i>Graphic technology - Process control for the production of half-tone colour separations, proof and production prints - Part 2: Offset lithographic processes.</i></p> <p>Date: 2004 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO13655:1996]	<p>ISO 13655:1996</p> <p><i>Graphic technology -- Spectral measurement and colorimetric computation for graphic arts images.</i></p> <p>Date: 1996 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO14977:1996]	<p>ISO 14977:1996(E)</p> <p><i>Information technology -- Syntactic metalanguage -- Extended BNF</i></p> <p>Date: 1996 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO15930-1:2001]	<p>ISO 15930-1:2001</p> <p><i>Graphic technology — Prepress digital data exchange — Use of PDF — Part 1: Complete exchange using CMYK data (PDF/X-1 and PDF/X-1a).</i></p> <p>Date: 2001 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO15930-4:2003]	<p>ISO 15930-4:2003</p> <p><i>Graphic technology — Prepress digital data exchange — Use of PDF — Part 1: Complete exchange using CMYK data (PDF/X-1 and PDF/X-1a).</i></p> <p>Date: 2003 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO15930-5:2003]	<p>ISO 15930-2:2003</p> <p><i>Graphic technology — Prepress digital data exchange — Use of PDF — Part 2: Partial exchange of printing data (PDF/X-2).</i></p> <p>Date: 2003 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[ISO15930-3:2002]	<p>ISO 15930-3:2002</p> <p><i>Graphic technology — Prepress digital data exchange — Use of PDF — Part 3: Complete exchange suitable for colour-managed workflows (PDF/X-3).</i></p> <p>Date: 2002 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>

Table L-1: References (Section 6 of 13)

Term	Definition
[ISO15930-6:2003]	<p><i>ISO 15930-6:2003</i> <i>Graphic technology — Prepress digital data exchange — Use of PDF — Part 3: Complete exchange suitable for colour-managed workflows (PDF/X-3).</i></p> <p>Date: 2003 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[JIS P0138:1998]	<p><i>JIS P 0138:1998</i> Writing paper and certain classes of printed matter -- Trimmed sizes -- A and B series.</p> <p>Date: 1998 Produced by: JIS Available at: http://www.webstore.jsa.or.jp/webstore or google title</p>
[japancolor]	<p><i>Japan Color 2001</i></p> <p>Date: 2001 Produced by: Japan Printing Machinery Manufacturers Association, Office of JNC for TC130 Available at: Call (81) 03-3434-4661</p>
[JDF12]	<p><i>Job Definition Format 1.2</i></p> <p>Date: 2004 Produced by: International Cooperation for Integration of Processes in Prepress, Press and Postpress (CIP4) Available at: http://www.cip4.org</p>
[K&R]	<p><i>C Programming Language</i>, by Brian W. Kernighan and Dennis M. Ritchie <i>Second Edition</i></p> <p>Date: March 22, 1988 Produced by: Prentice Hall Available at: (Book only. Look for ISBN 0131103628.)</p>
[macbinary]	<p><i>Macintosh Binary Transfer Format ("MacBinary III") Standard Proposal.</i></p> <p>Date: December 1996 Produced by: Macintosh Internet Developer Association Available at: http://www.lazerware.com/formats/</p>
[opentypefont]	<p><i>OpenType specification</i> <i>v.1.4</i></p> <p>Date: 11 October 2002 Produced by: Microsoft Corporation Available at: http://www.microsoft.com/typography/specs/</p>
[PDF1.6]	<p><i>PDF reference : Adobe portable document format version 1.6 / Adobe Systems Incorporated.</i> <i>Version 1.6</i></p> <p>Date: November 2004 Produced by: Addison-Wesley Available at: http://partners.adobe.com/public/developer/pdf/index_reference.html</p>
[PJTF]	<p><i>The Portable Job Ticket Format</i> <i>Version 1.1</i></p> <p>Date: 2 April 1999 Produced by: Adobe Systems Inc. Available at: http://partners.adobe.com/asn/developer/pdfs/tn/5620.pjtf.pdf</p>

Table L-1: References (Section 7 of 13)

Term	Definition
[PPD]	<p><i>Adobe PostScript Printer Description File Format Specification</i> <i>Version 4.3</i></p> <p>Date: 9 February 1996 Produced by: Adobe Systems Inc. Available at: http://www.cip4.org/documents/technical_info/cip3v3_0.pdf.</p>
[PPF]	<p><i>Print Production Format</i> <i>Version 3.0</i></p> <p>Date: 2 June 1998 Produced by: The International Cooperation for Integration of Prepress, Press and Postpress Available at: http://www.cip4.org/documents/technical_info/cip3v3_0.pdf.</p>
[PPML]	<p><i>PPML]</i> <i>Personal Print Markup Language (PPML)</i> <i>Version 2.1</i></p> <p>Produced by: Print On Demand Initiative (PODi) Available at: http://www.podi.org</p>
[PrintTalk]	<p><i>PrintTalk Implementation</i> <i>Version 1.1</i></p> <p>Produced by: PrintTalk Consortium Available at: http://www.printtalk.org/.</p>
[PS]	<p><i>PostScript Language Reference (Redbook)</i> <i>Third Edition</i></p> <p>Date: — Produced by: Adobe Systems, Inc. Available at: http://partners.adobe.com/asn/developer/pdfs/tn/PLRM.pdf</p>
[PWG]	<p><i>The Printer Working Group</i></p> <p>Date: — Produced by: IEEE-ISTO Available at: http://www.pwg.org</p>
[PWGFIMIB]	<p><i>Printer Finishing MIB</i> (draft-ietf-printmib-finishing-16.txt — work in progress.)</p> <p>Date: February 2003 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: IETF Internet-Drafts have a six month life-time. They are available at: https://datatracker.ietf.org/public/pidtracker.cgi</p>
[Quark]	<p>See http://www.quark.com.</p>
[RFC1738]	<p><i>RFC 1738</i> <i>Uniform Resource Locators (URL)</i></p> <p>Date: 1994 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC1741]	<p><i>RFC 1741</i> <i>MIME Content Type for BinHex Encoded Files, by Falstrom, P., Crocker, D. and Fair, E.</i></p> <p>Date: December 1994 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>

Table L-1: References (Section 8 of 13)

Term	Definition
[RFC1759]	<p><i>RFC 1759</i> <i>Printer MIB, Version 2.0</i> 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: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC1766]	<p><i>RFC 1766</i> <i>Tags for the Identification of Languages</i>, by H. Alvestrand. Date: March 1995 Produced by: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC1950]	<p><i>RFC 1950</i> <i>ZLIB Compressed Data Format Specification version 3.3</i>, by P. Deutsch. Date: May 1996 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC1951]	<p><i>RFC 1951</i> <i>DEFLATE Compressed Data Format Specification version 1.3</i>, by Deutsch, P. Date: May 1996 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC1952]	<p><i>RFC 1952</i> <i>GZIP file format specification version 4.3</i>, by Deutsch, P. Date: May 1996 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC1977]	<p><i>RFC 1977</i> <i>PPP BSD Compression Protocol</i>, by Schryver, V. Date: August 1996 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC2045]	<p><i>RFC 2045</i> <i>Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies</i>, 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: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>

Table L-1: References (Section 9 of 13)

Term	Definition
[RFC2046]	<p><i>RFC 2046</i> <i>Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types</i>, by Freed, N. and Borenstein, N. (Updated by RFC2646)</p> <p>Date: November 1996 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC2183]	<p><i>RFC 2183</i> <i>Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field</i></p> <p>Date: August 1997 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC2231]	<p><i>RFC 2231</i> <i>MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations</i></p> <p>Date: November 1997 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC2368]	<p><i>RFC 2368</i> <i>The mailto URL scheme</i> by P. Hoffman, L. Masinter and J. Zawinski.</p> <p>Date: July 1998 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC2387]	<p><i>RFC 2387</i> <i>The MIME Multipart/Related Content-type</i>, by Levinson, E.</p> <p>Date: August 1998 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC2392]	<p><i>RFC 2392</i> <i>Content-ID and Message-ID Uniform Resource Locators</i>, by Levinson, E.</p> <p>Date: August 1998 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC2616]	<p><i>RFC 2616</i> <i>Hypertext Transfer Protocol — HTTP/1.1</i></p> <p>Date: June 1999 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>

Table L-1: References (Section 10 of 13)

Term	Definition
[RFC2822]	<p><i>RFC 2822</i> <i>Internet Message Format</i></p> <p>Date: April 2001 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC2911]	<p><i>RFC 2911</i> <i>Internet Printing Protocol/1.1: Model and Semantics</i>, by T. Hastings, R. Herriot, R. deBry, S. Isaacson and P. Powell.</p> <p>Date: September 2000 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC3302]	<p><i>RFC 3302</i> <i>Tag Image File Format (TIFF) — image/tiff MIME Sub-type Registration</i>, by Parsons, G., Rafferty, J.</p> <p>Date: September 2002 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC3381]	<p><i>RFC 3381</i> <i>Internet Printing Protocol (IPP): Job Progress Attributes</i> by T. Hastings, H. Lewis and R. Bergman.</p> <p>Date: September 2002 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC3382]	<p><i>RFC 3382</i> <i>Internet Printing Protocol (IPP): The 'collection' attribute syntax</i> by R. deBry, R. Herriot, T. Hastings, K. Ocke and P. Zehler.</p> <p>Date: September 2002 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC3548]	<p><i>RFC 3548</i> <i>The Base16, Base32, and Base64 Data Encodings</i>, by S. Josefsson</p> <p>Date: July 2003 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC3966]	<p><i>RFC 3966</i> <i>The tel URI for Telephone Numbers</i> by H. Schulzrinne</p> <p>Date: December 2004 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>

Table L-1: References (Section 11 of 13)

Term	Definition
[RFC3986]	<p><i>FC 3986</i> Uniform Resource Identifier (URI): Generic Syntax by T. Berners-Lee, R. Fielding and L. Masinter</p> <p>Date: January 2005 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[RFC3987]	<p><i>RFC 3987</i> <i>Internationalized Resource Identifiers (IRIs)</i>, by M. Duerst and M. Suignard</p> <p>Date: January 2005 Produced by: Internet Engineering Task Force (IETF), Network Working Group Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: http://www.rfc-editor.org/rfcsearch.html.</p>
[SNAP]	<p><i>Specifications for Newsprint Advertising Production (SNAP)</i></p> <p>Date: 2000 Produced by: Printing Industries of America, Inc. (SNAP Committee) Available at: http://www.gain.net/store/item.cfm?productid=488</p>
[SSL3]	<p><i>SSL Specification</i> Netscape, The SSL Protocol, Version 3 (Text version 3.0.2), November 1996. http://wp.netscape.com/eng/ssl3/draft302.txt</p>
[TAPPI T480]	<p>TAPPI T480 <i>Specular Gloss of Paper and Paperboard at 75 Degrees, Test Method T 519 om-99</i></p> <p>Date: not stated Produced by: TAPPI. Available at: http://www.tappi.org</p>
[TAPPI T519]	<p>TAPPI T519 <i>Diffuse Opacity of Paper (d/0 paper backing), Test Method T 519 om-02</i></p> <p>Date: not stated Produced by: TAPPI. Available at: http://www.tappi.org</p>
[TAPPI T527]	<p>TAPPI T527 <i>Color of Paper and Paperboard (d/0, C/2), Test Method T 527 om-02</i></p> <p>Date: not stated Produced by: TAPPI. Available at: http://www.tappi.org</p>
[TAPPI T560]	<p>TAPPI T560 <i>CIE Whiteness and Tint of Paper and Paperboard (Using d/0°, Diffuse Illumination and Normal Viewing), Test Method T 560 wd-03</i></p> <p>Date: not stated Produced by: TAPPI. Available at: http://www.tappi.org</p>
[TIFF6]	<p><i>TIFF Revision 6.0</i></p> <p>Date: June 1992 Produced by: Adobe Systems, Inc. Available at: http://partners.adobe.com/asn/tech/tiff/specification.jsp</p>

Table L-1: References (Section 12 of 13)

Term	Definition
[TIFFPS]	<p><i>Adobe Photoshop TIFF Technical Notes</i></p> <p>Date: March 2002 Produced by: Adobe Systems, Inc. Available at: http://partners.adobe.com/asn/tech/tiff/specification.jsp</p>
[truetypefont]	<p><i>TrueType font file and TrueType Open specification</i></p> <p>Date: August 1995 Produced by: Microsoft Corporation Available at: http://www.microsoft.com/typography/specs/</p>
[type1font]	<p><i>Adobe Type 1 Font Format</i> <i>Adobe Systems, Inc.</i></p> <p>Date: 1990 Produced by: Addison-Wesley Publishing Company, Inc. Available at: http://partners.adobe.com/asn/developer/pdfs/tn/T1_SPEC.PDF</p>
[uencode]	<p><i>Unix Uencode, The Single UNIX ® Specification, Version 2</i> (Converts binary into the local character set that is suitable to pass through email systems.)</p> <p>Date: 1997 Produced by: The Open Group Available at: http://www.opengroup.org/onlinepubs/007908799/xcu/uencode.html</p>
[UPNP]	<p><i>Printer Device and Printer Basic Service</i> <i>Version 1.0</i></p> <p>Date: 2002 Produced by: Universal Plug N Play Forum Available at: http://www.upnp.org/standardizeddcp/printers.asp</p>
[WINZip]	<p><i>APPNOTE.TXT — .ZIP File Format Specification</i> <i>Version 5.2</i></p> <p>Date: 16 July 2003 Produced by: PKWARE Inc. Available at: http://www.pkware.com/products/enterprise/white_papers/appnote.html</p>
[XML]	<p><i>XML Specification *</i> <i>Version 1.0 (Second Edition)</i></p> <p>Date: 6 October 2000 Produced by: World Wide Web Consortium (W3C) Available at: http://www.w3.org/TR/REC-xml.</p>
[XMLNS]	<p><i>Namespaces in XML</i> <i>Version (W3C Recommendation of 14 January 1999)</i></p> <p>Date: 14 January 1999 Produced by: World Wide Web Consortium (W3C) Available at: http://www.w3.org/TR/REC-xml-names/</p>
[XMLSchema]	<p><i>XML Schema Part 0+1+2: Primer, Structures and Datatypes *</i> <i>Version (W3C Recommendation of 02 May 2001)</i></p> <p>Date: 02 May 2001 Produced by: World Wide Web Consortium (W3C) XML Schema working group Available at: http://www.w3.org/TR/xmlschema-0/, http://www.w3.org/TR/xmlschema-1/ and http://www.w3.org/TR/xmlschema-2/.</p>

Table L-1: References (Section 13 of 13)

Term	Definition
[XPath]	<i>XML Path Language (XPath) Version 1.0</i> <i>Version W3C Recommendation 16 November 1999</i> Date: 16 November 1999 Produced by: World Wide Web Consortium (W3C) Available at: http://www.w3.org/TR/xpath.html .
[X.509]	The Public-Key Infrastructure (X.509) (pkix) standard is defined at http://www.ietf.org/html.charters/pkix-charter.html

Appendix M JDF/CIP4 Hole Pattern Catalog

The following table defines the specifics of the predefined holes in **HoleMakingParams** and **HoleMakingParams**.

Notes:

- 1 All patterns are centered on the sheet along the process edge.
- 2 Process Edge is always defined relative to a portrait orientation of the medium, regardless of the orientation of the printed image or processing path.
- 3 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.
- 4 Single thumbcuts appear always in the center of the process edge.
- 5 Oval shape holes actually look sometimes more like rectangular holes with rounded corners.

Sources:

- 1 [PWGFINMIB]

Naming Scheme:

Table M-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

Table M-2: Hole Details for R2 Series (Section 1 of 2)

JDF Hole Pattern Catalog ID	Description	# Holes	Hole Shape	Hole Extent	Pattern Geometry	Pattern Axis Offset from Process Edge	JDF Default Pattern Axis Offset from Process Edge in pt (!)	Default Process Edge	Usage Notes	Source Standard
RING BINDING (R...)										
2 Holes (R2...)										
R2-generic	Generic request of a 2-hole pattern	2	●	5 - 13 mm 0.2-0.51"	N/A	4.5 - 13 mm 0.18 - 0.51"	34.02 (≅ 12 mm)	Left	See note (7).	N/A
R2m-DIN	DIN 2-hole MIB: 6 = twoHoleDIN and 10 = twoHoleMetric	2	●	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	2	●	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	2	●	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 = twoHoleUSTop and 12 = twoHoleUSSide	2	●	0.2 - 0.32"	2.75"	0.18 - 0.51"	29.25 (≅ 13/32")	Left for letter Top for ledger		Printer Finishing MIB

Table M-2: Hole Details for R2 Series (Section 2 of 2)

JDF Hole Pattern Catalog ID	Description	# Holes	Hole Shape	Hole Extent	Pattern Geometry	Pattern Axis Offset from Process Edge	JDF Default Pattern Axis Offset from Process Edge in pt (!)	Default Process Edge	Usage Notes	Source Standard
R2i-US-b	US 2-hole, Variant B	2	●	0.2-0.5" default: 5/16" typical: 1/4", 9/32", 11/32", 3/8", 13/32", 1/2"	6"	0.25" + 1/2 diameter range: 6/16" - 1/2"	29.25 (≡ 13/32")	Left		

Table M-3: Hole Details for R3 and R4 Series (Section 1 of 2)

JDF Hole Pattern Catalog ID	Description	# Holes	Hole Shape	Hole Extent	Pattern Geometry	Pattern Axis Offset from Process Edge	JDF Default Pattern Axis Offset from Process Edge in pt (!)	Default Process Edge	Usage Notes	Source Standard
RING BINDING (R...)										
3 Holes (R3...)										
R3-generic	Generic request of a 3-hole pattern.	3	●	5 - 13 mm 0.2-0.51"	N/A	4.5 - 13 mm 0.18 - 0.51"	29.25 (≡ 13/32")	Left	See note (7).	N/A
R3i-US	US 3-hole MIB: 5 = threeHoleUS	3	●	std: 5/16" mg: 0.2-0.5" typ: 1/4", 9/32", 11/32", 3/8", 13/32", 1/2"	4.25"	0.25" + 1/2 diameter range: 6/16" - 1/2"	29.25 (≡ 13/32")	Left		Printer Finishing MIB
4 Holes (R4...)										
R4-generic	Generic request of a 4-hole pattern.	4	●	5 - 13 mm 0.2-0.51"	N/A	4.5 - 13 mm 0.18 - 0.51"	31.18 (≡ 11 mm)	Left	See note (7).	N/A

Table M-3: Hole Details for R3 and R4 Series (Section 2 of 2)

JDF Hole Pattern Catalog ID	Description	# Holes	Hole Shape	Hole Extent	Pattern Geometry	Pattern Axis Offset from Process Edge	JDF Default Pattern Axis Offset from Process Edge in pt (I)	Default Process Edge	Usage Notes	Source Standard
R4m-DIN-A4	DIN 4-hole for A4	4	●	5.5 ± 0.1 mm	80 ± 0.1 mm	7 or 11 ± 0.3 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	4	●	5.5 ± 0.1 mm	45-65-45 mm	7 or 11 ± 0.3 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	4	●	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
R4i-US	US 4-hole	4	●	0.2 - 0.5" std: 5/16" typ: 1/4", 9/32", 11/32", 3/8", 13/32", 1/2"	1.375-4.25-1.375"	0.25" + ½ diame-ter range: 6/16" - 1/2"	29.25 (≅ 0.25" + ½ x 5/16" = 13/32")	Left		

Table M-4: Hole Details for R5 and R6 Series (Section 1 of 2)

JDF Hole Pattern Catalog ID	Description	# Holes	Hole Shape	Hole Extent	Pattern Geometry	Pattern Axis Offset from Process Edge	JDF Default Pattern Axis Offset from Process Edge in pt (I)	Default Process Edge	Usage Notes	Source Standard
RING BINDING (R...)										
5 Holes (R5...)										
R5-generic	Generic request of a 5-hole pattern.	5	●	5 - 13 mm 0.2-0.51"	N/A	4.5 - 13 mm 0.18 - 0.51"	29.25 (≅ 13/32")	Left	See note (7).	N/A

Table M-4: Hole Details for R5 and R6 Series (Section 2 of 2)

JDF Hole Pattern Catalog ID	Description	# Holes	Hole Shape	Hole Extent	Pattern Geometry	Pattern Axis Offset from Process Edge	JDF Default Pattern Axis Offset from Process Edge in pt (I)	Default Process Edge	Usage Notes	Source Standard
R5i-US-a	US 5-hole, Variant A MIB: 13 = fiveHoleUS	5	●	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	5	●	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" + 1/2 diameter 0.375 - 0.5"	29.25 (≅ 0.25" + 1/2 x 5/16" = 13/32")	Left		
R5i-US-c	Combination of R2i-US-a and R3i-US	5	●	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" + 1/2 diameter 0.375 - 0.5"	29.25 (≅ 0.25" + 1/2 x 5/16" = 13/32")	Left		
6 Holes (R6...)										
R6-generic	Generic request of a 6-hole pattern.	6	●	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 (7).	N/A
R6m-4h2s	Norwegian 4-hole (round) mixed with 2 slots (rectangular) MIB: 16 = norwegian6Hole	6	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
R6m-DIN-A5	DIN 6-hole for A5	6	I	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

Table M-5: Hole Details for R7 and R11 Series (Section 1 of 2)

JDF Hole Pattern Catalog ID	Description	# Holes	Hole Shape	Hole Extent	Pattern Geometry	Pattern Axis Offset from Process Edge	JDF Default Pattern Axis Offset from Process Edge in pt (!)	Default Process Edge	Usage Notes	Source Standard
RING BINDING (R....)										
7 Holes (R7...)										
R7-generic	Generic request of a 7-hole pattern.	7	●	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 (7).	N/A
R7i-US-a	US 7-hole, Variant A MIB: 14 = seven-HoleUS	7	●	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	7	●	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 diame- ter 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	7	●	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 diame- ter 0.375 - 0.5"	29.25 (≅ 13/32")	Left for letter Top for ledger		
11 Holes (R11...)										

Table M-5: Hole Details for R7 and R11 Series (Section 2 of 2)



JDF Hole Pattern Catalog ID	Description	# Holes	Hole Shape	Hole Extent	Pattern Geometry	Pattern Axis Offset from Process Edge	JDF Default Pattern Axis Offset from Process Edge in pt (!)	Default Process Edge	Usage Notes	Source Standard
R11m-7h4s	7-hole (round) mixed with 4 slots (rectangular) MIB: 15 = mixed7H4S	11	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

Table M-6: Hole Details for P, W, C and S Series (Section 1 of 3)



JDF Hole Pattern Catalog ID	Description	# Holes	Hole Shape	Hole Extent	Pattern Geometry	Pattern Axis Offset from Process Edge	JDF Default Pattern Axis Offset from Process Edge in pt (!)	Default Process Edge	Usage Notes	Source Standard
PLASTIC COMB BINDING (P...)										
P16_9i-rect-0t	US spacing, no thumbcut MIB: 9 = nineteen-HoleUS	A4: 21 Letter: 19		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		
WIRE COMB BINDING (W...)										

Table M-6: Hole Details for P, W, C and S Series (Section 2 of 3)

JDF Hole Pattern Catalog ID	Description	# Holes	Hole Shape	Hole Extent	Pattern Geometry	Pattern Axis Offset from Process Edge	JDF Default Pattern Axis Offset from Process Edge in pt (!)	Default Process Edge	Usage Notes	Source Standard
W2_i-round-0t	2:1, round, no thumbcut MIB: 8 = twentyT-woHoleUS	A4: 23 Letter: 21	●	0.2 - 0.32" std: 1/4" Europe typ: 6 or 6.4 mm	1/2"	3 mm + 1/2 diameter 0.318 - 0.438" Europe: 6 - 6.2 mm	17.50 (≅ 0.243")	Left		Printer Finishing MIB
W2_i-square-0t	2:1, square, no thumbcut	A4: 23 Letter: 21	■	0.2 - 0.32" std: 1/4" Europe typ: 6 or 6.4 mm	1/2"	3 mm + 1/2 diameter 0.318 - 0.438" Europe: 6 - 6.2 mm	17.50 (≅ 0.243")	Left		
W3_i-square-0t	3:1, square, no thumbcuts	A4: 34 A5: 24 Letter: 32	■	5/32 x 5/32" (4x4 mm)	1/3"	0.2"	14.40 (≅ 0.2")	Left		
COIL/SPIRAL BINDING (C...)										
C9.5m-round-0t	9.5 mm, round, no thumbcut MIB: 17 - metric26Hole and 18 - metric30Hole	A4/A3: 30 JIS B5/B4: 26	●	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
SPECIAL (S...)										
S-generic	Generic request of a hole pattern with an arbitrary or unknown number of holes, e.g. an inline shotgun.	1 or more	●	5 - 13 mm 0.2-0.51"	N/A	N/A	N/A	Any		N/A

Table M-6: Hole Details for P, W, C and S Series (Section 3 of 3)

JDF Hole Pattern Catalog ID	Description	# Holes	Hole Shape	Hole Extent	Pattern Geometry	Pattern Axis Offset from Process Edge	JDF Default Pattern Axis Offset from Process Edge in pt (!)	Default Process Edge	Usage Notes	Source Standard
SI-generic	Generic request of a hole pattern with 1 hole.	1	●	5 - 13 mm 0.2-0.51"	N/A	N/A	N/A	Any		N/A

Appendix N Examples

Note that these examples were generated using prototype tools and are intended to be used for general overview only. The emphasis is *not* on the individual bytes, (e.g., capitalization or exact keywords). Normative examples will be provided at <http://www.CIP4.org> when available.

N.1 Brief Example

N.1.1 Before Processing

This is a simple example of a JDF that describes color conversion for one file.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="ColorTest" JobID="ColorJob"
Status="Waiting" Type="ColorSpaceConversion" Version="1.3">
  <ResourcePool>
    <RunList Class="Parameter" ID="Link0003" Pages="0 ~ -1" Status="Available">
      <LayoutElement>
        <FileSpec URL="File:///in/colortest.pdf"/>
      </LayoutElement>
    </RunList>
    <ColorSpaceConversionParams Class="Parameter" ID="Link0004" Status="Available">
      <FileSpec ResourceUsage="FinalTargetDevice" URL="File:///SMProcessCMYK.icc"/>
      <ColorSpaceConversionOp Operation="Convert" RenderingIntent="Perceptual"
SourceCS="RGB" SourceObjects="ImagePhotographic ImageScreenShot SmoothShades">
        <FileSpec ResourceUsage="SourceProfile" URL="File:///image.icc"/>
      </ColorSpaceConversionOp>
      <ColorSpaceConversionOp Operation="Convert" RenderingIntent="Perceptual"
SourceCS="RGB" SourceObjects="Text LineArt">
        <FileSpec ResourceUsage="SourceProfile" URL="File:///text.icc"/>
      </ColorSpaceConversionOp>
    </ColorSpaceConversionParams>
    <ColorPool Class="Parameter" ID="Link0005" Status="Available">
      <Color CMYK="1 0 0 0" Name="Cyan"/>
      <Color CMYK="0 1 0 0" Name="Magenta"/>
      <Color CMYK="0 0 1 0" Name="Yellow"/>
      <Color CMYK="0 0 0 1" Name="Black"/>
      <Color CMYK="0.8 0.8 0 0" Name="Blue"/>
    </ColorPool>
    <ColorantControl Class="Parameter" ID="Link0006" ProcessColorModel="DeviceCMYK"
Status="Available">
      <ColorPoolRef rRef="Link0005"/>
    </ColorantControl>
    <RunList Class="Parameter" ID="Link0007" Pages="0 ~ -1" Status="Unavailable">
      <LayoutElement>
        <FileSpec URL="File:///out/colortest.pdf"/>
      </LayoutElement>
    </RunList>
  </ResourcePool>
  <ResourceLinkPool>
    <RunListLink Usage="Input" rRef="Link0003"/>
    <ColorSpaceConversionParamsLink Usage="Input" rRef="Link0004"/>
    <ColorPoolLink Usage="Input" rRef="Link0005"/>
    <ColorantControlLink Usage="Input" rRef="Link0006"/>
    <RunListLink Usage="Output" rRef="Link0007"/>
  </ResourceLinkPool>
  <AuditPool>
    <Created AgentName="Rainer's JDFWriter 0.2000"
TimeStamp="2005-06-01T10:26:11+01:00"/>
  </AuditPool>
</JDF>
```

```
</JDF>
```

N.1.2 After Processing

This is a simple example of a JDF that describes color conversion for one file after the color conversion process has been executed.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="ColorTest " JobID="ColorJob"
Status="Completed" Type="ColorSpaceConversion" Version="1.3">
  <ResourcePool>
    <RunList Class="Parameter" ID="Link0003" Pages="0 ~ -1" Status="Available">
      <LayoutElement>
        <FileSpec URL="File:///in/colortest.pdf"/>
      </LayoutElement>
    </RunList>
    <ColorSpaceConversionParams Class="Parameter" ID="Link0004" Status="Available">
      <FileSpec ResourceUsage="FinalTargetDevice" URL="File:///SMProcessCMYK.icc"/>
      <ColorSpaceConversionOp Operation="Convert" RenderingIntent="Perceptual"
SourceCS="RGB" SourceObjects="ImagePhotographic ImageScreenShot SmoothShades">
        <FileSpec ResourceUsage="SourceProfile" URL="File:///image.icc"/>
      </ColorSpaceConversionOp>
      <ColorSpaceConversionOp Operation="Convert" RenderingIntent="Perceptual"
SourceCS="RGB" SourceObjects="Text LineArt">
        <FileSpec ResourceUsage="SourceProfile" URL="File:///text.icc"/>
      </ColorSpaceConversionOp>
    </ColorSpaceConversionParams>
    <ColorPool Class="Parameter" ID="Link0005" Status="Available">
      <Color CMYK="1 0 0 0" Name="Cyan"/>
      <Color CMYK="0 1 0 0" Name="Magenta"/>
      <Color CMYK="0 0 1 0" Name="Yellow"/>
      <Color CMYK="0 0 0 1" Name="Black"/>
      <Color CMYK="0.8 0.8 0 0" Name="Blue"/>
    </ColorPool>
    <ColorantControl Class="Parameter" ID="Link0006" ProcessColorModel="DeviceCMYK"
Status="Available">
      <ColorPoolRef rRef="Link0005"/>
    </ColorantControl>
    <RunList Class="Parameter" ID="Link0007" Pages="0 ~ -1" Status="Available">
      <LayoutElement>
        <FileSpec URL="File:///out/colortest.pdf"/>
      </LayoutElement>
    </RunList>
  </ResourcePool>
  <ResourceLinkPool>
    <RunListLink Usage="Input" rRef="Link0003"/>
    <ColorSpaceConversionParamsLink Usage="Input" rRef="Link0004"/>
    <ColorPoolLink Usage="Input" rRef="Link0005"/>
    <ColorantControlLink Usage="Input" rRef="Link0006"/>
    <RunListLink Usage="Output" rRef="Link0007"/>
  </ResourceLinkPool>
  <AuditPool>
    <Created Author="Rainer's JDFWriter 0.2000" TimeStamp="2005-06-01T10:26:11+01:00"/>
    <Modified Author="EatJDF Complete: task=*" TimeStamp="2005-06-01T10:26:57+01:00"/>
    <PhaseTime End="2005-06-01T10:26:57+01:00" Start="2005-06-01T10:26:57+01:00"
Status="Setup" TimeStamp="2005-06-01T10:26:57+01:00"/>
    <PhaseTime End="2005-06-01T10:26:57+01:00" Start="2005-06-01T10:26:57+01:00"
Status="InProgress" TimeStamp="2005-06-01T10:26:57+01:00"/>
    <PhaseTime End="2005-06-01T10:26:57+01:00" Start="2005-06-01T10:26:57+01:00"
Status="Cleanup" TimeStamp="2005-06-01T10:26:57+01:00"/>
  </AuditPool>
```

```

    <ProcessRun End="2005-06-01T10:26:57+01:00" EndStatus="Completed" Start="2005-06-
01T10:26:57+01:00" TimeStamp="2005-06-01T10:26:57+01:00"/>
  </AuditPool>
</JDF>

```

N.2 Product JDF

The following example describe a pair of college textbooks, one teachers edition and one students edition as product intent. Most intent resources are intentionally left empty.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="bookTest" JobID="bookJob"
Status="Waiting" Type="Product" Version="1.3">
  <ResourcePool>
    <Component Amount="100" Class="Quantity" DescriptiveName="Teacher's Book"
ID="Link0003" Status="Unavailable"/>
    <Component Amount="2000" Class="Quantity" DescriptiveName="Cover" ID="Link0005"
Status="Unavailable">
      <!--This cover is reused by both-->
    </Component>
    <LayoutIntent Class="Intent" ID="Link0006" Status="Available">
      <Dimensions DataType="XYPairSpan" Preferred="612 792" Range="576 756 ~ 648 828"/>
    </LayoutIntent>
    <LayoutIntent Class="Intent" ID="Link0008" Status="Available">
      <Dimensions DataType="XYPairSpan" Preferred="612 792" Range="576 756 ~ 648 828"/>
      <Pages DataType="IntegerSpan" Preferred="240"/>
    </LayoutIntent>
    <Component Amount="1000" Class="Quantity" DescriptiveName="Student's Book"
ID="Link0011" Status="Unavailable">
      <!--Students Book Intent-->
    </Component>
    <LayoutIntent Class="Intent" ID="Link0014" Status="Available">
      <Dimensions DataType="XYPairSpan" Preferred="612 792" Range="576 756 ~ 648 828"/>
      <Pages DataType="IntegerSpan" Preferred="198"/>
    </LayoutIntent>
  </ResourcePool>
  <AuditPool>
    <Created Author="Rainer's JDFWriter 0.2000" TimeStamp="2000-11-01T12:46:56+01:00"/>
  </AuditPool>
  <JDF DescriptiveName="Teacher's Edition" ID="Link0002" JobPartID="0" Status="Waiting"
Type="Product">
    <ResourcePool>
      <Component Amount="100" Class="Quantity" DescriptiveName="Insert" ID="Link0009"
Status="Unavailable"/>
    </ResourcePool>
    <ResourceLinkPool>
      <ComponentLink Amount="100" Usage="Output" rRef="Link0003"/>
      <ComponentLink Amount="100" Usage="Input" rRef="Link0009"/>
      <ComponentLink Amount="100" Usage="Input" rRef="Link0005"/>
    </ResourceLinkPool>
    <JDF DescriptiveName="Teacher's Insert" ID="Link0007" JobPartID="2"
Status="Waiting" Type="Product">
      <ResourceLinkPool>
        <LayoutIntentLink Usage="Input" rRef="Link0008"/>
        <ComponentLink Amount="100" Usage="Output" rRef="Link0009"/>
      </ResourceLinkPool>
    </JDF>
  </JDF>
  <JDF DescriptiveName="Cover" ID="Link0004" JobPartID="1" Status="Waiting"
Type="Product">

```

```

    <ResourceLinkPool>
      <ComponentLink Amount="2000" Usage="Output" rRef="Link0005"/>
      <LayoutIntentLink Usage="Input" rRef="Link0006"/>
    </ResourceLinkPool>
  </JDF>
  <JDF DescriptiveName="Student's Edition" ID="Link0010" JobPartID="3" Status="Waiting"
Type="Product">
    <ResourcePool>
      <Component Amount="1000" Class="Quantity" DescriptiveName="Insert" ID="Link0013"
Status="Unavailable"/>
    </ResourcePool>
    <ResourceLinkPool>
      <ComponentLink Amount="1000" Usage="Output" rRef="Link0011"/>
      <ComponentLink Amount="1000" Usage="Input" rRef="Link0013"/>
      <ComponentLink Amount="1000" Usage="Input" rRef="Link0005"/>
    </ResourceLinkPool>
    <JDF DescriptiveName="Student's Insert" ID="Link0012" JobPartID="4"
Status="Waiting" Type="Product">
      <ResourceLinkPool>
        <ComponentLink Amount="1000" Usage="Output" rRef="Link0013"/>
        <LayoutIntentLink Usage="Input" rRef="Link0014"/>
      </ResourceLinkPool>
    </JDF>
  </JDF>
</JDF>

```

N.3 Spawning and Merging

The following set of examples show a JDF job in the relevant stages of spawning and merging. One example defines a simple brochure with a cover and an insert. The node in green emphasis, which defines the cover, is spawned, modified, and subsequently merged. Elements in red emphasis represent metadata that apply to spawning and merging.

N.3.1 Example 2 Component JDF before Spawning

The following JDF file describes a two-component brochure. The resources are not fleshed out.

```

<JDF ID="SpawnTest" Type="Product" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" Version="1.3" JobPartID="Part1">
  <AuditPool>
    <Created Author="CIP4 JDFWriter 1.0.1 beta" TimeStamp="2002-04-05T15:27:58+02:00"/>
  </AuditPool>
  <ResourcePool>
    <Component ID="r0043" Class="Quantity" Amount="10000" Status="Unavailable"/>
    <BindingIntent ID="r0044" Class="Intent" Status="Available"/>
    <ProductionIntent ID="r0045" Class="Intent" Status="Available">
      <PrintProcess Range="Gravure" DataType="EnumerationSpan"/>
    </ProductionIntent>
    <Component ID="r0047" Class="Quantity" Status="Unavailable"/>
    <Component ID="r0051" Class="Quantity" Status="Unavailable"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ComponentLink rRef="r0043" Usage="Output"/>
    <BindingIntentLink rRef="r0044" Usage="Input"/>
    <ProductionIntentLink rRef="r0045" Usage="Input"/>
    <ComponentLink rRef="r0047" Usage="Input"/>
    <ComponentLink rRef="r0051" Usage="Input"/>
  </ResourceLinkPool>
  <JDF ID="n0046" Type="Product" Status="Waiting" JobPartID="Part2"
DescriptiveName="Cover">
    <ResourceLinkPool>

```



```

    <ComponentLink rRef="r0047" Usage="Output"/>
    <LayoutIntentLink rRef="r0048" Usage="Input"/>
    <ColorIntentLink rRef="r0049" Usage="Input"/>
  </ResourceLinkPool>
  <ResourcePool>
    <LayoutIntent ID="r0048" Class="Intent" Status="Available"/>
    <ColorIntent ID="r0049" Class="Intent" Status="Available"/>
  </ResourcePool>
</JDF>
<JDF ID="n0050" Type="Product" Status="Waiting" JobPartID="Part3"
DescriptiveName="Insert">
  <ResourceLinkPool>
    <ComponentLink rRef="r0051" Usage="Output"/>
    <LayoutIntentLink rRef="r0052" Usage="Input"/>
    <ColorIntentLink rRef="r0053" Usage="Input"/>
  </ResourceLinkPool>
  <ResourcePool>
    <LayoutIntent ID="r0052" Class="Intent" Status="Available"/>
    <ColorIntent ID="r0053" Class="Intent" Status="Available"/>
  </ResourcePool>
</JDF>
</JDF>

```

N.3.2 Example 2 Component JDF Parent after spawning the cover node

The following JDF is the parent JDF after spawning. The **Component** that describes the cover is marked as *SpawnedRW*, since it was copied into the spawned node and can be modified. A **Spawned** audit was inserted into the Cover nodes parent's **AuditPool**, and the Spawned node itself has a *Status* of *Spawned*.

```

<JDF ID="SpawnTest" Type="Product" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" Version="1.3" JobPartID="Part1">
  <AuditPool>
    <Created Author="CIP4 JDFWriter 1.0.1 beta" TimeStamp="2002-04-05T15:27:58+02:00"/>
    <Spawned URL="File:///spawn.jdf" jRef="n0046" TimeStamp="2002-04-05T15:34:43+02:00"
NewSpawnID="Sp0057" rRefsRWCopied="r0047"/>
  </AuditPool>
  <ResourcePool>
    <Component ID="r0043" Class="Quantity" Amount="10000" Status="Unavailable"/>
    <BindingIntent ID="r0044" Class="Intent" Status="Available"/>
    <ProductionIntent ID="r0045" Class="Intent" Status="Available">
      <PrintProcess Range="Gravure" DataType="EnumerationSpan"/>
    </ProductionIntent>
    <Component ID="r0047" Class="Quantity" Status="Unavailable" SpawnIDs="Sp0057"
SpawnStatus="SpawnedRW"/>
    <Component ID="r0051" Class="Quantity" Status="Unavailable"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ComponentLink rRef="r0043" Usage="Output"/>
    <BindingIntentLink rRef="r0044" Usage="Input"/>
    <ProductionIntentLink rRef="r0045" Usage="Input"/>
    <ComponentLink rRef="r0047" Usage="Input"/>
    <ComponentLink rRef="r0051" Usage="Input"/>
  </ResourceLinkPool>
  <JDF ID="n0046" Type="Product" Status="Spawned" JobPartID="Part2"
DescriptiveName="Cover">
    <ResourceLinkPool>
      <ComponentLink rRef="r0047" Usage="Output"/>
      <LayoutIntentLink rRef="r0048" Usage="Input"/>
      <ColorIntentLink rRef="r0049" Usage="Input"/>
    </ResourceLinkPool>

```

Appendix N Examples

```
<ResourcePool>
  <LayoutIntent ID="r0048" Class="Intent" Status="Available" SpawnIDs="Sp0057"
SpawnStatus="SpawnedRO"/>
  <ColorIntent ID="r0049" Class="Intent" Status="Available" SpawnIDs="Sp0057"
SpawnStatus="SpawnedRO"/>
</ResourcePool>
</JDF>
<JDF ID="n0050" Type="Product" Status="Waiting" JobPartID="Part3"
DescriptiveName="Insert">
  <ResourceLinkPool>
    <ComponentLink rRef="r0051" Usage="Output"/>
    <LayoutIntentLink rRef="r0052" Usage="Input"/>
    <ColorIntentLink rRef="r0053" Usage="Input"/>
  </ResourceLinkPool>
  <ResourcePool>
    <LayoutIntent ID="r0052" Class="Intent" Status="Available"/>
    <ColorIntent ID="r0053" Class="Intent" Status="Available"/>
  </ResourcePool>
</JDF>
</JDF>
```

N.3.3 Example 2 Component JDF spawned node

The Component that represents the cover was copied into the spawned node, since it is the output resource. It is not locked, since it was spawned in RW mode. The existence of an AncestorPool denotes the node as spawned and defines the parent node

```
<JDF ID="n0046" Type="Product" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" SpawnID="Sp0057" Version="1.3" JobPartID="Part2"
DescriptiveName="Cover">
  <AuditPool>
    <Created Author="CIP4 JDFWriter 1.0.1 beta" TimeStamp="2002-04-05T15:34:43+02:00"/>
  </AuditPool>
  <ResourceLinkPool>
    <ComponentLink rRef="r0047" Usage="Output"/>
    <LayoutIntentLink rRef="r0048" Usage="Input"/>
    <ColorIntentLink rRef="r0049" Usage="Input"/>
  </ResourceLinkPool>
  <ResourcePool>
    <LayoutIntent ID="r0048" Class="Intent" Status="Available"/>
    <ColorIntent ID="r0049" Class="Intent" Status="Available"/>
    <Component ID="r0047" Class="Quantity" Status="Available" SpawnIDs="Sp0057"/>
  </ResourcePool>
  <AncestorPool>
    <Ancestor NodeID="SpawnTest" FileName="testjdf4.jdf"/>
  </AncestorPool>
</JDF>
```

N.3.4 Example 2 Component JDF after merging

In this example, it is assumed that the cover output component was created by some processor that processed the spawned node. This resulted in the Component becoming available. The Component was also removed from the copy of the spawned node, since it would otherwise exist twice.

```
<JDF ID="SpawnTest" Type="Product" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" Version="1.2" JobPartID="Part1">
  <AuditPool>
    <Created Author="CIP4 JDFWriter 1.0.1 beta" TimeStamp="2002-04-05T15:27:58+02:00"/>
    <Spawned URL="File:///spawn.jdf" jRef="n0046" TimeStamp="2002-04-05T15:34:43+02:00"
NewSpawnID="Sp0057" rRefsRWCopied="r0047"/>
  </AuditPool>
</JDF>
```

```

    <Merged URL="File:///spawn.jdf" jRef="n0046" MergeID="Sp0057" TimeStamp="2002-04-
05T15:40:20+02:00" rRefsOverwritten="r0047"/>
  </AuditPool>
  <ResourcePool>
    <Component ID="r0043" Class="Quantity" Amount="10000" Status="Unavailable"/>
    <BindingIntent ID="r0044" Class="Intent" Status="Available"/>
    <ProductionIntent ID="r0045" Class="Intent" Status="Available">
      <PrintProcess Range="Gravure" DataType="EnumerationSpan"/>
    </ProductionIntent>
    <Component ID="r0047" Class="Quantity" Status="Available"/>
    <Component ID="r0051" Class="Quantity" Status="Unavailable"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ComponentLink rRef="r0043" Usage="Output"/>
    <BindingIntentLink rRef="r0044" Usage="Input"/>
    <ProductionIntentLink rRef="r0045" Usage="Input"/>
    <ComponentLink rRef="r0047" Usage="Input"/>
    <ComponentLink rRef="r0051" Usage="Input"/>
  </ResourceLinkPool>
  <JDF ID="n0046" Type="Product" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" Version="1.2" JobPartID="Part2" DescriptiveName="Cover">
    <AuditPool>
      <Created Author="CIP4 JDFWriter 1.0.1 beta" TimeStamp="2002-04-
05T15:34:43+02:00"/>
    </AuditPool>
    <ResourceLinkPool>
      <ComponentLink rRef="r0047" Usage="Output"/>
      <LayoutIntentLink rRef="r0048" Usage="Input"/>
      <ColorIntentLink rRef="r0049" Usage="Input"/>
    </ResourceLinkPool>
    <ResourcePool>
      <LayoutIntent ID="r0048" Class="Intent" Status="Available"/>
      <ColorIntent ID="r0049" Class="Intent" Status="Available"/>
    </ResourcePool>
  </JDF>
  <JDF ID="n0050" Type="Product" Status="Waiting" JobPartID="Part3"
DescriptiveName="Insert">
    <ResourceLinkPool>
      <ComponentLink rRef="r0051" Usage="Output"/>
      <LayoutIntentLink rRef="r0052" Usage="Input"/>
      <ColorIntentLink rRef="r0053" Usage="Input"/>
    </ResourceLinkPool>
    <ResourcePool>
      <LayoutIntent ID="r0052" Class="Intent" Status="Available"/>
      <ColorIntent ID="r0053" Class="Intent" Status="Available"/>
    </ResourcePool>
  </JDF>
</JDF>

```

N.3.5 Example of a Partitioned ImageSetting Node before Spawning

The following example shows a simple ImageSetting node that is partitioned by Separation. The resources are not filled with data. The input resources are Available.

```

<JDF ID="n20020701190951" Type="ImageSetting" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" Version="1.3">
  <ResourcePool>
    <ImageSetterParams ID="r0052" Class="Parameter" Locked="false" Status="Available"/>
    <Media ID="r0053" Class="Consumable" Locked="false" Status="Available"/>
  </ResourcePool>

```

```

    <ExposedMedia ID="r0054" Class="Handling" Locked="false" Status="Unavailable"
PartIDKeys="Separation">
    <MediaRef rRef="r0053"/>
    <ExposedMedia Separation="Cyan"/>
    <ExposedMedia Separation="Magenta"/>
    <ExposedMedia Separation="Yellow"/>
    <ExposedMedia Separation="Black"/>
    </ExposedMedia>
    <RunList ID="r0055" Class="Parameter" Locked="false" Status="Available"/>
</ResourcePool>
<ResourceLinkPool>
    <ImageSetterParamsLink rRef="r0052" Usage="Input"/>
    <MediaLink rRef="r0053" Usage="Input"/>
    <ExposedMediaLink rRef="r0054" Usage="Output"/>
    <RunListLink rRef="r0055" Usage="Input"/>
</ResourceLinkPool>
</JDF>

```

N.3.6 The Spawned Cyan Partition of the ImageSetting Node

The following example shows the spawned Cyan partition of the ImageSetting node from the previous example.

```

<JDF ID="n20020701190951" Type="ImageSetting" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" SpawnID="Sp0059" Version="1.3">
  <AuditPool/>
  <ResourcePool>
    <ImageSetterParams ID="r0052" Class="Parameter" Locked="true" Status="Available"/>
    <Media ID="r0053" Class="Consumable" Locked="true" Status="Available"/>
    <ExposedMedia ID="r0054" Class="Handling" Locked="true" Status="Unavailable"
PartIDKeys="Separation">
      <ExposedMedia Separation="Cyan"/>
    </ExposedMedia>
    <RunList ID="r0055" Class="Parameter" Locked="true" Status="Available"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ImageSetterParamsLink rRef="r0052" Usage="Input"/>
    <MediaLink rRef="r0053" Usage="Input">
      <Part Separation="Cyan"/>
    </MediaLink>
    <ExposedMediaLink rRef="r0054" Usage="Output">
      <Part Separation="Cyan"/>
    </ExposedMediaLink>
    <RunListLink rRef="r0055" Usage="Input"/>
  </ResourceLinkPool>
  <AncestorPool>
    <Part Separation="Cyan"/>
    <Ancestor Type="ImageSetting" xmlns="http://www.CIP4.org/JDFSchema_1_1"
NodeID="n20020701190951" Status="Waiting" Version="1.2" FileName="testjdf5.jdf"/>
  </AncestorPool>
</JDF>

```

N.3.7 The Root Partitioned ImageSetting Node after Spawning

```

<JDF ID="n20020701190951" Type="ImageSetting" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Pool" Version="1.3">
  <AuditPool>
    <Spawned URL="File:///spawnIS.jdf" jRef="n20020701190951" Status="Waiting"
TimeStamp="2002-07-01T19:18:03+02:00" NewSpawnID="Sp0059">
      <Part Separation="Cyan"/>
    </Spawned>
  </AuditPool>

```

```

<ResourcePool>
  <NodeInfo Class="Parameter" ID="r050722154232_0045" NodeStatus="Waiting"
  PartIDKeys="Separation" Status="Available">
    <NodeInfo NodeStatus="Spawned" Separation="Cyan"/>
  </NodeInfo>
  <ImageSetterParams ID="r0052" Class="Parameter" Locked="false" Status="Available"
  SpawnIDs="Sp0059" SpawnStatus="SpawnedRO"/>
  <Media ID="r0053" Class="Consumable" Locked="false" Status="Available"
  SpawnIDs="Sp0059"/>
  <ExposedMedia ID="r0054" Class="Handling" Locked="false" Status="Unavailable"
  SpawnIDs="Sp0059" PartIDKeys="Separation">
    <MediaRef rRef="r0053"/>
    <ExposedMedia Locked="true" Separation="Cyan" SpawnStatus="SpawnedRW"/>
    <ExposedMedia Separation="Magenta"/>
    <ExposedMedia Separation="Yellow"/>
    <ExposedMedia Separation="Black"/>
  </ExposedMedia>
  <RunList ID="r0055" Class="Parameter" Locked="false" Status="Available"
  SpawnIDs="Sp0059" SpawnStatus="SpawnedRO"/>
</ResourcePool>
<ResourceLinkPool>
  <ImageSetterParamsLink rRef="r0052" Usage="Input"/>
  <MediaLink rRef="r0053" Usage="Input"/>
  <ExposedMediaLink rRef="r0054" Usage="Output"/>
  <RunListLink rRef="r0055" Usage="Input"/>
  <NodeInfoLink Usage="Input" rRef="r050722154232_0045"/>
</ResourceLinkPool>
</JDF>

```

N.3.8 The Merged ImageSetting Node

The Node has now been executed and merged.

```

<JDF ID="n20020701190951" Type="ImageSetting" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Pool" Version="1.3">
  <AuditPool>
    <Spawned URL="File:///spawnIS.jdf" jRef="n20020701190951" Status="Waiting"
    TimeStamp="2002-07-01T20:25:03+02:00" NewSpawnID="Sp0059">
      <Part Separation="Cyan"/>
    </Spawned>
    <Merged URL="File:///spawnIS2.jdf" jRef="n20020701190951" MergeID="Sp0059"
    TimeStamp="2002-07-01T20:27:51+02:00">
      <Part Separation="Cyan"/>
    </Merged>
  </AuditPool>
  <ResourcePool>
    <ImageSetterParams ID="r0052" Class="Parameter" Status="Available"/>
    <Media ID="r0053" Class="Consumable" Status="Available"/>
    <ExposedMedia ID="r0054" Class="Handling" Status="Unavailable"
    PartIDKeys="Separation">
      <MediaRef rRef="r0053"/>
      <ExposedMedia Status="Available" Separation="Cyan"/>
      <ExposedMedia Separation="Magenta"/>
      <ExposedMedia Separation="Yellow"/>
      <ExposedMedia Separation="Black"/>
    </ExposedMedia>
    <RunList ID="r0055" Class="Parameter" Status="Available"/>
    <NodeInfo Class="Parameter" ID="r050722154949_0045" NodeStatus="Waiting"
    PartIDKeys="Separation" Status="Available">
      <NodeInfo NodeStatus="Completed" Separation="Cyan"/>
    </NodeInfo>
  </ResourcePool>
</JDF>

```

```

    </NodeInfo>
  </ResourcePool>
  <ResourceLinkPool>
    <ImageSetterParamsLink rRef="r0052" Usage="Input"/>
    <MediaLink rRef="r0053" Usage="Input"/>
    <ExposedMediaLink rRef="r0054" Usage="Output"/>
    <RunListLink rRef="r0055" Usage="Input"/>
    <NodeInfoLink Usage="Input" rRef="r050722154232_0045"/>
  </ResourceLinkPool>
</JDF>

```

N.4 RunList

The following example shows the various separation types, all mixed into one big RunList. Both in-line and ResourceRef versions of **LayoutElement** are used.

```

<ResourcePool>
  <Runlist Class="Parameter" ID="Link0003" NPage="10" PartIDKeys="Run Separation"
    Status="Available">
    <Comment>Preseparated Runs in multiple files
    All LayoutElements are inline resources
    </Comment>
    <RunList FirstPage="0" NPage="1" Run="1">
      <RunList Separation="Cyan">
        <LayoutElement Status="Unavailable">
          <FileSpec URL="File:///Cyan.pdf"/>
        </LayoutElement>
      </RunList>
      <RunList Separation="Magenta">
        <LayoutElement Status="Unavailable">
          <FileSpec URL="File:///Magenta.pdf"/>
        </LayoutElement>
      </RunList>
      <RunList Separation="Yellow">
        <LayoutElement Status="Unavailable">
          <FileSpec URL="File:///Yellow.pdf"/>
        </LayoutElement>
      </RunList>
      <RunList Separation="Black">
        <LayoutElement Status="Unavailable">
          <FileSpec URL="File:///Black.pdf"/>
        </LayoutElement>
      </RunList>
      <RunList Separation="SpotGreen">
        <LayoutElement Status="Unavailable">
          <FileSpec URL="File:///Green.pdf"/>
        </LayoutElement>
      </RunList>
    </RunList>
    <RunList NPage="2" Run="2" SkipPage="4">
      <Comment>
        Preseparated Runs in one file CMYKGCMYKG
        LayoutElements are inter-resource links
      </Comment>
      <RunList FirstPage="0" Separation="Cyan">
        <LayoutElementRef rRef="Link0004"/>
      </RunList>
      <RunList FirstPage="1" Separation="Magenta">
        <LayoutElementRef rRef="Link0004"/>
      </RunList>
    </RunList>
  </Runlist>
</ResourcePool>

```

```

<RunList FirstPage="2" Separation="Yellow">
  <LayoutElementRef rRef="Link0004"/>
</RunList>
<RunList FirstPage="3" Separation="Black">
  <LayoutElementRef rRef="Link0004"/>
</RunList>
<RunList FirstPage="4" Separation="SpotGreen">
  <LayoutElementRef rRef="Link0004"/>
</RunList>
</RunList>
<RunList NPage="1" Run="3" SkipPage="3">
  <Comment>
    No Magenta, the missing sep does not exist as a page
  </Comment>
  <RunList FirstPage="10" Separation="Cyan">
    <LayoutElementRef rRef="Link0004"/>
  </RunList>
  <RunList FirstPage="11" Separation="Yellow">
    <LayoutElementRef rRef="Link0004"/>
  </RunList>
  <RunList FirstPage="12" Separation="Black">
    <LayoutElementRef rRef="Link0004"/>
  </RunList>
  <RunList FirstPage="13" Separation="Green">
    <LayoutElementRef rRef="Link0004"/>
  </RunList>
</RunList>
<RunList NPage="2" Run="4" SkipPage="4">
  <Comment>
    Continuation of Preseparated Runs in one file CMYKGCMYKG -
    the missing sep of the previous page does not exist as a page
  </Comment>
  <RunList FirstPage="14" Separation="Cyan">
    <LayoutElementRef rRef="Link0004"/>
  </RunList>
  <RunList FirstPage="15" Separation="Magenta">
    <LayoutElementRef rRef="Link0004"/>
  </RunList>
  <RunList FirstPage="16" Separation="Yellow">
    <LayoutElementRef rRef="Link0004"/>
  </RunList>
  <RunList FirstPage="17" Separation="Black">
    <LayoutElementRef rRef="Link0004"/>
  </RunList>
  <RunList FirstPage="18" Separation="SpotGreen">
    <LayoutElementRef rRef="Link0004"/>
  </RunList>
</RunList>
<RunList NPage="2" Run="5">
  <Comment>
    Preseparated Runs in one file CCMMYYKKGG
  </Comment>
  <RunList FirstPage="0" Separation="Cyan">
    <LayoutElementRef rRef="Link0005"/>
  </RunList>
  <RunList FirstPage="2" Separation="Magenta">
    <LayoutElementRef rRef="Link0005"/>
  </RunList>
  <RunList FirstPage="4" Separation="Yellow">

```

```

    <LayoutElementRef rRef="Link0005"/>
  </RunList>
  <RunList FirstPage="6" Separation="Black">
    <LayoutElementRef rRef="Link0005"/>
  </RunList>
  <RunList FirstPage="8" Separation="SpotGreen">
    <LayoutElementRef rRef="Link0005"/>
  </RunList>
</RunList>
<RunList NPage="2" Run="6">
  <Comment>
    Combined Runs in one file
  </Comment>
  <LayoutElement ElementType="document">
    <FileSpec URL="File:///Combined.pdf"/>
  </LayoutElement>
</RunList>
</Runlist>
<LayoutElement Class="Parameter" ID="Link0004" Status="Available">
  <FileSpec URL="File:///PreSepCMYKG.pdf"/>
</LayoutElement>
<LayoutElement Class="Parameter" ID="Link0005" Status="Available">
  <FileSpec URL="File:///PreSepCCMMYYKGG.pdf"/>
</LayoutElement>
</ResourcePool>

```

N.5 Messages

N.5.1 Simple KnownMessages

The following simple example shows a KnownMessages Query and the Response sent by a fairly dumb controller:

Query:

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFClient" TimeStamp="2000-
11-07T13:15:56+01:00" Version="1.3">
  <Query ID="Q0001" Type="KnownMessages">
    <KnownMsgQuParams ListCommands="true" ListQueries="true" ListSignals="false"/>
  </Query>
</JMF>

```

Response:

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFClient #2"
TimeStamp="2000-11-07T13:15:56+01:00" Version="1.3">
  <Response ID="R0001" Type="KnownMessages" refID="Q0001">
    <KnownMessages>
      <MessageService Query="true" Type="KnownMessages"/>
      <MessageService Persistent="true" Query="true" Type="Status"/>
      <MessageService Command="true" Type="StopPersistentChannel"/>
    </KnownMessages>
  </Response>
</JMF>

```

N.5.2 Simple persistent channel

The following query requests a persistent channel for Status messages. An update is requested whenever an attribute changes.


```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFCClient"
  TimeStamp="2000-11-07T16:02:09+01:00" Version="1.3">
  <Query ID="Q0011" Type="Status">
    <Subscription URL="http://123.123.123/message/recipient">
      <ObservationTarget ObservationPath="//*/@*" />
    </Subscription>
    <StatusQuParams JobDetails="brief" />
  </Query>
</JMF>
```

The following four examples are a set of typical, simple responses that are emitted whenever *DeviceStatus* changes.

This is the **Response** that is sent immediately within the same HTTP connection as the **Query**.

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFCClient #2"
  TimeStamp="2000-11-07T16:02:19+01:00" Version="1.2">
  <Response ID="R0013" Type="Status" refID="Q0011">
    <DeviceInfo DeviceStatus="Idle" />
  </Response>
</JMF>
```

This is an intermediate **Signal** that was emitted when *DeviceStatus* changed.

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFCClient #2"
  TimeStamp="2000-11-07T17:02:19+01:00" Version="1.2">
  <Signal ID="Q0015" Type="Status" refID="Q0011">
    <DeviceInfo DeviceStatus="Setup" />
  </Signal>
</JMF>
```

This is an intermediate **Signal** that was emitted when *DeviceStatus* changed.

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFCClient #2" TimeStamp="2000-
11-07T17:08:19+01:00" Version="1.2">
  <Signal ID="Q0017" Type="Status" refID="Q0011">
    <DeviceInfo DeviceStatus="Running" />
  </Signal>
</JMF>
```

This is the last **Signal** of the persistent channel.

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFCClient #2" TimeStamp="2000-
11-07T19:02:19+01:00" Version="1.2">
  <Signal ID="Q0017" Type="Status" refID="Q0011" LastRepeat="true">
    <DeviceInfo DeviceStatus="Idle" />
  </Signal>
</JMF>
```

N.6 Stripping

N.6.1 Using Position

The following example illustrates the more advanced use of the *Position* object. Note that the two B2x2 signatures are filled independently.



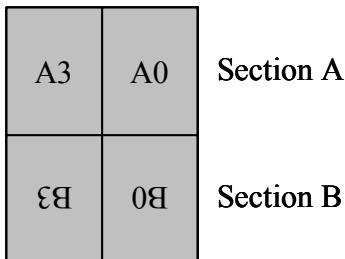
```

<BinderySignature Class="Parameter" ID="B4x2" NumberUp="4 2" Status="Available"/>
<BinderySignature Class="Parameter" ID="B2x2" NumberUp="2 2" Status="Available"/>
<StrippingParams Class="Parameter" ID="L1" PartIDKeys="SheetName
BinderySignatureName" Status="Available" WorkStyle="WorkAndBack">
  <StrippingParams SheetName="Sheet1">
    <StrippingParams BinderySignatureName="B4x2">
      <BinderySignatureRef rRef="B4x2"/>
      <Position RelativeBox="0 0 0.5 1" Orientation="Rotate270"/>
    </StrippingParams>
    <StrippingParams BinderySignatureName="B2x2-1">
      <BinderySignatureRef rRef="B2x2"/>
      <Position RelativeBox="0.5 0 1 0.5"/>
    </StrippingParams>
    <StrippingParams BinderySignatureName="B2x2-2">
      <BinderySignatureRef rRef="B2x2"/>
      <Position RelativeBox="0.5 0.5 1 1"/>
    </StrippingParams>
  </StrippingParams>
</StrippingParams>

```

N.6.2 Multiple Bindery Signatures

The following example illustrates how two identical **BinderySignature** resources that represent the same sections are placed onto a surface. It also shows how **StripCellParams** are overwritten for various sections.



```

<BinderySignature Class="Parameter" ID="B4x2" NumberUp="4 2" Status="Available"/>
<BinderySignature Class="Parameter" ID="B2x2" NumberUp="2 2" Status="Available"/>
<StrippingParams Class="Parameter" ID="L1" PartUsage="Implicit" Status="Available"
PartIDKeys="SheetName BinderySignatureName CellIndex" WorkStyle="WorkAndBack">
  <StrippingParams JobID="Customer Job 1" SheetName="Sheet1">
    <StrippingParams BinderySignatureName="B4x2">
      <BinderySignatureRef rRef="B4x2"/>
      <StripCellParams BleedFace="42" BleedSpine="0" MillingDepth="21"/>
      <Position RelativeBox="0 0 0.5 1" Orientation="Rotate270"/>
    </StrippingParams>
    <StrippingParams BinderySignatureName="B2x2">
      <BinderySignatureRef rRef="B2x2"/>
    </StrippingParams>
  </StrippingParams>

```

```

    <StripCellParams BleedFace="42" BleedSpine="20" MillingDepth="84"/>
    <Position RelativeBox="0.5 0 1 0.5"/>
    <Position RelativeBox="0.5 0.5 1 1"/>
    <StrippingParams CellIndex="3">
      <StripCellParams BleedFace="10" BleedSpine="10" MillingDepth="48"/>
    </StrippingParams>
  </StrippingParams>
</StrippingParams>
</StrippingParams>

```

N.6.3 Multisection Bindery Signatures

The following example illustrates the imposition of a job containing 80 pages using *ComeAndGo*. Five sheets need to be produced each containing two sections.

B5	B4	B1	B8
A4	A5	A8	A1

```

<BinderySignature Class="Parameter" ID="ComeAndGo" NumberUp="4 2" Status="Available">
  <SignatureCell BackPages="1 6 5 2" FrontPages="3 4 7 0" Orientation="Up"
  SectionIndex="0"/>
  <SignatureCell BackPages="6 1 2 5" FrontPages="4 3 0 7" Orientation="Down"
  SectionIndex="1"/>
</BinderySignature>
<StrippingParams Class="Parameter" ID="L1" PartIDKeys="SheetName" Status="Available"
  WorkStyle="WorkAndBack">
  <BinderySignatureRef rRef="ComeAndGo"/>
  <StrippingParams SectionList="0 9" SheetName="Sheet1"/>
  <StrippingParams SectionList="1 8" SheetName="Sheet2"/>
  <StrippingParams SectionList="2 7" SheetName="Sheet3"/>
  <StrippingParams SectionList="3 6" SheetName="Sheet4"/>
  <StrippingParams SectionList="4 5" SheetName="Sheet5"/>
</StrippingParams>

```

N.6.4 Multiple job parts in one imposition

The following example illustrates partitioning by *SectionIndex*. We reuse the *ComeAndGo* **BinderySignature** from the previous example, but map the **BinderySignature** to sections of different job parts.

```

<StrippingParams Class="Parameter" ID="L1" JobID="MyJob" PartIDKeys="SheetName
  SectionIndex" Status="Available" WorkStyle="WorkAndBack">
  <BinderySignatureRef rRef="ComeAndGo"/>
  <StrippingParams SheetName="Sheet1">
    <StrippingParams AssemblyIDs="Book1" SectionIndex="0" SectionList="0"/>
    <StrippingParams AssemblyIDs="Book2" SectionIndex="1" SectionList="9"/>
  </StrippingParams>
  <StrippingParams SheetName="Sheet2">
    <StrippingParams AssemblyIDs="Book1" SectionIndex="0" SectionList="1"/>
    <StrippingParams AssemblyIDs="Book2" SectionIndex="1" SectionList="8"/>
  </StrippingParams>
  <StrippingParams SheetName="Sheet3">
    <StrippingParams AssemblyIDs="Book1" SectionIndex="0" SectionList="2"/>
    <StrippingParams AssemblyIDs="Book2" SectionIndex="1" SectionList="7"/>
  </StrippingParams>

```

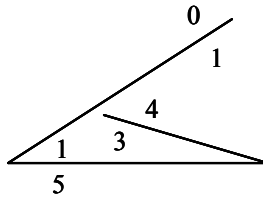
```

</StrippingParams>
<StrippingParams SheetName="Sheet4">
  <StrippingParams AssemblyIDs="Book1" SectionIndex="0" SectionList="3"/>
  <StrippingParams AssemblyIDs="Book2" SectionIndex="1" SectionList="6"/>
</StrippingParams>
<StrippingParams SheetName="Sheet5">
  <StrippingParams AssemblyIDs="Book1" SectionIndex="0" SectionList="4"/>
  <StrippingParams AssemblyIDs="Book2" SectionIndex="1" SectionList="5"/>
</StrippingParams>
</StrippingParams>

```

N.6.5 FoldOuts

The following example illustrates the use of foldouts. The same foldout is placed twice on a press sheet.



4	5	0
4	5	0

```

<BinderySignature Class="Parameter" ID="foldout" NumberUp="2 1" Status="Available">
  <SignatureCell BackFacePages="3" BackPages="2" FrontFacePages="4" FrontPages="5"
Orientation="Up"/>
  <SignatureCell BackPages="1" FrontPages="0" Orientation="Up"/>
</BinderySignature>
<StrippingParams SheetName="Cover" WorkStyle="WorkAndBack">
  <BinderySignatureRef rRef="foldout"/>
  <Position RelativeBox="0 0 1 0.5"/>
  <Position RelativeBox="0 0.5 1 1"/>
</StrippingParams>

```

N.6.6 Multiple Web Layout

The following example illustrates a regular double-web layout. A double-web **BinderySignature** is used in two signatures. This results in four sheets.

91	51	8	£7
31	0	7	24

Web1

81	£1	01	17
29	2	5	26

Web2

```

<BinderySignature Class="Parameter" ID="B001" NumberUp="4 2" PartIDKeys="WebName"
Status="Available">
  <BinderySignature WebName="Web1">
    <SignatureCell BackPages="22 9 14 17" FrontPages="31 0 7 24" Orientation="Up"/>
    <SignatureCell BackPages="25 6 1 30" FrontPages="16 15 8 23" Orientation="Down"/>
  </BinderySignature>
  <BinderySignature WebName="Web2">
    <SignatureCell BackPages="20 11 12 19" FrontPages="29 2 5 26" Orientation="Up"/>
    <SignatureCell BackPages="27 4 3 28" FrontPages="18 13 10 21" Orientation="Down"/>
  </BinderySignature>
</BinderySignature>
<StrippingParams Class="Parameter" ID="MultiWeb1" PartIDKeys="SignatureName SheetName"
Status="Available" WorkStyle="WorkAndBack">
  <StrippingParams SignatureName="Signature1">
    <StrippingParams SheetName="Sheet1">
      <BinderySignatureRef rRef="B001">
        <Part WebName="Web1"/>
      </BinderySignatureRef>
    </StrippingParams>
    <StrippingParams SheetName="Sheet2">
      <BinderySignatureRef rRef="B001">
        <Part WebName="Web2"/>
      </BinderySignatureRef>
    </StrippingParams>
  </StrippingParams>
  <StrippingParams SignatureName="Signature2">
    <StrippingParams SheetName="Sheet3">
      <BinderySignatureRef rRef="B001">
        <Part WebName="Web1"/>
      </BinderySignatureRef>
    </StrippingParams>
    <StrippingParams SheetName="Sheet4">
      <BinderySignatureRef rRef="B001">
        <Part WebName="Web2"/>
      </BinderySignatureRef>
    </StrippingParams>
  </StrippingParams>

```

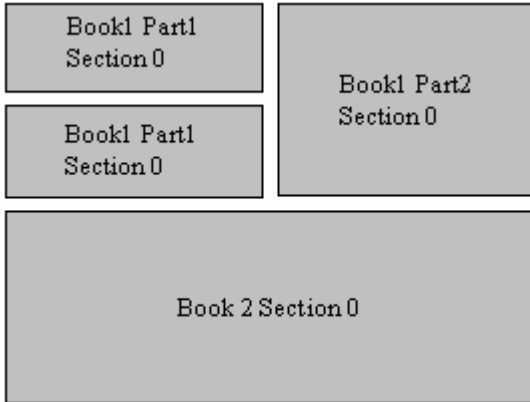
```

    </StrippingParams>
  </StrippingParams>
</StrippingParams>

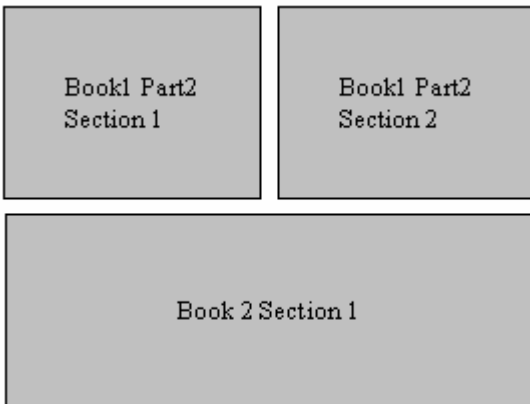
```

N.6.7 Stripping Process

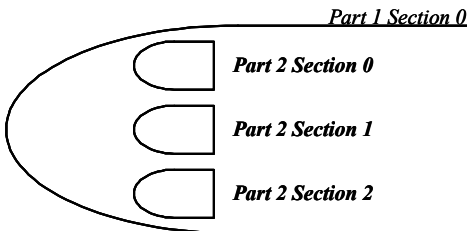
The next sample illustrates the **Stripping** process and its **StrippingParams** and **Assembly** resources. Sheet1:



Sheet2:



Assembly 1



```

<JDF ID="n001" Type="Stripping" Version="1.2">
  <ResourcePool>
    <BinderySignature Class="Parameter" FoldCatalog="F4-1" ID="F4-1"
      Status="Available"/>

```

```

    <BinderySignature Class="Parameter" FoldCatalog="F16-6" ID="F16-6"
Status="Available"/>
    <BinderySignature Class="Parameter" FoldCatalog="F8-7" ID="F8-7"
Status="Available"/>
    <StrippingParams Class="Parameter" ID="L1" PartIDKeys="SheetName
BinderySignatureName" Status="Available">
      <StrippingParams SheetName="Sheet1">
        <StrippingParams AssemblyIDs="Part1" BinderySignatureName="F4-1" JobID="Book1"
SectionList="0">
          <BinderySignatureRef rRef="F4-1"/>
          <Position RelativeBox="0 0.5 0.5 0.75"/>
          <Position RelativeBox="0 0.75 0.5 1"/>
        </StrippingParams>
        <StrippingParams AssemblyIDs="Part2" BinderySignatureName="F8-7" JobID="Book1"
SectionList="0">
          <BinderySignatureRef rRef="F8-7"/>
          <Position RelativeBox="0.5 0.5 1 1"/>
        </StrippingParams>
        <StrippingParams BinderySignatureName="F16-6" JobID="Book2" SectionList="0">
          <BinderySignatureRef rRef="F16-6"/>
          <Position RelativeBox="0 0 1 0.5"/>
        </StrippingParams>
      </StrippingParams>
      <StrippingParams SheetName="Sheet2">
        <StrippingParams AssemblyIDs="Part2" BinderySignatureName="F8-7_1"
JobID="Book1" SectionList="1">
          <BinderySignatureRef rRef="F8-7"/>
          <Position RelativeBox="0 0.5 0.5 1"/>
        </StrippingParams>
        <StrippingParams AssemblyIDs="Part2" BinderySignatureName="F8-7_2"
JobID="Book1" SectionList="2">
          <BinderySignatureRef rRef="F8-7"/>
          <Position RelativeBox="0.5 0.5 1 1"/>
        </StrippingParams>
        <StrippingParams BinderySignatureName="F16-6" JobID="Book2" SectionList="1">
          <BinderySignatureRef rRef="F16-6"/>
          <Position RelativeBox="0 0 1 0.5"/>
        </StrippingParams>
      </StrippingParams>
    </StrippingParams>
  </AssemblySection>
</Assembly>
  <Assembly Class="Parameter" ID="A2" JobID="Book2" Order="Collecting"
Status="Available"/>
  <Layout Class="Parameter" ID="L2" Status="Unavailable"/>
</ResourcePool>
<ResourceLinkPool>
  <StrippingParamsLink Usage="Input" rRef="L1"/>
  <AssemblyLink Usage="Input" rRef="A1"/>
  <AssemblyLink Usage="Input" rRef="A2"/>
  <LayoutLink Usage="Output" rRef="L2"/>
</ResourceLinkPool>
</JDF>

```

N.7 DigitalDelivery Examples

Example 1: Instruct the digital delivery device to compress the files delivered in gzip compression.

Before the delivery:

```
<ResourcePool>
  <RunList ID="SourceFilesLink" Class="Parameter" Status="Available">
    <LayoutElement>
      <FileSpec URL="File:///e:/ToSend/xxx.pdf"/>
    </LayoutElement>
  </RunList>
  <RunList ID="TargetFilesLink" Class="Parameter" Status="Unavailable">
    <LayoutElement>
      <FileSpec Compression="Gzip"/>
    </LayoutElement>
  </RunList>
</ResourcePool>
...
<ResourceLinkPool>
  <RunListLink rRef="SourceFilesLink" Usage="Input"/>
  <RunListLink rRef="TargetFilesLink" Usage="Output"/>
</ResourceLinkPool>
```

Since the input **RunList** resource is without *Compression* and the output **RunList** resource is with *Compression* — it will instruct the digital delivery device to compress the files delivered.

After the delivery:

```
<RunList ID="TargetFilesLink" Class="Parameter" Status="Available">
  <LayoutElement>
    <FileSpec Compression="Gzip" URL="File:///FileServer1/ComingJobs/job702555.gz"/>
  </LayoutElement>
</RunList>
```

Full example of ArtDeliveryIntent translated to Delivery and DigitalDelivery processes

The following example describes:

- 1 Intent with upload file through www form and instruction to return the intermediate files in digital media together with the final product.
- 2 **DigitalDelivery** process sub-jdf describing the upload to ftp server + compression + storage + subscription to get CustomerMessage notification on files when delivered.
- 3 **Delivery** process sub-jdf describing the return of final product and digital media via Fedex with values for service level and tracking id.

```
<JDF xmlns="http://www.CIP4.org/JDFSchemas_1_1" DescriptiveName="ArtDeliveryIntent
translated to Delivery and DigitalDelivery processes" ID="ID000" Status="InProgress"
Type="Product" Version="1.3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <NodeInfo JobPriority="100"/>
  <CustomerInfo CustomerJobName="Job title ...">
    <Contact ContactTypes="Customer">
      <Address City="Alta" PostalCode="36930" Region="AV" Street="123 Gibrish Street"/>
      <Person FamilyName="Spencer" FirstName="Ron"/>
      <ComChannel ChannelType="Phone" ChannelUsage="DayTime"
Locator="+44 019-1234-4567"/>
      <ComChannel ChannelType="Fax" ChannelUsage="Business DayTime NightTime"
Locator="+44 019-1234-4567"/>
    </Contact>
  </CustomerInfo>
  <ResourcePool>
    <ArtDeliveryIntent Class="Intent" ID="Link002" ReturnList="DigitalMedia"
Status="Available">
      <ArtHandling DataType="EnumerationSpan" Range="Return ReturnWithProduct"/>
```



```

    <ReturnMethod DataType="NameSpan" Preferred="FedEx"/>
    <ArtDelivery ArtDeliveryType="DigitalNetwork">
      <Contact ContactTypes="Delivery">
        <ComChannel ChannelType="WWW" ChannelTypeDetails="Form"
          Locator="http://www.server.com/uploader.aspx"/>
      </Contact>
      <RunList>
        <LayoutElement>
          <FileSpec URL="file:///D:/WINNT/Profiles/23423/Desktop/test.pdf"/>
        </LayoutElement>
      </RunList>
    </ArtDelivery>
  </ArtDeliveryIntent>
  <Contact Class="Parameter" ContactTypes="Delivery" ID="Shipping001"
    Status="Available">
    <Address City="Alta" PostalCode="36930" Region="AV" Street="123 Gibrish Street"/>
    <Person FamilyName="Jones" FirstName="Bill"/>
    <ComChannel ChannelType="Phone" ChannelTypeDetails="Mobile"
      Locator="+44 078-1234-4567"/>
  </Contact>
  <Component Amount="500" Class="Quantity" ComponentType="FinalProduct"
    ID="ItemFinal" Status="Unavailable"/>
</ResourcePool>
<ResourceLinkPool>
  <ArtDeliveryIntentLink Usage="Input" rRef="Link002"/>
  <ComponentLink Amount="500" Usage="Output" rRef="ItemFinal"/>
</ResourceLinkPool>
<JDF ID="J171373" Status="Completed" Type="DigitalDelivery">
  <CustomerInfo CustomerJobName="Job title ...">
    <CustomerMessage Language="FR" MessageEvents="Delivered"
      ShowList="StartTime EndTime Error">
      <ComChannel ChannelType="Email" Locator="sender@Email.com"/>
      <ComChannel ChannelType="InstantMessaging" ChannelTypeDetails="Yahoo!"
        Locator="bill_jones"/>
    </CustomerMessage>
  </CustomerInfo>
  <ResourcePool>
    <RunList Class="Parameter" ID="FileListLink1" Status="Available">
      <LayoutElement>
        <FileSpec URL="file:///D:/WINNT/Profiles/23423/Desktop/test.pdf"/>
      </LayoutElement>
    </RunList>
    <DigitalDeliveryParams Class="Parameter" DigitalDeliveryDirection="Push"
      DigitalDeliveryProtocol="FTP" ID="DestinationLink" Method="WebServer"
      Status="Available">
      <Contact ContactTypes="Delivery">
        <ComChannel ChannelType="WWW" ChannelTypeDetails="Form"
          Locator="http://www.server.com/uploader.aspx"/>
      </Contact>
      <Contact ContactTypes="Sender">
        <ComChannel ChannelType="Email" Locator="sender@Email.com"/>
      </Contact>
    </DigitalDeliveryParams>
    <RunList Class="Parameter" ID="FileListLink2" Status="Available">
      <Disposition MinDuration="P30D"/>
      <LayoutElement>
        <FileSpec Compression="Deflate" URL="test.pdf">
          <Container>
            <FileSpec MimeType="application/zip"

```

```

        URL="file://network_share/uploaded%20files/test.zip"/>
    </Container>
</FileSpec>
</LayoutElement>
</RunList>
</ResourcePool>
<ResourceLinkPool>
    <DigitalDeliveryParamsLink Usage="Input" rRef="DestinationLink"/>
    <RunListLink Usage="Input" rRef="FileListLink1"/>
    <RunListLink Usage="Output" rRef="FileListLink2"/>
</ResourceLinkPool>
<AuditPool>
    <PhaseTime DescriptiveName="Upload of Job 171373 to Server"
        End="2003-01-08T12:27:56Z" Start="2003-01-08T12:27:40Z" Status="InProgress"
        TimeStamp="2003-01-08T12:27:56Z"/>
    <Created Author="Server uploader 1.51" TimeStamp="2003-01-08T12:27:40Z"/>
    <ProcessRun End="2003-01-08T12:27:56Z" EndStatus="Completed"
        Start="2003-01-08T12:27:40Z" TimeStamp="2003-01-08T12:27:56Z"/>
</AuditPool>
</JDF>
<JDF DescriptiveName="The Return of product and digital media with intermediate
materials" ID="X00000" Status="Waiting" Type="Delivery">
    <ResourceLinkPool>
        <ComponentLink Usage="Output" rRef="Item001"/>
        <DigitalMediaLink Usage="Output" rRef="Item002"/>
        <DeliveryParamsLink Usage="Input" rRef="Delivery001"/>
    </ResourceLinkPool>
    <ResourcePool>
        <RunList Class="Parameter" ID="FileListLink0" PartIDKeys="Run"
            Status="Available">
            <RunList Run="1">
                <LayoutElement>
                    <FileSpec URL="./ForReturn/Intermediate/test.pdf"/>
                </LayoutElement>
            </RunList>
            <RunList Run="2">
                <LayoutElement>
                    <FileSpec URL="./ForReturn/Final/test.pdf"/>
                </LayoutElement>
            </RunList>
        </RunList>
        <Component Amount="500" Class="Quantity" ComponentType="FinalProduct"
            ID="Item001" ProductID="AG5678" Status="Available" Unit="1"/>
        <DigitalMedia Amount="1" Capacity="700" Class="Handling" ID="Item002"
            MediaLabel="TempResults" MediaType="CD" Status="Available">
            <RunListRef rRef="FileListLink0"/>
        </DigitalMedia>
        <DeliveryParams Class="Parameter" ID="Delivery001" Status="Available">
            <Drop Method="FedEx" ServiceLevel="Ground" TrackingID="1234567890Z">
                <ContactRef rRef="Shipping001"/>
                <DropItem Amount="500" Unit="1">
                    <ComponentRef rRef="Item001"/>
                </DropItem>
                <DropItem Amount="1">
                    <DigitalMediaRef rRef="Item002"/>
                </DropItem>
            </Drop>
        </DeliveryParams>
    </ResourcePool>

```

```
</JDF>
</JDF>
```

Full example of digital delivery through central server

The following example describes:

- 1 Upload of files to server by FTP protocol
- 2 Request for 10 days storage on server
- 3 Request to Mac Binary encode the files on server
- 4 Send to multiple destinations: 1 is email address and 1 is registered address in a private directory
- 5 Download of files by HTTP protocol
- 6 Decode from Mac Binary when downloading to target
- 7 Download by 1 destination out of the 2.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" DescriptiveName="Digital Delivery
through central server - example with process group" ID="ID000" Status="InProgress"
Type="ProcessGroup" Version="1.3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <NodeInfo JobPriority="60"/>
  <ResourcePool>
    <Contact Class="Parameter" ContactTypes="Delivery" ID="DestLink"
      PartIDKeys="Location" Status="Available">
      <Contact Location="Dest1">
        <ComChannel ChannelType="Email" Locator="Reciever1@hotmail.com"/>
      </Contact>
      <Contact Location="Dest2">
        <ComChannel ChannelType="PrivateDirectory" ChannelTypeDetails="VioAddress"
          Locator="Best Workgroup@Best Company"/>
      </Contact>
    </Contact>
  </ResourcePool>
  <RunList Class="Parameter" ID="TempFileListLink" PartIDKeys="Run"
    Status="Available">
    <Disposition MinDuration="P10D"/>
    <RunList Run="1">
      <LayoutElement>
        <FileSpec Compression="MacBinary" URL="./Atlas/Europe.bmp.bin"/>
      </LayoutElement>
    </RunList>
    <RunList Run="2">
      <LayoutElement>
        <FileSpec Compression="MacBinary" URL="./Atlas/America.jpg.bin"/>
      </LayoutElement>
    </RunList>
  </RunList>
  <JDF DescriptiveName="Upload Job to Server" ID="ID001" JobID="J702555"
    Status="Completed" Type="DigitalDelivery">
    <CustomerInfo CustomerJobName="World atlas maps #2">
      <CustomerMessage Language="FR" MessageEvents="Delivered Accepted">
        <ComChannel ChannelType="Email" Locator="sender@email.com"/>
      </CustomerMessage>
    </CustomerInfo>
  </ResourcePool>
  <RunList Class="Parameter" Directory="file:///c:/MyDir/JobForSend"
    ID="SourceFileListLink0" PartIDKeys="Run" Status="Available">
    <RunList Run="1">
```

```

    <LayoutElement>
      <FileSpec FileSize="240066" URL="./Atlas/Europe.bmp"/>
    </LayoutElement>
  </RunList>
</RunList Run="2">
  <LayoutElement>
    <FileSpec FileSize="33947" URL="./Atlas/America.jpg"/>
  </LayoutElement>
</RunList>
</RunList>
<Contact Class="Parameter" ContactTypes="Sender" ID="SendLink"
  Status="Available">
  <ComChannel ChannelType="Email" Locator="sender@email.com"/>
</Contact>
<DigitalDeliveryParams Class="Parameter" DigitalDeliveryDirection="Push"
  DigitalDeliveryProtocol="FTP" ID="DestinationLink0" Method="Vio"
  PartIDKeys="Location" Status="Available">
  <Comment Name="Instruction">
    Please take these maps and add them to the rest ...
  </Comment>
  <DigitalDeliveryParams Location="SenderToDest1">
    <ContactRef rRef="SendLink"/>
    <ContactRef rRef="DestLink">
      <Part Location="Dest1"/>
    </ContactRef>
  </DigitalDeliveryParams>
  <DigitalDeliveryParams Location="SenderToDest2">
    <ContactRef rRef="SendLink"/>
    <ContactRef rRef="DestLink">
      <Part Location="Dest2"/>
    </ContactRef>
  </DigitalDeliveryParams>
</DigitalDeliveryParams>
</ResourcePool>
<ResourceLinkPool>
  <DigitalDeliveryParamsLink Usage="Input" rRef="DestinationLink0"/>
  <RunListLink Usage="Input" rRef="SourceFileListLink0"/>
  <RunListLink Usage="Output" rRef="TempFileListLink"/>
</ResourceLinkPool>
<AuditPool>
  <ProcessRun DescriptiveName="Upload of Job 702555 to Vio Server"
    End="2002-07-21T10:47:11Z" EndStatus="Completed" Start="2002-07-21T10:45:52Z"
    TimeStamp="2002-07-21T10:47:11Z"/>
  <Created Author="Vio Server 4.3" TimeStamp="2002-07-21T10:45:52Z"/>
</AuditPool>
</JDF>
<JDF DescriptiveName="Download Job from Server to destination" ID="ID002"
  JobID="J702555" Status="Pool" Type="DigitalDelivery">
  <ResourcePool>
    <RunList Class="Parameter" Directory="File:///e:/My%20Download"
      ID="TargetFileListLink1" PartIDKeys="Run" Status="Available">
      <RunList Run="1">
        <LayoutElement>
          <FileSpec FileSize="240066" URL="./Atlas/Europe.bmp"/>
        </LayoutElement>
      </RunList>
      <RunList Run="2">
        <LayoutElement>
          <FileSpec FileSize="33947" URL="./Atlas/America.jpg"/>
        </LayoutElement>
      </RunList>
    </RunList>
  </ResourcePool>
</JDF>

```

```

    </LayoutElement>
  </RunList>
</RunList>
<DigitalDeliveryParams Class="Parameter" DigitalDeliveryDirection="Pull"
DigitalDeliveryProtocol="HTTP" ID="DestinationLink1" Method="Vio"
PartIDKeys="Location" Status="Available">
  <DigitalDeliveryParams Location="ToDest1">
    <ContactRef rRef="DestLink">
      <Part Location="Dest1"/>
    </ContactRef>
  </DigitalDeliveryParams>
  <DigitalDeliveryParams Location="ToDest2">
    <ContactRef rRef="DestLink">
      <Part Location="Dest2"/>
    </ContactRef>
  </DigitalDeliveryParams>
</DigitalDeliveryParams>
</ResourcePool>
<ResourceLinkPool>
  <DigitalDeliveryParamsLink Usage="Input" rRef="DestinationLink1"/>
  <RunListLink Usage="Input" rRef="TempFileListLink"/>
  <RunListLink Usage="Output" rRef="TargetFileListLink1"/>
</ResourceLinkPool>
<StatusPool Status="InProgress">
  <PartStatus Status="Completed">
    <Part Location="ToDest2"/>
  </PartStatus>
</StatusPool>
<AuditPool>
  <Created Author="Vio Server 4.3" TimeStamp="2002-07-21T10:48:57Z"/>
  <ProcessRun DescriptiveName="HTTP Download of Job by Best Workgroup@Best Company"
    End="2002-07-21T10:50:11Z" EndStatus="Completed" Start="2002-07-21T10:48:57Z"
    TimeStamp="2002-07-21T10:50:11Z">
    <Part Location="ToDest2"/>
  </ProcessRun>
</AuditPool>
</JDF>
</JDF>

```


Appendix O New, Deprecated, Modified, & Removed Items

The first column of each table contains an XPath of the changed item. However, for brevity the XPath of an input or output resource in a process is expressed with just the process name and resource name. For example, **DigitalDelivery/RunList** rather than as a correct XPath of `JDF [/@Type = "DigitalDelivery"]//RunList`. Likewise, a message is abbreviated. For example, **UpdateJDF** rather than the correct XPath of `Message [/@Type = "UpdateJDF"]`. When the XPath specifies that an attribute is equal ("=") to some enumeration value, the attribute is in the context of the cited value. Finally Span elements are treated as if they were attributes when "=" is used to specify a value in the XPath notation.

O.1 Compatibility Warnings

Table O-1: Compatibility Warnings

XPath	Table and Page	Description
<code>LayoutElement/@ElementType = "Auxiliary"</code>	Table 7-241 on page 500	was misspelled as <i>Auxilliary</i>
<code>PRItem/@Occurrences</code>	Table 7-281 on page 543	was misspelled as <i>Occurences</i>
<code>PRGroup/@Occurrences</code>	Table 7-283 on page 544	was misspelled as <i>Occurences</i>
<code>PRGroup/PRGroupOccurrence</code>	Table 7-283 on page 544	was misspelled as <i>PRGroupOccurence</i>
<code>PRGroup/PROccurrence</code>	Table 7-283 on page 544	was misspelled as <i>PROccurrence</i>
<code>Abstract PRGroupOccurrenceBase</code>	Table 7-284 on page 546	was misspelled as <i>PRGroupOccurenceBase</i>
<code>PROccurrence/@Occurrences</code>	Table 7-288 on page 547	was misspelled as <i>Occurences</i>
<code>PreflightReportRulePool/@MaxOccurrences</code>	Table 7-289 on page 547	was misspelled as <i>MaxOccurences</i>
<code>OverPrintFlag</code>	Table 7-446 on page 670	was misspelled as <i>OverprintFlag</i>
<code>OverPrintMode</code>	Table 7-446 on page 670	was misspelled as <i>OverprintMode</i>
<code>Continuous</code>	Table C-20 on page 714	was misspelled as <i>Continous</i>

O.2 New Items

Table O-2: New Items (Section 1 of 13)

XPath	Table and Page	Description
<code>Comment/@AgentName</code>	Table 3-2 on page 38	
<code>Comment/@AgentVersion</code>	Table 3-2 on page 38	
<code>Comment/@Author</code>	Table 3-2 on page 38	
<code>Comment/@ID</code>	Table 3-2 on page 38	
<code>Comment/@TimeStamp</code>	Table 3-2 on page 38	
<code>JDF /@Category = "Cutting"</code>	Table 3-4 on page 41	
<code>JDF /@Category = "PublishingPreparation"</code>	Table 3-4 on page 41	
<code>JDF /@Category = "RIPing"</code>	Table 3-4 on page 41	
<code>JDF /@Status = "Suspended"</code>	Table 3-4 on page 41	
<code>JDF /@Status = "Part"</code>	Table 3-4 on page 41	
<code>Resource/@PartUsage = "Sparse"</code>	Table 3-9 on page 54	
<code>Resource/GeneralID</code>	Table 3-9 on page 54	

Table O-2: New Items (Section 2 of 13)

XPath	Table and Page	Description
Resource/SourceResource	Table 3-9 on page 54	
PhysicalResource/@GrossWeight	Table 3-12 on page 61	
PhysicalResource/@LotControl	Table 3-12 on page 61	
PhysicalResource/@Manufacturer	Table 3-12 on page 61	
ResourceLink/@MinLateStatus	Table 3-15 on page 66	
ResourceLink/@MinStatus	Table 3-15 on page 66	
PartAmount/@MinLateStatus	Table 3-17 on page 69	
PartAmount/@MinStatus	Table 3-17 on page 69	
PhysicalLink/@MaxAmount	Table 3-19 on page 70	
PhysicalLink/@MinAmount	Table 3-19 on page 70	
PhysicalLink/Lot	Table 3-19 on page 70	
	Section 3.8.4.1 on page 72	Identification of Physical Resources
Lot	Table 3-20 on page 72	
	Section 3.9.5.5 on page 83	Logical partitions and the Identical element
Identical	Table 3-24 on page 83	
Part/@DeliveryUnit0	Table 3-26 on page 87	For this row and following ones for Table 3-26, also a value of Resource/@PartIDKeys
Part/@DeliveryUnit9	Table 3-26 on page 87	Values from 1 to 8 are also new.
Part/@DocTags	Table 3-26 on page 87	
Part/@Edition	Table 3-26 on page 87	
Part/@EditionVersion	Table 3-26 on page 87	
Part/@PageTags	Table 3-25 on page 86	
Part/@PlateLayout	Table 3-26 on page 87	
Part/@ProductionRun	Table 3-26 on page 87	
Part/@RunSet	Table 3-26 on page 87	
Part/@SetTags	Table 3-26 on page 87	
Part/@SubRun	Table 3-26 on page 87	
Part/@WebProduct	Table 3-26 on page 87	
Notification/@JobID	Table 3-32 on page 103	
Notification/@JobPartID	Table 3-32 on page 103	
PhaseTime/@Status = “Suspended”	Table 3-33 on page 104	
ModulePhase/@CombinedProcessIndex	Table 3-34 on page 106	
ResourceAudit/MISDetails	Table 3-35 on page 107	
JMF/@ICSVersions	Table 5-1 on page 144	
JMF/@MaxVersion	Table 5-1 on page 144	
Query/@AcknowledgeFormat	Table 5-3 on page 148	
Query/@AcknowledgeTemplate	Table 5-3 on page 148	

Table O-2: New Items (Section 3 of 13)

XPath	Table and Page	Description
Query/@AcknowledgeURL	Table 5-3 on page 148	
Query/@AcknowledgeType	Table 5-3 on page 148	
Registration	Section 5.2.6 on page 154	
Registration	Table 5-10 on page 154	
	Section 5.3.3.2 on page 154	Persistent Channels for Commands
Subscription/@MinDelayTime	Table 5-11 on page 155	
KnownMsgQuParams/@ListRegistrations	Table 5-24 on page 163	
MessageService/@GenericAttributes	Table 5-25 on page 163	
MessageService/@JMFRole	Table 5-25 on page 163	
MessageService/@Registration	Table 5-25 on page 163	
MessageService/@URLSchemes	Table 5-25 on page 163	
MessageService/ActionPool	Table 5-25 on page 163	
MessageService/DevCapPool	Table 5-25 on page 163	
MessageService/ModulePool	Table 5-25 on page 163	
MsgFilter/@Family = "Command"	Table 5-27 on page 165	
ModifyNode	Table 5-30 on page 167	
UpdateJDF	Table 5-30 on page 167	
ModifyNode	Section 5.8.2 on page 168	
ModifyNode	Table 5-34 on page 168	
ResourceQuParams/@ResourceID	Table 5-46 on page 173	
ResourceCmdParams/@ResourceID	Table 5-48 on page 175	
ResourceCmdParams/@UpdateMethod	Table 5-48 on page 175	
ResourceInfo/@ModuleID	Table 5-49 on page 177	
ResourceInfo/AmountPool	Table 5-49 on page 177	
DeviceInfo/@DeviceID	Table 5-56 on page 184	
JobPhase/ModuleStatus	Table 5-57 on page 185	
ModuleStatus/@CombinedProcessIndex	Table 5-58 on page 187	
ModuleStatus/@ModuleID	Table 5-58 on page 187	
ModuleStatus/@ModuleIndex	Table 5-58 on page 187	
UpdateJDF	Section 5.8.11 on page 190	
UpdateJDF	Table 5-62 on page 190	New section
RequestQueueEntryParams/@SubmitPolicy	Table 5-82 on page 200	
QueueSubmissionParams/@GangName	Table 5-93 on page 204	
QueueSubmissionParams/@GangPolicy	Table 5-93 on page 204	
QueueEntry/@GangName	Table 5-109 on page 213	
QueueEntry/@GangPolicy	Table 5-109 on page 213	
QueueEntry/@Status= "PendingReturn"	Table 5-109 on page 213	
QueueFilter/@GangNames	Table 5-111 on page 214	
	Section 5.14 on page 215	Gang Jobs
ForceGang	Table 5-112 on page 215	

Table O-2: New Items (Section 4 of 13)

XPath	Table and Page	Description
GangStatus	Table 5-112 on page 215	
ForceGang	Section 5.14.1 on page 215	
ForceGang	Table 5-113 on page 215	
GangCmdFilter element	Table 5-114 on page 215	
GangStatus	Section 5.14.2 on page 216	
GangStatus	Table 5-115 on page 216	
GangQuFilter element	Table 5-116 on page 216	
ResourcePool/ CustomerInfo	Table 6-1 on page 219	
ResourcePool/ MiscConsumable	Table 6-1 on page 219	
ResourcePool/ NodeInfo	Table 6-1 on page 219	
ResourcePool/ UsageCounter	Table 6-1 on page 219	
Bending	Section 6.4.2 on page 226	new process
CylinderLayoutPreparation	Section 6.4.7 on page 228	new process
ImageSetting/ExposedMedia	Table 6-48 on page 231	
LayoutElementProduction/ LayoutElementProductionParams	Table 6-56 on page 234	
PDLCreation	Section 6.4.21 on page 235	new process
RasterReading	Section 6.4.26 on page 238	new process
Stripping/ColorantControl	Table 6-80 on page 242	
ConventionalPrinting/ExposedMedia (Cylinder)	Table 6-86 on page 245	
BoxFolding	Section 6.6.3 on page 249	new process
BoxPacking/Media (Tie)	Table 6-94 on page 249	
BoxPacking/Media (Underlay)	Table 6-94 on page 249	
Collecting/Assembly	Table 6-106 on page 252	
Gathering/Assembly	Table 6-122 on page 257	
WebInlineFinishing	Section 6.6.47 on page 266	new process
DurationSpan/OfferRange	Table 7-4 on page 273	
EnumerationSpan/OfferRange	Table 7-5 on page 274	
IntegerSpan/OfferRange	Table 7-6 on page 274	
NameSpan/OfferRange	Table 7-7 on page 274	
NumberSpan/OfferRange	Table 7-8 on page 275	
OptionSpan/OfferRange	Table 7-9 on page 275	
ShapeSpan/OfferRange	Table 7-10 on page 276	
StringSpan/OfferRange	Table 7-11 on page 276	
TimeSpan/OfferRange	Table 7-12 on page 276	
XYPairSpan/OfferRange	Table 7-13 on page 277	
ArtDeliveryIntent/@Method = "Local"	Table 7-14 on page 277	
ChannelBinding/ChannelBrand	Table 7-20 on page 286	
CoilBinding/CoilBrand	Table 7-21 on page 287	
HardCoverBinding/CoverStyle	Table 7-23 on page 287	

Table O-2: New Items (Section 5 of 13)

XPath	Table and Page	Description
HardCoverBinding/JacketFoldingWidth	Table 7-23 on page 287	
PlasticCombBinding/CombBrand	Table 7-24 on page 289	
RingBinding/BinderBrand	Table 7-25 on page 290	
SoftCoverBinding/EndSheets	Table 7-29 on page 291	
SoftCoverBinding/FoldingWidth	Table 7-29 on page 291	
SoftCoverBinding/FoldingWidthBack	Table 7-29 on page 291	
Tabs/TabBrand	Table 7-32 on page 293	
WireCombBinding/WireCombBrand	Table 7-35 on page 294	
ColorsUsed/ SeparationSpec / <i>@Name = "Varnish"</i>	Table 7-38 on page 297	
DropIntent/ <i>AdditionalAmount</i>	Table 7-40 on page 300	
EmbossingItem/Direction= <i>"Flat"</i>	Table 7-43 on page 303	
LaminatingIntent /Texture	Table 7-49 on page 307	
MediaIntent /MediaType= <i>"CorrugatedBoard"</i>	Table 7-51 on page 311	
MediaIntent /MediaType= <i>"SelfAdhesive"</i>	Table 7-51 on page 311	
MediaIntent /MediaTypeDetails	Table 7-51 on page 311	
PackingIntent /PalletCornerBoards	Table 7-54 on page 317	
ProductionIntent / Resource	Table 7-55 on page 318	
PublishingIntent	Section 7.1.17 on page 320	
ShapeCut/ShapeType= <i>"RoundedRectangle"</i>	Table 7-61 on page 321	
ApprovalPerson/ <i>@ApprovalRole = "Approvinator"</i>	Table 7-65 on page 325	
ApprovalPerson/ <i>@ApprovalRoleDetails</i>	Table 7-65 on page 325	
ApprovalSuccess /ApprovalDetails	Table 7-66 on page 325	
ApprovalDetails	Table 7-67 on page 326	
Assembly / <i>@AssemblyIDs</i>	Table 7-68 on page 326	
Assembly / <i>@JogSide</i>	Table 7-68 on page 326	
Assembly / <i>@PhysicalSection</i>	Table 7-68 on page 326	
Assembly / PageList	Table 7-68 on page 326	
Assembly /PageAssignedList	Table 7-68 on page 326	
AssemblySection / <i>@AssemblyIDs</i>	Table 7-69 on page 328	
AssemblySection /PageAssignedList	Table 7-69 on page 328	
PageAssignedList	Table 7-70 on page 328	
AutomatedOverPrintParams / <i>@KnockOutCMYKWhite</i>	Table 7-72 on page 330	
BarcodeCompParams	Section 7.2.9 on page 330	
BarcodeReproParams	Section 7.2.10 on page 331	
BendingParams	Section 7.2.11 on page 331	
BinderySignature / <i>@BinderySignatureType</i>	Table 7-76 on page 332	
BinderySignature / <i>@BindingOrientation</i>	Table 7-76 on page 332	

Table O-2: New Items (Section 6 of 13)

XPath	Table and Page	Description
BinderySignature/@JogEdge	Table 7-76 on page 332	
BinderySignature/@OutsideGutter	Table 7-76 on page 332	
BinderySignature/@StaggerColumns	Table 7-76 on page 332	
BinderySignature/@StaggerContinuous	Table 7-76 on page 332	
BinderySignature/@StaggerRows	Table 7-76 on page 332	
BinderySignature/DieLayout	Table 7-76 on page 332	
SignatureCell/@Orientation = "Left"	Table 7-77 on page 335	
SignatureCell/@Orientation = "Right"	Table 7-77 on page 335	
SignatureCell/@StationName	Table 7-77 on page 335	
BoxFoldingParams	Section 7.2.14 on page 337	
BoxPackingParams/@ComponentsPerRow	Table 7-83 on page 342	
BoxPackingParams/@Layers	Table 7-83 on page 342	
BoxPackingParams/@Rows	Table 7-83 on page 342	
BoxPackingParams/@Ties	Table 7-83 on page 342	
BoxPackingParams/@UnderLays	Table 7-83 on page 342	
ColorSpaceConversionOp/@SourceCS = "CalGray"	Table 7-112 on page 372	
ColorSpaceConversionOp/@SourceCS = "CalRGB"	Table 7-112 on page 372	
ColorSpaceConversionOp/@SourceCS = "ICCMYK"	Table 7-112 on page 372	
ColorSpaceConversionOp/@SourceCS = "ICCGray"	Table 7-112 on page 372	
ColorSpaceConversionOp/@SourceCS = "ICCLAB"	Table 7-112 on page 372	
ColorSpaceConversionOp/@SourceCS = "ICCRGB"	Table 7-112 on page 372	
Component/@AssemblyIDs	Table 7-119 on page 382	
Component/@CartonTopFlaps	Table 7-119 on page 382	
Component/@ComponentType = "Other"	Table 7-119 on page 382	
Component/@ComponentType = "Proof"	Table 7-119 on page 382	
Component/@PageListIndex	Table 7-119 on page 382	
Component/@ProductType = "BlankBox"	Table 7-119 on page 382	
Component/@ProductType = "FlatBox"	Table 7-119 on page 382	
Component/@ProductType = "Pallet"	Table 7-119 on page 382	
Component/@ProductType = "Newspaper"	Table 7-119 on page 382	
Component/@ProductTypeDetails	Table 7-119 on page 382	
Component/Assembly	Table 7-119 on page 382	
Component/PageList	Table 7-119 on page 382	
ContentList	Section 7.2.40 on page 387	
ConventionalPrintingParams/@NonPrintableMarginBottom	Table 7-125 on page 389	

Table O-2: New Items (Section 7 of 13)

XPath	Table and Page	Description
ConventionalPrintingParams/ <i>@NonPrintableMarginLeft</i>	Table 7-125 on page 389	
ConventionalPrintingParams/ <i>@NonPrintableMarginRight</i>	Table 7-125 on page 389	
ConventionalPrintingParams/ <i>@NonPrintableMarginTop</i>	Table 7-125 on page 389	
Component/ <i>@PrintingType = "WebMultiple"</i>	Table 7-125 on page 389	
Component/ <i>@PrintingType = "WebSingle"</i>	Table 7-125 on page 389	
CutBlock/ <i>@AssemblyIDs</i>	Table 7-134 on page 397	
CutBlock/Assembly	Table 7-134 on page 397	
CylinderLayout	Section 7.2.49 on page 400	
CylinderLayoutPreparationParams	Section 7.2.50 on page 403	
DropItem/ <i>@ActualAmount</i>	Table 7-148 on page 407	
DropItem/ <i>@ActualTotalAmount</i>	Table 7-148 on page 407	
DropItem/ <i>@TotalAmount</i>	Table 7-148 on page 407	
DropItem/ <i>@TotalDimensions</i>	Table 7-148 on page 407	
DropItem/ <i>@TotalVolume</i>	Table 7-148 on page 407	
DropItem/ <i>@TotalWeight</i>	Table 7-148 on page 407	
Device/ <i>@ICSVersions</i>	Table 7-151 on page 410	
Device/ <i>@SecureJMFURL</i>	Table 7-151 on page 410	
Device/Module	Table 7-151 on page 410	
Module	Table 7-154 on page 412	
DieLayout	Section 7.2.61 on page 414	
DigitalPrintingParams/ <i>@Sides</i>	Table 7-161 on page 417	
DigitalPrintingParams/Ink	Table 7-161 on page 417	
Emboss/ <i>@Direction = "Flat"</i>	Table 7-166 on page 423	
ExposedMedia/ <i>@PageListIndex</i>	Table 7-170 on page 426	
ExposedMedia/ <i>@PlateType</i>	Table 7-170 on page 426	
ExposedMedia/PageList	Table 7-170 on page 426	
ExternalImpositionTemplate	Section 7.2.73 on page 426	
FileSpec/ <i>@Password</i>	Table 7-176 on page 430	
FileSpec/ <i>@RequestQuality</i>	Table 7-176 on page 430	
GeneralID	Section 7.2.83 on page 447	
GlueLine/ <i>@GluingPattern</i>	Table 7-191 on page 448	
GluingParams/ <i>@GluingProductionID</i>	Table 7-192 on page 450	
Glue/GlueLine	Table 7-193 on page 450	
IdentificationField/ <i>@Encoding = "Barcode"</i>	Table 7-199 on page 455	
IdentificationField/ <i>@Encoding = "RFID"</i>	Table 7-199 on page 455	
IdentificationField/ <i>@PurposeDetails</i>	Table 7-199 on page 455	
IdentificationField/ <i>@ValueFormat</i>	Table 7-199 on page 455	
IdentificationField/ <i>@ValueTemplate</i>	Table 7-199 on page 455	

Table O-2: New Items (Section 8 of 13)

XPath	Table and Page	Description
IdentificationField/@BarcodeDetails	Table 7-199 on page 455	
IdentificationField/@ExtraValues	Table 7-199 on page 455	
ImageCompression/@ImageFilter = "JBIG2Encode"	Table 7-205 on page 460	
ImageCompression/JBIG2Params	Table 7-205 on page 460	
ImageCompression/JPEG2000params	Table 7-205 on page 460	
JBIG2Params	Table 7-211 on page 466	
JPEG2000params	Table 7-212 on page 466	
ImageSetterParams/@NonPrintableMarginBottom	Table 7-215 on page 469	
ImageSetterParams/@NonPrintableMarginLeft	Table 7-215 on page 469	
ImageSetterParams/@NonPrintableMarginRight	Table 7-215 on page 469	
ImageSetterParams/@NonPrintableMarginTop	Table 7-215 on page 469	
InsertSheet/Layout	Table 7-222 on page 476	
PDFInterpretingParams/@OCGDefault	Table 7-225 on page 481	
PDFInterpretingParams/@OCGIntent	Table 7-225 on page 481	
PDFInterpretingParams/@OCGProcess	Table 7-225 on page 481	
PDFInterpretingParams/@OCGZoom	Table 7-225 on page 481	
PDFInterpretingParams/OCGControl	Table 7-225 on page 481	
OCGControl	Table 7-226 on page 482	
JobField/@ShowList = "LayPartCount2in1"	Table 7-228 on page 484	
JobField/@ShowList = "LayPartIndex2in1"	Table 7-228 on page 484	
LaminatingParams/@LaminatingMethod = "Fusing"	Table 7-230 on page 487	
LaminatingParams/@NipWidth	Table 7-230 on page 487	
Layout/@LockOrigins	Table 7-231 on page 487	
Layout/@SourceWorkStyle	Table 7-231 on page 487	
Layout/@SurfaceContentsBox	Table 7-231 on page 487	
Layout/PlacedObject	Table 7-231 on page 487	
PlacedObject/@ClipPath	Table 7-234 on page 489	
	Section 7.2.108.1 on page 492	Migrating from a Pre-JDF 1.3 Layout to a Partitioned Layout
LayoutElement/@ElementType = "Barcode"	Table 7-240 on page 498	
LayoutElement/@ElementType = "IdentificationField"	Table 7-240 on page 498	
LayoutElementProductionParams	Section 7.2.110 on page 501	
LayoutPreparationParams/@BindingEdge	Table 7-246 on page 502	
LayoutPreparationParams/@FoldCatalogOrientation	Table 7-246 on page 502	

Table O-2: New Items (Section 9 of 13)

XPath	Table and Page	Description
LayoutPreparationParams/ @GutterMinimumLimit	Table 7-246 on page 502	
LayoutPreparationParams/ @ImplicitGutter	Table 7-246 on page 502	
LayoutPreparationParams/ @ImplicitGutterMinimumLimit	Table 7-246 on page 502	
LayoutPreparationParams/ ExternalImpositionTemplate	Table 7-246 on page 502	
Media/ @CoreWeight	Table 7-251 on page 514	
Media/ @Flute	Table 7-251 on page 514	
Media/ @FluteDirection	Table 7-251 on page 514	
Media/ @FrontCoatings = "Polymer"	Table 7-251 on page 514	
Media/ @FrontCoatings = "Silver"	Table 7-251 on page 514	
Media/ @GrainDirection = "XDirection"	Table 7-251 on page 514	
Media/ @GrainDirection = "YDirection"	Table 7-251 on page 514	
Media/ @InsideLoss	Table 7-251 on page 514	
Media/ @MediaType = "CorrugatedBoard"	Table 7-251 on page 514	
Media/ @MediaType = "GravureCylinder"	Table 7-251 on page 514	
Media/ @MediaType = "ImagingCylinder"	Table 7-251 on page 514	
Media/ @MediaType = "SelfAdhesive"	Table 7-251 on page 514	
Media/ @OuterCoreDiameter	Table 7-251 on page 514	
Media/ @OutsideGain	Table 7-251 on page 514	
Media/ @PlateTechnology	Table 7-251 on page 514	
Media/ @WrapperWeight	Table 7-251 on page 514	
Media/ MediaLayers	Table 7-251 on page 514	
Media/ @MediaTypeDetails = "CD"	Table 7-251 on page 514	
Media/ @MediaTypeDetails = "DoubleWall"	Table 7-251 on page 514	
Media/ @MediaTypeDetails = "DVD"	Table 7-251 on page 514	
Media/ @MediaTypeDetails = "SingleFace"	Table 7-251 on page 514	
Media/ @MediaTypeDetails = "SingleWall"	Table 7-251 on page 514	
Media/ @MediaTypeDetails = "TripleWall"	Table 7-251 on page 514	
MiscConsumable	Section 7.2.116 on page 523	
NodeInfo/ @NodeStatus	Table 7-256 on page 525	
NodeInfo/ @NodeStatusDetails	Table 7-256 on page 525	
PageList/ @AssemblyIDs	Table 7-261 on page 528	
PageList/ Assembly	Table 7-261 on page 528	
PageList/ ContentList	Table 7-261 on page 528	
PageData/ @AssemblyIDs	Table 7-262 on page 530	
PageData/ @PageFormat	Table 7-262 on page 530	
PageData/ @PageStatus	Table 7-262 on page 530	
PageData/ @PageElement	Table 7-262 on page 530	
PageElement	Table 7-263 on page 531	

Table O-2: New Items (Section 10 of 13)

XPath	Table and Page	Description
PDLCreationParams	Section 7.2.127 on page 536	
PreflightParams/TestPool	Table 7-274 on page 540	
ProductionPath	Section 7.2.144 on page 554	
RasterReadingParams	Section 7.2.149 on page 562	
ShapeCuttingParams/@DeliveryMode	Table 7-325 on page 579	
ShapeCuttingParams/@SheetLay	Table 7-325 on page 579	
ShapeCuttingParams/DieLayout	Table 7-325 on page 579	
Shape/@StationName	Table 7-326 on page 580	
StackingParams/@UnderLays	Table 7-332 on page 585	
StrappingParams/@StrapPositions	Table 7-335 on page 590	
StrippingParams/@AssemblyIDs	Table 7-337 on page 591	
StrippingParams/ ExternalImpositionTemplate	Table 7-337 on page 591	
StrippingParams/StripMark	Table 7-337 on page 591	
Position/@AbsoluteBox	Table 7-338 on page 593	
Position/@BlockName	Table 7-338 on page 593	
StripCellParams/@Mask	Table 7-339 on page 594	
StripCellParams/@MaskBleed	Table 7-339 on page 594	
StripCellParams/@MaskSeparation	Table 7-339 on page 594	
StripMark	Table 7-340 on page 596	
TrimmingParams/@TrimCover	Table 7-356 on page 609	
UsageCounter	Section 7.2.187 on page 609	
WebInlineFinishingParams	Section 7.2.189 on page 611	
DeviceCap/@CombinedMethod = "GrayBox"	Table 7-363 on page 615	
DeviceCap/DevCapPool	Table 7-363 on page 615	
DeviceCap/ModulePool	Table 7-363 on page 615	
DeviceCap/State	Table 7-363 on page 615	
DevCapPool	Section 7.3.3 on page 618	
ModulePool	Section 7.3.4 on page 618	
ModuleCap	Table 7-368 on page 618	and containing section
DevCaps/@Availability = "Module"	Table 7-369 on page 619	
DevCaps/@DevCapRef	Table 7-369 on page 619	
DevCaps/@ModuleRefs	Table 7-369 on page 619	
DevCaps/@ProcessUsage	Table 7-369 on page 619	
DevCaps/@ResourceUsage	Table 7-369 on page 619	
DevCap/@Availability = "Module"	Table 7-371 on page 622	
DevCap/@DevCapRefs	Table 7-371 on page 622	
DevCap/@ID	Table 7-371 on page 622	
DevCap/@ModuleRefs	Table 7-371 on page 622	
DevCap/@ResourceUsage	Table 7-371 on page 622	
State/@Availability = "Module"	Table 7-372 on page 625	

Table O-2: New Items (Section 11 of 13)

XPath	Table and Page	Description
<i>State/@ListType = "Range"</i>	Table 7-372 on page 625	
<i>State/@ModuleRefs</i>	Table 7-372 on page 625	
	Section 8.2.3 on page 680	HTTPS-Based Protocol – SSL with two-way authentication
<i>StatusDetails = "AbortedBySystem"</i>	Table C-1 on page 703	
<i>StatusDetails = "CoverOpen"</i>	Table C-1 on page 703	
<i>StatusDetails = "DocumentAccessError"</i>	Table C-1 on page 703	
<i>StatusDetails = "DoorOpen"</i>	Table C-1 on page 703	
<i>StatusDetails = "InputTrayMissing"</i>	Table C-1 on page 703	
<i>StatusDetails = "InterlockOpen"</i>	Table C-1 on page 703	
<i>StatusDetails = "JobCanceledByOperator"</i>	Table C-1 on page 703	
<i>StatusDetails = "JobCanceledByUser"</i>	Table C-1 on page 703	
<i>StatusDetails = "JobCompletedSuccessfully"</i>	Table C-1 on page 703	
<i>StatusDetails = "JobCompletedWithErrors"</i>	Table C-1 on page 703	
<i>StatusDetails = "JobCompletedWithWarnings"</i>	Table C-1 on page 703	
<i>StatusDetails = "JobHeldOnCreate"</i>	Table C-1 on page 703	
<i>StatusDetails = "JobHeld"</i>	Table C-1 on page 703	
<i>StatusDetails = "JobIncoming"</i>	Table C-1 on page 703	
<i>StatusDetails = "JobResuming"</i>	Table C-1 on page 703	
<i>StatusDetails = "JobScheduling"</i>	Table C-1 on page 703	
<i>StatusDetails = "JobStreaming"</i>	Table C-1 on page 703	
<i>StatusDetails = "JobSuspended"</i>	Table C-1 on page 703	
<i>StatusDetails = "JobSuspending"</i>	Table C-1 on page 703	
<i>StatusDetails = "MovingToPaused"</i>	Table C-1 on page 703	
<i>StatusDetails = "OutputAreaFull"</i>	Table C-1 on page 703	
<i>StatusDetails = "OutputTrayMissing"</i>	Table C-1 on page 703	
<i>StatusDetails = "ProcessingToStopPoint"</i>	Table C-1 on page 703	
<i>ModuleType = "FarmPrinter"</i>	Table C-6 on page 707	for DigitalPrinting
<i>ModuleType = "ChillUnit"</i>	Table C-7 on page 707	Possible modules of a PrintingUnitWebPath in this row and following
<i>ModuleType = "ImprintUnit"</i>	Table C-7 on page 707	
<i>ModuleType = "PrintUnit"</i>	Table C-7 on page 707	
<i>ModuleType = "RemoisteningModule"</i>	Table C-7 on page 707	
<i>ModuleType = "Rollstand"</i>	Table C-7 on page 707	
<i>ModuleType = "UVCoater"</i>	Table C-7 on page 707	

Table O-2: New Items (Section 12 of 13)

XPath	Table and Page	Description
<i>ModuleType</i> = "CrossCutter"	Table C-8 on page 708	Possible Modules of a FolderSuperstructure WebPath for this and following rows
<i>ModuleType</i> = "Delivery"	Table C-8 on page 708	
<i>ModuleType</i> = "Folder"	Table C-8 on page 708	
<i>ModuleType</i> = "Former"	Table C-8 on page 708	
<i>ModuleType</i> = "GluingAndSofteningModule"	Table C-8 on page 708	
<i>ModuleType</i> = "MoebiusDeinfinitizer"	Table C-8 on page 708	
<i>ModuleType</i> = "PerforatingModule"	Table C-8 on page 708	
<i>ModuleType</i> = "PlanoModule"	Table C-8 on page 708	
<i>ModuleType</i> = "PloughFoldModule"	Table C-8 on page 708	
<i>ModuleType</i> = "Rewinder"	Table C-8 on page 708	
<i>ModuleType</i> = "RibbonCompensator"	Table C-8 on page 708	
<i>ModuleType</i> = "Slitter"	Table C-8 on page 708	
<i>ModuleType</i> = "Stitcher"	Table C-8 on page 708	
<i>ModuleType</i> = "Superstructure"	Table C-8 on page 708	
<i>ModuleType</i> = "TurnerBar"	Table C-8 on page 708	
<i>ModuleType</i> = "TurnerBarUnit"	Table C-8 on page 708	
<i>ModuleType</i> = "BundlingModule"	Table C-9 on page 709	Possible Modules of a PostPressComponent Path for this and following rows.
<i>ModuleType</i> = "LabelingModule"	Table C-9 on page 709	
<i>ModuleType</i> = "PalletizingModule"	Table C-9 on page 709	
<i>ModuleType</i> = "PrintRoll"	Table C-9 on page 709	
<i>ModuleType</i> = "Stacker"	Table C-9 on page 709	
<i>ModuleType</i> = "Trimmer"	Table C-9 on page 709	
Error/@Resend	Table C-15 on page 710	
Error/Error	Table C-15 on page 710	
Milestone	Section C.3.1.7 on page 711	A Milestone element
<i>ReturnCode</i> = "10"	Table D-1 on page 717	
<i>ReturnCode</i> = "116"	Table D-1 on page 717	
<i>ReturnCode</i> = "120"	Table D-1 on page 717	
<i>ReturnCode</i> = "121"	Table D-1 on page 717	
<i>ReturnCode</i> = "130"	Table D-1 on page 717	
<i>ReturnCode</i> = "131"	Table D-1 on page 717	
<i>ReturnCode</i> = "204"	Table D-1 on page 717	Could not create JDF node.
<i>ReturnCode</i> = "300"	Table D-1 on page 717	
<i>ReturnCode</i> = "301"	Table D-1 on page 717	
<i>ReturnCode</i> = "302"	Table D-1 on page 717	
<i>ReturnCode</i> = "303"	Table D-1 on page 717	

Table O-2: New Items (Section 13 of 13)

XPath	Table and Page	Description
	Section F.2 on page 722	Japanese Media Weight
	Section J on page 743	Generating strings with Format and Template
<i>LayPartCount2in1</i>	Table J-1 on page 743	
<i>LayPartIndex2in1</i>	Table J-1 on page 743	
<i>SheetNum</i>	Table J-1 on page 743	
<i>TotalPagesInDoc</i>	Table J-1 on page 743	

O.3 Deprecated Items

Table O-3: Deprecated Items (Section 1 of 2)

XPath	Table and Page	Description
JDF /@ <i>Status</i> = "Pool"	Table 3-4 on page 41	
JDF/ CustomerInfo	Table 3-4 on page 41	
JDF/ NodeInfo	Table 3-4 on page 41	
JDF/ <i>StatusPool</i>	Table 3-4 on page 41	
<i>CustomerInfo</i>	Section 3.4 on page 54	
<i>NodeInfo</i>	Section 3.5 on page 54	
<i>StatusPool</i>	Section 3.6 on page 54	
Resource /@ <i>UpdateID</i>	Table 3-9 on page 54	
<i>ResourceUpdate</i>	Section 3.7.5 on page 63	
<i>ResourceLink</i> /@ <i>DraftOK</i>	Table 3-15 on page 66	
<i>PartAmount</i> /@ <i>DraftOK</i>	Table 3-17 on page 69	
<i>ProcessRun</i> /@ <i>EndStatus</i> = "Stopped"	Table 3-31 on page 102	
<i>PhaseTime</i> /@ <i>Status</i> = "Spawned"	Table 3-33 on page 104	
<i>NotificationFilter</i> /@ <i>DeviceID</i>	Table 5-16 on page 159	
<i>NodeInfo</i>	Table 5-30 on page 167	
<i>NodeInfo</i>	Section 5.8.4 on page 171	
<i>ResourceCmdParams</i> /@ <i>UpdateIDs</i>	Table 5-48 on page 175	
<i>ResourceInfo</i> /Part	Table 5-49 on page 177	
DeliveryIntent /@ <i>AdditionalAmount</i>	Table 7-39 on page 297	
DeliveryIntent /Pricing	Table 7-39 on page 297	
<i>DropIntent</i> /@ <i>AdditionalAmount</i>	Table 7-40 on page 300	
<i>DropIntent</i> /Pricing	Table 7-40 on page 300	
<i>DropItemIntent</i> /@ <i>AdditionalAmount</i>	Table 7-41 on page 302	
<i>DropItemIntent</i> /Pricing	Table 7-41 on page 302	
<i>Pricing</i>	below Table 7-41 on page 302	
<i>Payment</i>	below Table 7-41 on page 302	
<i>CreditCard</i>	below Table 7-41 on page 302	
SizeIntent	Section 7.1.20 on page 322	
ApprovalSuccess /Contact	Table 7-66 on page 325	

Table O-3: Deprecated Items (Section 2 of 2)

XPath	Table and Page	Description
ApprovalSuccess/FileSpec	Table 7-66 on page 325	Deprecated after IP review start.
Assembly/@AssemblyID	Table 7-68 on page 326	
AssemblySection/@AssemblyID	Table 7-69 on page 328	
Component/@SourceRibbon	Table 7-119 on page 382	
Component/@SourceSheet	Table 7-119 on page 382	
Component/@SourceWeb	Table 7-119 on page 382	
Component/@PrintingType = "WebFed"	Table 7-125 on page 389	
CuttingParams/CutMark	Table 7-137 on page 399	
IdentificationField/@Encoding = "Barcode1D"	Table 7-199 on page 455	
IdentificationField/@Encoding = "Barcode2D"	Table 7-199 on page 455	
IdentificationField/@EncodingDetails = "EAN"	Table 7-199 on page 455	
ImageSetterParams/@Punch	Table 7-215 on page 469	
ImageSetterParams/@PunchType	Table 7-215 on page 469	
InsertSheet/Sheet	Table 7-222 on page 476	
Layout/Signature	Table 7-231 on page 487	
MarkObject/DeviceMark	Table 7-238 on page 496	
	Section , Structure of Signature Subelement	A deprecated section
ImageShift/@PositionX = "None"	Table 7-249 on page 511	
ImageShift/@PositionY = "None"	Table 7-249 on page 511	
Media/@MediaTypeDetails = "CtPVisiblePhotoPolymer"	Table 7-251 on page 514	
Media/@MediaTypeDetails = "CtPVisibleSilver"	Table 7-251 on page 514	
Media/@MediaTypeDetails = "CtPThermal"	Table 7-251 on page 514	
Media/@MediaTypeDetails = "PlateUV"	Table 7-251 on page 514	
PageList/@AssemblyID	Table 7-261 on page 528	
PageData/@AssemblyID	Table 7-262 on page 530	
PSToPDFConversionParams/@EndPage	Table 7-299 on page 555	
PSToPDFConversionParams/@StartPage	Table 7-299 on page 555	
Sheet	Section 7.2.164 on page 580	
StrippingParams/@AssemblyID	Table 7-337 on page 591	
Surface	Section 7.2.175 on page 597	
Tool/@ToolAmount	Table 7-346 on page 600	
Tool/@ToolID	Table 7-346 on page 600	
DeviceCap/@CombinedMethod = "CombinedProcessGroup"	Table 7-363 on page 615	
DevCaps/@ResourceUpdate	Table 7-369 on page 619	
<i>surf</i>	Table J-1 on page 743	

O.4 Modified Items

Table O-4: Modified Items (Section 1 of 3)

XPath	Table and Page	Description
URI	Table 1-6 on page 10	Includes Internationalized Resource Identifier (IRI).
URL	Table 1-6 on page 10	Includes Internationalized Resource Identifier (IRI).
JDF/@Status	Table 3-4 on page 41	
Ancestor/ CustomerInfo	Table 3-7 on page 53	
Ancestor/ NodeInfo	Table 3-7 on page 53	
Resource /@PartUsage	Table 3-9 on page 54	
PhysicalResource /@Brand	Table 3-12 on page 61	
Resource /@PartIDKeys	Table 3-25 on page 86	
Part/@TileID	Table 3-26 on page 87	Also a value of Resource /@PartIDKeys
PhaseTime/@End	Table 3-33 on page 104	
PhaseTime/@Status	Table 3-33 on page 104	
ModuleStatus/@ModuleIndex	Table 5-58 on page 187	
QueueEntry/@Status	Table 5-109 on page 213	
ManualLabor / Resource	Table 6-12 on page 222	
DigitalDelivery / RunList	Table 6-42 on page 229	
DigitalDelivery / RunList	Table 6-43 on page 229	
InkZoneCalculation / InkZoneCalculationParams	Table 6-52 on page 232	
ConventionalPrinting / ExposedMedia (Plate)	Table 6-86 on page 245	
Cutting / Component	Table 6-113 on page 254	
Cutting / Media	Table 6-113 on page 254	
ShapeCutting / ShapeCuttingParams	Table 6-150 on page 262	
ShapeCutting / Tool	Table 6-150 on page 262	
ShapeCutting / Component	Table 6-151 on page 262	
ArtDeliveryIntent / DeliveryCharge	Table 7-14 on page 277	
ArtDeliveryIntent / Method	Table 7-14 on page 277	
ArtDelivery/ DeliveryCharge	Table 7-15 on page 280	
ColorIntent / Coatings	Table 7-36 on page 294	
EmbossingItem/ Direction	Table 7-43 on page 303	
LaminatingIntent / Temperature	Table 7-49 on page 307	
MediaIntent / MediaType	Table 7-51 on page 311	
ShapeCut/ ShapeType	Table 7-61 on page 321	
ApprovalPerson/@ApprovalRole	Table 7-65 on page 325	
BinderySignature /@NumberUp	Table 7-76 on page 332	
SignatureCell/@Orientation	Table 7-77 on page 335	

Table O-4: Modified Items (Section 2 of 3)

XPath	Table and Page	Description
ColorSpaceConversionOp/@RenderingIntent	Table 7-111 on page 370	
ColorSpaceConversionOp/@RenderingIntent ="ColorSpaceDependent"	Table 7-111 on page 370	
ColorSpaceConversionOp/@SourceCS	Table 7-111 on page 370	
ColorSpaceConversionOp/ @SourceRenderingIntent ="ColorSpaceDependent"	Table 7-111 on page 370	
Component/@ComponentType	Table 7-119 on page 382	
Component/@ComponentType = "Proof"	Table 7-119 on page 382	
Component/@ProductType	Table 7-119 on page 382	
Component/@PrintingType	Table 7-125 on page 389	
Component/@Speed	Table 7-125 on page 389	
Emboss/@Direction	Table 7-166 on page 423	
GlueLine/@GluingPattern	Table 7-191 on page 448	
Glue/GlueApplication	Table 7-193 on page 450	
IdentificationField/@Encoding	Table 7-199 on page 455	
ImageCompression/@ImageFilter	Table 7-206 on page 461	
JobField/@ShowList	Table 7-228 on page 484	
JobField/DeviceMark	Table 7-228 on page 484	
Layout/Media	Table 7-231 on page 487	
PlacedObject/@SourceClipPath	Table 7-234 on page 489	
LayoutElement/@ElementType	Table 7-240 on page 498	
ManualLaborParams/@LaborType	Table 7-250 on page 514	
Media/@FrontCoatings	Table 7-251 on page 514	
Media/@GrainDirection	Table 7-251 on page 514	
Media/@MediaType	Table 7-251 on page 514	
Media/@MediaTypeDetails	Table 7-251 on page 514	
Media/@MediaTypeDetails = "Aluminum"	Table 7-251 on page 514	
Media/@MediaTypeDetails = "Polyester"	Table 7-251 on page 514	
NodeInfo/@JobPriority	Table 7-251 on page 514	
Preview/@CTM	Table 7-251 on page 514	
ResourceDefinitionParams/ResourceParam	Table 7-313 on page 566	
Tool/@ToolType	Table 7-346 on page 600	
TransferFunctionControl/ @TransferFunctionSource	Table 7-350 on page 603	
DeviceCap/@CombinedMethod	Table 7-363 on page 615	
DevCaps/@Context	Table 7-369 on page 619	
DevCaps/@Name	Table 7-369 on page 619	
DevCaps/@DevCap	Table 7-369 on page 619	
State/@ListType	Table 7-372 on page 625	
data type: PDFPath	Section A.2.30 on page 693	

Table O-4: Modified Items (Section 3 of 3)

XPath	Table and Page	Description
data type: URI	Section A.2.42 on page 695	
data type: URL	Section A.2.43 on page 696	
Error/@ <i>ErrorID</i>	Table C-15 on page 710	

Appendix P Deprecated Elements, JMF Messages, Processes and Resources

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.

P.1 Deprecated Structures of JDF Nodes and Jobs

P.1.1 ResourceUpdate Elements

[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** MUST 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 MUST 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 MUST NOT be applied to product 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 P-1: Contents of the abstract ResourceUpdate Element

Name	Data Type	Description
<i>UpdateID</i> 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 <i>UpdateID</i> may occur per JDF document, even though the scope of the ResourceUpdate is local to the resource that it is defined in.

P.1.2 StatusPool and PartStatus

[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 par-

titions. It is an error to define `PartStatus` elements that reference identical or overlapping parts within one `StatusPool`. Partitioned resources are described in Section 3.9.5, *Description of Partitioned Resources*.

Table P-2: Contents of the `StatusPool` element

Name	Data Type	Description
<code>Status</code> ?	enumeration	Identifies the <i>Status</i> of the node when <code>JDF/@Status = "Pool"</code> . Individual <code>PartStatus</code> elements MAY override this value for the partitions they represent. <i>Status</i> applies to all partitions of the node except where it is overridden by <code>PartStatus/@Status</code> . Possible values are all valid <i>Status</i> attributes of a JDF node except <code>Pool</code> are valid as defined in Table 3-4, "JDF node," on page 41, <i>Status</i> .
<code>StatusDetails</code> ? New in JDF 1.2	string	Identifies the <i>StatusDetails</i> of the node when <code>JDF/@Status = "Pool"</code> . Individual <code>PartStatus</code> elements MAY override this value for the partitions they represent. <i>StatusDetails</i> applies to all partitions of the node except where it is overridden by <code>PartStatus/@StatusDetails</code> . For a list of supported values, see "Supported String and NMTOKEN values" on page 703.
<code>PartStatus</code> *	element	Element that defines the node's status for a set of parts.

The following table describes the `PartStatus` element.

Table P-3: Contents of the `PartStatus` element

Name	Data Type	Description
<code>Status</code> ?	enumeration	Identifies the status of an individual part of the node. If not specified, defaults to <code>StatusPool/@Status</code> . Possible values are identical to those defined in defined in Table 3-4, "JDF node," on page 41, <i>Status</i> . In JDF 1.3 and beyond, <i>Status</i> has been replaced by <code>NodeInfo/@NodeStatus</code> .
<code>StatusDetails</code> ? New in JDF 1.2	string	Description of the status that provides details beyond the enumerative values given by the <i>Status</i> attribute. If not specified, defaults to <code>StatusPool/@StatusDetails</code> . For a list of supported values, see "Supported String and NMTOKEN values" on page 703. In JDF 1.3 and beyond, <i>StatusDetails</i> has been replaced by <code>NodeInfo/@NodeStatus</code> .
<code>Part</code> ^a Modified in JDF 1.2	element	Specifies the selected part that the <code>PartStatus</code> is valid for. This MUST be a leaf or intermediate partition of the node's output resource. Thus, if the node's output resource is partitioned by <i>Side</i> and <i>Separation</i> , the <code>Part</code> may contain either <i>Side</i> only or <i>Side</i> and <i>Separation</i> , but not <i>Separation</i> only. See Table 3-26, "Part element," on page 87 for details of the <code>Part</code> element. For details on partitioned resources, see "Description of Partitioned Resources" on page 79.

- a. The cardinality of `Part` in `PartStatus` has been changed from * to none, (i.e., exactly one element) in version 1.1 of the JDF specification.

P.2 JMF Messaging Elements

P.2.1 Signal

— Element: Trigger

The following 3 elements were deprecated from `Signal/Trigger` in JDF 1.2.

— Element: Added[Deprecated in JDF 1.2](#)**Table P-4: Added element**

Name	Data Type	Description
AddedElement *	element	<p>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.</p> <p>This is an abstract element. It is a placeholder for a ResponseTypeObj like NotificationDef, a JDFController, a Device, a JDFService or a MessageService.</p> <p>For details on these elements see Section 5.7, Messages for Events and Capabilities.</p>

— Element: ChangedAttribute[Deprecated in JDF 1.2](#)**Table P-5: ChangedAttribute element**

Name	Data Type	Description
<i>AttributeName</i>	NMTOKEN	Name of the attribute that changed.
<i>ElementID ?</i>	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.
<i>ElementType</i>	NMTOKEN	Name of the element which contains the changed attribute.
<i>OldValue</i>	string	Old value. The string MUST be cast to the appropriate data type that depends on the attribute's data type.
<i>NewValue</i>	string	New value of the attribute.

— Element: Removed[Deprecated in JDF 1.2](#)**Table P-6: Removed element**

Name	Data Type	Description
RemovedElement *	element	<p>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.</p> <p>This is an abstract element. It is a placeholder for a ResponseTypeObj like NotificationDef, a JDFController, a Device, a JDFService or a MessageService.</p> <p>For details on these elements see Section 5.7, Messages for Events and Capabilities.</p>

P.2.2 NodeInfo[New in JDF 1.2](#)[Deprecated in JDF 1.3](#)

The **NodeInfo** message can be used as a command or a query to modify or to query JDF **NodeInfo** elements. The query 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.

P.2.2.1 NodeInfo Query

The `NodeInfo` query is made selective by specifying a `NodeInfoQuParams` element. The query response 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 P-7: Contents of the `NodeInfo` query message

Object Type	Element Name	Description
QueryTypeObj	NodeInfoQuParams	Specifies the node queried.
ResponseTypeObj	NodeInfoResp *	Details of the <code>NodeInfo</code> elements

— Element: NodeInfoQuParams

Table P-8: Contents of the `NodeInfoQuParams` element

Name	Data Type	Description
<code>JobID</code>	string	Job ID of the JDF node that is being queried.
<code>JobPartID ?</code>	string	Job part ID of the JDF node that is being queried.
<code>QueueEntryID ?</code>	string	<code>QueueEntryID</code> of the job that is currently being executed. If <code>QueueEntryID</code> is specified, <code>JobID</code> , <code>JobPartID</code> , and <code>Part</code> are ignored. If none of <code>JobID</code> , <code>JobPartID</code> , <code>Part</code> , or <code>QueueEntryID</code> are specified, <code>ResourceQuParams</code> applies to all jobs.
<code>Part *</code>	element	<code>Part</code> elements that describe the partition of the job whose <code>NodeInfo</code> is modified. For details on Node partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 112.

P.2.2.2 NodeInfo Command

The `NodeInfo` command 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 contains a list of `NodeInfoResp` elements with a copy of `NodeInfo` after the changes have been applied. If the `NodeInfo` command 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 P-9: Contents of the `NodeInfo` Command Message

Object Type	Element name	Description
CommandTypeObj	NodeInfoCmdParams	Specifies the <code>NodeInfo</code> elements to be modified.
ResponseTypeObj	NodeInfoResp *	Contains information about the <code>NodeInfo</code> and the <code>NodeInfo</code> after modification.

— Element: NodeInfoCmdParams

Table P-10: Contents of the `NodeInfoCmdParams` element

Name	Data Type	Description
<code>JobID</code>	string	Job ID of the JDF node that is being modified.
<code>JobPartID ?</code>	string	Job part ID of the JDF node that is being modified.

Table P-10: Contents of the *NodeInfoCmdParams* element

Name	Data Type	Description
<i>QueueEntryID</i> ?	string	<i>QueueEntryID</i> of the job that is currently being executed. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> , and <i>Part</i> are ignored. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> , or <i>QueueEntryID</i> are specified, <i>NodeInfoCmdParams</i> applies to all jobs.
<i>UpdateMethod</i> = "Complete"	enumeration	Method how <i>NodeInfo</i> is applied to the JDF. Values are: <i>Complete</i> – The <i>NodeInfo</i> in the JDF is completely overwritten by <i>NodeInfo</i> in this message. <i>Incremental</i> – The <i>NodeInfo</i> in the JDF is incrementally updated by the values that are explicitly set in <i>NodeInfo</i> in this message.
<i>Part</i> *	element	<i>Part</i> elements that describe the partition of the job whose <i>NodeInfo</i> is modified. For details on Node partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 112.
<i>NodeInfo</i> ?	element	<i>NodeInfo</i> to be uploaded to the Device.

— Element: NodeInfoRespTable P-11: Contents of the *NodeInfoResp* element

Name	Data Type	Description
<i>JobID</i>	string	Job ID of the JDF node that is being modified.
<i>JobPartID</i> ?	string	Job part ID of the JDF node that is being modified.
<i>QueueEntryID</i> ?	string	<i>QueueEntryID</i> of the job that is currently being executed. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> , and <i>Part</i> are ignored. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> , or <i>QueueEntryID</i> are specified, <i>NodeInfoResp</i> applies to all jobs.
<i>Part</i> *	element	<i>Part</i> elements that describe the partition of the job <i>NodeInfo</i> is modified. For details on Node partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 112.
<i>NodeInfo</i> ?	element	<i>NodeInfo</i> 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 to the query above:

```
<Response ID="M1" Type="NodeInfo" refID="Q1">
  <NodeInfoResp JobID="J1" JobPartID="P1">
    <NodeInfo/>
  </NodeInfoResp>
  <NodeInfoResp JobID="J1" JobPartID="P2">
    <NodeInfo/>
  </NodeInfoResp>
</Response>
```

P.2.3 KnownJDFServices

[Deprecated in JDF 1.2](#)

In JDF 1.2 and beyond, KnownJDFServices has been replaced with KnownDevices and DeviceDetails = "Capabilities".

Table P-12: Contents of the KnownJDFServices message

Object Type	Element name	Description
QueryTypeObj	—	—
ResponseTypeObj	JDFService *	Processes that the controller or device can execute.

The KnownJDFServices query 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.

JDFService

JDFService elements define the node types that can be processed by the controller. A JDF processor SHOULD be capable of processing *Combined* 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 P-13: Contents of the JDFService element

Name	Data Type	Description
CombinedMethod ? New in JDF 1.1	enumeration	Specifies how the processes specified in <i>Types</i> may be specified. One of: <i>Combined</i> – The list of processes in <i>Types</i> MUST be specified as a <i>Combined</i> process. <i>ProcessGroup</i> – The list of processes in <i>Types</i> MUST be specified as a <i>ProcessGroup</i> of individual processes. <i>CombinedProcessGroup</i> – The list of processes in <i>Types</i> may be specified either as a <i>Combined</i> process or as a <i>ProcessGroup</i> of individual processes. <i>None</i> – No support for <i>Combined</i> or <i>ProcessGroup</i> . Only the individual process type defined in <i>Types</i> is supported. The default.
<i>Type</i>	NMTOKEN	JDF <i>Type</i> attribute of the supported process. Extension types may be specified by stating the namespace in the value.
TypeOrder ? New in JDF 1.1	enumeration	Ordering restriction for combined nodes. <i>Fixed</i> – The order of process types specified in the <i>Types</i> attribute is ordered and each type can be specified only once, e.g., Cutting, Folding; order does matter. The default. <i>Unordered</i> – The order of process types specified in the <i>Types</i> attribute is unordered and each type can be specified only once, e.g., DigitalPrinting, Screening, Trapping; order does not matter. <i>Unrestricted</i> – The order of process types specified in the <i>Types</i> 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.
<i>Types ?</i>	NMTOKENS	If <i>Type</i> = <i>Combined</i> , or <i>Type</i> = <i>ProcessGroup</i> 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 3.2.3, Combined Process Nodes.

The following is an example of a response to a KnownJDFServices query:

```
<Response ID="M1" refID="Q1" Type="KnownJDFServices">
  <JDFService Type="Rendering" />
  <JDFService Type="Folding" />
  <JDFService Type="Combined" Types="Gathering Stitching"/>
  <JDFService Type="AnyCompaniesNamespace:MyFolding" />
  ...
</Response>
```

P.2.4 QueueEntryStatus

[Deprecated in JDF 1.2](#)

In JDF 1.2 and beyond, use QueueStatus with an appropriate QueueFilter instead of QueueEntryStatus.

Table P-14: Contents of the 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 5.13, 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.

Structure of the QueueEntryDefList Element

[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 P-15: Contents of the QueueEntryDefList element

Name	Data Type	Description
QueueEntryDef *	element	Defines the addressed queue entries.

P.3 Deprecated Processes

P.3.1 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 6.7.5, Packaging Processes.

Input Resources

Name	Description
PackingParams	Necessary information about the packing process.
PhysicalResource	All kinds of physical resources can be packed.

Output Resources

Name	Description
PhysicalResource	The packaged physical resources. Note that <i>Amount</i> attributes referring to this resource still refer to individual products and not to boxes, cartons or pallets.

P.3.2 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.

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.

Output Resources

Name	Description
ExposedMedia	The resulting exposed plate.

P.3.3 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
Resource referenced by:	—
Example Partition:	—
Input of processes:	—
Output of processes:	Preflight

Resource Structure

Name	Data Type	Description
ColorsResultsPool ?	element	A pool of PreflightDetail and PreflightInstance subelements that provides analysis about color.
DocumentResultsPool ?	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.
FileTypeResultsPool ?	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.

Structure of PreflightDetail Subelement

PreflightDetail subelements are used to describe one property within the **PreflightAnalysis** category in which they occur. This subelement is also used by **PreflightInventory** resource.

Name	Data Type	Description
<i>PageRefs</i>	IntegerRangeList	Identifies the set of pages in a RunList resource that exhibit the characteristic identified by the combination of the <i>Property</i> attribute and the <i>Value</i> element.
<i>Property</i> ?	string	Identifies the property described by this element.
<i>Status</i> ?	enumeration	Possible values are: <i>Error</i> – Value violates the <i>ConstraintValue</i> specified in the associated PreflightConstraint element. The constraint was flagged as an Error in the profile. <i>Warning</i> – Value violates the <i>ConstraintValue</i> specified in the associated PreflightConstraint element. The constraint was flagged as a Warning in the profile. <i>Ignore</i> – The constraint is ignored, and no PreflightDetail or PreflightInstanceDetail elements are created for this constraint. <i>IgnoreValue</i> – No comparison was made against a <i>ConstraintValue</i> . In other words, either the <i>Status</i> for the PreflightConstraint was <i>Ignore</i> or <i>IgnoreValue</i> , or this PreflightDetail is part of a PreflightInventory hierarchy.
<i>Value</i> ?	element	Identifies the value of the property. The semantics are PDL-specific.

Structure of PreflightInstance Subelement

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.

Name	Data Type	Description
<i>Identifier</i> ?	string	Identifies the instance this element collects PreflightInstanceDetail elements.
<i>PageRefs</i> Modified in JDF 1.1	IntegerRange-List	Identifies the set of finished pages in a RunList on which the instance occurs.

Name	Data Type	Description
PreflightInstanceDetail * Modified in JDF 1.1	element	A pool of PreflightInstanceDetail elements that describe the properties for this instance

Structure of PreflightInstanceDetail Subelement

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

Name	Data Type	Description
<i>Status ?</i>	enumeration	Specifies the results of the comparison between the value of the property for this instance with the ConstraintValue for the associated PreflightConstraint element. Possible values are: <i>Error</i> – Value violates the ConstraintValue specified. The constraint was flagged as an Error in the profile. <i>Warning</i> – Value violates the ConstraintValue specified. The constraint was flagged as a Warning in the profile. <i>IgnoreValue</i> – No comparison was made against a ConstraintValue. In other words, either the <i>Status</i> for the Constraint was <i>Ignore</i> or <i>IgnoreValue</i> , or this PreflightInstanceDetail is part of a PreflightInventory hierarchy.
<i>Property ?</i>	string	Identifies the property described by this element.
<i>Value ?</i>	element	Identifies the value of the property. The semantics are PDL-specific.

P.3.4 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 PreflightInstanceDetails.

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
Resource referenced by:	—
Example Partition:	—
Input of processes:	Preflight
Output of processes:	Preflight

Resource Structure

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.

Name	Data Type	Description
FontsResultsPool ?	element	A pool of PreflightDetail and PreflightInstance subelements that provides a font inventory.
FileTypeResultsPool ?	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.

P.3.5 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 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
Resource referenced by:	—
Example Partition:	—
Input of processes:	Preflight
Output of processes:	—

Resource Structure

Name	Data Type	Description
ColorsConstraintsPool ?	element	A pool of PreflightConstraint subelements. Each element in this pool identifies a specific constraint concerning colors against which to test the file
DocumentConstraintsPool ?	element	A pool of PreflightConstraint subelements. Each element in this pool identifies a specific constraint concerning documents against which to test the file
FontsConstraintsPool ?	element	A pool of PreflightConstraint subelements. Each element in this pool identifies a specific constraint concerning fonts against which to test the file
FileTypeConstraintsPool ?	element	A Preflight constraint. The Type attribute MUST have a value of <i>array</i> and MUST contain string objects that identify the allowable types of data in the file. The strings in the Value array MUST be MIME-file types as recorded by the Internet Assigned Numbers Authority (IANA). IANA has procedures for registering new file types if needed.
ImagesConstraintsPool ?	element	A pool of PreflightConstraint subelements. Each element in this pool identifies a specific constraint concerning images against which to test the file
PagesConstraintsPool ?	element	A pool of PreflightConstraint subelements. Each element in this pool identifies a specific constraint concerning finished pages against which to test the file

Structure of PreflightConstraint Subelement

Name	Data Type	Description
<i>AttemptFixupErrors</i> = "false"	boolean	If <i>true</i> , the device performing preflight SHOULD attempt to fix errors that are identified during preflight. Errors that are corrected are not given a <i>Status</i> attribute. Default = "false"
<i>AttemptFixupWarnings</i> = "false"	boolean	If <i>true</i> , the device performing preflight SHOULD attempt to fix warnings that are identified during preflight. Warnings that are corrected are not given a <i>Status</i> attribute. Default = "false"
<i>Constraint</i> ?	string	Describes the specific file characteristic to be checked.
<i>Status</i>	enumeration	Possible values are: <i>Error</i> – Values that violate the <i>ConstraintValue</i> specified are flagged as Errors in <i>PreflightDetail</i> and <i>PreflightInstanceDetail</i> elements. <i>Warning</i> – Values that violate the <i>ConstraintValue</i> specified are flagged as Warnings in <i>PreflightDetail</i> and <i>PreflightInstanceDetail</i> elements. <i>Ignore</i> – The constraint is ignored, and no <i>PreflightDetail</i> or <i>PreflightInstanceDetail</i> elements are created for this constraint. <i>IgnoreValue</i> – No comparison is made against the <i>ConstraintValue</i> .
<i>ConstraintValue</i> ?	element	Provides a value against which to test occurrences of the characteristic in the file. Note: The semantics of the <i>ConstraintValue</i> element depend on the PDL characteristic in question.

P.3.6 Proofing

[Deprecated in JDF 1.2](#)

The **Proofing** process is deprecated in JDF/1.2. Instead, use a combined process to produce 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.

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.

Name	Description
RunList (Document)	Identifies the pages to be proofed. When the Layout resource is present in the ProofingParams resource, <i>Ord</i> 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, <i>Ord</i> values from MarkObject subelements refer to pages in this RunList .

Output Resources

Name	Description
ExposedMedia	The resulting physical proof.

P.3.7 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 **MUST** 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 **MUST** be paired with the identified final target device profile to create the transformation.

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, <i>Ord</i> values from ContentObject subelements refer to pages in this RunList .

Name	Description
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, <i>Ord</i> values from MarkObject subelements refer to pages in this RunList .

Output Resources

None.

P.3.8 IDPrinting

[Deprecated in JDF 1.1](#)

The IDPrinting process was deprecated in JDF/1.1. Instead, implementations SHOULD use the **DigitalPrinting** process combined with 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**.

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.

Name	Description
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 ByteMaps 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.

Output Resources

Name	Description
Component (Good)	Components are produced for other printing processes or postpress processes. Note that the <i>Amount</i> 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.

P.3.9 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.

P.3.10 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**.

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.

Output Resources

Name	Description
Component	One Component is produced: either the divided web or ribbon.

P.3.11 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 node with *ConventionalPrinting*).

Input Resources

Name	Description
Component	The Component can consist of more than one web or ribbon that has been combined with the <i>Gathering</i> process.
LongitudinalRibbonOperationParams	Specific parameters to set up the machinery tools for the <i>LongitudinalRibbonOperations</i> process.

Output Resources

Name	Description
Component +	A ribbon is produced that is used in other postpress processes. If the <i>LongitudinalRibbonOperations</i> process was slitting, more than one Component is produced.

P.3.12 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.

Input Resources

Name	Description
Component	The only REQUIRED Component is the collected pile.
SaddleStitchingParams	Specific parameters to set up the machinery.

Output Resources

Name	Description
Component	The stitched-together components.

P.3.13 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 6.7.2, 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*.

Input Resources

Name	Description
Component	The only REQUIRED Component is the gathered sheets.
SideSewingParams	Specific parameters to set up the machinery.

Output Resources

Name	Description
Component	The Component is produced.

P.4 Deprecated Resources

P.4.1 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.

Structure of the AdhesiveBinding Subelement.

[Deprecated in JDF 1.1](#)

Name	Data Type	Description
<i>Scoring ?</i>	EnumerationSpan	Scoring option for AdhesiveBinding . Possible values are: <i>TwiceScored</i> <i>QuadScored</i> <i>None</i> Values are based on viewing the cover in its flat pre-binding state.
<i>SpineGlue ?</i>	EnumerationSpan	Glue type used to define AdhesiveBinding procedures. Possible values are: <i>ColdGlue</i> <i>Hotmelt</i> <i>PUR</i> – Polyurethane Rubber
<i>TapeBinding ?</i>	OptionSpan	If " <i>true</i> ", 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).

Structure of the BookCase Subelement.

[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.

Name	Data Type	Description
<i>HeadBands ?</i>	OptionSpan	The following CaseBinding choice specifies the use of headbands on a case bound book. If " <i>true</i> ", headbands are inserted both top and bottom.
<i>Shape ?</i>	EnumerationSpan	Indicates the shape of the "back" or spine of a case bound book. Possible values are: <i>RoundedBack</i> <i>SquareBack</i>
<i>Thickness ?</i>	NumberSpan	Specifies thickness of board which is wrapped as front and back covers of a case bound book, in points.

P.4.2 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].

Contents of the Pricing Subelement

[Deprecated in JDF 1.3](#)

Name	Data Type	Description
<i>AdditionalPrice</i> ?	double	Price for ordering the number of copies specified in the <i>AdditionalAmount</i> attribute as specified in the parent element of the Pricing.
<i>Currency</i> ?	NMTOKEN	Three digit currency definition according to [ISO4217:2001]. It defaults to the currency defined in the parent quote.
<i>HasPrice</i> = "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 MUST be added. If "true", the line item is included in the parent quote.
<i>Item</i> ?	string	Name of the item that this particular quote element describes. If not specified, Pricing applies to the entire DropItemIntent.
<i>Price</i> ?	double	Price for ordering the number of copies specified in the <i>Amount</i> 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.
<i>Payment</i> ? New in JDF 1.1	element	Details of the payment method.
<i>Pricing</i> *	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 <i>HasPrice</i> = "false"). The sum of line items need not be identical to the parent quote.

Contents of the Payment Subelement

[New in JDF 1.1](#)

[Deprecated in JDF 1.3](#)

Name	Data Type	Description
<i>PayTerm</i> ?	telem	Describes the payment terms & conditions.
<i>CreditCard</i> ?	element	Specifies credit card information

Contents of the CreditCard Subelement

[New in JDF 1.1](#)

[Deprecated in JDF 1.3](#)

Name	Data Type	Description
<i>Authorization</i> ?	String	Authorization code for this transaction.
<i>AuthorizationExpires</i> ?	gYearMonth	Expiration date of the <i>Authorization</i> .
<i>Expires</i>	gYearMonth	Expiration date of the credit card.
<i>Number</i>	NMTOKEN	Credit card number. The format is specified without blanks or any other separator characters.

Name	Data Type	Description
<i>Type</i>	NMTOKEN	Credit card brand. Possible values include: <i>Amex</i> <i>DinersClub</i> <i>Discovery</i> <i>MasterCard</i> – This includes derived brands, (e.g., EuroCard). <i>Visa</i>

P.4.3 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
Resource referenced by:	—
Process Resource Pairing:	CutMark, CuttingParams, Layout, LayoutPreparationParams, Sheet, Surface, TrimmingParams
Example Partition:	<i>Option</i>
Input of processes:	Any Product Node
Output of processes:	—

Resource Structure

Name	Data Type	Description
<i>Dimensions</i>	XYPairSpan	Specifies the height and width of the product component in points. Note: Height and width are ambiguously specified in JDF 1.0.
<i>Pages ?</i>	IntegerSpan	Specifies the number of pages of the product component.
<i>Type = "Folded"</i>	enumeration	Specifies whether the product component referred to is flat or finished. Possible values are: <i>Folded</i> = Size of the product after folding. The default value <i>Flat</i> = 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.

P.4.4 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
Resource referenced by:	—

Example Partition: —
 Input of processes: **AdhesiveBinding**
 Output of processes: —

Resource Structure

Name	Data Type	Description
<i>FlexValue ?</i>	double	Flex quality parameter given in [N/cm].
<i>PullOutValue ?</i>	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.

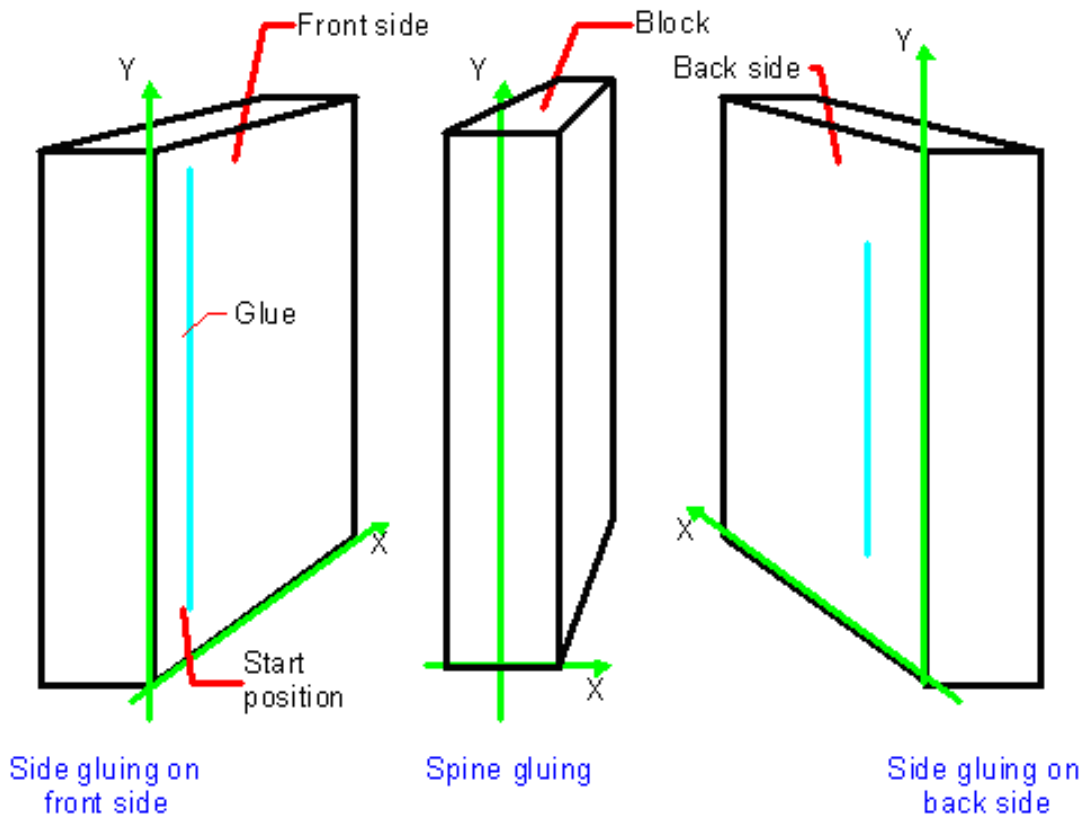


Figure P-1: Parameters and coordinate system for glue application

P.4.5 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
 Resource referenced by: —

Example Partition: *RibbonName, SheetName, SignatureName, WebName*

Input of processes: **Dividing**

Output of processes: —

Resource Structure

Name	Data Type	Description
<i>DividePositions</i>	DoubleList	Array containing the cross cut positions in y-direction (direction of web traveling).

P.4.6 IDPrintingParams

[Deprecated in JDF 1.1](#)

This resource contains the parameters needed to control the **IDPrinting** process.

Resource Properties

Resource class: **Parameter**

Resource referenced by: —

Example Partition: *DocIndex, DocRunIndex, DocSheetIndex, PartVersion, Run, RunIndex, RunTags, SheetIndex, SheetName, Side*

Input of processes: **IDPrinting**

Output of processes: —

Resource Structure

Name	Data Type	Description
<i>AttributesNaturalLang</i> = "US English"	language	Language selected for communicating attributes. Default = "US English"
<i>IDPAttributeFidelity</i> = "false"	boolean	Indicates whether or not the device MUST reject the job if there are attribute values or elements that it does not support. Default = "false"
<i>IPPJobPriority</i> = "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 MUST be printed before any lower priority ones. Default = 50
<i>IPPVersion</i> ?	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.

Name	Data Type	Description
<i>OutputBin ?</i>	NMTOKEN	<p>Specifies the bin to which the finished document is to be output. Possible values are:</p> <p><i>Top</i> – The bin that, when facing the device, can best be identified as “top”.</p> <p><i>Middle</i> – The bin that, when facing the device, can best be identified as “middle”.</p> <p><i>Bottom</i> – The bin that, when facing the device, can best be identified as “bottom”.</p> <p><i>Side</i> – The bin that, when facing the device, can best be identified as “side”.</p> <p><i>Left</i> – The bin that, when facing the device, can best be identified as “left”.</p> <p><i>Right</i> – The bin that, when facing the device, can best be identified as “right”.</p> <p><i>Center</i> – The bin that, when facing the device, can best be identified as “center”.</p> <p><i>Rear</i> – The bin that, when facing the device, can best be identified as “rear”.</p> <p><i>FaceUp</i> – The bin that can best be identified as “face up” with respect to the device.</p> <p><i>FaceDown</i> – The bin that can best be identified as “face down” with respect to the device.</p> <p><i>FitMedia</i> – Requests the device to select a bin based on the size of the media.</p> <p><i>LargeCapacity</i> – The bin that can best be identified as the “large capacity” bin (in terms of the number of sheets) with respect to the device.</p> <p><i>Mailbox-N</i> – The job will be output to the bin that is best identified as “Mailbox-1”, “Mailbox-2” ...etc.</p> <p><i>Stacker-N</i> – The job will be output to the bin that is best identified as “Stacker-1”, “Stacker-2” ...etc.</p> <p><i>Tray-N</i> – The job will be output to the tray that is best identified as “Tray-1”, “Tray-2” ... etc.</p>
<i>PageDelivery ?</i>	enumeration	<p>Indicates how pages are to be delivered to the output bin or finisher. Possible values are:</p> <p><i>SameOrderFaceUp</i> – Order as defined by the RunList, with the “front” sides of the media up.</p> <p><i>SameOrderFaceDown</i> – Order as defined by the RunList, with the “front” sides of the media up.</p> <p><i>ReverseOrderFaceUp</i> – Order reversed, as defined by the RunList, with the “front” sides of the media up.</p> <p><i>ReverseOrderFaceDown</i> – Order reversed, as defined by the RunList, with the “front” sides of the media down.</p>
<i>PrintQuality ?</i>	enumeration	<p>Indicates how pages are to be delivered to the output bin or finisher. Possible values are:</p> <p><i>High</i> – Highest quality available on the printer.</p> <p><i>Normal</i> – The default quality provided by the printer.</p> <p><i>Draft</i> – Lowest quality available on the printer.</p>

Name	Data Type	Description
<i>SheetCollate</i> ?	boolean	Determines whether the sequencing of the leaves in the output of the job. If <i>true</i> , sheets for each copy of the document are sequenced together, followed by the sheets for the next copy. If <i>false</i> , all copies of the first sheet are sequenced, followed by the second and subsequent sheet. <i>SheetCollate</i> 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 MUST 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.

Structure of the Cover Subelement

[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.

Name	Data Type	Description
<i>BackSide</i> = "false"	boolean	The next page from the RunList is imaged onto the back of this cover. This would be the inside of a <i>Front</i> cover and outside of a <i>Back</i> cover. Default = "false"
<i>CoverType</i> = "Front"	enumeration	Specifies whether this Cover element specifies the front or back cover. <i>Front</i> – The front cover. <i>Back</i> – The back cover. Default = "Front"
<i>FrontSide</i> = "false"	boolean	The next page from the RunList is imaged onto the front of this cover. This would be the outside of a <i>Front</i> cover and inside of a <i>Back</i> 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.
MediaSource ?	refelement	Describes the source and physical orientation of the media to be used.

Properties of the IDPFinishing Subelement

[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

Name	Data Type	Description
<i>Finishings ?</i>	IntegerList	<p>A set of finishing operations to apply to the job. The operations are encoded as an enumeration. Possible values are:</p> <p>3 – (none) Perform no finishing</p> <p>4 – (staple) Bind the document(s) with one or more staples. The exact number and placement of the staples is site-defined.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>12 – (bale) Bale the document(s). The type of baling is implementation and/or site-defined.</p> <p>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.</p> <p>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.</p> <p>50 – (bind-left) Bind the document(s) along the left edge. The type of the binding is site-defined.</p> <p>51 – (bind-top) Bind the document(s) along the top edge. The type of the binding is site-defined.</p> <p>52 – (bind-right) Bind the document(s) along the right edge. The type of the binding is site-defined.</p> <p>53 – (bind-bottom) Bind the document(s) along the bottom edge. The type of the binding is site-defined.</p>

Name	Data Type	Description
IDPFolding ?	refelement	Provides details of how to fold the set of pages (or document). When this element is present, <i>Finishings</i> is ignored.
IDPHoleMaking ?	refelement	Provides details of how to punch holes in the set of pages (or document). When this element is present, <i>Finishings</i> is ignored.
IDPStitching ?	refelement	Provides details of how to stitch the set of pages (or document). When this element is present, <i>Finishings</i> is ignored.
IDPTrimming ?	refelement	Provides details of how to trim the set of pages (or document). When this element is present, <i>Finishings</i> is ignored.

Structure of IDPFolding Subelement

[Deprecated in JDF 1.1](#)

This element describes the folding requested for a set of pages in the document.

Name	Data Type	Description
FoldingParams ?	Refelement	Describes the details of how to fold the media.

Structure of IDPHoleMaking Subelement

[Deprecated in JDF 1.1](#)

This element describes the hole making requested for a set of pages in the document.

Name	Data Type	Description
HoleMakingParams ?	refelement	Describes the details of the holes to be punched into the Media.

Structure of the IDPLayout Subelement

[Deprecated in JDF 1.1](#)

Name	Data Type	Description
<i>Border</i> = "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).
<i>FinishedPageOrientation</i> = "Portrait"	enumeration	Indicates the desired orientation of the finished page. This value is used with <i>PresentationDirection</i> to determine how pages will be imaged onto the media. Possible values are: <i>Portrait</i> – The short edges of the media are the top and bottom. <i>Landscape</i> – The long edges of the media are the top and bottom. Default = "Portrait".
<i>ForceFrontSide</i> ?	NumberRange List	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.
<i>ImageShift</i> ?	element	Element which describes how page images are to be placed onto the media. When <i>NumberUp</i> is present and is not "1,1", <i>NumberUp</i> is applied before the <i>ImageShift</i> , and all contents for each surface are shifted the same amount.

Name	Data Type	Description
<i>NumberUp</i> ?	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 <i>FinishedPageOrientation</i> and <i>PresentationDirection</i> . <i>FinishedPageOrientation</i> indicates how the page will be oriented, and <i>PresentationDirection</i> indicates how page images will be distributed, given that orientation.
<i>PresentationDirection</i> ?	enumeration	Indicates the order in which the requested <i>NumberUp</i> pages will be imaged onto the media. The value of <i>FinishedPageOrientation</i> is used to define “top”, “left”, “right” and “bottom” for the media. Possible values are: <i>ToBottomToRight</i> – Pages are imaged in successive columns, from left to right, starting at the top of each column. <i>ToBottomToLeft</i> – Pages are imaged in successive columns, from right to left, starting at the top of each column. <i>ToTopToRight</i> – Pages are imaged in successive columns, from left to right, starting at the bottom of each column. <i>ToTopToLeft</i> – Pages are imaged in successive columns, from right to left, starting at the bottom of each column. <i>ToRightToBottom</i> – Pages are imaged in successive rows, from top to bottom, starting at the left of each row. <i>ToRightToTop</i> – Pages are imaged in successive rows, from bottom to top, starting at the left of each row. <i>ToLeftToBottom</i> – Pages are imaged in successive rows, from top to bottom, starting at the right of each row. <i>ToLeftToTop</i> – Pages are imaged in successive rows, from bottom to top, starting at the right of each row.
<i>Rotate</i> = “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”
<i>Sides</i> = “OneSided”	enumeration	Indicates how pages are to be imposed onto sides of the medium. Possible values are: <i>OneSided</i> – Page contents will only be imaged on one side of the media. The default. <i>TwoSidedLongEdge</i> – 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”. <i>TwoSidedShortEdge</i> – 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”.

Structure of IDPStitching Subelement

[Deprecated in JDF 1.1](#)

This element describes the stitching requested for a set of pages in the document

Name	Data Type	Description
<i>StitchingPosition</i> ?	enumeration	<p>Specifies the location for stitching. All locations are interpreted as if the document were a portrait document. Ignored if <i>StitchingParams</i> is present. Possible values are:</p> <p><i>None</i> – The document is not to be stitched.</p> <p><i>TopLeft</i> – Bind the document with one or more staples in the top left corner.</p> <p><i>BottomLeft</i> – Bind the document with one or more staples in the Bottom left corner.</p> <p><i>TopRight</i> – Bind the document with one or more staples in the top right corner.</p> <p><i>BottomRight</i> – Bind the document with one or more staples in the bottom right corner.</p> <p><i>LeftEdge</i> – Bind the document with one or more staples across the left edge.</p> <p><i>TopEdge</i> – Bind the document with one or more staples across the top edge.</p> <p><i>RightEdge</i> – Bind the document with one or more staples across the right edge.</p> <p><i>BottomEdge</i> – Bind the document with one or more staples across the bottom edge.</p> <p><i>DualLeftEdge</i> – Bind the document with two staples across the left edge.</p> <p><i>DualTopEdge</i> – Bind the document with two staples across the top edge.</p> <p><i>DualRightEdge</i> – Bind the document with two staples across the right edge.</p> <p><i>DualBottomEdge</i> – Bind the document with two staples across the bottom edge.</p>
<i>StitchingReferenceEdge</i> ?	enumeration	<p>The edge of the output media relative to which the stapling or stitching MUST be applied. If <i>StitchingParams</i> is present, <i>StitchingReferenceEdge</i> defines the <i>BindingEdge</i>. Possible values are:</p> <p><i>Bottom</i> – The bottom edge coincides with the x-axis of the coordinate system.</p> <p><i>Top</i> – The top edge is opposite and parallel to the bottom edge.</p> <p><i>Left</i> – The left edge coincides with the y-axis of the coordinate system.</p> <p><i>Right</i> – The right edge is opposite and parallel to the left edge.</p>
<i>StitchingParams</i> ?	refelement	<p>A <i>StitchingParams</i> element which provides detailed control of the stitching. <i>StitchingReferenceEdge</i> MUST be present if <i>StitchingParams</i> is provided.</p>

Structure of IDPTrimming Subelement

[Deprecated in JDF 1.1](#)

This element describes the trimming requested for a set of pages in the document.

Name	Data Type	Description
TrimmingParams ?	refelement	Describes the details of how to trim the media.

Structure of the ImageShift Subelement

[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

Name	Data Type	Description
<i>PositionX</i> = “None”	enumeration	Indicates how finished page images are to be positioned horizontally on the surface. Shifts are applied after positioning. Values are: <i>Center</i> – Center the page images horizontally on the surface without regard to limitations of the printable area. <i>Left</i> – Position the left edge of the page images so they is coincident with the left edge of the printable area of the surface. <i>None</i> – Place the page images wherever the print data specifies (the default). <i>Right</i> – Position the right edge of the page images so they is coincident with the right edge of the printable area of the surface.
<i>PositionY</i> = “None”	enumeration	Indicates how finished page images are to be positioned vertically on the surface. Shifts are applied after positioning. Values are: <i>Bottom</i> – Position the bottom edge of the page images so they is coincident with the bottom edge of the printable area of the surface. <i>Center</i> – Center the page images horizontally on the surface without regard to limitations of the printable area. <i>None</i> – Place the page images wherever the print data specifies (the default). <i>Top</i> – Position the top edge of the page images so they is coincident with the top edge of the printable area of the surface.
<i>ShiftX</i> ?	integer	The image is to be shifted along the x axis on both sides of the media.
<i>ShiftY</i> ?	integer	The image is to be shifted along the y axis on both sides of the media.
<i>ShiftXSide1</i> ?	integer	The image is to be shifted along the x axis on the front side of the media.
<i>ShiftXSide2</i> ?	integer	The image is to be shifted along the x axis on the back side of the media.
<i>ShiftYSide1</i> ?	integer	The image is to be shifted along the y axis on the front side of the media.
<i>ShiftYSide2</i> ?	integer	The image is to be shifted along the y axis on the back side of the media.

Structure of the JobSheet Subelement

[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 MUST have a *Name* attribute with the value ‘SheetMessage’.

Name	Data Type	Description
<i>SheetFormat</i> = “Standard”	NMTOKEN	Identifies the format of the JobSheet. The default is “Standard”, but site-specific values may be defined.

Name	Data Type	Description
<i>SheetOccurrence</i>	enumeration	Indicates when the sheet is to be produced and inserted into the set of output pages. Possible values are: <i>Always</i> – Valid for <i>ErrorSheet</i> or <i>AccountingSheet</i> . The sheet is always produced at the end of the job. <i>End</i> – Valid for <i>JobSheet</i> or <i>SeparatorSheet</i> . The sheet is produced at the end of the job (for <i>JobSheet</i>) or at the end of each copy of each instance document (for <i>SeparatorSheet</i>). <i>OnError</i> – Valid for <i>ErrorSheet</i> . The sheet is produced at the end of the job when an error or warning occurs. <i>Slip</i> – Valid for <i>SeparatorSheet</i> . The sheet is produced between each copy of each instance document. <i>Start</i> – Valid for <i>JobSheet</i> or <i>SeparatorSheet</i> . The sheet is produced at the start of the job (for <i>JobSheet</i>) or at the start of each copy of each instance document (for <i>SeparatorSheet</i>). <i>Both</i> – Valid for <i>JobSheet</i> or <i>SeparatorSheet</i> . The sheet is produced at the beginning and end of the job (for <i>JobSheets</i>) or at the beginning and end of each copy of each instance document (for <i>SeparatorSheets</i>). <i>None</i> – Valid for any <i>SheetType</i> .
<i>SheetType</i>	enumeration	Identifies the type of sheet. Possible values are: <i>AccountingSheet</i> – A sheet that reports accounting information for the job. <i>ErrorSheet</i> – A sheet that reports errors for the job. <i>JobSheet</i> – A sheet that delimits the job. <i>SeparatorSheet</i> – 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 3.9.5, 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>

```

P.4.7 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.

Structure of Signature Subelement

[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.

Name	Data Type	Description
<i>Name</i> ?	string	Unique name of the signature. <i>Name</i> 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.

P.4.8 LongitudinalRibbonOperationParams

[Deprecated in JDF 1.1.](#)

This resource provides the parameters of the **LongitudinalRibbonOperation** process. It is defined as a list of abstract *LROperation* elements.

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>RibbonName, SheetName, SignatureName, WebName</i>
Input of processes:	LongitudinalRibbonOperations
Output of processes:	—

Resource Structure

Name	Data Type	Description
<i>LROperation</i> +	element	Abstract element which is a placeholder for a longitudinal ribbon operation.

Structure of LongitudinalRibbonOperationParams Elements

LROperation

[Deprecated in JDF 1.1.](#)

LROperation is an abstract element that describes the **LongitudinalRibbonOperation** process. The defined instances (subclasses) of LROperation are LongFold, LongGlue, LongPerforate, and LongSlit. All instances of LROperation have the following common contents

Name	Data Type	Description
<i>WorkingList</i> = "0 1000000000"	NumberList	List of lengths of the <i>Operation</i> 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).
<i>XOffset</i>	double	Position of the tool for longitudinal action along the cylinder axis.

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.

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.

Name	Data Type	Description
<i>GlueBrand</i> ?	string	Glue brand. Use only when <i>Operation</i> = <i>Glue</i> .
<i>GlueType</i> ?	Enumeration	If <i>Operation</i> = <i>Glue</i> , the following values can be used: <i>ColdGlue</i> <i>Hotmelt</i> <i>PUR</i> – Polyurethane
<i>LineWidth</i> ?	double	Width of the <i>Operation</i> line.
<i>MeltingTemperature</i> ?	integer	Temperature needed for melting the glue (in degrees centigrade). Use only when <i>GlueType</i> = <i>Hotmelt</i> and <i>Operation</i> = <i>Glue</i> .

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.

Name	Data Type	Description
<i>TeethPerDimension</i> ?	integer	If <i>Operation</i> = <i>Perforate</i> , the number of teeth in a given perforation extent is defined in teeth/point. <i>MicroPerforation</i> is defined by specifying a large number of teeth (n>1000).

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.

P.4.9 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
Example Partition:	—
Input of processes:	DigitalPrinting, IDPrinting
Output of processes:	—

Resource Structure

Name	Data Type	Description
<i>LeadingEdge</i> ?	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 <i>Dimension</i> parameter for the Media .
<i>MediaLocation</i> ?	String	Identifies the location, such as a slot name or ID, of the media in the device. If the media resource is partitioned by <i>Location</i> (see also Section 3.9.6.2, Locations of Physical Resources) there SHOULD be a match between one <i>Location</i> partition key and this <i>MediaLocation</i> value.
<i>ManualFeed</i> = "false"	boolean	Indicates whether the media will be fed manually. Default = "false"
<i>SheetLay</i> ? New in JDF 1.1	enumeration	Lay of input media. Reference edge of where paper is placed in feeder. Possible values are: <i>Left</i> <i>Right</i> <i>Center</i>
Component ? New in JDF 1.1	refelement	A Component resource which identifies the preprinted media to be used. Only one of Component or Media SHOULD be specified.
Media ?	refelement	A Media resource which identifies the media to be used. Only one of Component or Media SHOULD be specified.

P.4.10 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 6.7.4, Numbering and Section 6.7.5, 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
Resource referenced by:	—
Example Partition:	—
Input of processes:	Packing

Output of processes: —

Resource Structure

Name	Data Type	Description
<i>BoxedQuantity</i> ?	integer	How many units of <i>product</i> in a box.
<i>BoxShape</i> ?	shape	Describes the length, width and height of the box in points.
<i>CartonQuantity</i> ?	integer	How many units of <i>product</i> in a carton.
<i>CartonShape</i> ?	shape	Describes the length, width and height of the carton in points, (e.g., 288 544 1012).
<i>CartonMaxWeight</i> ?	double	Maximum weight of an individual carton in kilograms.
<i>CartonStrength</i> ?	double	Strength of the carton in Newtons per square meter.
<i>PalletQuantity</i> ?	integer	Number of <i>product</i> per pallet
<i>PalletSize</i> ?	XYPair	Describes the length and width of the pallet in points, (e.g., 3500 3500).
<i>PalletMaxHeight</i> ?	double	Maximum height of a loaded pallet in points.
<i>PalletMaxWeight</i> ?	double	Maximum weight of a loaded pallet in kilograms.
<i>PalletType</i> ?	enumeration	Type of pallet used. Examples include: <i>2Way</i> – Two-way entry <i>4Way</i> – Four-way entry <i>Euro</i> – Standard 1*1 m Euro pallet
<i>PalletWrapping</i> = "None"	enumeration	Wrapping of the completed pallet. Examples include: <i>StretchWrap</i> <i>Banding</i> <i>None</i> – The default.
<i>WrappedQuantity</i> ?	integer	Number of units of <i>product</i> per wrapped package.
<i>WrappingMaterial</i> = "None"	name	Examples include: <i>RubberBand</i> <i>ShrinkWrap</i> <i>PaperBand</i> <i>Polyethylene</i> <i>None</i> – The default.

P.4.11 PlateCopyParams[Deprecated in JDF 1.1](#)This resource specifies the parameters of the **FilmToPlateCopying** process.**Resource Properties**

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>FilmToPlateCopying</i>
Output of processes:	—

Resource Structure

Name	Data Type	Description
<i>Cycle</i> ?	integer	Number of exposure light units to be used. The amount depends on the subject to be exposed.

Name	Data Type	Description
<i>Diffusion ?</i>	enumeration	The diffusion foil setting. Possible values are: <i>On</i> <i>Off</i>
<i>Vacuum ?</i>	double	Amount of vacuum pressure to be used. Measured in bars.

P.4.12 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
Resource referenced by:	—
Example Partition:	<i>DocIndex, RunIndex, RunTags, SheetName, Side, SignatureName</i>
Input of processes:	<i>Proofing, SoftProofing</i>
Output of processes:	—

Resource Structure

Name	Data Type	Description
<i>ColorType ?</i>	enumeration	Color quality of the proof. Possible values are: <i>Monochrome</i> – Black and white. <i>BasicColor</i> – Color does not match precisely. This implies the absence of a color matching system. <i>MatchedColor</i> – Color is matched to the output of the press using a color matching system.
<i>DisplayTraps = "false"</i>	boolean	If <i>true</i> , the trap networks are shown in the proof. Default = <i>"false"</i>
<i>HalfTone = "false"</i>	boolean	Specifies whether the proof is to emulate halftone screens. Default = <i>"false"</i>
<i>ImageViewingStrategy = "NoImages"</i>	string	Identifies which images will be displayed during the <i>SoftProofing</i> process. Possible values are: <i>NoImages</i> – Default value. <i>OmitReference</i> – Displays only images actually embedded in the file. <i>UseProxies</i> – Displays images embedded in the file and proxy versions of referenced data. <i>UseReplacements</i> – Displays embedded images plus the full resolution version of referenced images.
<i>ManualFeed = "false"</i> New in JDF 1.1	boolean	Indicates whether the media will be fed manually. Default = <i>"false"</i>
<i>ProofRenderingIntent = "Perceptual"</i> New in JDF 1.1	enumeration	Identifies the rendering intents associated with the proof. Possible ICC-defined rendering intent values are: <i>Saturation</i> <i>Perceptual</i> – The default. <i>RelativeColorimetric</i> <i>AbsoluteColorimetric</i>

Name	Data Type	Description
<i>ProofType</i> = "None"	enumeration	Describes the type of the proof. Possible values are: <i>None</i> – Default value. Not a proof or the type is unknown. <i>Page</i> – Page proof <i>Imposition</i> – Imposition proof.
<i>Resolution</i> ?	XYPair	Resolution of the output.
<i>FileSpec</i> ?	reference	A FileSpec resource pointing to an ICC profile that describes the proofer device. The <i>ResourceUsage</i> attribute of the <i>FileSpec</i> MUST be " <i>ProoferProfile</i> ".
Media ?	reference	Describes the media to be used.

P.4.13 SaddleStitchingParams

This resource provides the parameters of the **SaddleStitching** process.

[Deprecated in JDF 1.1](#)

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	SaddleStitching
Output of processes:	—

Resource Structure

Name	Data Type	Description
<i>NumberOfStitches</i>	integer	The number of stitches that will be made.
<i>StitchPositions</i> ?	NumberList	Array containing the stitch positions along the saddle. The center of the stitch MUST be specified, and the number of entries MUST match the number given in the <i>NumberOfStitches</i> attribute.
<i>StapleShape</i> ?	enumeration	Shape of staples. Possible values are: <i>Crown</i> <i>Overlap</i> <i>Butted</i> <i>ClinchOut</i> <i>Eyelet</i> These values are displayed in Figure P-2, below.
<i>StitchWidth</i> ?	double	Width of each stitch.
<i>WireGauge</i> ?	double	Gauge of the wire being used.
<i>WireBrand</i> ?	string	Brand of wire being used.

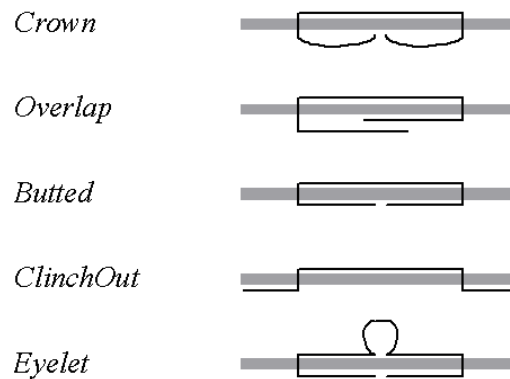


Figure P-2: Staple shapes

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.

P.4.14 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:	<i>SheetName</i> .
Input of processes:	—
Output of processes:	—

Resource Structure

Name	Data Type	Description
<i>LockOrigins = "false"</i>	boolean	Determines the relationship of the coordinate systems for front and back surfaces. When <i>"false"</i> , 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 <i>"true"</i> , 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.
<i>Name ?</i> Clarified in JDF 1.2	string	Name of the sheet. <i>Name</i> MUST be unique within a given Layout . <i>Name</i> is used for external reference to a sheet in, for example, a Part element.
<i>SurfaceContentsBox ?</i>	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 MarkObjects or ContentObject elements transform page contents or marks into this rectangle.

Name	Data Type	Description
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 <i>Side</i> attribute of the Surface resource. Surface/@Side MUST be <i>Front</i> .
Surface (Back) ?	refelement	Describes the back surface to be used. Surface/@Side MUST be <i>Back</i> .

P.4.15 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).

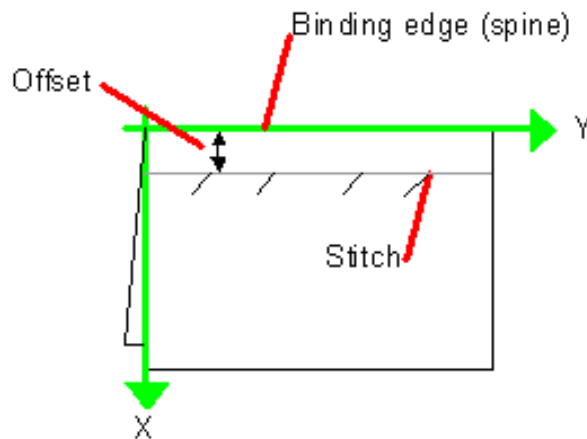


Figure P-3: Parameters and coordinate system used for side sewing

Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	SideSewing
Output of processes:	—

Resource Structure

Name	Data Type	Description
<i>NumberOfNeedles</i>	integer	Specifies the number of needles to be used.

Name	Data Type	Description
<i>NeedlePositions</i> ?	NumberList	Array containing the Y-coordinates of the needle positions. The number of entries MUST match the number given in <i>NumberOfNeedles</i> .
<i>Offset</i>	double	Specifies the distance between the stitch and the binding edge.
<i>SewingPattern</i> ?	enumeration	Specifies the sewing pattern to be used. Possible values are: <i>Normal</i> <i>Staggered</i> <i>CombinedStaggered</i>
<i>ThreadMaterial</i> ?	enumeration	Specifies the thread material to be used. Possible values are: <i>Cotton</i> <i>Nylon</i> <i>Polyester</i>
<i>ThreadThickness</i> ?	double	The thickness of the thread to be used.
<i>ThreadBrand</i> ?	string	The brand of thread to be used.

P.4.16 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**.

Input of processes: —

Output of processes: —

Resource Structure

Name	Data Type	Description
<i>Side</i>	enumeration	The side of the Sheet that the Surface describes. Possible values are: <i>Front</i> <i>Back</i>
<i>SurfaceContentsBox</i> ?	rectangle	This rectangle provides the region of the surface into which the contents of ContentObject elements and MarkObjects are to be imaged. Note: The <i>SurfaceContentsBox</i> also provides a translation for an object's <i>CTM</i> .
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.

Appendix Q List of Figures

Figure 1-1 Handling of default values of JDF attributes.	9
Figure 2-1 Example of JDF and JMF workflow interactions	18
Figure 2-2 JDF tree structure	19
Figure 2-3 Example of a hierarchical tree structure of JDF nodes	21
Figure 2-4 Example of a process chain linked by input and output resources	22
Figure 2-5 Standard coordinate system	23
Figure 2-6 Relation between resource and process coordinate systems	24
Figure 2-7 Layout of simple saddle stitched brochure (product example)	27
Figure 2-8 Equation for Surface Coordinate System Transformations	27
Figure 2-9 Surface coordinate system	28
Figure 2-10 Press coordinate system used for sheet-fed printing	28
Figure 2-11 Press coordinate system used for web printing	29
Figure 2-12 Coordinate systems after Folding (product example)	29
Figure 2-13 Coordinate systems after Collecting (product example)	30
Figure 2-14 Examples of Transformations and Coordinate Systems in JDF.	31
Figure 2-15 Transforming a point (example)	33
Figure 3-1 JDF node – a diagram of its structure	36
Figure 3-2 Any-element (generic content) – a diagram of its structure	40
Figure 3-3 Job hierarchy with process, process group and product intent nodes	46
Figure 3-4 Combined node dependencies	51
Figure 3-5 Abstract Resource element – a diagram of its structure	59
Figure 3-6 Nodes linked by a resource	64
Figure 3-7 ResourceLink elements and ResourceRef elements	64
Figure 3-8 Abstract ResourceLink element – a diagram of its structure	66
Figure 3-9 Amount handling	78
Figure 3-10 Workflow for splitting shared input resources	98
Figure 3-11 Workflow for combining shared output resources	99
Figure 3-12 Workflow for splitting independent input resources	99
Figure 3-13 Workflow for combining independent output resources	99
Figure 3-14 Abstract Audit element – a diagram of its structure	101
Figure 4-1 Simplified PrintTalk workflow (negotiation phase)	120
Figure 4-2 Life Cycle of a JDF node	123
Figure 4-3 Example of a simple process chain linked by resources	125
Figure 4-4 Example of a Pipe resource linking two processes	128
Figure 4-5 Example of status transitions in case of overlapping processing	129
Figure 4-6 The spawning and merging mechanism and its phases	133
Figure 4-7 JDF node structure that requires resource copying during spawning and merging	135
Figure 4-8 Example for a JDF node structure with nested spawning	137
Figure 4-9 Example of the spawning and merging of independent jobs	138
Figure 4-10 Parameter space in device capabilities	141
Figure 5-1 JMF root element – a diagram of its structure	146
Figure 5-2 Interaction of Messages with a subscription	147
Figure 5-3 Interaction of Command and Acknowledge Messages	153
Figure 5-4 Without UpdateJDF Message	190
Figure 5-5 With UpdateJDF Message	190
Figure 5-6 Mechanism of a PipePull message	194
Figure 5-7 Mechanism of a PipePush message	196

Figure 5-8 JMF QueueEntry Status transition diagram	198
Figure 5-9 Effects of the global queue messages on the queue Status	207
Figure 6-1 Worst case scenario for area coverage calculation	236
Figure 6-2 Overview of Web Printing	245
Figure 6-3 Bundle Creation	250
Figure 6-4 Bundle Transport	250
Figure 6-5 Combined Process with Feeding Process	255
Figure 6-6 Input Components	255
Figure 6-7 Output Component	255
Figure 6-8 Packaging Process Coordinate System	269
Figure 7-1 Structure of a normal hardcover book	289
Figure 7-2 Structure of a padded hardcover book	289
Figure 7-3 Structure of a book with GlueProcedure = "SideOnly" (Layflat)	292
Figure 7-4 Folding examples for some values of BoxFoldAction/@Action	340
Figure 7-5 BoxFoldingType attribute for values of Type00, Type01 and Type02	340
Figure 7-6 BoxFoldingType attribute for values of Type03, Type04 and Type10	341
Figure 7-7 BoxFoldingType attribute for values of Type 11, Type12 and Type13	341
Figure 7-8 BoxFoldingType attribute for values of Type15 and Type20	342
Figure 7-9 Box packing	343
Figure 7-10 CaseMakingParams	348
Figure 7-11 Parameters and coordinate system for CasingIn	349
Figure 7-12 Parameters used for channel binding	350
Figure 7-13 Coordinate systems used for collecting	353
Figure 7-14 Component – terms and definitions	382
Figure 7-15 Parameters and coordinate system for cover application	393
Figure 7-16 Definition of the PlatePosition attribute on a newspaper-web press	401
Figure 7-17 Example of a single physical section of eight pages	401
Figure 7-18 Parameters and coordinate system used for end-sheet gluing	425
Figure 7-19 Names of the reference edges of a sheet in the FoldingParams resource	438
Figure 7-20 Fold catalog part 1	440
Figure 7-21 Fold catalog part 2	441
Figure 7-22 Coordinate system used for gathering	446
Figure 7-23 Parameters and coordinate system for glue application	448
Figure 7-24 Hole line parameters	452
Figure 7-25 Line hole punching for multiple webs	452
Figure 7-26 Parameters and coordinate system used for Inserting	473
Figure 7-27 Parameters and coordinate system for jacketing	483
Figure 7-28 Diagram of a 4-up cross-folded saddle-stitch imposition with vertical gutter creep	512
Figure 7-29 Diagram of a step-and-repeat 2-up saddle-stitch imposition with vertical spine gutter creep	512
Figure 7-30 a paper-roll with some roll-specific information	521
Figure 7-31 InsideLoss, OutsideGain	522
Figure 7-32 PRGroup – a diagram of its structure	544
Figure 7-33 RegisterRibbon lengths and coordinate system for BlockPreparation	564
Figure 7-34 Parameters and coordinate systems for the SpinePreparation process	582
Figure 7-35 Parameters and coordinate system for the SpineTaping process	583
Figure 7-36 Stacking Layers	584
Figure 7-37 Pile Patterns	584
Figure 7-38 Odd count handling for a Bundle	585

Figure 7-39 Odd count handling for a Layer	585
Figure 7-40 Staple shapes	586
Figure 7-41 Parameters and coordinate system used for saddle stitching	587
Figure 7-42 Parameters and coordinate system used for Stitching	587
Figure 7-43 Strapped bundle	590
Figure 7-44 Strapped bundle with subbundles	590
Figure 7-45 RelativeBox including margins	593
Figure 7-46 Definition of margins in StripCellParams	596
Figure 7-47 Parameters and coordinate system used for thread sewing	598
Figure 7-48 Parameters and coordinate system used for side sewing	598
Figure 7-49 Parameters and coordinate system used for trimming	608
Figure 7-50 DeviceCap – a diagram of its structure	614
Figure 7-51 Abstract State element – a diagram of its structure	624
Figure 7-52 macro element – a diagram of its structure	640
Figure 7-53 Abstract Term element – a diagram of its structure	645
Figure 7-54 Abstract Evaluation element – a diagram of its structure	647
Figure P-1 Parameters and coordinate system for glue application	834
Figure P-2 Staple shapes	851
Figure P-3 Parameters and coordinate system used for side sewing	852

Appendix R List of Tables

Table 1-1 Basic references	1
Table 1-2 Callouts	3
Table 1-3 Cardinality symbols	4
Table 1-4 Glossary	4
Table 1-5 Conformance terminology	7
Table 1-6 JDF data types	10
Table 1-7 Units used in JDF	13
Table 2-1 Information contained in JDF nodes, arranged numerically	19
Table 2-2 Information contained in JDF nodes, arranged by group	20
Table 2-3 Data types for specifying coordinates and transformation	24
Table 2-4 Matrices and Orientation values for describing the orientation of a Component	26
Table 2-5 JDF processes used for the production of the simple brochure	27
Table 3-1 Any element (generic content)	37
Table 3-2 Comment element	38
Table 3-3 Definition of “Scope” Terms used in Table 3-4 on page 41	40
Table 3-4 JDF node	41
Table 3-5 Status attribute – possible values	45
Table 3-6 AncestorPool element	52
Table 3-7 Ancestor element	53
Table 3-8 ResourcePool element	54
Table 3-9 Abstract Resource element	54
Table 3-10 SourceResource element	58
Table 3-11 Abstract Parameter Resource element	60
Table 3-12 Abstract Physical Resource element	61
Table 3-13 Location element	62
Table 3-14 ResourceLinkPool element	63
Table 3-15 Abstract ResourceLink element	66
Table 3-16 AmountPool element	69
Table 3-17 PartAmount element	69
Table 3-18 Abstract ImplementationLink or //AmountPool/PartAmount element	70
Table 3-19 Abstract PhysicalLink or //AmountPool/PartAmount element	70
Table 3-20 Lot element	72
Table 3-21 Abstract ResourceElement	74
Table 3-22 Abstract ResourceRef element	74
Table 3-23 Example of actual amount and amount handling	78
Table 3-24 Identical element	83
Table 3-25 Partitionable resource element	86
Table 3-26 Part element	87
Table 3-27 Condition attribute – possible values	93
Table 3-28 PartUsage attribute examples	97
Table 3-29 AuditPool element	100
Table 3-30 Abstract Audit element	102
Table 3-31 ProcessRun audit element	102
Table 3-32 Notification audit element	103
Table 3-33 PhaseTime audit element	104
Table 3-34 ModulePhase audit element	106
Table 3-35 ResourceAudit element	107

Table 3-36 Created audit element	109
Table 3-37 Deleted audit element	109
Table 3-38 Modified audit element	109
Table 3-39 Spawned audit element	110
Table 3-40 Merged audit element	110
Table 3-41 Example from TrappingParams	113
Table 4-1 Business objects as defined by PrintTalk	118
Table 4-2 Examples of resource and process states in the case of simple process routing ..	125
Table 4-3 Examples of partitioning across multiple resources	126
Table 4-4 Actions generated when a dynamic-pipe buffer passes various levels	130
Table 4-5 Class attribute values for the Notification audit element	140
Table 5-1 JMF element	144
Table 5-2 Abstract message element	145
Table 5-3 Query message element	148
Table 5-4 Response message element	149
Table 5-5 Signal message element	150
Table 5-6 Trigger element	150
Table 5-7 ChangedPath element	151
Table 5-8 Command message element	152
Table 5-9 Acknowledge message element	153
Table 5-10 Registration message element	154
Table 5-11 Subscription element	155
Table 5-12 ObservationTarget element	155
Table 5-13 Template for message tables	158
Table 5-14 Messages for events and capabilities	158
Table 5-15 Events message	159
Table 5-16 NotificationFilter element	159
Table 5-17 NotificationDef element	160
Table 5-18 KnownControllers message	161
Table 5-19 JDFController element	161
Table 5-20 KnownDevices message	161
Table 5-21 DeviceFilter element	162
Table 5-22 DeviceList element	162
Table 5-23 KnownMessages message	163
Table 5-24 KnownMsgQuParams element	163
Table 5-25 MessageService element	163
Table 5-26 RepeatMessages message	165
Table 5-27 MsgFilter element	165
Table 5-28 StopPersistentChannel message	166
Table 5-29 StopPersChParams element	166
Table 5-30 Messages to query/affect a job, device or controller	167
Table 5-31 FlushResources message	168
Table 5-32 FlushResourceParams element	168
Table 5-33 FlushedResources element	168
Table 5-34 ModifyNode message	168
Table 5-35 ModifyNodeCmdParams element	169
Table 5-36 NewComment element	169
Table 5-37 NewJDF query message	170
Table 5-38 NewJDFQuParams element	170
Table 5-39 NewJDF command message	170

Table 5-40 NewJDFCmdParams element	170
Table 5-41 IDInfo element	171
Table 5-42 Occupation message	171
Table 5-43 EmployeeDef element	171
Table 5-44 Occupation element	172
Table 5-45 Resource query message	173
Table 5-46 ResourceQuParams element	173
Table 5-47 Resource command message	175
Table 5-48 ResourceCmdParams element	175
Table 5-49 ResourceInfo element	177
Table 5-50 ResourcePull message	179
Table 5-51 ResourcePullParams element	180
Table 5-52 ShutDown message	181
Table 5-53 ShutDownCmdParams element	182
Table 5-54 Status message	182
Table 5-55 StatusQuParams element	183
Table 5-56 DeviceInfo element	184
Table 5-57 JobPhase element	185
Table 5-58 ModuleStatus element	187
Table 5-59 Track message	188
Table 5-60 TrackFilter element	189
Table 5-61 TrackResult element	189
Table 5-62 UpdateJDF message	190
Table 5-63 UpdateJDFCmdParams element	190
Table 5-64 CreateLink element	191
Table 5-65 CreateResource element	191
Table 5-66 MoveResource element	191
Table 5-67 RemoveLink element	192
Table 5-68 WakeUp message	193
Table 5-69 WakeUpCmdParams element	193
Table 5-70 Messages for control of dynamic pipes	193
Table 5-71 PipeClose message	193
Table 5-72 PipePull message	194
Table 5-73 PipeParams element	194
Table 5-74 PipePush message	195
Table 5-75 PipePause message	196
Table 5-76 Messages for queue entry handling	197
Table 5-77 Status transitions for QueueEntry handling messages	197
Table 5-78 AbortQueueEntry message	199
Table 5-79 HoldQueueEntry message	199
Table 5-80 RemoveQueueEntry message	200
Table 5-81 RequestQueueEntry message	200
Table 5-82 RequestQueueEntryParams element	200
Table 5-83 ResubmitQueueEntry message	201
Table 5-84 ResubmissionParams element	201
Table 5-85 ResumeQueueEntry message	202
Table 5-86 ReturnQueueEntry message	202
Table 5-87 ReturnQueueEntryParams element	202
Table 5-88 SetQueueEntry message	203
Table 5-89 QueueEntryPosParams element	203

Table 5-90 SetQueueEntryPriority message	203
Table 5-91 QueueEntryPriParams element	204
Table 5-92 SubmitQueueEntry message	204
Table 5-93 QueueSubmissionParams element	204
Table 5-94 SuspendQueueEntry message	206
Table 5-95 Messages for global handling of queues	206
Table 5-96 Definition of the Queue Status Attribute values	207
Table 5-97 CloseQueue message	208
Table 5-98 FlushQueue command message	208
Table 5-99 FlushQueueParams element	208
Table 5-100 FlushQueue query message	209
Table 5-101 FlushQueueInfo element	209
Table 5-102 HoldQueue message	209
Table 5-103 OpenQueue message	209
Table 5-104 QueueStatus message	210
Table 5-105 ResumeQueue message	210
Table 5-106 SubmissionMethods message	210
Table 5-107 SubmissionMethods element	210
Table 5-108 Queue element	211
Table 5-109 QueueEntry element	213
Table 5-110 QueueEntryDef element	214
Table 5-111 QueueFilter element	214
Table 5-112 Messages for Gang Jobs	215
Table 5-113 Contents of the ForceGang Command message	215
Table 5-114 GangCmdFilter element	215
Table 5-115 GangStatus message	216
Table 5-116 GangQuFilter element	216
Table 5-117 GangInfo element	216
Table 6-1 Template for input resources	219
Table 6-3 Approval – Input resources	220
Table 6-4 Approval – Output resources	220
Table 6-2 Template for output resources	220
Table 6-5 Buffer – Input resources	221
Table 6-6 Buffer – Output resources	221
Table 6-7 Combine – Input resources	221
Table 6-8 Combine – Output resources	221
Table 6-9 Delivery – Input resources	221
Table 6-11 ManualLabor – Input resources	222
Table 6-12 ManualLabor – Output resources	222
Table 6-13 Ordering – Input resources	222
Table 6-14 Ordering – Output resources	222
Table 6-10 Delivery – Output resources	222
Table 6-15 QualityControl – Input resources	223
Table 6-16 QualityControl – Output resources	223
Table 6-17 ResourceDefinition – Input resources	223
Table 6-18 ResourceDefinition – Output resources	223
Table 6-19 Split – Input resources	223
Table 6-20 Split – Output resources	223
Table 6-21 Verification – Input resources	224
Table 6-22 Verification – Output resources	224

Table 6-23 Product Intent – Input resources	224
Table 6-25 AssetListCreation – Input resources	225
Table 6-26 AssetListCreation – Output resources	225
Table 6-24 Product Intent – Output resources	225
Table 6-27 Bending – Input resources	226
Table 6-28 Bending – Output resources	226
Table 6-29 ColorCorrection – Input resources	226
Table 6-30 ColorCorrection – Output resources	226
Table 6-31 ColorSpaceConversion – Input resources	227
Table 6-32 ColorSpaceConversion – Output resources	227
Table 6-33 ContactCopying – Output resources	227
Table 6-34 ContoneCalibration – Input resources	228
Table 6-35 ContoneCalibration – Output resources	228
Table 6-36 CylinderLayoutPreparation – Input resources	228
Table 6-37 CylinderLayoutPreparation – Output resources	228
Table 6-38 DBDocTemplateLayout – Input resources	228
Table 6-40 DBTemplateMerging – Input resources	229
Table 6-41 DBTemplateMerging – Output resources	229
Table 6-42 DigitalDelivery – Input resources	229
Table 6-43 DigitalDelivery – Output resources	229
Table 6-39 DBDocTemplateLayout – Output resources	229
Table 6-44 FormatConversion – Input resources	230
Table 6-45 FormatConversion – Output resources	230
Table 6-46 ImageReplacement – Input resources	230
Table 6-47 ImageReplacement – Output resources	230
Table 6-48 ImageSetting – Input resources	231
Table 6-49 ImageSetting – Output resources	231
Table 6-50 Imposition – Input resources	232
Table 6-51 Imposition – Output resources	232
Table 6-52 InkZoneCalculation – Input resources	232
Table 6-54 Interpreting – Input resources	233
Table 6-55 Interpreting – Output resources	233
Table 6-53 InkZoneCalculation – Output resources	233
Table 6-56 LayoutElementProduction – Input resources	234
Table 6-57 LayoutElementProduction – Output resources	234
Table 6-58 LayoutPreparation – Input resources	234
Table 6-59 LayoutPreparation – Output resources	234
Table 6-60 PDFToPSConversion – Input resources	235
Table 6-61 PDFToPSConversion – Output resources	235
Table 6-62 PDLCreation – Input resources	235
Table 6-63 PDLCreation – Output resources	235
Table 6-64 Preflight – Input resources	235
Table 6-65 Preflight – Output resources	236
Table 6-66 PreviewGeneration – Input resources	237
Table 6-68 PSToPDFConversion – Input resources	238
Table 6-69 PSToPDFConversion – Output resources	238
Table 6-70 RasterReading – Input resources	238
Table 6-67 PreviewGeneration – Output resources	238
Table 6-72 Rendering – Input resources	239
Table 6-73 Rendering – Output resources	239

Table 6-71 RasterReading – Output resources	239
Table 6-74 Scanning – Input resources	240
Table 6-75 Scanning – Output resources	240
Table 6-76 Screening – Input resources	241
Table 6-77 Screening – Output resources	241
Table 6-78 Separation – Input resources	241
Table 6-79 Separation – Output resources	241
Table 6-80 Stripping – Input resources	242
Table 6-81 Stripping – Output resources	242
Table 6-82 Tiling – Input resources	243
Table 6-83 Tiling – Output resources	243
Table 6-84 Trapping – Input resources	244
Table 6-85 Trapping – Output resources	244
Table 6-86 ConventionalPrinting – Input resources	245
Table 6-87 ConventionalPrinting – Output resources	246
Table 6-88 DigitalPrinting – Input resources	247
Table 6-89 DigitalPrinting – Output resources	248
Table 6-90 BlockPreparation – Input resources	249
Table 6-91 BlockPreparation – Output resources	249
Table 6-92 BoxFolding – Input resources	249
Table 6-93 BoxFolding – Output resources	249
Table 6-94 BoxPacking – Input resources	249
Table 6-96 Bundling – Input resources	250
Table 6-97 Bundling – Output resources	250
Table 6-98 CaseMaking – Input resources	250
Table 6-95 BoxPacking – Output resources	250
Table 6-100 CasingIn – Input resources	251
Table 6-101 CasingIn – Output resources	251
Table 6-102 ChannelBinding – Input resources	251
Table 6-103 ChannelBinding – Output resources	251
Table 6-99 CaseMaking – Output resources	251
Table 6-104 CoilBinding – Input resources	252
Table 6-105 CoilBinding – Output resources	252
Table 6-106 Collecting – Input resources	252
Table 6-107 Collecting – Output resources	252
Table 6-108 CoverApplication – Input resources	253
Table 6-109 CoverApplication – Output resources	253
Table 6-110 Creasing – Input resources	253
Table 6-111 Creasing – Output resources	253
Table 6-112 Cutting – Input resources	253
Table 6-114 Embossing – Input resources	254
Table 6-115 Embossing – Output resources	254
Table 6-116 EndSheetGluing – Input resources	254
Table 6-113 Cutting – Output resources	254
Table 6-117 EndSheetGluing – Output resources	255
Table 6-118 Feeding – Input resources	256
Table 6-119 Feeding – Output resources	256
Table 6-120 Folding – Input resources	256
Table 6-121 Folding – Output resources	256
Table 6-122 Gathering – Input resources	257

Table 6-123 Gathering – Output resources	257
Table 6-124 Gluing – Input resources	257
Table 6-125 Gluing – Output resources	257
Table 6-126 HeadBandApplication – Input resources	257
Table 6-127 HeadBandApplication – Output resources	257
Table 6-128 HoleMaking – Input resources	258
Table 6-129 HoleMaking – Output resources	258
Table 6-130 Inserting – Input resources	258
Table 6-131 Inserting – Output resources	258
Table 6-132 Jacketing – Input resources	259
Table 6-133 Jacketing – Output resources	259
Table 6-134 Labeling – Input resources	259
Table 6-135 Labeling – Output resources	259
Table 6-136 Laminating – Input resources	259
Table 6-137 Laminating – Output resources	259
Table 6-138 Numbering – Input resources	260
Table 6-139 Numbering – Output resources	260
Table 6-140 Palletizing – Input resources	260
Table 6-141 Palletizing – Output resources	260
Table 6-142 Perforating – Input resources	260
Table 6-143 Perforating – Output resources	260
Table 6-144 PlasticCombBinding – Input resources	261
Table 6-145 PlasticCombBinding – Output resources	261
Table 6-146 PrintRolling – Input resources	261
Table 6-147 PrintRolling – Output resources	261
Table 6-148 RingBinding – Input resources	261
Table 6-150 ShapeCutting – Input resources	262
Table 6-151 ShapeCutting – Output resources	262
Table 6-152 Shrinking – Input resources	262
Table 6-153 Shrinking – Output resources	262
Table 6-149 RingBinding – Output resources	262
Table 6-154 SpinePreparation – Input resources	263
Table 6-155 SpinePreparation – Output resources	263
Table 6-156 SpineTaping – Input resources	263
Table 6-157 SpineTaping – Output resources	263
Table 6-158 Stacking – Input resources	263
Table 6-159 Stacking – Output resources	263
Table 6-160 Stitching – Input resources	264
Table 6-161 Stitching – Output resources	264
Table 6-162 Strapping – Input resources	264
Table 6-163 Strapping – Output resources	264
Table 6-164 StripBinding – Input resources	265
Table 6-165 StripBinding – Output resources	265
Table 6-166 ThreadSealing – Input resources	265
Table 6-167 ThreadSealing – Output resources	265
Table 6-168 ThreadSewing – Input resources	265
Table 6-169 ThreadSewing – Output resources	265
Table 6-170 Trimming – Input resources	266
Table 6-171 Trimming – Output resources	266
Table 6-172 WebInlineFinishing – Input resources	266

Table 6-173 WebInlineFinishing – Output resources	266
Table 6-174 WireCombBinding – Input resources	267
Table 6-175 WireCombBinding – Output resources	267
Table 6-176 Wrapping – Input resources	267
Table 6-177 Wrapping – Output resources	267
Table 7-1 Template for intent resources	272
Table 7-2 List of span elements	272
Table 7-3 Abstract span element	273
Table 7-4 DurationSpan element	273
Table 7-5 EnumerationSpan element	274
Table 7-6 IntegerSpan element	274
Table 7-7 NameSpan element	274
Table 7-8 NumberSpan element	275
Table 7-9 OptionSpan element	275
Table 7-10 ShapeSpan element	276
Table 7-11 StringSpan element	276
Table 7-12 TimeSpan element	276
Table 7-13 XYPairSpan element	277
Table 7-14 ArtDeliveryIntent resource	277
Table 7-15 ArtDelivery element	280
Table 7-16 BindingIntent resource	283
Table 7-17 BindingType attribute – possible values	284
Table 7-18 BindList element	285
Table 7-19 BindItem element	285
Table 7-20 ChannelBinding element	286
Table 7-21 CoilBinding element	287
Table 7-22 EdgeGluing element	287
Table 7-23 HardcoverBinding element	287
Table 7-24 PlasticCombBinding element	289
Table 7-25 RingBinding element	290
Table 7-26 SaddleStitching element	291
Table 7-27 SideSewing element	291
Table 7-28 SideStitching element	291
Table 7-29 SoftCoverBinding element	291
Table 7-30 StripBinding element	292
Table 7-31 Tape element	293
Table 7-32 Tabs element	293
Table 7-33 ThreadSealing element	293
Table 7-34 ThreadSewing element	293
Table 7-35 WireCombBinding element	294
Table 7-36 ColorIntent resource	294
Table 7-37 ColorStandard attribute – possible values	296
Table 7-38 ColorsUsed element	297
Table 7-39 DeliveryIntent resource	297
Table 7-40 DropIntent element	300
Table 7-41 DropItemIntent element	302
Table 7-42 EmbossingIntent resource	303
Table 7-43 EmbossingItem element	303
Table 7-44 FoldingIntent resource	304
Table 7-45 HoleMakingIntent resource	305

Table 7-46 InsertingIntent resource	306
Table 7-47 InsertList element	306
Table 7-48 Insert element	306
Table 7-49 LaminatingIntent resource	307
Table 7-50 LayoutIntent resource	308
Table 7-51 MedialIntent resource	311
Table 7-52 NumberingIntent resource	316
Table 7-53 NumberItem element	316
Table 7-54 PackingIntent resource	317
Table 7-55 ProductionIntent resource	318
Table 7-56 ProofingIntent resource	319
Table 7-57 ProofItem element	319
Table 7-58 PublishingIntent resource	320
Table 7-59 ScreeningIntent resource	321
Table 7-60 ShapeCuttingIntent resource	321
Table 7-61 ShapeCut element	321
Table 7-62 Template for process resource	323
Table 7-63 Address resource	324
Table 7-64 ApprovalParams resource	325
Table 7-65 ApprovalPerson element	325
Table 7-66 ApprovalSuccess resource	325
Table 7-67 ApprovalDetails element	326
Table 7-68 Assembly resource	326
Table 7-69 AssemblySection element	328
Table 7-70 PageAssignedList element	328
Table 7-71 AssetListCreationParams resource	329
Table 7-72 AutomatedOverPrintParams resource	330
Table 7-73 BarcodeCompParams resource	330
Table 7-74 BarcodeReproParams resource	331
Table 7-75 BendingParams resource	332
Table 7-76 BinderySignature resource	332
Table 7-77 SignatureCell element	335
Table 7-78 BlockPreparationParams resource	336
Table 7-79 BoxFoldingParams resource	337
Table 7-80 BoxApplication element	338
Table 7-81 BoxFoldAction element	339
Table 7-82 Action attribute – possible values	339
Table 7-83 BoxPackingParams resource	342
Table 7-84 BufferParams resource	343
Table 7-85 Bundle resource	344
Table 7-86 BundleItem element	345
Table 7-87 BundlingParams resource	346
Table 7-88 ByteMap resource	347
Table 7-89 Band element	347
Table 7-90 PixelColorant element	347
Table 7-91 CaseMakingParams resource	348
Table 7-92 CasingInParams resource	349
Table 7-93 ChannelBindingParams resource	350
Table 7-94 CIELABMeasuringField resource	351
Table 7-95 CoilBindingParams resource	352

Table 7-96 CollectingParams resource	352
Table 7-97 Color resource	354
Table 7-98 DeviceNColor element	357
Table 7-99 PrintConditionColor element	357
Table 7-100 ColorantAlias resource	359
Table 7-101 ColorantControl resource	360
Table 7-102 ColorantOrder, ColorantParams and DeviceColorantOrder element	362
Table 7-103 ColorSpaceSubstitute element	362
Table 7-104 Sample output for different values of ProcessColorModel, ColorantParams, ColorantOrder, ColorantControlLink and DeviceColorantOrder Elements.	363
Table 7-105 ColorControlStrip resource	364
Table 7-106 ColorCorrectionParams resource	365
Table 7-107 ColorCorrectionOp element	365
Table 7-108 ColorMeasurementConditions resource	367
Table 7-109 ColorPool resource	368
Table 7-110 ColorSpaceConversionParams resource	368
Table 7-111 ColorSpaceConversionOp resource	370
Table 7-112 SourceCS attribute – possible values	372
Table 7-113 Mapping of SourceCS enumeration values to color spaces in the most common input file formats	373
Table 7-114 Effect of color space conversion operations on color spaces	375
Table 7-115 ComChannel resource	379
Table 7-116 ChannelTypeDetails attribute – predefined values for certain ChannelType values .	380
Table 7-117 Company resource	381
Table 7-118 Component – terms and definitions	382
Table 7-119 Component resource	382
Table 7-120 Contact resource	386
Table 7-121 ContactCopyParams resource	387
Table 7-122 ContentList resource	388
Table 7-123 ContentData element	388
Table 7-124 ContentType attribute – possible values	388
Table 7-125 ConventionalPrintingParams resource	389
Table 7-126 WorkStyle attribute – possible values	391
Table 7-127 CostCenter resource	392
Table 7-128 CoverApplicationParams resource	393
Table 7-129 Score element	393
Table 7-130 CreasingParams resource	394
Table 7-131 Crease element	394
Table 7-132 CustomerInfo resource	395
Table 7-133 CustomerMessage element	396
Table 7-134 CutBlock resource	397
Table 7-135 CutMark resource	397
Table 7-136 Cut mark types as specified by CutMark/@MarkType	398
Table 7-137 CuttingParams resource	399
Table 7-138 Cut element	399
Table 7-139 CylinderLayout resource	400
Table 7-140 CylinderPosition element	400
Table 7-141 CylinderLayoutPreparationParams resource	403
Table 7-142 DBMergeParams resource	403

Table 7-143 DBRules resource	404
Table 7-144 DBSchema resource	404
Table 7-145 DBSelection resource	404
Table 7-146 DeliveryParams resource	405
Table 7-147 Drop element	406
Table 7-148 Dropltem element	407
Table 7-149 DensityMeasuringField resource	408
Table 7-150 DevelopingParams resource	409
Table 7-151 Device resource	410
Table 7-152 IconList element	411
Table 7-153 Icon element	411
Table 7-154 Module element	412
Table 7-155 DeviceMark resource	413
Table 7-156 DeviceNSpace resource	414
Table 7-157 DieLayout resource	414
Table 7-158 Station element	414
Table 7-159 DigitalDeliveryParams resource	415
Table 7-160 DigitalMedia resource	416
Table 7-161 DigitalPrintingParams resource	417
Table 7-162 Disjointing resource	419
Table 7-163 Disposition resource	420
Table 7-164 ElementColorParams resource	421
Table 7-165 EmbossingParams resource	422
Table 7-166 Emboss element	423
Table 7-167 Employee resource	424
Table 7-168 EndSheetGluingParams resource	424
Table 7-169 EndSheet element	424
Table 7-170 ExposedMedia resource	426
Table 7-171 ExternallmpositionTemplate resource	427
Table 7-172 FeedingParams resource	427
Table 7-173 Feeder element	427
Table 7-174 FeederQualityParams element	429
Table 7-175 CollatingItem element	429
Table 7-176 FileSpec resource	430
Table 7-177 Container element	435
Table 7-178 FileAlias element	436
Table 7-179 FitPolicy resource	436
Table 7-180 Fold resource	437
Table 7-181 FoldingParams resource	439
Table 7-182 FontParams resource	442
Table 7-183 FontPolicy resource	443
Table 7-184 FormatConversionParams resource	443
Table 7-185 TIFFFormatParams element	444
Table 7-186 TIFFtag element	445
Table 7-187 TIFFEmbeddedFile element	446
Table 7-188 GatheringParams resource	447
Table 7-189 GeneralID element.....	447
Table 7-190 GlueApplication resource	448
Table 7-191 GlueLine resource	448
Table 7-192 GluingParams resource	450

Table 7-193 Glue element	450
Table 7-194 HeadBandApplicationParams resource	450
Table 7-195 Hole resource	451
Table 7-196 HoleLine resource	452
Table 7-197 HoleList resource	453
Table 7-198 HoleMakingParams resource	453
Table 7-199 IdentificationField resource	455
Table 7-200 EncodingDetails attribute – possible values	457
Table 7-201 BarcodeDetails element	457
Table 7-202 ExtraValues element	458
Table 7-203 Usage of barcode attributes for certain barcode types	458
Table 7-204 BarcodeDetails/@BarcodeVersion – possible values	459
Table 7-205 ImageCompressionParams resource	460
Table 7-206 ImageCompression element	461
Table 7-207 CCITTFaxParams element	463
Table 7-208 DCTParams element	464
Table 7-209 SourceCSs attribute – possible values	464
Table 7-210 FlateParams element	465
Table 7-211 JBIG2Params element	466
Table 7-212 JPEG2000Params element	466
Table 7-213 LZWParams element	466
Table 7-214 ImageReplacementParams resource	467
Table 7-215 ImageSetterParams resource	469
Table 7-216 Sides attribute – possible values	470
Table 7-217 Ink resource	471
Table 7-218 InkZoneCalculationParams resource	472
Table 7-219 InkZoneProfile resource	472
Table 7-220 InsertingParams resource	474
Table 7-221 Location of Inserts	474
Table 7-222 InsertSheet resource	476
Table 7-223 SheetUsage attribute – possible values	478
Table 7-224 InterpretingParams resource	479
Table 7-225 PDFInterpretingParams element	481
Table 7-226 OCGControl element	482
Table 7-227 JacketingParams resource	484
Table 7-228 JobField resource	484
Table 7-229 LabelingParams resource	486
Table 7-230 LaminatingParams resource	487
Table 7-231 Layout resource	487
Table 7-232 LayerList element	489
Table 7-233 LayerDetails element	489
Table 7-234 Abstract PlacedObject element	489
Table 7-235 ContentObject element	491
Table 7-236 Example (1) of Ord attribute in PlacedObject elements	494
Table 7-237 Example (2) of Ord attribute in PlacedObject elements	495
Table 7-238 MarkObject element	496
Table 7-239 DynamicField element	497
Table 7-240 LayoutElement resource	498
Table 7-241 ElementType attribute – possible values	500
Table 7-242 Dependencies element	501

Table 7-243 LayoutElementProductionParams resource	501
Table 7-244 LayoutElementPart element	501
Table 7-245 BarcodeProductionParams element	501
Table 7-246 LayoutPreparationParams resource	502
Table 7-247 FrontMarkList attribute – possible values	509
Table 7-248 PageCell element	510
Table 7-249 ImageShift element	511
Table 7-250 ManualLaborParams resource	514
Table 7-251 Media resource	514
Table 7-252 MediaTypeDetails attribute – possible values	520
Table 7-253 MediaLayers element	521
Table 7-254 MiscConsumable resource	523
Table 7-255 MISDetails element	524
Table 7-256 NodeInfo resource	525
Table 7-257 NumberingParams resource	527
Table 7-258 NumberingParam element	527
Table 7-259 ObjectResolution resource	527
Table 7-260 OrderingParams resource	528
Table 7-261 PageList resource	528
Table 7-262 PageData element	530
Table 7-263 PageElement element	531
Table 7-264 Pallet resource	532
Table 7-265 PalletizingParams resource	532
Table 7-266 PDFToPSConversionParams resource	533
Table 7-267 PDLCreationParams resource	536
Table 7-268 PDLResourceAlias resource	537
Table 7-269 PerforatingParams resource	537
Table 7-270 Perforate element	537
Table 7-271 Person resource	538
Table 7-272 PlasticCombBindingParams resource	539
Table 7-273 PlasticCombBindingParams/@Type attribute – possible values	540
Table 7-274 PreflightParams resource	540
Table 7-275 PreflightAction element	541
Table 7-276 BasicPreflightTest element	541
Table 7-277 PreflightArgument element	542
Table 7-278 BoxArgument element	542
Table 7-279 BoxToBoxDifference element	542
Table 7-280 PreflightReport resource	543
Table 7-281 PRItem element	543
Table 7-282 PRError element	544
Table 7-283 PRGroup element	544
Table 7-284 Abstract PRGroupOccurrenceBase element	546
Table 7-285 PRGroupOccurrence element	546
Table 7-286 StringListValue element	546
Table 7-287 ArgumentValue element	546
Table 7-288 PROccurrence element	547
Table 7-289 PreflightReportRulePool resource	547
Table 7-290 PRRule element	547
Table 7-291 PRRuleAttr element	547
Table 7-292 Contingent Report Behavior	548

Table 7-293 Preview resource	549
Table 7-294 PreviewGenerationParams resource	551
Table 7-295 PrintCondition resource	552
Table 7-296 PrintRollingParams resource	553
Table 7-297 ProductionPath resource	554
Table 7-298 FolderSuperstructureWebPath, PostPressComponentPath and PrintingUnitWeb- Path element	554
Table 7-299 PSToPDFConversionParams resource	555
Table 7-300 AdvancedParams element	557
Table 7-301 PDFXParams element	559
Table 7-302 PDFXOutputIntentProfile attribute – possible values	560
Table 7-303 ThinPDFParams element	560
Table 7-304 QualityControlParams resource	561
Table 7-305 BindingQualityParams element	561
Table 7-306 QualityControlResult resource	561
Table 7-307 QualityMeasurement element	562
Table 7-308 BindingQualityMeasurement element	562
Table 7-309 RasterReadingParams resource	562
Table 7-310 RegisterMark resource	564
Table 7-311 RegisterRibbon resource	565
Table 7-312 RenderingParams resource	565
Table 7-313 ResourceDefinitionParams resource	566
Table 7-314 ResourceParam element	566
Table 7-315 RingBindingParams resource	567
Table 7-316 RollStand resource	568
Table 7-317 RunList resource	569
Table 7-318 DynamicInput element	573
Table 7-319 ScanParams resource	575
Table 7-320 ScavengerArea resource	576
Table 7-321 ScreeningParams resource	576
Table 7-322 ScreenSelector element	577
Table 7-323 SeparationControlParams resource	579
Table 7-324 SeparationSpec resource	579
Table 7-325 ShapeCuttingParams resource	579
Table 7-326 Shape element	580
Table 7-327 ShrinkingParams resource	581
Table 7-328 SpinePreparationParams resource	581
Table 7-329 Operations attribute – possible values	582
Table 7-330 SpineTapingParams resource	583
Table 7-331 Parameters in Stacking	584
Table 7-332 StackingParams resource	585
Table 7-333 StitchingParams resource	588
Table 7-334 Strap resource	589
Table 7-335 StrappingParams resource	590
Table 7-336 StripBindingParams resource	590
Table 7-337 StrippingParams resource	591
Table 7-338 Position element	593
Table 7-339 StripCellParams element	594
Table 7-340 StripMark element	596
Table 7-341 MarkName attribute – included values	597

Table 7-342 MarkSide attribute – possible values	597
Table 7-343 ThreadSealingParams resource	597
Table 7-344 ThreadSewingParams resource	599
Table 7-345 Tile resource	600
Table 7-346 Tool resource	600
Table 7-347 TransferCurve resource	601
Table 7-348 TransferCurvePool resource	602
Table 7-349 TransferCurveSet element	602
Table 7-350 TransferFunctionControl resource	603
Table 7-351 TrappingDetails resource	603
Table 7-352 TrappingOrder element	604
Table 7-353 TrappingParams resource	604
Table 7-354 ColorantZoneDetails element	607
Table 7-355 TrapRegion resource	608
Table 7-356 TrimmingParams resource	609
Table 7-357 UsageCounter resource	609
Table 7-358 VerificationParams resource	611
Table 7-359 WebInlineFinishingParams resource	611
Table 7-360 FolderProduction element	612
Table 7-361 WireCombBindingParams resource	612
Table 7-362 WrappingParams resource	613
Table 7-363 DeviceCap element	615
Table 7-364 ActionPool element	617
Table 7-365 Action element	617
Table 7-366 DevCapPool element	618
Table 7-367 ModulePool element	618
Table 7-368 ModuleCap element	618
Table 7-369 DevCaps element	619
Table 7-370 Loc element	621
Table 7-371 DevCap element	622
Table 7-372 Abstract State element	625
Table 7-373 ListType attribute – possible values	627
Table 7-374 List of State elements	627
Table 7-375 BooleanState element	628
Table 7-376 ValueLoc element	628
Table 7-377 DateTimeState element	628
Table 7-378 DurationState element	629
Table 7-379 EnumerationState element	629
Table 7-380 IntegerState element	630
Table 7-381 MatrixState element	631
Table 7-382 MatrixState/Value element	632
Table 7-383 NameState element	633
Table 7-384 NumberState element	633
Table 7-385 PDFPathState element	634
Table 7-386 PDFPathState/Value element	634
Table 7-387 RectangleState element	635
Table 7-388 ShapeState element	635
Table 7-389 StringState element	636
Table 7-390 StringState/Value element	637
Table 7-391 XYPairState element	637

Table 7-392 DisplayGroupPool element	638
Table 7-393 DisplayGroup element	639
Table 7-394 FeaturePool element	639
Table 7-395 MacroPool element	639
Table 7-396 macro element	640
Table 7-397 choice element	641
Table 7-398 otherwise element	641
Table 7-399 when element	641
Table 7-400 set element	641
Table 7-401 FeatureAttribute element	641
Table 7-402 call element	642
Table 7-403 Performance element	642
Table 7-404 TestPool element	643
Table 7-405 Test element	643
Table 7-406 List of abstract Term elements	643
Table 7-407 and element	645
Table 7-408 or element	645
Table 7-409 xor element	646
Table 7-410 not element	646
Table 7-411 Mapping of Evaluation elements to State element	646
Table 7-412 Abstract Evaluation element	648
Table 7-413 BooleanEvaluation element	648
Table 7-414 DateTimeEvaluation element	648
Table 7-415 DurationEvaluation element	648
Table 7-416 EnumerationEvaluation element	649
Table 7-417 IntegerEvaluation element	649
Table 7-418 MatrixEvaluation element	649
Table 7-419 MatrixEvaluation/Value element	650
Table 7-420 NameEvaluation element	650
Table 7-421 NumberEvaluation element	650
Table 7-422 PDFPathEvaluation element	650
Table 7-423 PDFPathEvaluation/Value element	651
Table 7-424 RectangleEvaluation element	651
Table 7-425 ShapeEvaluation element	651
Table 7-426 StringEvaluation element	651
Table 7-427 StringEvaluation/Value element	652
Table 7-428 XYPairEvaluation element	652
Table 7-429 TestRef element	652
Table 7-430 Object Classes for a document	660
Table 7-431 Properties Implemented by Class	661
Table 7-432 Mapping between property types (in the preflight spec) and evaluations	662
Table 7-433 Annotation properties	662
Table 7-434 AnnotationType attribute – possible values	663
Table 7-435 Box properties and Box attribute – possible values	663
Table 7-436 Box properties – other ones	664
Table 7-437 Class properties	664
Table 7-438 ClassName attribute – possible values	664
Table 7-439 PropertyList attribute – possible values	665
Table 7-440 Colorant properties	665
Table 7-441 Document properties	665

Table 7-442 Fill properties	669
Table 7-443 FillColorType attribute – possible values	669
Table 7-444 Font properties	669
Table 7-445 FontType attribute – possible values	670
Table 7-446 Graphic properties	670
Table 7-447 Image properties	672
Table 7-448 Logical properties	674
Table 7-449 PageBox properties	674
Table 7-450 Pages properties	674
Table 7-451 PDLObject properties	675
Table 7-452 Reference properties	676
Table 7-453 Shading properties	676
Table 7-454 Stroke properties	676
Table 7-455 Text properties	677
Table 7-456 Vector properties	678
Table 8-1 MIME Content-Types	681
Table A-1 JDFJMFVersion enumeration Values	697
Table A-2 Named Colors	698
Table A-3 Page Orientation	698
Table A-4 Side enumeration Values	698
Table A-5 WorkStyle enumeration Values	698
Table A-6 XYRelation enumeration Values	699
Table C-1 StatusDetails and Status mapping for generic devices	703
Table C-2 Printing Device specific StatusDetails	705
Table C-3 Postpress Device specific StatusDetails	706
Table C-4 ModuleType definition for conventional printing devices	707
Table C-5 ModuleType definition Gathering / Collecting	707
Table C-6 ModuleType definition for DigitalPrinting	707
Table C-7 Module Type definition for web printing devices; Possible Modules of a PrintingUnitWebPath	707
Table C-8 Module Type definition for web printing devices; Possible Modules of a FolderSuperstructureWebPath	708
Table C-9 Module Type definition for web printing devices; Possible Modules of a PostPressComponentPath	709
Table C-10 Type attribute	709
Table C-11 Barcode element	710
Table C-12 FCNKey element	710
Table C-13 SystemTimeSet element	710
Table C-14 CounterReset element	710
Table C-15 Error element	710
Table C-16 ErrorData element	711
Table C-17 Event element	711
Table C-18 Milestone element	711
Table C-19 MessageEvents and MilestoneType Values	712
Table C-20 Locations within Printers	714
Table D-1 Return codes for JMF	717
Table E-1 Attributes for Color Space Adjustment	719
Table F-1 Conversion Factor from USWeight (lbs) to Weight (g/m ²)	721
Table F-2 Grammage Equivalents for Common (US) Basis Weights	721
Table F-3 Japanese Media Weight	722

Table G-1 Media Sizes	725
Table H-1 MimeType (IANA Registered) and MimeTypeVersion Combinations	729
Table H-2 MimeType (File Type) and MimeTypeVersion combinations	732
Table I-1 Use Cases showing MimeType, URL and Compression attribute values	735
Table I-2 AppOS and OSVersion Examples	742
Table J-1 Predefined variables used in @FileTemplate and @ValueTemplate	743
Table L-1 References	747
Table M-1 Naming Scheme for Hole Patterns	761
Table M-2 Hole Details for R2 Series	762
Table M-3 Hole Details for R3 and R4 Series	763
Table M-4 Hole Details for R5 and R6 Series	764
Table M-5 Hole Details for R7 and R11 Series	766
Table M-6 Hole Details for P, W, C and S Series	767
Table O-1 Compatibility Warnings	797
Table O-2 New Items	797
Table O-3 Deprecated Items	809
Table O-4 Modified Items	811
Table P-1 Contents of the abstract ResourceUpdate Element	815
Table P-2 Contents of the StatusPool element	816
Table P-3 Contents of the PartStatus element	816
Table P-4 Added element	817
Table P-5 ChangedAttribute element	817
Table P-6 Removed element	817
Table P-7 Contents of the NodeInfo query message.....	818
Table P-8 Contents of the NodeInfoQuParams element	818
Table P-9 Contents of the NodeInfo Command Message	818
Table P-10 Contents of the NodeInfoCmdParams element.....	818
Table P-11 Contents of the NodeInfoResp element	819
Table P-12 Contents of the KnownJDFServices message	820
Table P-13 Contents of the JDFService element	820
Table P-14 Contents of the QueueEntryStatus message	821
Table P-15 Contents of the QueueEntryDefList element	821