

JDF Specification

Release 1.4

November 10, 2008



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 © 2000-2008, 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 © 2000-2008, International Cooperation for the Integration of Processes in Prepress, Press and Postpress (CIP4) with registered office in Zurich, Switzerland.

Trademarks and Tradenames

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

Except as contained in this legal notice or as allowed by membership in CIP4, the name of CIP4 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	xxxv
Chapter 1 JDF Specification Release 1.4	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
1.3.3.1 Location of Callouts.....	4
1.3.4 Specification of Cardinality.....	4
1.4 Glossary	5
1.4.1 Conformance Terminology.....	10
1.4.2 Conformance Requirements for JDF Entities	11
1.4.2.1 Conformance Requirements for Support of Attributes and Attribute Values.....	11
1.4.2.2 Conformance Requirements for Support of Elements	12
1.4.2.3 Conformance Requirements for Support of Processes	12
1.4.2.4 Conformance Requirements for Support of Combined Processes.....	12
1.4.3 Conformance to Settings Policy.....	13
1.5 Data Structures	13
1.6 Units	16
1.6.1 Counting in JDF	17
Chapter 2 Overview of JDF	19
2.1 System Components	19
2.1.1 Job Components.....	19
2.1.1.1 Jobs and Nodes	19
2.1.1.2 Elements.....	19
2.1.1.3 Attributes	19
2.1.1.4 Relationships.....	19
2.1.1.5 Links	20
2.1.2 Workflow Component Roles	20
2.1.2.1 Machines.....	20
2.1.2.2 Devices.....	20

2.1.2.3 Agents	20
2.1.2.4 Controllers	21
2.1.2.5 Management Information Systems—MIS	21
2.1.2.6 System Interaction	21
2.2 JDF Workflow	22
2.2.1 Job Structure.....	23
2.3 Hierarchical Tree Structure and Networks in JDF	25
2.4 Role of Messaging in JDF	26
2.5 Coordinate Systems in JDF	27
2.5.1 Introduction	27
2.5.1.1 Source Coordinate Systems	28
2.5.2 Coordinates and Transformations.....	28
2.5.3 Coordinate Systems of Resources and Processes.....	29
2.5.3.1 Coordinate Systems of Combined Processes.....	29
2.5.3.2 Coordinate System Transformations	29
2.5.4 Product Example: Simple Brochure	31
2.5.5 General Rules.....	36
2.5.6 Homogeneous Coordinates	36
Chapter 3 Structure of JDF Nodes and Jobs	39
3.1 Generic Contents of All Elements	40
3.1.1 Comment	42
3.1.2 GeneralID.....	43
3.1.3 Structure Diagram.....	44
3.2 JDF Node	45
3.2.1 Structure Diagram.....	52
3.3 Common Node Types	53
3.3.1 Product Intent Nodes	54
3.3.2 Process Group Nodes.....	55
3.3.2.1 Use of the Types Attribute in Process Group Nodes – Gray Boxes	55
3.3.2.2 Use of the NamedFeatures Attribute in Product and Process Group Nodes.....	56
3.3.2.3 ResourceLink Structure in Process Group Nodes.....	56
3.3.3 Combined Process Nodes	57
3.3.3.1 Combined Process Nodes with Multiple Processes of the Same Type.....	58
3.3.3.2 Specifying non-linear dependencies in a Combined Process Node.....	59
3.3.4 Process Nodes.....	60
3.4 AncestorPool	60
3.4.1 Ancestor.....	61
3.5 CustomerInfo	62

3.6 NodeInfo	62
3.7 StatusPool	62
3.8 ResourcePool and its Resource Children	62
3.8.1 ResourcePool	62
3.8.2 Resource.....	62
3.8.3 Abstract Resource	63
3.8.3.1 SourceResource	67
3.8.4 Structure Diagram.....	67
3.8.5 Resource Classes.....	69
3.8.5.1 Parameter Resource	69
3.8.5.1.1 Abstract Parameter Resource	69
3.8.5.2 Intent Resource	69
3.8.5.3 Implementation Resource	69
3.8.5.3.1 ImplementationResource	69
3.8.5.4 Consumable Resource.....	69
3.8.5.5 Quantity Resource.....	70
3.8.5.6 Handling Resource.....	70
3.8.5.7 Physical Resource	70
3.8.5.7.1 PhysicalResource	70
3.8.5.7.2 Abstract Physical Resource	70
3.8.5.7.3 Location	71
3.8.5.8 Placeholder Resource.....	72
3.8.6 Position of Resources within JDF Nodes.....	72
3.8.7 Pipe Resources.....	72
3.8.8 ResourceUpdate	72
3.9 ResourceLinkPool and ResourceLink	73
3.9.1 ResourceLinkPool.....	73
3.9.2 ResourceLink	73
3.9.3 Structure Diagram.....	75
3.9.4 Abstract ResourceLink.....	76
3.9.4.1 AmountPool and PartAmount.....	80
3.9.4.1.1 AmountPool	80
3.9.4.1.2 PartAmount	81
3.9.5 ParameterLink.....	82
3.9.6 ImplementationLink.....	82
3.9.6.1 Abstract ImplementationLink	82
3.9.7 ConsumableLink	82
3.9.8 HandlingLink	82
3.9.9 QuantityLink	82
3.9.10 PhysicalLink	82
3.9.10.1 Abstract PhysicalLink.....	83
3.9.10.2 Identification of Physical Resources.....	85
3.9.10.2.1 Lot	85

3.9.11 PlaceholderLink.....	86
3.9.12 IntentLink	86
3.10 ResourcePool and ResourceLinkPool – Deep Structure	86
3.10.1 ResourceElement – Subelement of a Resource.....	86
3.10.1.1 ResourceElement	86
3.10.1.2 Abstract ResourceElement.....	86
3.10.2 ResourceRef – Element for Inter-Resource Linking and refElement.....	87
3.10.2.1 ResourceRef.....	87
3.10.2.2 Abstract ResourceRef	87
3.10.2.3 ResourceRef Elements in the AncestorPool/Ancestor Element	88
3.10.2.4 Status of a Resource that Contains an rRef Reference	88
3.10.2.5 Alignment of ResourceLink and ResourceRef	89
3.10.3 Set of Resources and Partitioned Subsets Thereof.....	90
3.10.4 Resource Amount	90
3.10.4.1 Evaluating and Updating Amount-Related Attributes in a Device.....	90
3.10.4.2 Specifying Amount for a Partially-Completed Process	91
3.10.5 Description of Partitioned Resources.....	93
3.10.5.1 Subelements in Partitioned Resources	94
3.10.5.2 Amount in Partitioned Resources	94
3.10.5.3 Relating PartIDKeys and Partitions.....	95
3.10.5.3.1 Incomplete Partitions	95
3.10.5.3.2 Number of Partition Keys per Partitioned Leaf or Node	96
3.10.5.3.3 Degenerate Partitions	96
3.10.5.4 Partitioning of Resource Subelements.....	97
3.10.5.5 Logical Partitions and the Identical Element.....	99
3.10.5.5.1 Identical	99
3.10.5.5.2 Restrictions when using Identical Elements	100
3.10.6 PartIDKeys Attribute and Partition Keys	102
3.10.6.1 Partitionable Resource	102
3.10.6.2 Part.....	104
3.10.6.3 Options in Intent Resources	113
3.10.6.4 Locations of Physical Resources	113
3.10.7 Linking to Resources	114
3.10.7.1 Linking to Subsets of Resources.....	115
3.10.7.2 Reordering the Processing of Resources	115
3.10.7.3 Handling Amount in a ResourceLink to a Partitioned Resource.....	115
3.10.7.4 Implicit, Sparse and Explicit PartUsage in Partitioned Resources	116
3.10.7.5 Referencing Multiple Resources of the Same Type	118
3.10.8 Splitting and Combining Resources.....	119
3.11 AuditPool and Audit	120
3.11.1 AuditPool.....	121
3.11.2 Structure Diagram.....	122
3.11.3 Abstract Audit.....	123
3.11.4 Audit.....	123

3.11.4.1 Created	124
3.11.4.2 Deleted	124
3.11.4.3 Merged	124
3.11.4.4 Modified.....	125
3.11.4.5 Notification	125
3.11.4.6 PhaseTime.....	127
3.11.4.6.1 ModulePhase	129
3.11.4.7 ProcessRun.....	130
3.11.4.8 ResourceAudit	131
3.11.4.8.1 Logging Machine Data by Using the ResourceAudit	132
3.11.4.8.2 Logging Changes in Product Descriptions by Using the ResourceAudit	133
3.11.4.9 Spawned.....	134
3.12 JDF Extensibility	135
3.12.1 Namespaces in XML.....	135
3.12.1.1 JDF Namespace	135
3.12.1.2 JDF Extension Namespace	135
3.12.2 Extending Process Types	135
3.12.2.1 Rules about Process Extension	136
3.12.3 Extending the NodeInfo and CustomerInfo Nodes	136
3.12.4 Extending Existing Resources	136
3.12.5 Extending NMTOKEN Lists.....	136
3.12.6 Creating New Resources	137
3.12.7 Future JDF Extensions	137
3.12.8 Maintaining Extensions	137
3.12.9 Processing Unknown Extensions.....	137
3.12.10 Derivation of Types in XML Schema.....	138
3.13 JDF Versioning	138
3.13.1 JDF Versioning Requirements	138
3.13.2 JDF Version Definition	138
3.13.3 JDF Version Policies.....	138
3.13.3.1 JDF Specification Version Policies	138
3.13.3.2 JDF Schema Version Policies.....	139
3.13.3.3 JDF Application Version Policies.....	139
3.13.3.3.1 JDF Agent Version Policies	139
3.13.3.3.2 JDF Device/Controller Version Policies	140
Chapter 4 Life Cycle of JDF	141
4.1 Creation and Modification	141
4.1.1 Product Intent Constructs	141
4.1.1.1 Representation of Product Intent	142
4.1.1.2 Representation of Product Binding.....	142
4.1.2 Defining Business Objects Using Intent Resources.....	142
4.1.3 Specification of Delivery of End Products	145

4.1.4 Specification of Process Specifics for Product Intent Nodes	145
4.2 Process Routing	146
4.2.1 Determining Executable Nodes	147
4.2.2 Distributing Processing to Work Centers or Devices	147
4.2.3 Device / Controller Selection.....	148
4.3 Execution Model	148
4.3.1 Serial Processing	148
4.3.2 Partial Processing of Nodes with Partitioned Resources.....	150
4.3.3 Overlapping Processing Using Pipes.....	151
4.3.3.1 Pipes of Partitionable Resources.....	153
4.3.3.2 Dynamic Pipes	154
4.3.3.3 Comparison of Non-Dynamic and Dynamic Pipes.....	155
4.3.4 Parallel Processing	155
4.3.5 Iterative Processing	155
4.3.5.1 Informal Iterative Processing.....	156
4.3.5.2 Formal Iterative Processing	156
4.3.6 Approval, Quality Control and Verification	156
4.4 Spawning and Merging	156
4.4.1 Case 1: Standard Spawning and Merging	158
4.4.2 Case 2: Spawning and Merging with Resource Copying.....	159
4.4.2.1 Spawning of Resources with Inter-Resource Links.....	159
4.4.3 Case 3: Parallel Spawning and Merging of Partitioned Resources	160
4.4.4 Case 4: Nested Spawning and Merging in Reverse Sequence	161
4.4.5 Case 5: Spawning and Merging of Independent Jobs	161
4.4.6 Case 6: Simultaneous Spawning and Merging of Multiple Nodes	163
4.5 Node and Resource IDs	163
4.6 Error Handling	164
4.6.1 Classification of Notifications	164
4.6.2 Event Description.....	164
4.6.3 Error Logging in the JDF File	164
4.6.4 Error Handling via Messaging (JMF)	164
4.7 Test Running	164
4.7.1 Resource Status During a Test Run	165
4.8 Capability and Constraint Definitions	165
Chapter 5 JDF Messaging with the Job Messaging Format	167
5.1 JMF Root	167
5.1.1 Message	169
5.1.2 Structure Diagram.....	170

5.2 List of All JMF Messages	172
5.3 JMF Message Families	174
5.3.1 Query	174
5.3.2 Response.....	176
5.3.3 Signal	177
5.3.3.1 Trigger	178
5.3.3.2 ChangedPath	179
5.3.4 Command	180
5.3.5 Acknowledge.....	181
5.3.6 Registration.....	183
5.4 JMF Handshaking	183
5.4.1 Single Query/Command Response Communication	183
5.4.2 Signal and Acknowledge Handshaking.....	183
5.4.3 Persistent Channels.....	184
5.4.3.1 Persistent Channels for Signals.....	184
5.4.3.2 Persistent Channels for Commands	184
5.4.4 Subscription	184
5.4.4.1 ObservationTarget	186
5.4.5 Creating Persistent Channels in a JDF Node	186
5.4.6 Deleting Persistent Channels.....	186
5.5 JMF Messaging Levels	187
5.6 Error and Event Messages	187
5.7 Message Template	188
5.7.1 Object Type Column	188
5.7.1.1 QueryTypeObj	188
5.7.1.2 CommandTypeObj.....	189
5.7.1.3 ResponseTypeObj.....	189
5.8 Messages for Events and Capabilities	189
5.8.1 Events	189
5.8.1.1 NotificationFilter	191
5.8.1.2 NotificationDef	192
5.8.2 KnownControllers.....	192
5.8.2.1 ControllerFilter	193
5.8.2.2 JDFController	193
5.8.3 KnownDevices	194
5.8.3.1 DeviceFilter	194
5.8.3.2 DeviceList.....	195
5.8.4 KnownJDFServices.....	196
5.8.5 KnownMessages.....	196
5.8.5.1 KnownMsgQuParams	196
5.8.5.2 MessageService	196

5.8.6 KnownSubscriptions	198
5.8.6.1 SubscriptionFilter.....	198
5.8.6.2 SubscriptionInfo.....	199
5.8.7 Notification	200
5.8.8 RepeatMessages	200
5.8.8.1 MsgFilter.....	200
5.8.9 RequestForAuthentication	202
5.8.9.1 RequestForAuthentication Command.....	202
5.8.9.1.1 AuthenticationCmdParams	203
5.8.9.1.2 Certificate	204
5.8.9.1.3 AuthenticationResp	204
5.8.9.2 RequestForAuthentication Query	204
5.8.9.2.1 AuthenticationQuParam	205
5.8.10 StopPersistentChannel	206
5.8.10.1 StopPersChParams.....	206
5.9 Messages to Query/Command a Job, Device or Controller	207
5.9.1 FlushResources	208
5.9.1.1 FlushResources Command	208
5.9.1.2 FlushResources Signal.....	208
5.9.1.2.1 FlushResourceParams	209
5.9.1.2.2 FlushedResources	209
5.9.2 ModifyNode.....	209
5.9.2.1 ModifyNode Command	209
5.9.2.2 ModifyNode Signal.....	209
5.9.2.2.1 ModifyNodeCmdParams	209
5.9.2.2.2 NewComment	210
5.9.3 NewJDF	210
5.9.3.1 NewJDF Query	211
5.9.3.1.1 NewJDFQuParams	211
5.9.3.2 NewJDF Command.....	211
5.9.3.2.1 NewJDFCmdParams	211
5.9.3.2.2 IDInfo	212
5.9.4 NodeInfo	212
5.9.5 Occupation.....	212
5.9.5.1 EmployeeDef	212
5.9.5.2 Occupation	212
5.9.6 Resource.....	214
5.9.6.1 Resource Query.....	214
5.9.6.1.1 ResourceQuParams	214
5.9.6.2 Resource Command.....	218
5.9.6.2.1 ResourceCmdParams	219
5.9.6.2.2 ResourceInfo	221
5.9.7 ResourcePull.....	225
5.9.7.1 ResourcePullParams	226
5.9.8 ShutDown	227

5.9.8.1 ShutDownCmdParams	228
5.9.9 Status	228
5.9.9.1 StatusQuParams	229
5.9.9.2 DeviceInfo	230
5.9.9.3 JobPhase.....	232
5.9.9.4 ModuleStatus	234
5.9.10 Track	236
5.9.10.1 TrackFilter	236
5.9.10.2 TrackResult.....	236
5.9.11 UpdateJDF	237
5.9.11.1 UpdateJDF Command.....	237
5.9.11.2 UpdateJDF Signal	238
5.9.11.2.1 UpdateJDFCmdParams	238
5.9.11.2.2 CreateLink	239
5.9.11.2.3 CreateResource	239
5.9.11.2.4 MoveResource	239
5.9.11.2.5 RemoveLink	240
5.9.12 WakeUp	241
5.9.12.1 WakeUpCmdParams.....	241
5.10 Messages for Pipe Control	241
5.10.1 PipeClose.....	241
5.10.2 PipePull.....	242
5.10.2.1 PipeParams.....	242
5.10.3 PipePush.....	243
5.10.4 PipePause.....	244
5.11 Queue Support	244
5.11.1 Queue Entry ID Generation	245
5.11.2 Use of QueueFilter in Queue Entry Handling commands	245
5.12 Messages for Queue Entry Handling	245
5.12.1 AbortQueueEntry	248
5.12.2 HoldQueueEntry	248
5.12.3 RemoveQueueEntry	249
5.12.4 RequestQueueEntry	249
5.12.4.1 RequestQueueEntryParams	250
5.12.5 ResubmitQueueEntry.....	250
5.12.5.1 ResubmissionParams	251
5.12.6 ResumeQueueEntry	251
5.12.7 ReturnQueueEntry	251
5.12.7.1 ReturnQueueEntryParams	251
5.12.8 SetQueueEntryPosition.....	252
5.12.8.1 QueueEntryPosParams.....	252
5.12.9 SetQueueEntryPriority	253

5.12.9.1 QueueEntryPriParams.....	253
5.12.10 SubmitQueueEntry.....	253
5.12.10.1 QueueSubmissionParams.....	253
5.12.11 SuspendQueueEntry.....	256
5.13 Messages for Global Handling of Queues	256
5.13.1 CloseQueue.....	257
5.13.2 FlushQueue	257
5.13.2.1 FlushQueue Command	257
5.13.2.1.1 FlushQueueParams	258
5.13.2.2 FlushQueue Query	258
5.13.2.2.1 FlushQueueInfo	258
5.13.3 HoldQueue.....	259
5.13.4 OpenQueue	259
5.13.5 QueueEntryStatus.....	259
5.13.6 QueueStatus	259
5.13.7 ResumeQueue.....	259
5.13.8 SubmissionMethods.....	260
5.13.8.1 SubmissionMethods.....	260
5.14 Elements for Queues	261
5.14.1 Queue	261
5.14.2 QueueEntry.....	262
5.14.3 QueueEntryDef	263
5.14.4 QueueFilter	264
5.15 Gang Jobs	265
5.15.1 ForceGang.....	266
5.15.1.1 GangCmdFilter	266
5.15.2 GangStatus.....	266
5.15.2.1 GangQuFilter	266
5.15.2.2 GangInfo	266
5.16 Extending Messages	266
5.16.1 IfraTrack Support.....	267
Chapter 6 Processes	269
6.1 Process Template	269
6.2 General Processes	270
6.2.1 Approval.....	270
6.2.2 Buffer	271
6.2.3 Combine.....	271
6.2.4 Delivery	271
6.2.5 ManualLabor	272

6.2.6 Ordering	272
6.2.7 Packing	273
6.2.8 QualityControl	273
6.2.9 ResourceDefinition.....	273
6.2.10 Split.....	273
6.2.11 Verification	274
6.3 Product Intent Descriptions	274
6.4 Prepress Processes	275
6.4.1 AssetListCreation.....	275
6.4.2 Bending.....	276
6.4.3 ColorCorrection.....	276
6.4.4 ColorSpaceConversion	277
6.4.5 ContactCopying	277
6.4.6 ContoneCalibration	278
6.4.7 CylinderLayoutPreparation	278
6.4.8 DBDocTemplateLayout.....	279
6.4.9 DBTemplateMerging	279
6.4.10 DieDesign	279
6.4.11 DieLayoutProduction.....	280
6.4.12 DigitalDelivery	282
6.4.13 FilmToPlateCopying.....	283
6.4.14 FormatConversion	283
6.4.15 ImageReplacement.....	283
6.4.16 ImageSetting.....	284
6.4.17 Imposition.....	284
6.4.17.1 Glossary for Automated Imposition	286
6.4.17.2 Variables for Automated Imposition	289
6.4.17.3 Execution Model for Automated Imposition	290
6.4.17.4 Configuration for Various Automated Impositions	292
6.4.17.4.1 Using Logical Stacks	292
6.4.17.4.1.1 Imposition for Cut and Stack	293
6.4.17.4.2 Imposition for Signatures with Saddle Stitching	294
6.4.17.4.3 Selecting from Multiple Imposition Templates When Processing an unstructured PDL	295
6.4.17.4.4 Imposition for Start of a Chapter	295
6.4.17.4.5 Imposition for Regenerating Sheet Surfaces	295
6.4.17.4.6 Imposition for Document-Major Processing of a VDP Structured PDL	296
6.4.18 InkZoneCalculation	296
6.4.19 Interpreting.....	296
6.4.20 LayoutElementProduction.....	297
6.4.21 LayoutPreparation.....	298
6.4.22 LayoutShifting	298
6.4.23 PageAssigning	299

6.4.24 PDFToPSConversion.....	299
6.4.25 PDLCreation	299
6.4.26 Preflight.....	300
6.4.27 PreviewGeneration	300
6.4.28 Proofing.....	302
6.4.29 PSToPDFConversion.....	302
6.4.30 RasterReading	303
6.4.31 Rendering	303
6.4.32 RIPing	304
6.4.33 Scanning.....	305
6.4.34 Screening.....	305
6.4.35 Separation.....	306
6.4.36 SoftProofing	306
6.4.37 Stripping.....	306
6.4.38 Tiling	308
6.4.39 Trapping.....	309
6.5 Press Processes	309
6.5.1 ConventionalPrinting.....	309
6.5.2 DigitalPrinting.....	312
6.5.3 Varnishing	314
6.5.4 IDPrinting	314
6.6 Postpress Processes	314
6.6.1 AdhesiveBinding	314
6.6.2 BlockPreparation.....	315
6.6.3 BoxFolding	315
6.6.4 BoxPacking	315
6.6.5 Bundling.....	316
6.6.6 CaseMaking	317
6.6.7 CasingIn.....	317
6.6.8 ChannelBinding.....	317
6.6.9 CoilBinding.....	318
6.6.10 Collecting	318
6.6.11 CoverApplication.....	319
6.6.12 Creasing.....	319
6.6.13 Cutting.....	320
6.6.14 DieMaking	320
6.6.15 Dividing	321
6.6.16 Embossing	321
6.6.17 EndSheetGluing.....	321
6.6.18 Feeding	322

6.6.19 Folding	323
6.6.20 Gathering	323
6.6.21 Gluing.....	324
6.6.22 HeadBandApplication	324
6.6.23 HoleMaking	325
6.6.24 Inserting	325
6.6.25 Jacketing.....	325
6.6.26 Labeling	326
6.6.27 Laminating	326
6.6.28 LongitudinalRibbonOperations.....	326
6.6.29 Numbering	327
6.6.30 Palletizing.....	327
6.6.31 Perforating	327
6.6.32 PlasticCombBinding.....	328
6.6.33 PrintRolling.....	328
6.6.34 RingBinding.....	329
6.6.35 SaddleStitching.....	329
6.6.36 ShapeCutting	329
6.6.37 ShapeDefProduction.....	329
6.6.38 Shrinking	330
6.6.39 SideSewing	330
6.6.40 SpinePreparation	330
6.6.41 SpineTaping.....	331
6.6.42 Stacking	331
6.6.43 StaticBlocking	331
6.6.44 Stitching	332
6.6.45 Strapping.....	332
6.6.46 StripBinding.....	333
6.6.47 ThreadSealing.....	333
6.6.48 ThreadSewing.....	333
6.6.49 Trimming	334
6.6.50 WebInlineFinishing.....	334
6.6.51 WireCombBinding	335
6.6.52 Wrapping.....	335
6.7 Postpress Processes Structure	335
6.7.1 Block Production	335
6.7.1.1 Block Compiling.....	335
6.7.1.2 Block Joining	336
6.7.1.2.1 Single-Leaf Binding Methods	336
6.7.1.2.2 Loose-Leaf Binding Method	336
6.7.1.2.3 Mechanical Binding Methods	336

6.7.2 HoleMaking	336
6.7.3 Laminating	336
6.7.4 Numbering	336
6.7.5 Packaging Processes	337
6.7.6 Processes in Hardcover Book Production	337
6.7.7 Sheet Processes.....	338
6.7.8 Tip-on/in.....	338
6.7.9 Trimming.....	338
6.7.10 Web Processes.....	338
Chapter 7 Resources	339
7.1 Intent Resources	339
7.1.1 Span Subelements of an Intent Resource	340
7.1.1.1 Abstract Span Element.....	340
7.1.1.2 Span Elements.....	341
7.1.1.2.1 DurationSpan	341
7.1.1.2.2 EnumerationSpan	342
7.1.1.2.3 IntegerSpan	343
7.1.1.2.4 NameSpan	343
7.1.1.2.4.1 Specifying New Values in a NameSpan Subelement	343
7.1.1.2.5 NumberSpan	343
7.1.1.2.6 OptionSpan	344
7.1.1.2.7 ShapeSpan	344
7.1.1.2.8 StringSpan	345
7.1.1.2.9 TimeSpan	345
7.1.1.2.10 XYPairSpan	345
7.1.2 ArtDeliveryIntent	346
7.1.2.1 ArtDelivery	349
7.1.3 BindingIntent	351
7.1.3.1 BindList	354
7.1.3.2 BindItem	355
7.1.3.3 AdhesiveBinding	356
7.1.3.4 BookCase	356
7.1.3.5 ChannelBinding	356
7.1.3.6 CoilBinding.....	356
7.1.3.7 EdgeGluing	357
7.1.3.8 HardcoverBinding.....	357
7.1.3.9 PlasticCombBinding	359
7.1.3.10 RingBinding.....	360
7.1.3.11 SaddleStitching	361
7.1.3.12 SideSewing	361
7.1.3.13 SideStitching.....	361
7.1.3.14 SoftCoverBinding	361
7.1.3.15 StripBinding.....	362
7.1.3.16 Tabs.....	362

7.1.3.17 Tape	363
7.1.3.18 ThreadSealing	363
7.1.3.19 ThreadSewing	364
7.1.3.20 WireCombBinding	364
7.1.4 ColorIntent	364
7.1.4.1 ColorsUsed	367
7.1.5 DeliveryIntent	367
7.1.5.1 DropIntent	371
7.1.5.2 DropItemIntent	373
7.1.5.3 Pricing	373
7.1.5.4 Payment	373
7.1.5.5 CreditCard	373
7.1.6 EmbossingIntent	374
7.1.6.1 EmbossingItem	375
7.1.7 FoldingIntent	376
7.1.8 HoleMakingIntent	376
7.1.9 InsertingIntent	377
7.1.9.1 InsertList	378
7.1.9.2 Insert	378
7.1.10 LaminatingIntent	379
7.1.11 LayoutIntent	380
7.1.12 MediaIntent	383
7.1.13 NumberingIntent	389
7.1.13.1 NumberItem	389
7.1.14 PackingIntent	390
7.1.15 ProductionIntent	391
7.1.16 ProofingIntent	392
7.1.16.1 ProofItem	392
7.1.17 PublishingIntent	394
7.1.18 ScreeningIntent	395
7.1.19 ShapeCuttingIntent	395
7.1.19.1 ShapeCut	396
7.1.20 SizeIntent	396
7.2 Process Resources	396
7.2.1 Process Resource Template	396
7.2.2 Address	398
7.2.3 AdhesiveBindingParams	399
7.2.4 ApprovalParams	399
7.2.4.1 ApprovalPerson	399
7.2.5 ApprovalSuccess	399
7.2.5.1 ApprovalDetails	400
7.2.6 Assembly	400

7.2.6.1 AssemblySection	402
7.2.6.2 PageAssignedList.....	403
7.2.7 AssetListCreationParams	404
7.2.8 AutomatedOverPrintParams	405
7.2.9 BarcodeCompParams.....	405
7.2.10 BarcodeReproParams	406
7.2.11 BendingParams	407
7.2.12 BinderySignature	407
7.2.12.1 SignatureCell	413
7.2.13 BlockPreparationParams	415
7.2.14 BoxFoldingParams.....	416
7.2.14.1 BoxApplication.....	418
7.2.14.2 BoxFoldAction.....	418
7.2.15 BoxPackingParams.....	422
7.2.16 BufferParams	424
7.2.17 Bundle.....	424
7.2.17.1 BundleItem.....	425
7.2.18 BundlingParams.....	426
7.2.19 ByteMap.....	427
7.2.19.1 Band	429
7.2.19.2 PixelColorant	429
7.2.20 CaseMakingParams.....	429
7.2.21 CasingInParams	431
7.2.22 ChannelBindingParams	431
7.2.23 CIELABMeasuringField.....	432
7.2.24 CoilBindingParams	433
7.2.25 CollectingParams.....	434
7.2.26 Color	435
7.2.26.1 DeviceNColor	439
7.2.26.2 PrintConditionColor.....	440
7.2.27 ColorantAlias.....	444
7.2.28 ColorantControl.....	445
7.2.28.1 ColorantConvertProcess	448
7.2.28.2 ColorantOrder	448
7.2.28.3 ColorantParams.....	448
7.2.28.4 DeviceColorantOrder.....	448
7.2.28.5 ColorSpaceSubstitute.....	448
7.2.29 ColorControlStrip	449
7.2.30 ColorCorrectionParams	450
7.2.30.1 ColorCorrectionOp	452
7.2.31 ColorMeasurementConditions	453
7.2.32 ColorPool	454

7.2.33 ColorSpaceConversionOp	455
7.2.34 ColorSpaceConversionParams.....	462
7.2.35 ComChannel.....	464
7.2.36 Company.....	466
7.2.37 Component	466
7.2.38 Contact.....	472
7.2.39 ContactCopyParams.....	473
7.2.40 ContentList.....	474
7.2.40.1 ContentData	474
7.2.40.2 ContentMetadata.....	476
7.2.41 ConventionalPrintingParams	479
7.2.42 CostCenter.....	482
7.2.43 CoverApplicationParams	482
7.2.43.1 Score	482
7.2.44 CreasingParams	483
7.2.44.1 Crease.....	483
7.2.45 CustomerInfo	484
7.2.45.1 CustomerMessage.....	485
7.2.46 CutBlock.....	486
7.2.47 CutMark	487
7.2.48 CuttingParams	488
7.2.48.1 Cut.....	489
7.2.49 CylinderLayout.....	489
7.2.49.1 CylinderPosition	490
7.2.50 CylinderLayoutPreparationParams	493
7.2.51 DBMergeParams	493
7.2.52 DBRules.....	494
7.2.53 DBSchema.....	494
7.2.54 DBSelection	495
7.2.55 DeliveryParams.....	495
7.2.55.1 Drop	496
7.2.55.2 DropItem.....	497
7.2.56 DensityMeasuringField	498
7.2.57 DevelopingParams.....	499
7.2.58 Device	499
7.2.58.1 IconList	501
7.2.58.2 Icon	502
7.2.58.3 Module	502
7.2.59 DeviceMark	503
7.2.60 DeviceNSpace	504
7.2.61 DieLayout.....	505
7.2.61.1 RuleLength.....	506

7.2.61.2 Station	506
7.2.62 DieLayoutProductionParams	506
7.2.62.1 ConvertingConfig	507
7.2.62.2 RepeatDesc	507
7.2.63 DigitalDeliveryParams.....	512
7.2.64 DigitalMedia	513
7.2.65 DigitalPrintingParams	514
7.2.65.1 Coordinate systems in DigitalPrinting	514
7.2.66 Disjointing	518
7.2.67 Disposition	519
7.2.68 DividingParams.....	520
7.2.69 ElementColorParams.....	520
7.2.70 EmbossingParams.....	521
7.2.70.1 Emboss.....	522
7.2.71 Employee	523
7.2.72 EndSheetGluingParams	523
7.2.72.1 EndSheet	524
7.2.73 ExposedMedia	525
7.2.74 ExternalImpositionTemplate	527
7.2.75 FeedingParams.....	527
7.2.75.1 Feeder.....	527
7.2.75.2 FeederQualityParams.....	529
7.2.75.3 CollatingItem	529
7.2.76 FileSpec.....	531
7.2.76.1 Container.....	537
7.2.76.2 FileAlias.....	537
7.2.77 FitPolicy	538
7.2.78 Fold	539
7.2.79 FoldingParams.....	540
7.2.80 FontParams	544
7.2.81 FontPolicy	544
7.2.82 FormatConversionParams	545
7.2.82.1 TIFFFormatParams	546
7.2.82.2 TIFFtag	547
7.2.82.3 TIFFEmbeddedFile.....	548
7.2.83 GatheringParams.....	548
7.2.84 GeneralID.....	549
7.2.85 GlueApplication.....	549
7.2.86 GlueLine.....	550
7.2.87 GluingParams	551
7.2.87.1 Glue.....	552
7.2.88 HeadBandApplicationParams	552

7.2.89 Hole.....	553
7.2.90 HoleLine.....	553
7.2.91 HoleList.....	554
7.2.92 HoleMakingParams.....	555
7.2.93 IdentificationField	557
7.2.93.1 BarcodeDetails.....	561
7.2.93.2 ExtraValues.....	562
7.2.94 IDPrintingParams.....	564
7.2.95 ImageCompressionParams	564
7.2.95.1 ImageCompression	565
7.2.95.2 CCITTFaxParams	567
7.2.95.3 DCTParams.....	568
7.2.95.4 FlateParams.....	569
7.2.95.5 JBIG2Params	570
7.2.95.6 JPEG2000Params.....	570
7.2.95.7 LZWParams	571
7.2.96 ImageReplacementParams	571
7.2.97 ImageSetterParams.....	573
7.2.98 Ink	575
7.2.99 InkZoneCalculationParams.....	576
7.2.100 InkZoneProfile.....	577
7.2.101 InsertingParams.....	578
7.2.102 InsertSheet.....	580
7.2.103 InterpretedPDLDData	584
7.2.104 InterpretingParams	584
7.2.104.1 PDFInterpretingParams	586
7.2.104.2 OCGControl.....	588
7.2.104.3 More about PDFInterpretingParams.....	588
7.2.104.3.1 PDF Optional Content Groups	588
7.2.105 JacketingParams	589
7.2.106 JobField	590
7.2.107 LabelingParams.....	590
7.2.108 LaminatingParams	591
7.2.109 Layout	592
7.2.109.1 LayerList.....	596
7.2.109.2 LayerDetails.....	596
7.2.109.3 LogicalStackParams.....	596
7.2.109.4 Stack.....	596
7.2.109.5 PageCondition.....	597
7.2.109.6 SheetCondition.....	599
7.2.109.7 PlacedObject	600
7.2.109.7.1 Abstract PlacedObject	600
7.2.109.8 ContentObject	602

7.2.109.9 MarkObject	603
7.2.109.10 MarkActivation	605
7.2.109.10.1	Dynamic Marks 605
7.2.109.11 DynamicField.....	606
7.2.109.12 More about Layout.....	607
7.2.109.12.1 Migrating from a Pre-JDF 1.3 Layout to a Partitioned Layout	607
7.2.109.12.1.1 Partition Key restrictions:	607
7.2.109.12.1.2 Position of PlacedObject Elements in Layout.....	608
7.2.109.12.2 CTM Definitions	609
7.2.109.12.3 Finding the Trim Box of an Object	609
7.2.109.12.4 Using Ord to Reference Elements in RunList Resources	609
7.2.109.12.5 Using Expressions in the OrdExpression Attribute	611
7.2.109.12.5.1 Structure of Signature Subelement (Deprecated).....	612
7.2.110 LayoutElement	612
7.2.110.1 Dependencies	615
7.2.111 LayoutElementProductionParams	615
7.2.111.1 LayoutElementPart	617
7.2.111.2 BarcodeProductionParams.....	617
7.2.111.3 PositionObj	618
7.2.112 LayoutPreparationParams	623
7.2.112.1 PageCell	632
7.2.112.2 ImageShift.....	633
7.2.113 LayoutShift	636
7.2.113.1 ShiftPoint	636
7.2.114 LongitudinalRibbonOperationParams	638
7.2.115 ManualLaborParams.....	638
7.2.116 Media	638
7.2.116.1 MediaLayers	649
7.2.116.2 TabDimensions	649
7.2.116.3 More about Media.....	652
7.2.116.3.1 Inside Loss and Outside Gain	652
7.2.116.3.2 Corrugated Media:	653
7.2.116.3.3 Self adhesive Media	654
7.2.116.3.4 Flexo Plate Media	654
7.2.116.3.5 Flexo Sleeve Media	655
7.2.117 MediaSource.....	655
7.2.118 MiscConsumable	655
7.2.119 MISDetails.....	656
7.2.120 NodeInfo	657
7.2.121 NumberingParams.....	660
7.2.121.1 NumberingParam	660
7.2.122 ObjectResolution.....	660
7.2.123 OrderingParams.....	661
7.2.124 PackingParams.....	662
7.2.125 PageAssignParams	662

7.2.126 PageList	662
7.2.126.1 PageData	664
7.2.126.2 PageElement	666
7.2.127 Pallet	667
7.2.128 PalletizingParams	668
7.2.129 PDFToPSConversionParams	668
7.2.130 PDLCreationParams	671
7.2.131 PDLResourceAlias	672
7.2.132 PerforatingParams	672
7.2.132.1 Perforate	673
7.2.133 Person	673
7.2.134 PlaceholderResource	674
7.2.135 PlasticCombBindingParams	674
7.2.136 PlateCopyParams	675
7.2.137 PreflightAnalysis	675
7.2.138 PreflightInventory	675
7.2.139 PreflightParams	676
7.2.139.1 PreflightAction	676
7.2.139.2 BasicPreflightTest	677
7.2.139.3 PreflightArgument	677
7.2.139.4 BoxArgument	678
7.2.139.5 BoxToBoxDifference	679
7.2.140 PreflightProfile	679
7.2.141 PreflightReport	679
7.2.141.1 PRItem	680
7.2.141.2 PRError	681
7.2.141.3 PRGroup	681
7.2.141.4 Abstract PRGroupOccurrenceBase	683
7.2.141.5 ArgumentValue	683
7.2.141.6 PRGroupOccurrence	684
7.2.141.7 StringListValue	684
7.2.141.8 PROccurrence	684
7.2.142 PreflightReportRulePool	684
7.2.142.1 PRRule	685
7.2.142.2 PRRuleAttr	685
7.2.143 Preview	687
7.2.144 PreviewGenerationParams	689
7.2.145 PrintCondition	691
7.2.146 PrintRollingParams	692
7.2.147 ProductionPath	693
7.2.147.1 FolderSuperstructureWebPath	694
7.2.147.2 PostPressComponentPath	694
7.2.147.3 PrintingUnitWebPath	694

7.2.148 ProofingParams	695
7.2.149 PSToPDFConversionParams	695
7.2.149.1 AdvancedParams	697
7.2.149.2 PDFXParams	699
7.2.149.3 ThinPDFParams	701
7.2.150 QualityControlParams	701
7.2.150.1 BindingQualityParams	701
7.2.151 QualityControlResult	702
7.2.151.1 QualityMeasurement	702
7.2.151.2 BindingQualityMeasurement	702
7.2.152 RasterReadingParams	703
7.2.153 RefAnchor	704
7.2.154 RegisterMark	704
7.2.155 RegisterRibbon	705
7.2.156 RenderingParams	706
7.2.157 ResourceDefinitionParams	707
7.2.157.1 ResourceParam	708
7.2.158 RingBindingParams	708
7.2.159 RollStand	710
7.2.160 RunList	710
7.2.160.1 DynamicInput	717
7.2.160.2 MetadataMap	717
7.2.160.3 Expr	720
7.2.161 SaddleStitchingParams	725
7.2.162 ScanParams	725
7.2.163 ScavengerArea	726
7.2.164 ScreeningParams	726
7.2.164.1 ScreenSelector	727
7.2.165 SeparationControlParams	729
7.2.166 SeparationSpec	730
7.2.167 Shape	730
7.2.168 ShapeCuttingParams	731
7.2.169 ShapeDef	732
7.2.170 ShapeDefProductionParams	733
7.2.170.1 ObjectModel	734
7.2.170.2 ShapeTemplate	734
7.2.171 Sheet	736
7.2.172 ShrinkingParams	736
7.2.173 SideSewingParams	736
7.2.174 SpinePreparationParams	737
7.2.175 SpineTapingParams	738
7.2.176 StackingParams	739

7.2.177 StaticBlockingParams	743
7.2.178 StitchingParams	743
7.2.179 Strap	746
7.2.180 StrappingParams	747
7.2.181 StripBindingParams	748
7.2.182 StrippingParams	748
7.2.182.1 Position	752
7.2.182.2 StripCellParams	753
7.2.182.3 StripMark	755
7.2.183 Surface.....	758
7.2.184 ThreadSealingParams	758
7.2.185 ThreadSewingParams	759
7.2.186 Tile	761
7.2.187 Tool.....	762
7.2.188 TransferCurve	763
7.2.189 TransferCurvePool.....	763
7.2.189.1 TransferCurveSet.....	763
7.2.190 TransferFunctionControl	764
7.2.191 TrappingDetails.....	765
7.2.191.1 TrappingOrder	765
7.2.192 TrappingParams	766
7.2.192.1 ColorantZoneDetails.....	769
7.2.193 TrapRegion	769
7.2.194 TrimmingParams.....	770
7.2.195 UsageCounter.....	771
7.2.196 VarnishingParams.....	772
7.2.197 VerificationParams.....	773
7.2.198 WebInlineFinishingParams	774
7.2.198.1 FolderProduction	774
7.2.199 WireCombBindingParams.....	775
7.2.200 WrappingParams	775
7.3 Device Capability Definitions	776
7.3.1 DeviceCap	776
7.3.2 ActionPool.....	780
7.3.2.1 Action.....	780
7.3.3 DevCapPool.....	781
7.3.4 ModulePool	781
7.3.4.1 ModuleCap.....	781
7.3.5 DevCaps	782
7.3.5.1 Loc	785
7.3.6 DevCap	785

7.3.7 State	787
7.3.7.1 Abstract State Element.....	789
7.3.7.2 State Elements.....	791
7.3.7.2.1 BooleanState	792
7.3.7.2.1.1 ValueLoc	792
7.3.7.2.2 DateTimeState	793
7.3.7.2.3 DurationState	793
7.3.7.2.4 EnumerationState	793
7.3.7.2.5 IntegerState	794
7.3.7.2.6 MatrixState	796
7.3.7.2.6.1 Value	796
7.3.7.2.7 NameState	797
7.3.7.2.8 NumberState	797
7.3.7.2.9 PDFPathState	798
7.3.7.2.9.1 Value	799
7.3.7.2.10 RectangleState	799
7.3.7.2.11 ShapeState	800
7.3.7.2.12 StringState	800
7.3.7.2.12.1 Value	801
7.3.7.2.13 XYPairState	801
7.3.8 DisplayGroupPool	802
7.3.8.1 DisplayGroup.....	803
7.3.9 FeaturePool	803
7.3.10 MacroPool	804
7.3.10.1 macro	804
7.3.10.2 choice	805
7.3.10.3 otherwise	805
7.3.10.4 when	805
7.3.10.5 set	805
7.3.10.6 FeatureAttribute	806
7.3.10.7 call.....	806
7.3.11 Performance	806
7.3.12 TestPool	807
7.3.12.1 Test.....	807
7.3.13 Term	807
7.3.13.1 and.....	809
7.3.13.2 or	809
7.3.13.3 xor	810
7.3.13.4 not	810
7.3.13.5 TestRef.....	810
7.3.13.6 Evaluation	810
7.3.13.6.1 Abstract Evaluation	811
7.3.13.6.2 Evaluation Elements	812
7.3.13.6.2.1 BooleanEvaluation	813
7.3.13.6.2.2 DateTimeEvaluation	813
7.3.13.6.2.3 DurationEvaluation	814
7.3.13.6.2.4 EnumerationEvaluation.....	814

7.3.13.6.2.5 IntegerEvaluation	814
7.3.13.6.2.6 IsPresentEvaluation.....	814
7.3.13.6.2.7 MatrixEvaluation	814
7.3.13.6.2.7.1 Value	815
7.3.13.6.2.8 NameEvaluation.....	815
7.3.13.6.2.9 NumberEvaluation	815
7.3.13.6.2.10 PDFPathEvaluation.....	816
7.3.13.6.2.10.1 Value	816
7.3.13.6.2.11 RectangleEvaluation	816
7.3.13.6.2.12 ShapeEvaluation.....	816
7.3.13.6.2.13 StringEvaluation.....	817
7.3.13.6.2.13.1 Value	817
7.3.13.6.2.14 XYPairEvaluation	817
7.3.14 Examples of Device Capabilities.....	817
7.3.14.1 Device Description of a Scanner.....	817
7.3.14.2 Device Description of a Scanner #2.....	820
7.4 Concept of the Preflight Process	825
7.4.1 Object Classes.....	826
7.4.1.1 Properties Implemented by each Class of Object.....	826
7.4.1.2 Checking for the Presence of a Property.....	827
7.4.1.3 Basic tests on set of objects	829
7.4.2 Properties.....	829
7.4.2.1 Annotation Properties	830
7.4.2.2 Box Properties.....	830
7.4.2.3 Class Properties.....	831
7.4.2.4 Colorant Properties	832
7.4.2.5 Document Properties.....	832
7.4.2.6 Fill Properties.....	836
7.4.2.7 Font Properties.....	836
7.4.2.8 Graphic Properties	838
7.4.2.9 Image Properties	839
7.4.2.10 Logical Properties	841
7.4.2.11 PageBox Properties.....	841
7.4.2.12 Pages Properties	842
7.4.2.13 PDLObject Properties	843
7.4.2.14 Reference Properties	843
7.4.2.15 Shading Properties	844
7.4.2.16 Stroke Properties.....	844
7.4.2.17 Text Properties	845
7.4.2.18 Vector Properties	846
Chapter 8 Building a System Around JDF	847
8.1 Implementation Considerations and Guidelines	847
8.2 JDF and JMF Interchange Protocol	847
8.2.1 File-Based Protocol (JDF + JMF)	847
8.2.1.1 JMF Transport Using a Hot Folder.....	847

8.2.1.2 JMF Transport Using the File Protocol	847
8.2.2 HTTP-Based Protocol (JDF + JMF).....	848
8.2.2.1 Protocol Implementation Details	848
8.2.3 HTTPS-Based Protocol – SSL with two-way authentication.....	848
8.2.3.1 Purpose.....	848
8.2.3.2 Certificates	848
8.2.3.2.1 Verification of Certificates	849
8.2.3.3 Exchange of Certificates	849
8.2.3.4 Standards.....	851
8.2.3.5 Implementation	851
8.2.3.5.1 Discovery Messages	851
8.2.3.5.2 Example of Sun Keytool Usage	851
8.3 JDF Packaging	851
8.3.1 MIME Basics	852
8.3.2 MIME Types and File Extensions	852
8.3.2.1 MIME Headers	852
8.3.2.1.1 Content-Type Header	852
8.3.2.1.2 Content-ID Header	852
8.3.2.1.3 Content-Transfer-Encoding	852
8.3.2.1.4 Content-Disposition Header	853
8.3.2.2 CID URL Scheme	853
8.3.2.3 Ordering of Body Parts in MIME Multipart/Related.....	854
8.4 MIS Requirements	855
8.5 Interoperability Conformance Specifications	855
Appendix A Encoding.....	857
A.1 Notes About Encoding	857
A.1.1 List, Range and Range List Data Types.....	857
A.1.2 Whitespace.....	857
A.1.3 Infinity Limits.....	857
A.2 Simple Types — Attribute Values	857
A.2.1 boolean.....	857
A.2.2 CMYKColor.....	857
A.2.3 date.....	858
A.2.4 dateTime.....	858
A.2.5 DateTimeRange	858
A.2.6 DateTimeRangeList.....	858
A.2.7 double.....	859
A.2.8 DoubleList.....	859
A.2.9 DoubleRange.....	859
A.2.10 DoubleRangeList.....	859
A.2.11 duration.....	860

A.2.12 DurationRange	860
A.2.13 DurationRangeList.....	860
A.2.14 gYearMonth	860
A.2.15 hexBinary.....	860
A.2.16 ID	861
A.2.17 IDREF	861
A.2.18 IDREFS	861
A.2.19 integer.....	861
A.2.20 IntegerList.....	861
A.2.21 IntegerRange.....	862
A.2.22 IntegerRangeList	862
A.2.23 LabColor	862
A.2.24 language.....	862
A.2.25 languages	863
A.2.26 matrix.....	863
A.2.27 NameRange	863
A.2.28 NameRangeList.....	863
A.2.29 NMTOKEN.....	864
A.2.30 NMTOKENS	864
A.2.31 PDFPath	864
A.2.32 rectangle.....	864
A.2.33 RectangleRange.....	865
A.2.34 RectangleRangeList	865
A.2.35 regExp	865
A.2.36 shape.....	865
A.2.37 ShapeRange.....	866
A.2.38 ShapeRangeList.....	866
A.2.39 sRGBColor	866
A.2.40 string.....	866
A.2.41 TimeRange	867
A.2.42 TransferFunction	867
A.2.43 URI	867
A.2.44 URL	867
A.2.45 XPath.....	868
A.2.46 XYPair	868
A.2.47 XYPairRange.....	869
A.2.48 XYPairRangeList	869
A.3 Enumerations and Lists	869
A.3.1 enumeration.....	869
A.3.2 enumerations.....	869

A.3.3 Defined JDF enumeration Data Types 870

A.3.3.1 Anchor..... 870

A.3.3.2 JDFJMFVersion..... 870

A.3.3.3 NamedColor..... 870

A.3.3.4 Orientation 871

A.3.3.5 WorkStyle 871

A.3.4 XYRelation..... 872

A.4 JDF File Formats 872

A.4.1 PNG Image Format 872

Appendix B Schema 873

B.1 Using xsi:type 873

B.1.1 Using xsi:type with JDF Nodes..... 873

B.1.2 Using xsi:type with JMF Messages 874

Appendix C Supported String and NMTOKEN values. 875

C.1 StatusDetails Supported Strings 875

C.2 ModuleType Supported Strings 879

C.3 NotificationDetails 883

C.3.1 Abstract NotificationDetails 883

C.3.2 NotificationDetails..... 883

C.3.2.1 Barcode 884

C.3.2.2 FCNKey 884

C.3.2.3 SystemTimeSet 884

C.3.2.4 CounterReset 884

C.3.2.5 Error 884

C.3.2.5.1 ErrorData 885

C.3.2.6 Event 885

C.3.2.7 Milestone..... 885

C.4 MessageEvents and Milestone Values 886

C.5 Input Tray and Output Bin Names 889

**Appendix D Supported Error Codes in JMF and Notification Elements
891**

Appendix E Color Adjustment Attribute Description and Usage . . . 893

E.1 Adjustment Using Direct Attributes 893

E.2 Adjustment using ICC Profile Attributes 894

E.3 Adjustment using an ICC Abstract Profile Attribute 894

E.4 Adjustment using an ICC DeviceLink Profile Attribute 894

Appendix F North American and Japanese Media Weight Explained . . .	895
F.1 North American Media Weight	895
F.2 Japanese Media Weight	896
Appendix G Media Sizes	899
Appendix H MimeType and MimeVersion Attributes	903
Appendix I Generating strings with Format and Template	909
Appendix J Resolving RunList/@Directory and FileSpec/@URL URI references	913
J.1 Semantics of the RunList/@Directory Attribute	913
Appendix K References	915
Appendix L JDF/CIP4 Hole Pattern Catalog	929
Appendix M FileSpec Attributes and Container Subelement	939
M.1 Examples of Attribute Values of FileSpec	939
M.2 Corresponding XML examples	940
M.3 Additional examples showing Partitioning of FileSpec	943
M.4 Example of an Intent Job Ticket with a doubly nested ZIP packaging file . .	949
M.5 AppOS and OSVersion Attributes	950
Appendix N Examples	953
N.1 Brief Example	953
N.1.1 Before and After Processing	953
N.2 Product JDF	955
N.3 Spawning and Merging	956
N.4 RunList	963
N.5 Messages	965
N.5.1 Simple KnownMessages	965
N.5.2 Simple persistent channel	966
N.6 Stripping	967
N.7 DigitalDelivery Examples	974
N.8 Automated Imposition	980

Appendix O New, Deprecated & Modified Items	987
O.1 Compatibility Warnings	987
O.2 Changed Items	987
Appendix P Deprecated Elements, JMF Messages, Processes and Resources	1005
P.1 Deprecated Structures of JDF Nodes and Jobs	1005
P.1.1 ResourceUpdate.....	1005
P.1.2 StatusPool and PartStatus	1005
P.1.2.1 PartStatus	1006
P.2 JMF Messaging Elements	1006
P.2.1 Signal.....	1006
P.2.1.1 Trigger	1006
P.2.1.2 Added.....	1007
P.2.1.3 ChangedAttribute.....	1007
P.2.1.4 Removed.....	1007
P.2.2 NodeInfo	1007
P.2.2.1 NodeInfo Query	1008
P.2.2.2 NodeInfo Command	1008
P.2.3 KnownJDFServices	1010
P.2.3.1 JDFService.....	1010
P.2.4 QueueEntryStatus	1011
P.2.4.1 QueueEntryDefList.....	1011
P.3 Deprecated Processes	1012
P.3.1 Packing.....	1012
P.3.2 FilmToPlateCopying	1012
P.3.3 PreflightAnalysis	1012
P.3.3.1 PreflightDetail	1013
P.3.3.2 PreflightInstance	1013
P.3.3.3 PreflightInstanceDetail	1014
P.3.4 PreflightInventory.....	1014
P.3.5 PreflightProfile	1015
P.3.5.1 PreflightConstraint.....	1016
P.3.6 Proofing	1016
P.3.7 SoftProofing.....	1017
P.3.8 IDPrinting.....	1018
P.3.9 AdhesiveBinding.....	1019
P.3.10 Dividing.....	1020
P.3.11 LongitudinalRibbonOperations	1020
P.3.12 SaddleStitching.....	1020
P.3.13 SideSewing.....	1021

P.4 Deprecated Resources	1021
P.4.1 BindingIntent Deprecated Subelements	1021
P.4.1.1 AdhesiveBinding	1022
P.4.1.2 BookCase	1022
P.4.2 DeliveryIntent Deprecated Subelements	1022
P.4.2.1 Pricing	1023
P.4.2.2 Payment	1023
P.4.2.3 CreditCard	1023
P.4.3 SizeIntent	1024
P.4.4 AdhesiveBindingParams	1024
P.4.5 DividingParams	1026
P.4.6 IDPrintingParams	1026
P.4.6.1 Cover	1028
P.4.6.2 IDPFinishing	1029
P.4.6.3 IDPFolding	1030
P.4.6.4 IDPHoleMaking	1030
P.4.6.5 IDPLayout	1030
P.4.6.6 IDPStitching	1032
P.4.6.7 IDPTrimming	1034
P.4.6.8 ImageShift	1034
P.4.6.9 JobSheet	1034
P.4.7 Layout Deprecated Subelement	1037
P.4.7.1 Signature	1037
P.4.8 LongitudinalRibbonOperationParams	1037
P.4.8.1 LROperation	1038
P.4.8.2 LongFold	1038
P.4.8.3 LongGlue	1038
P.4.8.4 LongPerforate	1039
P.4.8.5 LongSlit	1039
P.4.9 MediaSource	1039
P.4.10 PackingParams	1039
P.4.11 PlateCopyParams	1041
P.4.12 ProofingParams	1041
P.4.13 SaddleStitchingParams	1042
P.4.14 Sheet	1043
P.4.15 SideSewingParams	1044
P.4.16 Surface	1046
Appendix Q List of Figures	1047
Appendix R List of Tables	1051
Appendix S List of Examples.	1073




JDF Preface and User Overview

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

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


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

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 fourth version of JDF (v.1.3) 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?

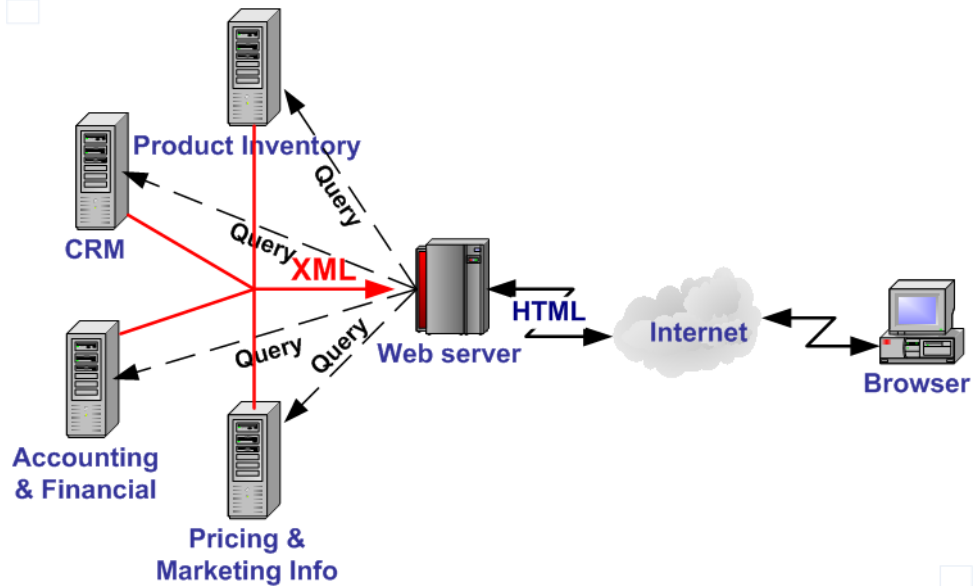
- A standard that can be applied to eliminate wasteful re-keying and redundancy of information; and
- A common computer language for printing and related industries as well as a platform for more effective communication.

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:

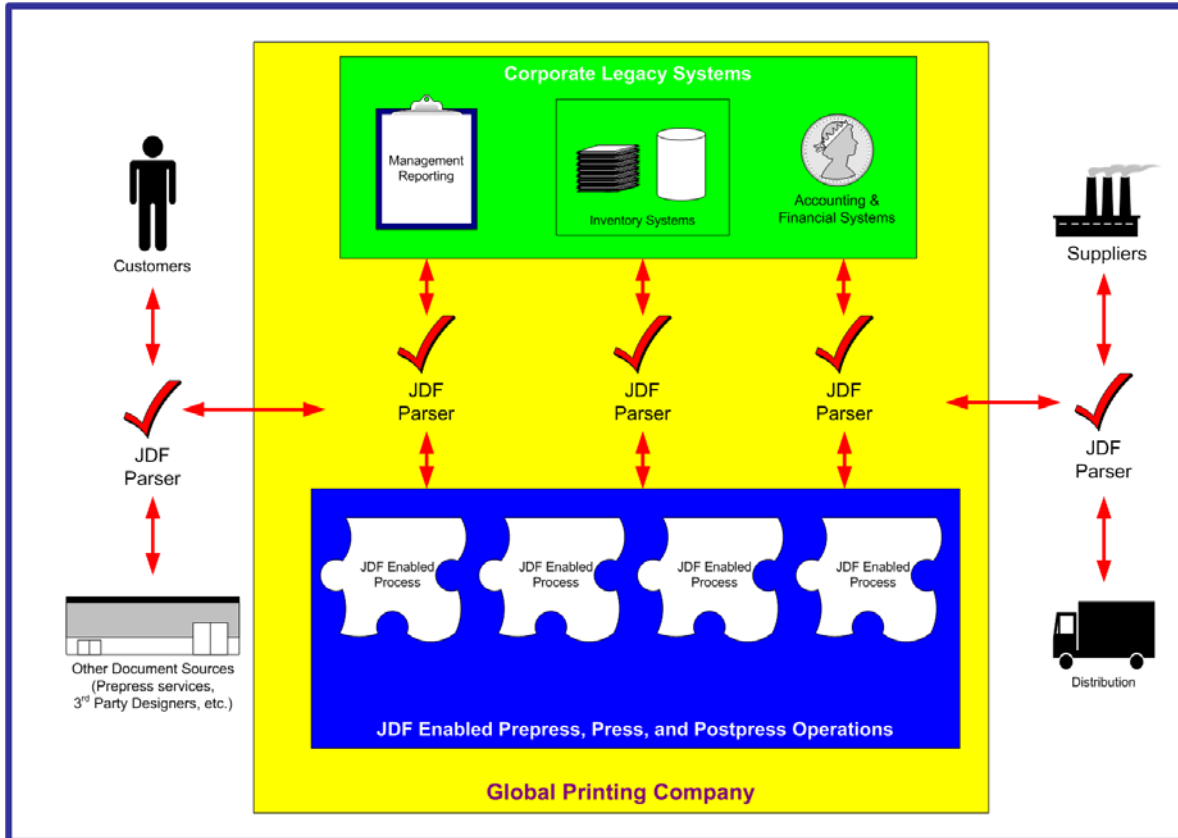
- 1 Acceptance checking of client job tickets;
- 2 Validation of JDF prior to or following transformation of data into and out of databases;
- 3 Ensuring that source job information is collected as a document is created (embedded in document layout software);
- 4 Determining if equipment reads and writes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



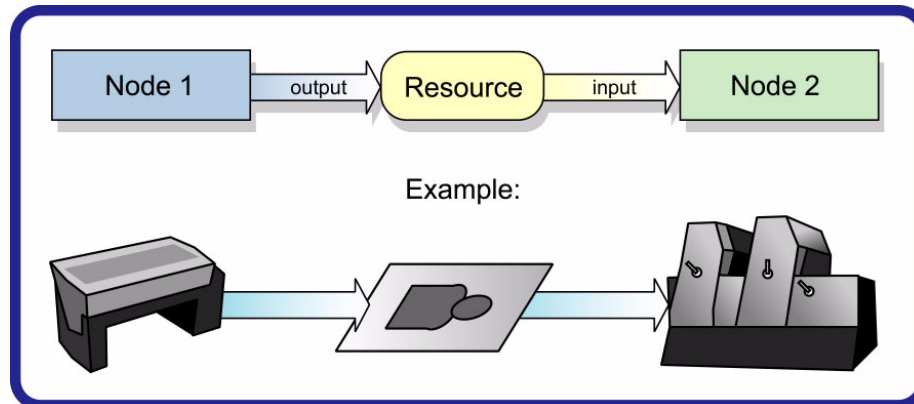
The JDF Concept. The JDF schema is quite complex and detailed—something best left to programmers, MIS personnel and XML experts. But the language and concepts behind JDF are quite simple and straightforward. The schema itself can be downloaded from the CIP4 Website, but is not part of this specification. 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 1.4

JDF 1.4 includes both some wholly new material, as well as many improvements and refinements to JDF 1.3. [A complete catalog of changes can be found in "New, Deprecated & Modified Items" on page 987.](#) You will also find [New in JDF 1.4](#), [Deprecated in JDF 1.4](#) and [Modified in JDF 1.4](#) flags throughout this document.

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

- **Content Creation:** See Section 6.4.20, "LayoutElementProduction" on page 297.
- **Reliable Signals:** See Section 5.4.2, "Signal and Acknowledge Handshaking" on page 183
- **Dynamic Marks:** See Section 7.2.182.3, "StripMark" on page 755 and Section 7.2.109, "Layout" on page 592.
- **Automated Layout and Variable Data:** See Section 6.4.17, "Imposition" on page 284, Section 7.2.109, "Layout" on page 592 and Section 7.2.109.11, "DynamicField" on page 606.

- **Structural Design in Packaging – Workflow Steps:** See Section 6.4.10, "DieDesign" on page 279, Section 6.4.11, "DieLayoutProduction" on page 280, Section 6.6.14, "DieMaking" on page 320 and Section 6.6.37, "ShapeDefProduction" on page 329.

For a complete list of modifications, see Section O, "New, Deprecated & Modified Items" on page 987

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 JDF Specification Release 1.4

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 915 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, BundleItem 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 **Ink**, **Layout** 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 141 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 1-1: XPath Expression

The XPath expression:

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

Means:

The value “Replacement Text for %s and %s go in here at %s on %s” of the *Format* Attribute of the DynamicField Subelement of the MarkObject Element of the **Layout** Resource Element.

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

1.3.3 Callouts

[New in JDF 1.2](#)

To help the reader familiar with earlier versions of JDF, this specification indicates additions, deprecations and clarifications using the callouts described in Table 1-2. Please note that not all changes are identified with modified callout flags. When modification occurs in multiple versions, only the most recent version is indicated. A few changes

have been made globally and are explained in the body of the document and only significant changes have been flagged with callouts, as determined by CIP4 Working Groups.

Table 1-2: Callouts

Example	Callout Meaning
New in JDF 1.x	New sections, Attributes/Elements and Attribute Values.
Deprecated in JDF 1.x	Deprecated sections, Attributes/Elements and Attribute Values. Usually there is a deprecation note describing the mechanism that replaces the deprecated item.
Modified in JDF 1.x	Changed syntax or semantics of sections or Attributes/Elements. Might include clarification as well. Frequently there is a modification note describing the change

1.3.3.1 Location of Callouts

[New in JDF 1.4](#)

A callout occurs after one of the following document elements.

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

1.3.4 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 11. 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 6)

Term	Definition
Abstract	Abstract is used as a modifier for Elements , Resources and ResourceLink Elements, e.g. Abstract Element , Abstract Resource , Abstract Physical Resource and Abstract ResourceLink .
Abstract Element	An Abstract Element is an abstract data type with Attributes and Elements that are inherited by subclass concrete Elements. For example, PlacedObject is an Abstract Element member of a Layout ; MarkObject and ContentObject Elements are concrete Elements of PlacedObject. that are potential members of Layout
Abstract Resource	An Abstract Resource is an abstract data type with Attributes and Elements that are inherited by all Resources. For example, Media , as a resource inherits all the Attributes and Elements of the Abstract Resource. Abstract Resource has subclasses, such as Abstract Physical Resource and Abstract Implementation Resource. See Resource .
Abstract ResourceLink	An Abstract ResourceLink is an abstract data type with Attributes and Elements that are inherited by all ResourceLink Elements. For example, MediaLink, as a ResourceLink inherits all the Attributes and Elements of the Abstract ResourceLink. Abstract ResourceLink has subclasses, such as Abstract PhysicalLink and Abstract ImplementationLink. See ResourceLink.
Acknowledge Message	An Acknowledge Message is a JMF Message that is delayed response to a Command Message or Query Message .
Agent	The component of a JDF-based workflow that writes JDF.
Attribute	An XML-based syntactic construct describing an unstructured characteristic of a JDF Node or Element .
Attribute Value	The value of an Attribute .
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 . Each Resource belongs to a Class : <i>Consumable, Handling, Implementation, Intent, Parameter, PlaceHolder, Quantity</i> . See Consumable Resource, etc. Each Notification Audit Element belongs to a Class : <i>Event, Information, Warning, Error, Fatal</i> .
Combined Process	A Process which is described by multiple simpler JDF Processes. See Process
Combined Process Node	A Node that represents a Combined Process , which is described by multiple simpler JDF Processes., See Combined Process, Node and "Common Node Types" on page 53.
Consumable Resource	A Consumable Resource is consumed during a Process. See Resource, Physical Resource and Section 3.8.5.4, Consumable Resource.
Command Message	.A Command Message is a JMF Message that requests its recipient to change its state.
Controller	The component of a JDF-based workflow that initiates Devices , routes JDF, and communicates status information.

Table 1-4: Glossary (Section 2 of 6)

Term	Definition
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 = "DefaultValue"</i> in this specification) from a JDF instance. See Section 1.4.2.1, Conformance Requirements for Support of Attributes and Attribute Values.
Default behavior	The phrase “ Default behavior: ” precedes a description of the default behavior for an Attribute or Span Element.
Default value is	The phrase “ Default value is: ” precedes the default value for an Attribute.
Default value is from	The phrase “ Default value is from: ” precedes a reference to a default value – usually an XPath.
Deprecated	Indicates that a JDF Element is being phased out of JDF usually in favor of newer JDF Element(s). It is RECOMMENDED that an Agent not include such a JDF Element in a JDF instance. Such an indicated JDF Element might be removed from a future version of the JDF specification. JDF Consumers SHOULD only support such JDF Elements for backward compatibility with previous versions of JDF. Deprecated items are flagged with Deprecated in JDF 1.X in this specification.
Device	The component of a JDF workflow part that interprets JDF and executes the instructions. If a Device controls a Machine , it does so in a proprietary manner.
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 <i>Types</i> Attribute and no child Nodes represents a Gray Box .
Handling Resource	A Handling Resource is used during a Process, but are not destroyed by that Process. See Resource , Physical Resource and Section 3.8.5.6, Handling Resource.
Implementation Resource	An Implementation Resource defines a Device or operator that executes a given Node. See Resource and Section 3.8.5.3, Implementation Resource.
Input Resource	A Resource that is an input to a Process . See Resource .
Instance Document	A document that is part of the output of a Job. This generally refers to personalized printing Jobs. Each of the individual documents produced from the same input template is referred to as an Instance Document. For example, in a credit card statement run, each statement is an Instance Document.

Table 1-4: Glossary (Section 3 of 6)

Term	Definition
Intent Resource	An Intent Resource defines the details of products to be produced without defining the process to produce them. See Resource and Section 3.8.5.2, Intent Resource.
JDF	Job Definition Format. The overall name of this specification. There is also a JDF Element, which is a top-level Element within JDF that encompasses a Node (see below.)
JDF Consumer	A Device , Controller or Agent that consumes JDF instances.
JDF Node	See Node .
JMF	Job Messaging Format. A communication format with multi-level capabilities. Structures information between MIS , Controllers and Devices . There is also the JMF Element, which is a top-level Element within JDF. See “JDF Messaging with the Job Messaging Format” on page 167.
JMF Message	A JMF Message is synonymous with Message . See Message
Job	A hierarchical tree structure comprised of Nodes . Describes the output that is desired by a customer.
Job Part	One or more Nodes which comprise the smallest level of control of interest to MIS.
Leaf	Both the recto and verso finished pages on one piece of paper with “leaves” being the plural usage.
Link	A pointer to information that is located elsewhere in a JDF document or that is located in another document.
Machine	The part of a device that does not know JDF and is controlled by a JDF Device in a proprietary manner.
Message	The XML element that Devices and Controller use to exchange queries, commands, responses, etc. among themselves using HTTP as the underlying protocol and JMF as the XML format. See JMF and Message Family .
Message Family	A Message Family is a set of Messages . The 6 Message Families are Query Message , Command Message , Registration Message , Response Message , Acknowledge Message and Signal Message .
MIS	Management Information 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. A Node is also called a JDF Node .
Operating Side	Operating Side is the side of a Machine , where the operator works. Operating Side is opposite to Gear Side .
Output Resource	A Resource that is an output from a Process . See Resource .
Parameter Resource	A Parameter Resource defines the details of Processes, as well as any non-physical computer data such as files used by a Process. See Resource and Section 3.8.5.1, Parameter Resource.
Partition	A Partition is a node of a Partitioned Resource structure. A leaf node Partition represents a single Resource . A non-leaf node Partition represents a set of Resources . Values of the <i>PartIDKeys</i> Attribute in the Partitioned Resource root specify the Attributes used to identify the individual Resources in the Partitioned Resource . Each Partition except the Partitioned Resource root has a Partition Key Attribute whose value identifies the Partition . (See Table 3-27, “Partitionable Resource Element,” on page 103.)
to Partition	The verb to Partition means to construct a Partitioned Resource from a set of Resources of the same Class. See Section 3.10.5.4, Partitioning of Resource Subelements
Partition Key	A Partition Key is an enumeration value of the <i>PartIDKeys</i> Attribute and a Partition Key is an Attribute that with can identify a Partition or can reference a Partition from within a Part Element. See Section 3.10.6, PartIDKeys Attribute and Partition Keys

Table 1-4: Glossary (Section 4 of 6)

Term	Definition
Partitionable Resource	A Resource that can become a Partitioned Resource .
Partitioned Resource	A Partitioned Resource is a structured Resource that describes a set of Resources , all of the same Class and representing multiple physical or logical entities (e.g. a set of ExposedMedia that represent multiple separated plates).
PDL	Page Description Language. A generic term for any language that describes pages which might be printed. Examples are PDF®, PostScript® or PCL®.
PlaceHolder Resource	A PlaceHolder Resource defines process linking and process ordering when the exact nature of interchange resources is still unknown. See Resource and Section 3.8.5.8, PlaceHolder Resource .
Physical Resource	A Physical Resource is a Resource whose <i>Class</i> is <i>Consumable</i> , <i>Quantity</i> or <i>Handling</i> is considered a Physical Resource . See Resource and Section 3.8.5.7, Physical Resource .
PhysicalLink	A ResourceLink element that links to a Physical Resource , i.e. a ConsumableLink , QuantityLink or HandlingLink . See ResourceLink , Physical Resource and Section 3.9.10, PhysicalLink
Process Group	A group of Processes . See Process and Process Group Node
Process Group Node	A Node that contains multiple child Nodes . See Process Group , Node and "Common Node Types" on page 53.
Process Node	A Node that describes an individual Process . See Node and see "Common Node Types" on page 53.
Process	An individual step in the workflow.
Product Intent	Describes the end result that a customer is requesting. See Product Intent Node .
Product Intent Node	A Node that describes intent rather than specifying the Process . See Node , Product Intent and see "Common Node Types" on page 53.
Quantity Resource	A Quantity Resource has been created by a Process from either a Consumable Resource or an earlier Quantity Resource . See Resource , Physical Resource and Section 3.8.5.5, Quantity Resource .
Query Message	A Query Message is a JMF Message that requests its recipient to provide information, but not change its state.
Queue	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).
Registration Message	A Registration Message is a JMF Message that requests its recipient to send a Command Message to some other recipient.
Resource	A physical or conceptual entity that is modified or used by a Node . Examples include paper, images or Process parameters.
ResourceLink	An element that links to a Resource . See Resource and Section 3.9.2, ResourceLink
Response Message	A Response Message is a JMF Message that functions as a response to a Command Message or Query Message .

Table 1-4: Glossary (Section 5 of 6)

Term	Definition
Roll	A Roll is media that is mainly used in connection with Web Printing. In British English the name “reel” for “roll” is in widespread use. Roll is used as synonym of reel. Also, see the term Web in this glossary.
Root Node	The top most JDF Node . See Node .
Sheet	The printer’s Roll of paper or paper cut for press size, with “recto” and “verso” forms for identification of orientation through the press (facing up versus facing down at the feeder or off the Roll.). The term “cut sheet” refers to an individual Sheet, typically in a phrase, such as “separately cut sheets of an opaque material”. The term “Sheet-Fed” is used to describe a press that consumes cut Sheets, typically in the phrase “Sheet-Fed Press”.
	A is a JMF Message that is sent asynchronously when some event occurs.
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 .
Subelement	A child Element of some other Element
Subnode	A Node that is a child of some other Node . See Node .
Support	A JDF Consumer supports a JDF syntactic construct (Processes, Resources, Elements, Attributes and Attribute Values) if the JDF Consumer performs the action defined in this specification for the JDF construct when consuming a JDF instance that includes the JDF syntactic construct. If the Machine that a Device is representing supports a feature which is represented by a JDF construct, then the Device SHOULD support that JDF syntactic construct.
Surface	A single side of either a Sheet or a Signature
Tag	A syntactic construct that marks the start or end of an Element .
Value format is:	The phrase “ Value format is: ” precedes description of the format.
Value format is from:	The phrase “ Value format is from: ” precedes a reference to some section which describes the format.
Values are	The phrase “ Values are: ” precedes the complete list of values for an Attribute whose values are enumeration or enumerations.
Values are from	The phrase “ Values are from: ” precedes a reference to the values. The reference may be an XPath to a value or a reference to a table of Attribute Values. The referenced values are as complete as the reference says they are.
Values include	The phrase “ Values include: ” precedes a list of values for an Attribute whose values are NMTOKEN, NMTOKENS, string, NameSpan or StringSpan. The list does not include potential vendor or customer extensions.
Values include those from	The phrase “ Values include those from: ” precedes a reference to the values. The reference may be an XPath to a value or a table of Attribute Values. The referenced values do not include potential vendor or customer extensions.

Table 1-4: Glossary (Section 6 of 6)

Term	Definition
Web	A Web is media that comes from a Roll and is mainly used in connection with Web Printing. This specification uses the word “Web-Fed” instead of “roll-fed” It uses the phrase “Web Printing” and “Web Press” to describe printing presses that consumes media that comes from a Roll.
Work Center	An organizational unit, such as a department or a subcontracting company, that can accomplish a task.



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.

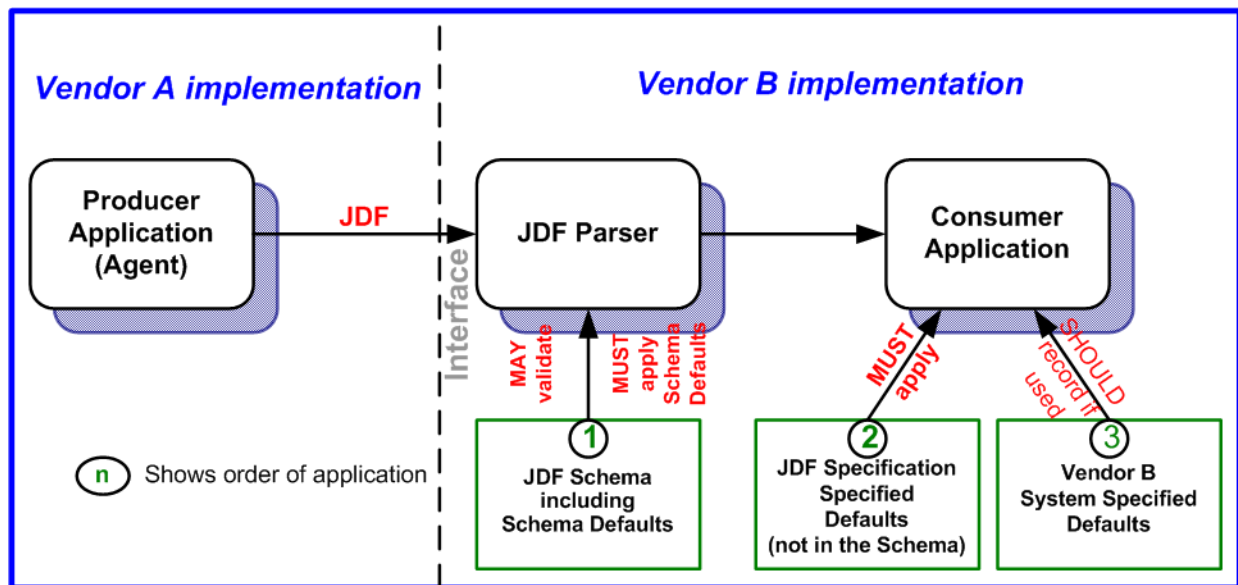


Figure 1-1: Handling of default values of JDF Attributes.

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, Generic Contents of All 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 Process 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 Settings Policy

The *SettingsPolicy*, *BestEffortExceptions*, *MustHonorExceptions* and *OperatorInterventionExceptions* Attributes are defined in Table 3-1, “Any Element (generic content),” on page 40. 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 857.

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 857 for all data type definitions.

Table 1-6: JDF data types (Section 1 of 4)

Data Type	Description
Anchor	Describe the 9 anchor points of a rectangle. See Table A-1, “Anchor Enumeration Values,” on page 870 for a list values.
boolean	Binary-valued logic: (true false).
CMYKColor	Represents a CMYK color specification.
date	Represents a time period that starts at midnight of a specified day and lasts for 24 hours.
dateTime	Represents a specific instant of time. It MUST be a UTC time or a local time that includes the time zone.
DateTimeRange	Two <i>dateTime</i> values separated by a “~” (tilde) character that defines the closed interval of the two. TimeRange corresponds semantically to the time interval (two time instants separated by a slash) defined in [ISO8601:2004].
DateTimeRangeList	Whitespace-separated list of <i>DateTimeRange</i> values.
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 <i>number</i> was used. The <i>double</i> and <i>number</i> data types are syntactically equivalent.
DoubleList New in JDF 1.2	Whitespace-separated list of <i>double</i> values. Note: this data type was named <i>NumberList</i> before JDF 1.2.

Table 1-6: JDF data types (Section 2 of 4)

Data Type	Description
DoubleRange New in JDF 1.2	Two <i>double</i> values separated by a “~” (tilde) character that define the closed interval of the two. Note: this data type was named <i>NumberRange</i> before JDF 1.2.
DoubleRangeList New in JDF 1.2	Whitespace-separated list of <i>double</i> and <i>DoubleRange</i> values. Note: this data type was named <i>NumberRangeList</i> before JDF 1.2.
duration	Represents a duration of time.
DurationRange	Describes a range of time durations. More specifically, it describes a time span that has a relative start and end.
DurationRangeList	Whitespace-separated list of <i>DurationRange</i> values.
element	Structured data. The specific data type is defined by the Element name.
enumeration	Limited set of <i>NMTOKEN</i> values (see below).
enumerations	Whitespace-separated list of <i>enumeration</i> values.
gYearMonth	Represents a specific Gregorian month in a specific Gregorian year.
hexBinary	Represents arbitrary hex encoded binary data.
hexBinaryList New in JDF 1.4	Whitespace-separated list of <i>hexBinary</i> values.
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 <i>integer</i> values.
IntegerRange	Two <i>integer</i> values separated by a “~” character that define a closed interval.
IntegerRangeList	Whitespace-separated list of <i>integer</i> values and <i>IntegerRange</i> values.
JDFJMFVersion	Version label of a JDF or JMF instance. See Section 3.13, JDF Versioning for a discussion of versioning in JDF. See Table A-2, “JDFJMFVersion Enumeration Values,” on page 870 for a list values.
JDFJMFVersions	Whitespace separated list of <i>JDFJMFVersion</i> .values
LabColor	Represents a Lab color specification.
language	Represents a language and country code (for example, en-US) for a natural language. Values MUST conform to [RFC1766].
languages New in JDF 1.4	Whitespace-separated list of <i>language</i> values.
LongInteger	Represents numerical integer values, including the special tokens INF and -INF. This corresponds to the standard XML integer with INF and -INF added. Values greater than +/-2**31 are expected to occur for this data type. For details, see [XMLSchema].
matrix	Whitespace-separated list of six doubles representing a coordinate transformation matrix.
NamedColor	Represents a color definition by name. See Table A-3, “NamedColor Enumeration Values,” on page 870 for a list values.
NameRange	Two <i>NMTOKEN</i> values separated by a “~” (tilde) character that define an interval of <i>NMTOKEN</i> values.
NameRangeList	Whitespace-separated list of <i>NMTOKEN</i> and <i>NameRange</i> values.

Table 1-6: JDF data types (Section 3 of 4)

Data Type	Description
NMTOKEN	A continuous sequence of special characters as defined by the [XML Specification 1.0].
NMTOKENS	Whitespace-separated list of <i>NMTOKEN</i> values.
Orientation New in JDF 1.2	Enumeration that specifies named orthogonal two-dimensional orientations. See Table A-4, "Orientation Enumeration Values," on page 871 for a list values.
Orientations New in JDF 1.2	Whitespace separated list of <i>Orientation</i> enumeration values that specify named orthogonal two-dimensional orientations.
PDFPath	Whitespace-separated list of path operators as defined in PDF.
rectangle	Whitespace-separated list of four doubles representing a rectangle.
refelement	ResourceElement or a reference to an Element. Used to define candidates for inter-Resource linking in Resources.
regExp New in JDF 1.2	Regular expression as defined by [XMLSchema]
shape	Whitespace-separated list of three doubles representing a three-dimensional shape consisting of a width, height and length. Unless specified otherwise in the Attribute description, these three numbers are an X-dimension, a Y-dimension and a Z-dimension, respectively.
ShapeRange	Two <i>shape</i> 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 <i>shape</i> values or <i>ShapeRange</i> 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. A telem contains text, but neither Attributes nor Subelements. A telem is defined when it appears as a Subelement. It is usually not defined with a table.
text	Text data contained in a telem (text element). Some Elements, such as Comment, have text, but are not telems because they haveAttributes and/or Subelements
TimeRange	Two <i>dateTime</i> values separated by a "~" (tilde) character that defines the closed interval of the two. TimeRange corresponds semantically to the time interval (two time instants separated by a slash) defined in [ISO8601:2004].
TransferFunction	Whitespace separated list of an even number of doubles representing a set of XY coordinates of a transfer function.
URI Modified in JDF 1.3	URI-reference. Represents a Uniform Resource Identifier (URI) Reference as defined in Section 4 of [RFC3986]. For the "file:" URL scheme, see [RFC3987]. URI was modified in JDF 1.3 to include Internationalized Resource Identifiers (IRI).
URL Modified in JDF 1.3	URL-reference. Represents a Uniform Resource Locator (URL) Reference as defined in Section 4 of [RFC3986]. For the "file:" URL scheme, see [RFC3987]. URL was modified in JDF 1.3 to include usage of Internationalized Resource Identifiers (IRI).
WorkStyle New in JDF 1.4	Specifies work styles of a press run. See Table A-5, "WorkStyle Enumeration Values," on page 871 for a list values.
XPath	Represents an XPath expression of an XML node set (Attributes or Elements), boolean, double or string.[XPath]
XYPair	Whitespace-separated list of two doubles. Unless specified otherwise in the Attribute Description, these two doubles are an X-dimension and a Y-dimension, respectively.

Table 1-6: JDF data types (Section 4 of 4)

Data Type	Description
XYPairRange	Two <i>XYPair</i> values separated by a “~” (tilde) character that defines a rectangle bounded by $x_1 y_1 \sim x_2 y_2$
XYPairRangeList	Whitespace-separated list of <i>XYPairRange</i> values.
XYRelation New in JDF 1.2	Defines the relationship between two ordered doubles. See Table A-6, “XYRelation Enumeration Values,” on page 872 for a list of NMTOKEN values.

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 to indicate the units used in the following JDF attributes:

- 1 The *Unit* Attribute in Resources (see Table 3-13, “Abstract Physical Resource Element,” on page 70)
- 2 The *Unit* Attribute in ResourceInfo (see Table 5-63, “ResourceInfo Element,” on page 221)
- 3 The *CounterUnit* Attribute in DeviceInfo (see Table 5-70, “DeviceInfo Element,” on page 230):

Table 1-7: Units used in JDF (Section 1 of 2)

Measurement	Unit	Representation	Remarks
Length	point (1/72 inch)	pt	Used for all except microscopic lengths (see below)
	micron (μ)	um	Used for microscopic lengths — where used (instead of points) it will be explicitly stated in the definition of the item. See Media/@Thickness .
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) NEED NOT be equivalent.
Paper weight	g/m^2	g/m2	—
Speed	units/hour	*/h	Replace the “*” in the representation with the appropriate unit

Table 1-7: Units used in JDF (Section 2 of 2)

Measurement	Unit	Representation	Remarks
Temperature	C° (Celsius)	C	degree centigrade
Angle	degrees°	degree	—
Countable Objects Modified in JDF 1.4	1	count	Countable objects, such as Sheets, MAY be specified as “count”.

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. Note that this restriction applies to the JDF representation only. Display of values, for instance in a user interface, is implementation defined.

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 3.2, JDF Node.

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

2.1.1.3 Attributes

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

For more information about Attributes, see Section 3.2, JDF Node. 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 3.11.3, Abstract Audit](#).) Alternatively, the completed **Job** can be passed to the **Job accounting** system, which examines the audit records to determine the costs of all the **Processes** in the **Job**.

2.1.2.6 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.

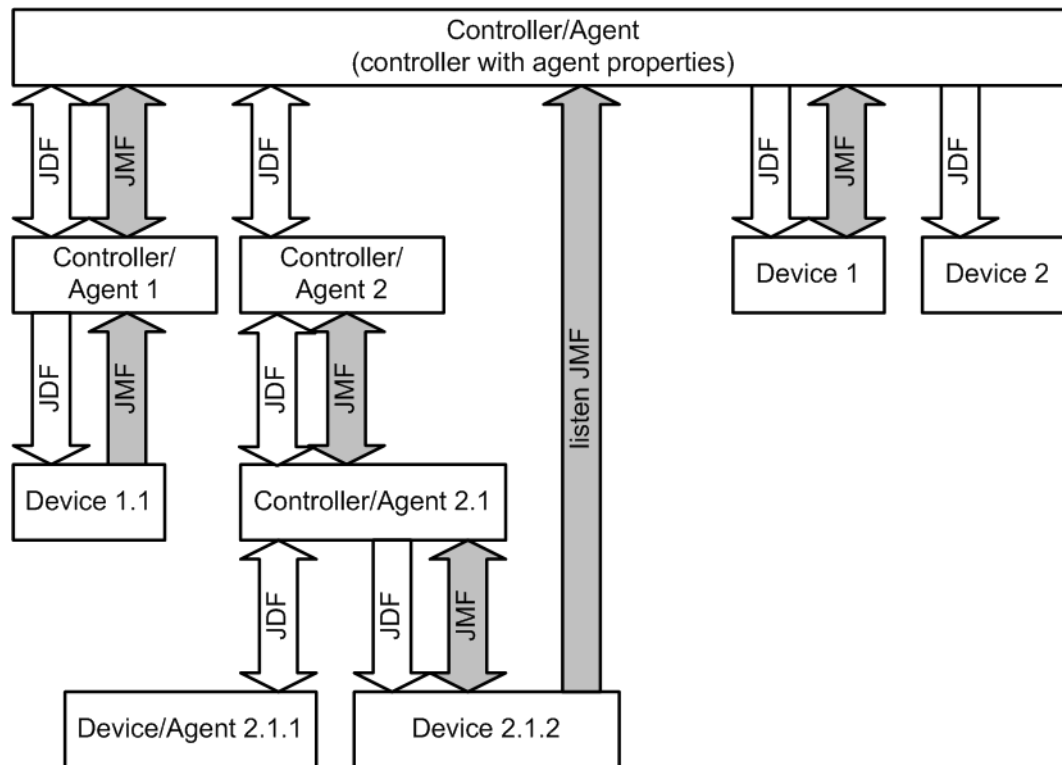


Figure 2-1: Example of JDF and JMF workflow interactions

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.

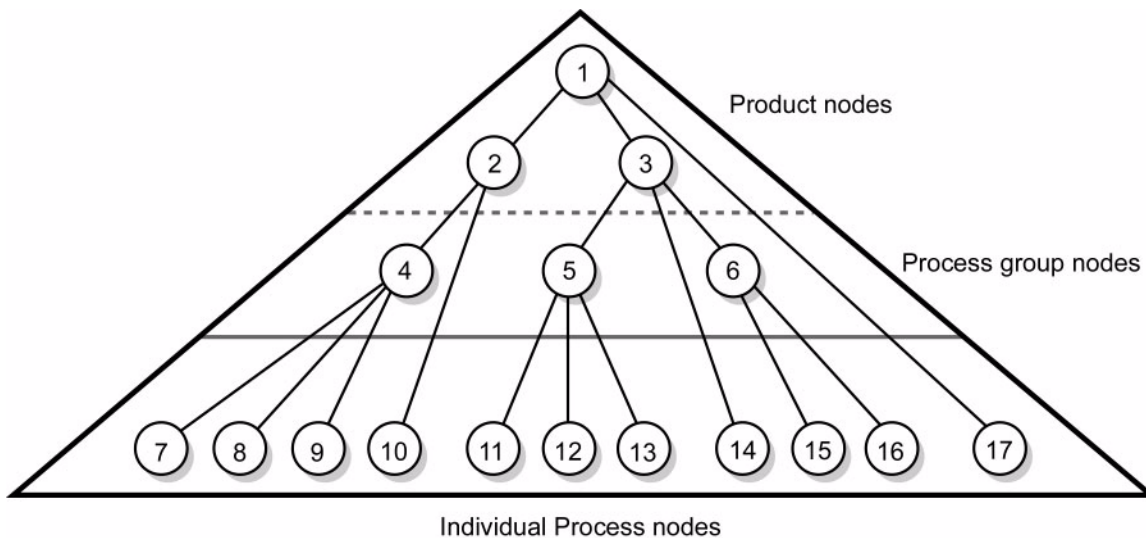


Figure 2-2: JDF tree structure

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?.

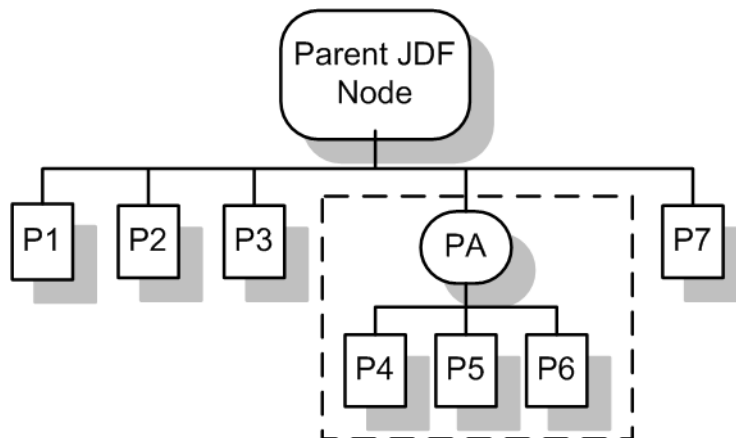


Figure 2-3: Example of a hierarchical tree structure of JDF Nodes

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.

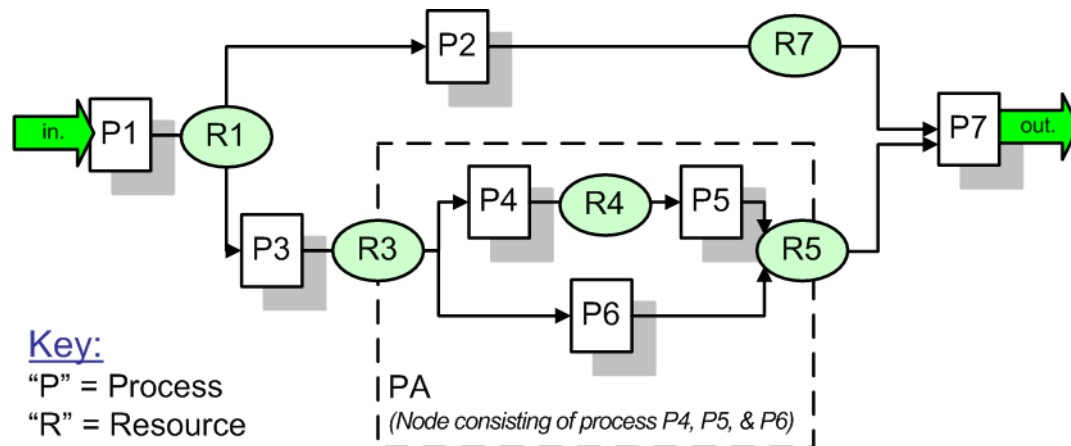


Figure 2-4: Example of a Process chain linked by Input Resources and Output Resources

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

The Agent or set of Agents employed by MIS to write the JDF Job 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 Message allows the Controller to determine what Processes can be executed by a particular Device or Work Center. These Processes are likely to be determined at system initialization time. The SubmitQueueEntry Message provides a means for the Controller to submit a Job ticket to individual work centers or Devices. And the Status, 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.)

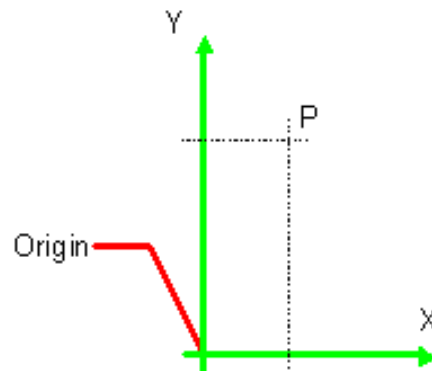


Figure 2-5: Standard coordinate system

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.

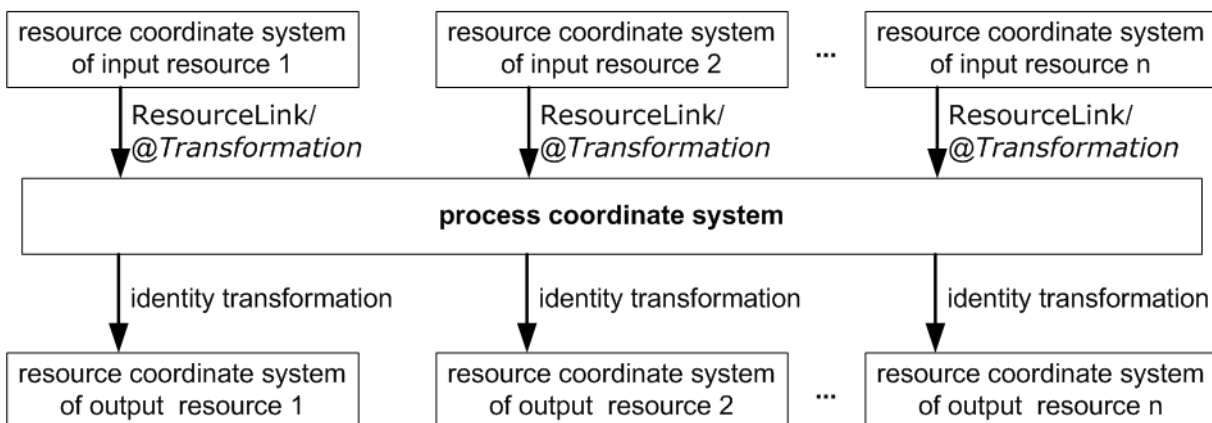


Figure 2-6: Relation between Resource and process coordinate systems

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	"Rotate180" or "Flip90"

Coordinates and transformations are used throughout JDF, to include:

Intent Resources, such as:

- **LayoutIntent**: specifies size of finished product

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

Process Resources, such as:

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

2.5.3 Coordinate Systems of Resources and Processes

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

It is often necessary to change the orientation of an Input Resource before executing the operation. This can be done by specifying a transformation matrix. It is stored in the *Orientation* or *Transformation* Attribute of the ResourceLink. This provides the ability to specify different matrices for the individual Resources of a Process. For details on ResourceLink Elements, see the section “ResourceLinkPool and ResourceLink” on page 73.

2.5.3.1 Coordinate Systems of Combined Processes

[New in JDF 1.2](#)

Combined Processes (See “Combined Process Nodes” on page 57.) 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 151.) 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 *Dimensions* Attribute. If these are unknown, it is still possible to define a general orientation in the *Orientation* Attribute of the ResourceLink. The naming of the Attribute reflects the state of the Resource and not necessarily the order of applied transformations. Thus *Rotate90* and *Flip90* specify that the original Y axis as represented by the spine is on top. In the case of *Flip90*, the **Component** is additionally flipped front to back.

Table 2-4: Matrices and Orientation values for describing the orientation of a Component

Orientation Value	Source Coordinate System	Transformation Matrix According Action	Target Coordinate System
<i>Rotate0</i>		$\begin{matrix} 1 & 0 & 0 & 1 & 0 & 0 \\ \text{No Action} \end{matrix}$	
<i>Rotate90</i>		$\begin{matrix} 0 & 1 & -1 & 0 & h & 0 \\ 90^\circ \text{ Counterclockwise Rotation} \end{matrix}$	
<i>Rotate180</i>		$\begin{matrix} -1 & 0 & 0 & -1 & w & h \\ 180^\circ \text{ Rotation} \end{matrix}$	
<i>Rotate270</i>		$\begin{matrix} 0 & -1 & 1 & 0 & 0 & w \\ 270^\circ \text{ Counterclockwise Rotation} \end{matrix}$	
<i>Flip0</i>		$\begin{matrix} 1 & 0 & 0 & -1 & 0 & h \\ \text{Flip around X} \end{matrix}$	
<i>Flip90</i>		$\begin{matrix} 0 & -1 & -1 & 0 & h & w \\ 90^\circ \text{ Counterclockwise Rotation} \\ + \text{ Flip around X} \end{matrix}$	
<i>Flip180</i>		$\begin{matrix} -1 & 0 & 0 & 1 & w & 0 \\ 180^\circ \text{ Rotation} + \text{ Flip around X} \end{matrix}$	
<i>Flip270</i>		$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 270^\circ \text{ Counterclockwise} \\ \text{Rotation} + \text{ Flip around X} \end{matrix}$	

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)	<i>Imposition</i>	RunList
RunList	<i>Interpreting</i>	RunList (of interpreted PDL data)
RunList (of interpreted PDL data) Media RenderingParams	<i>Rendering</i>	RunList (of rasterized byte maps)
RunList (of rasterized byte maps)	<i>Screening</i>	RunList (of bit maps)
ImageSetterParams Media (of film) RunList (of bit maps)	<i>ImageSetting</i> (to film)	ExposedMedia (of film)
ExposedMedia (of film)	<i>ContactCopying</i>	ExposedMedia (of plate)
ExposedMedia (of plate) ConventionalPrintingParams	<i>ConventionalPrinting</i>	Component
FoldingParams Component	<i>Folding</i>	Component
CollectingParams Component	<i>Collecting</i>	Component
SaddleStitchingParams Component	<i>SaddleStitching</i>	Component
TrimmingParams Component	<i>Trimming</i>	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.)

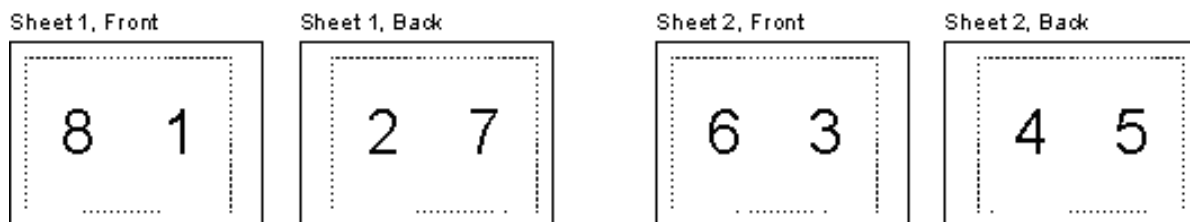


Figure 2-7: Layout of simple saddle stitched brochure (product example)

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.

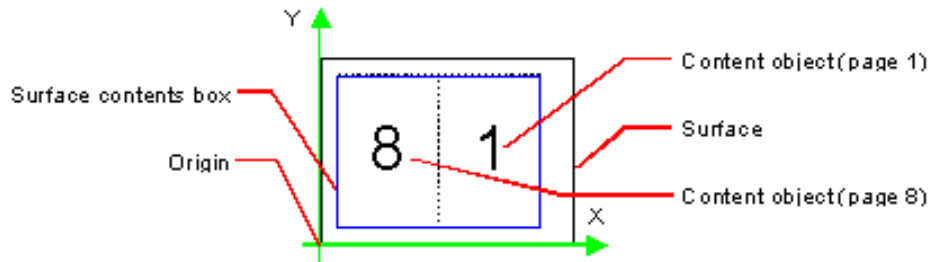


Figure 2-9: Surface coordinate system

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**.

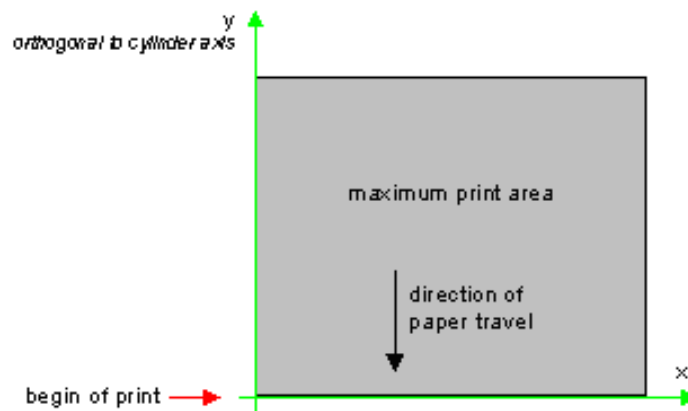


Figure 2-10: Press coordinate system used for Sheet-Fed Printing

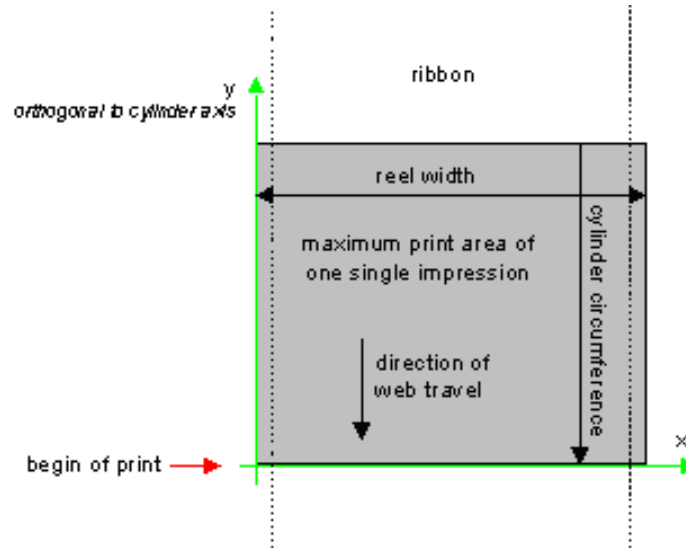


Figure 2-11: Press coordinate system used for Web Printing

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.)

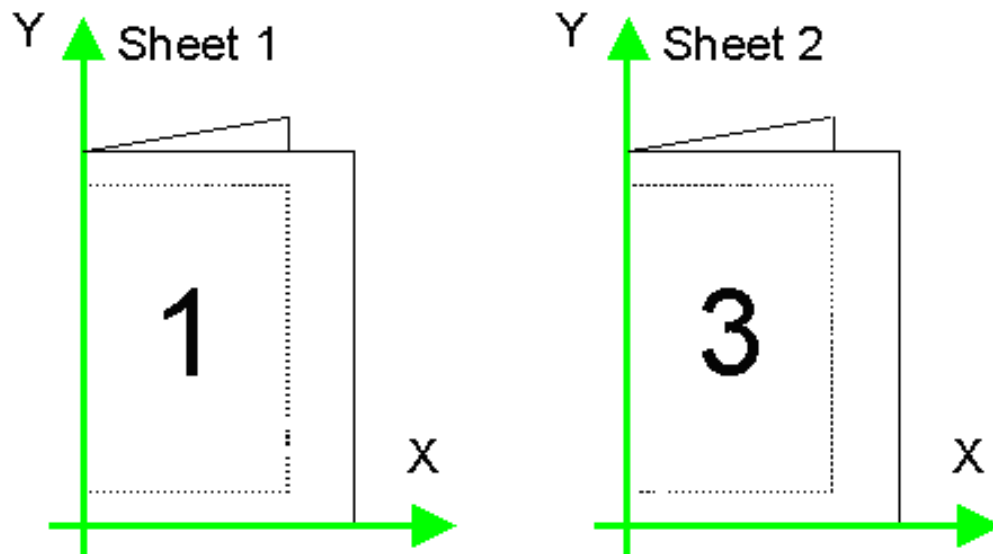


Figure 2-12: Coordinate systems after Folding (product example)

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.

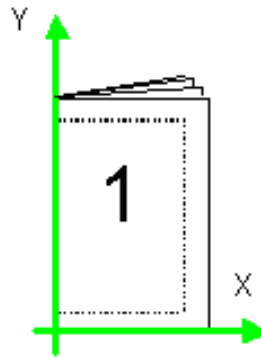


Figure 2-13: Coordinate systems after Collecting (product example)

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.

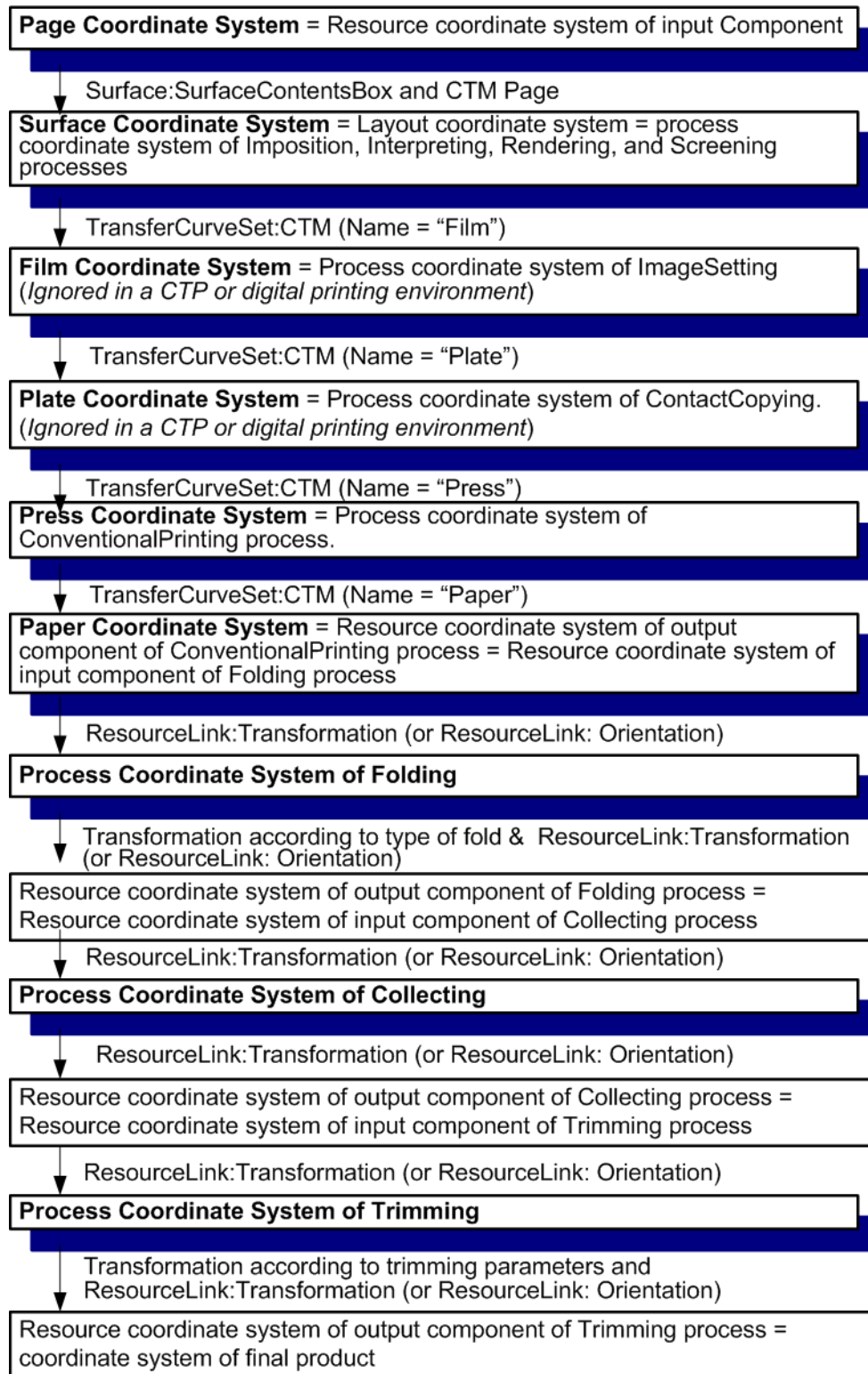


Figure 2-14: Examples of Transformations and Coordinate Systems in JDF.

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

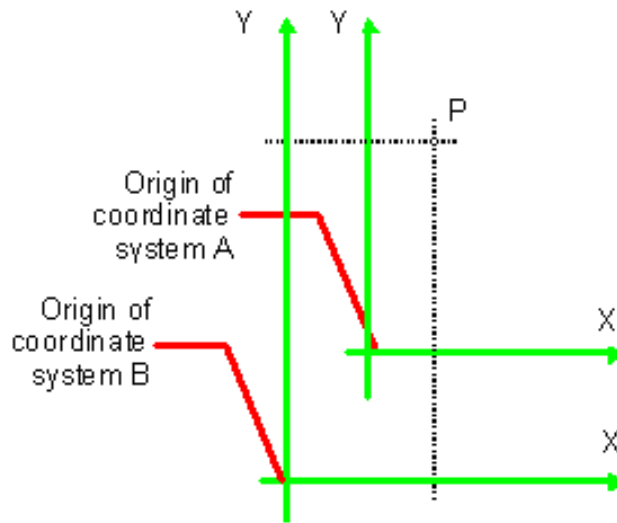


Figure 2-15: Transforming a point (example)

$$P_A = (30, 100)$$

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

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

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 Intent 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 269.

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 3.9, *ResourceLinkPool* and *ResourceLink*.

3.1 Generic Contents of All Elements

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

Table 3-1: Any Element (generic content) (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 Intent). It is strongly RECOMMENDED to supply <i>DescriptiveName</i> in all JDF Nodes, <i>Quantity Resources</i> (for example: Component Resources) and <i>Handling Resources</i> (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
<p><i>SettingsPolicy</i>?</p> <p>New in JDF 1.2</p>	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.</p> <p>Default value is from: parent's @<i>SettingsPolicy</i>. If not specified in the parent Element or further superior Elements, the default value is "<i>BestEffort</i>".</p> <p>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>Note: for additional details on <i>SettingsPolicy</i>, see "Conformance to Settings Policy" on page 13.</p>
Comment *	element	<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>
<p>GeneralID *</p> <p>New in JDF 1.4</p>	element	<p>Additional identifiers related to the Element.</p> <p>Creation note: starting with JDF 1.4, GeneralID has been promoted from being only in a Resource to being in any JDF Element,</p>
<p>Preview *</p> <p>New in JDF 1.4</p>	refelement	<p>Provides a Preview Resource for thumbnails or other images. MUST not provide multiple Preview Resources with the same Preview/@PreviewUsage values</p> <p>Creation note: starting with JDF 1.4, Preview has been moved from Table 6-1, "Template for Input Resources".</p>

3.1.1 Element: Comment

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	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 <i>Media/@Dimensions</i> or the name <i>Media/@Weight</i> MAY be localized. 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.
<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.

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

Name	Data Type	Description
<i>Name</i> = "Description" Modified in JDF 1.4	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.</p> <p>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 3-38, "Notification Audit Element," on page 126).</p> <p><i>DeviceText</i> – Human readable description created by the Device that provides details beyond the value of <i>@StatusDetails</i>. New in JDF 1.4</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 62. 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.

3.1.2 Element: GeneralID

[New in JDF 1.3](#)

[Modified in JDF 1.3](#)

Modification note: starting with JDF 1.4, GeneralID becomes an Element, and is no longer a Resource. See Section 7.2.84, GeneralID. GeneralID becomes a child of any Element. See Table 3-1, "Any Element (generic content)," on page 40.

GeneralID describes a generic variable. The name or usage of the variable is specified in GeneralID/*@IDUsage* and the specific value of the variable is specified in GeneralID/*@IDValue*. The data type is specified in GeneralID/*@DataType*.

Table 3-3: GeneralID Element

Name	Data Type	Description
<i>DataType</i> ? New in JDF 1.4	enumeration	Data type of the variable. Values are: <i>string</i> <i>integer</i> <i>double</i> <i>NMTOKEN</i> <i>boolean</i> <i>dateTime</i> <i>duration</i>
<i>IDUsage</i> Modified in JDF 1.4	NMTOKEN	Usage of the GeneralID. There are no predefined values in JDF. Values below with "AdsML" prefix are defined in [AdsML]. Values include: <i>DeviceProductID</i> – An ID of the resource as defined in the Device namespace. For instance Media catalogs of a press may provide Media identifiers that are different from those defined by the MIS (which are identified with ProductID values). New in JDF 1.4 <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>LineID</i> – an ID for PrintTalk which associates a PrintTalk//Pricing/Price[@LineID = "someValue"] Element with a JDF Element embedded in PrintTalk, such as PrintTalk//jdf:DeliveryParams/Drop/DropItem[GeneralID/@IDUsage = "LineID" and GeneralID/@IDValue = "someValue"].
<i>IDValue</i>	string	Value of the GeneralID. The data type of the value MUST correspond to GeneralID/ <i>@DataType</i> .

3.1.3 Structure Diagram

Figure 3-1 below shows the structure of the generic content defined above. Figure 3-1 and other similar diagrams describe JDF structure using the following notation.

- Each box represents an Element, with the Element's name in the rounded box at the top and its Attributes if any, listed below. A rounded box with a dashed line represents an Abstract Element

- A solid line connects an Element to its Subelement, where the Subelement is at the arrowhead. The cardinality of the Subelement is specified after its name. Cardinality in the line overrides that in the box.
- A dashed line connects an Element to its Abstract Element, where the superclass Element is at the arrowhead.

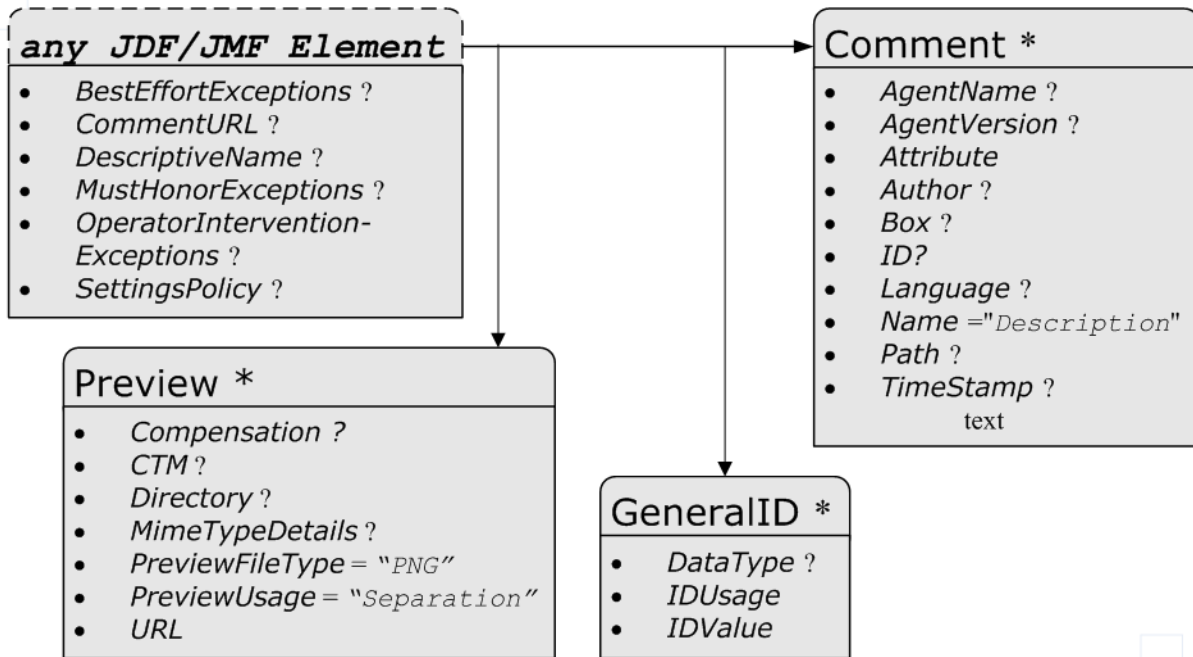


Figure 3-1: Any-Element (generic content) – a diagram of its structure

3.2 JDF Node

The top-level Element of a JDF instance is a JDF Element. JDF Elements MAY also be nested within other JDF Elements. The individual JDF Elements are referred to as “Nodes” and Nodes, in turn, contain various Attributes and further Subelements, including nested JDF Nodes.

The following table presents the Attributes and Elements likely to be found in any given JDF Node. Three of the Attributes in Table 3-5, 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 Intent or Process the JDF Node represents. As is detailed in Section 3.3, Common Node Types, all Nodes fall into one of the following four general categories: Process, Process Group, Combined Processes and Product Intent. Each Node is identified as belonging to one of these categories by the value of its *Type* Attribute, as described in the table below. For example, if *Type* = “*Product*”, the Node is a Product Intent Node. Each of these categories is described in greater detail in the sections that follow.

The table “JDF Node” on page 47 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 3-4: Definition of “Scope” Terms used in Table 3-5 on page 47

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, Spawning and Merging. All Attributes and Elements listed in subsequent chapters SHOULD be considered local unless otherwise noted.

Table 3-5: JDF Node (Section 1 of 5)

Name	Data Type	S	Description																					
<p><i>Activation?</i> Modified in JDF 1.1</p>	enumeration	D	<p>Describes the activation status of the JDF Node. Allows for a range of activity, including deactivation and test running.</p> <p>A child Node inherits the value of the <i>Activation</i> Attribute from its parent. The value of <i>Activation</i> corresponds to the least active value of <i>Activation</i> of any ancestor, including itself. Therefore, if any ancestor has an <i>Activation</i> of <i>Inactive</i>, the Node itself is <i>Inactive</i>.</p> <p>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 <i>Activation</i>.</p> <p>The values are ordered from least to most active</p> <p>Values 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 <i>Activation</i>= "<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 <i>Activation</i> SHOULD also be <i>Informative</i>. When a new Job ticket with an identical <i>ID</i> Attribute and a higher <i>Activation</i> is submitted to a Device, that JDF Job ticket MUST replace the JDF Job ticket that was submitted to the Device with an <i>Activation</i> 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 <i>Activation</i> 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 147.</p> <table border="1" data-bbox="646 1661 1430 1934"> <thead> <tr> <th data-bbox="646 1661 1008 1703">Activation</th> <th data-bbox="1008 1661 1219 1703">Test Node</th> <th data-bbox="1219 1661 1430 1703">Execute Node</th> </tr> </thead> <tbody> <tr> <td data-bbox="646 1703 1008 1738"><i>Inactive</i></td> <td data-bbox="1008 1703 1219 1738"><i>false</i></td> <td data-bbox="1219 1703 1430 1738"><i>false</i></td> </tr> <tr> <td data-bbox="646 1738 1008 1774"><i>Informative</i></td> <td data-bbox="1008 1738 1219 1774"><i>false</i></td> <td data-bbox="1219 1738 1430 1774"><i>false</i></td> </tr> <tr> <td data-bbox="646 1774 1008 1810"><i>Held</i></td> <td data-bbox="1008 1774 1219 1810"><i>false</i></td> <td data-bbox="1219 1774 1430 1810"><i>false</i></td> </tr> <tr> <td data-bbox="646 1810 1008 1845"><i>Active</i></td> <td data-bbox="1008 1810 1219 1845"><i>false</i></td> <td data-bbox="1219 1810 1430 1845"><i>true</i></td> </tr> <tr> <td data-bbox="646 1845 1008 1881"><i>TestRun</i></td> <td data-bbox="1008 1845 1219 1881"><i>true</i></td> <td data-bbox="1219 1845 1430 1881"><i>false</i></td> </tr> <tr> <td data-bbox="646 1881 1008 1934"><i>TestRunAndGo</i></td> <td data-bbox="1008 1881 1219 1934"><i>true</i></td> <td data-bbox="1219 1881 1430 1934"><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 3-5: JDF Node (Section 2 of 5)

Name	Data Type	S	Description
<p><i>Category?</i></p> <p>New in JDF 1.2</p> <p>Modified in JDF 1.4</p>	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.</p> <p>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>Newsprinting</i> – A press run on a newsprinting Web Press. New in JDF 1.4</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 6.4.32, 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>Note: 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>
<p><i>ICSVersions?</i></p> <p>New in JDF 1.2</p>	NMTOKENS	D	<p>CIP4 Interoperability Conformance Specification (ICS) Versions that this JDF Node complies with.</p> <p>Value format is: <ICSName>_L<ICSLevel>-<ICSVersion>.</p> <p>Example: MISPRE_L1-1.3 for the MIS to Prepress ICS. If there is a revision to that ICS: "MISPRE_L1-1.3.1". See "Interoperability Conformance Specifications" on page 855 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>

Table 3-5: JDF Node (Section 3 of 5)

Name	Data Type	S	Description
<i>JobPartID</i> ?	string	D	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 <i>Resource/@ProductID</i> and not by <i>JobPartID</i> .
<i>MaxVersion</i> ? 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 3.13, JDF Versioning for a discussion of versioning in JDF.
<i>NamedFeatures</i> ? 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 776. <i>NamedFeatures</i> MUST be specified only in Process Group Nodes, typically with a <i>Types</i> Attribute supplied, or Product Intent Nodes. See "Use of the NamedFeatures Attribute in Product and Process Group Nodes" on page 56 for details.
<i>ProjectID</i> ? 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.
<i>RelatedJobID</i> ? 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.
<i>RelatedJobPartID</i> ? 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.
<i>SpawnID</i> ? 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 156.
<i>Status</i> Modified in JDF 1.3	enumeration	L	Identifies the status of the Node. Derivation of the <i>Status</i> of a parent Node from the <i>Status</i> of child Nodes is non-trivial and implementation-dependent. Values are from: Table 3-6, "Status Attribute Values," on page 52).
<i>StatusDetails</i> ? New in JDF 1.2	string	L	Description of the status phase that provides details beyond the enumerative values given by the <i>Status</i> Attribute. Values include those from: "StatusDetails Supported Strings" on page 875.
<i>Template</i> = "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".
<i>TemplateID</i> ? 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.

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

Name	Data Type	S	Description
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.
<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 269, although the flexibility of JDF allows anyone to create Processes. In addition to these, there are three values which are described in greater detail in the sections that follow. Values include: <i>Combined</i> <i>ProcessGroup</i> <i>Product</i> – Identifies a Product Intent Node. Values include those from: "Processes" on page 269.
Types? 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> = "Combined", OPTIONAL when <i>Type</i> = "ProcessGroup", and is ignored if <i>Type</i> equals any other value. For details on using Combined Process Nodes, see Section 3.3.3, Combined Process Nodes. If the <i>Types</i> Attribute is specified, that JDF Node MUST NOT contain child JDF Nodes. For details on using Process Group Nodes, see Section 3.3.2, Process Group Nodes. If <i>Type</i> = "ProcessGroup", the tokens MAY also be the name of a Gray Box that needs expansion. See <i>Category</i> for more details. Values include those from: "Processes" on page 269.
Version? 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 138.
xmlns? 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 versions 1.1 through 1.4 of JDF, <i>xmlns</i> = "http://www.CIP4.org/JDFSchema_1_1".
xsi:type? 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 Node" on page 45.
<i>AncestorPool?</i>	element	R	If this Element is present, the current JDF Node has been spawned, and this Element contains a list of all <i>Ancestor</i> Elements prior to spawning. See Section 3.4, <i>AncestorPool</i> .
<i>AuditPool?</i>	element	L	List of Elements that contains all relevant audit information. <i>Audit</i> Elements are intended to serve the requirements of MIS for evaluation and post calculation. See Section 3.11, <i>AuditPool</i> and <i>Audit</i> .

Table 3-5: JDF Node (Section 5 of 5)

Name	Data Type	S	Description
CustomerInfo ? Deprecated in JDF 1.3	element	D	Container Element for customer-specific information. See Section 3.5, CustomerInfo. In JDF 1.3 and beyond, CustomerInfo is a Resource that is referenced through a CustomerInfoLink in the ResourceLinkPool.
JDF *	element	L	Child JDF Nodes. The nesting of JDF Nodes defines the JDF tree.
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 3.11, AuditPool and Audit. In JDF 1.3 and beyond, NodeInfo is a Resource that is referenced through a NodeInfoLink in the ResourceLinkPool.
ResourceLinkPool ?	element	L	Container Element for ResourceLink Elements, which describe the input and Output Resources of the Node. See Section 3.9, ResourceLinkPool and ResourceLink.
ResourcePool ?	element	L	Container Element for Resources. See Section 3.8, ResourcePool and its Resource Children. Note: Resources are local in a ResourcePool but MAY be referenced from ResourceLink Elements in descendent Nodes. For details see "ResourceLinkPool and ResourceLink" on page 73.
StatusPool ? Deprecated in JDF 1.3	element	L	Container for PartStatus Elements that specify the details of a Node's Partition dependent <i>Status</i> related Attributes if the <i>Status</i> of the Node is "Pool". Deprecation note: starting with JDF 1.3, StatusPool/PartStatus/@Status is replaced by NodeInfo/@NodeStatus in the respective Partition of NodeInfo .

— Attribute: Status

Table 3-6: Status Attribute Values

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 3.11, AuditPool and Audit .
<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> New in JDF 1.3 Modified in JDF 1.4	Execution has been stopped. If a Job is <i>Suspended</i> , running will be resumed later. Unlike <i>Stopped</i> this <i>Status</i> indicates that the Job has been taken off the Device to execute another Job or perform some other action that is not related to this Job. When resumed, the Job MAY go into <i>Status</i> = " <i>Setup</i> " before changing to <i>InProgress</i> again. <i>Suspended</i> is also used to describe iterations. In an iterative environment, <i>Suspended</i> specifies that at least one iteration cycle has completed but additional iteration cycles MAY still occur. In this use case, <i>StatusDetails</i> SHOULD be set to " <i>IterationPaused</i> "
<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.
<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> New in JDF 1.3	Indicates that the Node is processing Partitioned Resources and that the Status varies depending on the Partition Keys. Details are provided in the NodeInfo Resource of the Node.
<i>Pool</i> Deprecated in JDF 1.3	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.

3.2.1 Structure Diagram

Figure 3-2 is a schematic structure of the JDF Node.

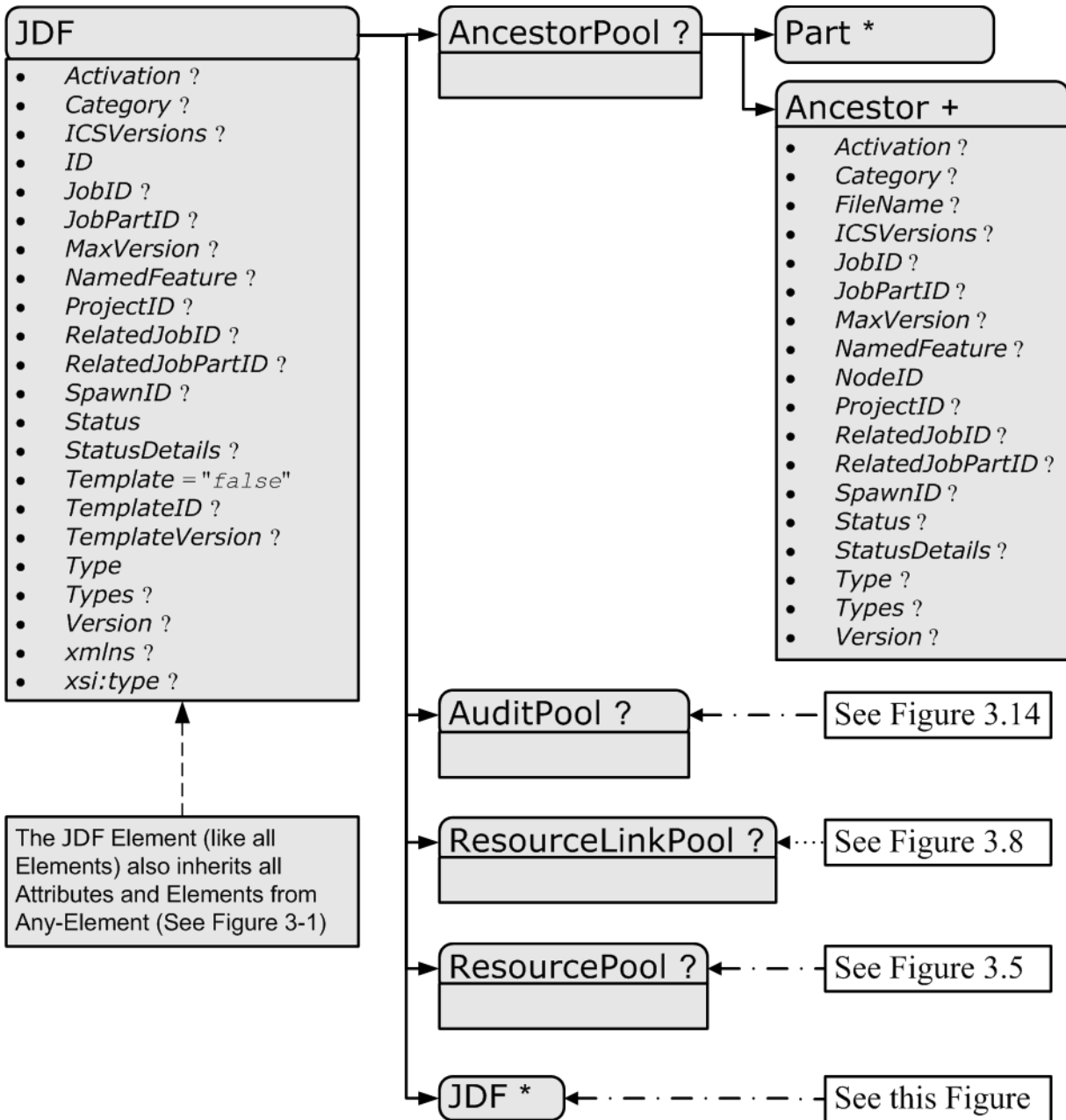


Figure 3-2: JDF Node – a Diagram of its Structure

3.3 Common Node Types

As was noted in the preceding section, the *Type* of a Node can fall into four categories. The first is comprised of the specific Processes of the kind delineated in "Processes" on page 269, known simply as Process Nodes. The other categories are made up of three enumerative values of the *Type* Attribute: *ProcessGroup*, *Combined* and *Product*, which is also known as Product Intent. These three Node types are described in this section.

The figure below, which was also presented as an illustration in Chapter 2, represents a theoretical Job hierarchy comprised of Product Intent Nodes, Process Group Nodes and Nodes that represent individual or Combined Processes. The diagram is divided into three levels to help illustrate the difference between the three kinds of Nodes, but

these levels do not dictate the hierarchical nesting mechanism of a Job. Note, however, that an individual Process Node MAY be the child of a Product Intent Node without first being the child of a Process Group Node. Likewise, a Process Group Node MAY have child Nodes that are also Process Groups.

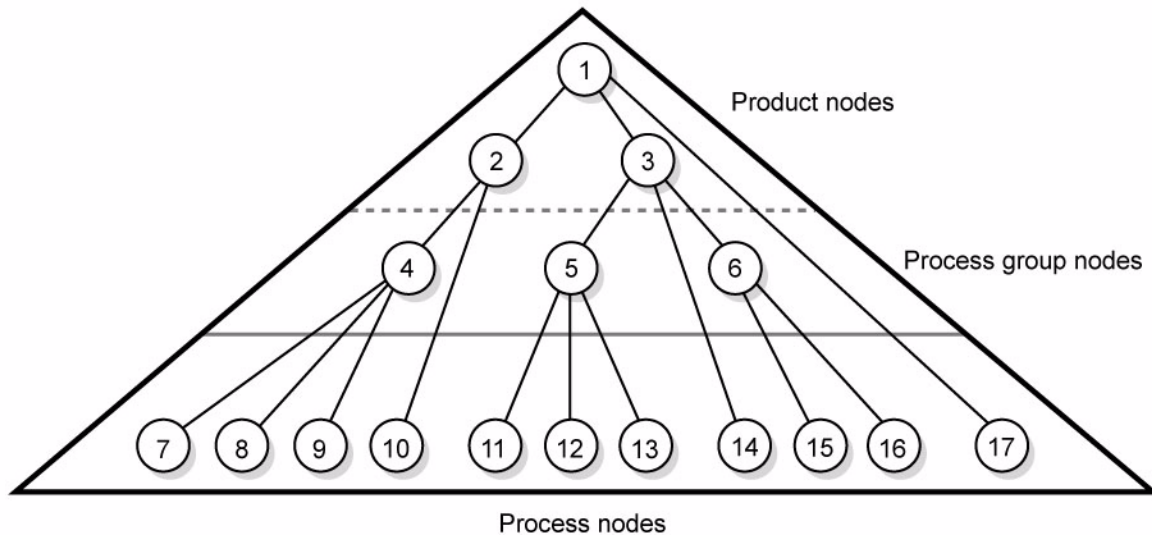


Figure 3-3: Job hierarchy with Process, Process Group and Product Intent Nodes

3.3.1 Product Intent Nodes

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

If this is the case, the initiating Agent creates a set of top-level Nodes to specify the Product Intent without providing any of the processing details. Subsequent Agents then add Nodes below these top-level Nodes to provide the processing details needed to fulfill the intent specified.

These top-level Nodes 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*.

3.3.2 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 “Process Group Nodes”.) These Nodes are used to describe multiple steps in a Process chain that have common Resources or scheduling data.

Since the Agent writing the Job has the option of grouping Processes in any way that seems logical, custom workflows can be modeled flexibly. Process Group Nodes MAY contain further Process Group Nodes, individual Process Nodes or a mixture of both Node types. Sequencing of Process Group Nodes SHOULD be defined by linking Resources of the appropriate leaves, or if the nature of the interchange Resources is unknown, by linking Placeholder Resources.

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

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

[New in JDF 1.2](#)

Process Group Nodes MAY contain an OPTIONAL *@Types* Attribute that allows a Controller (e.g., an MIS system) to specify a minimum set of Processes to be executed without specifying the complete list of Processes or the exact structure or grouping of these Processes into individual JDF Nodes. Process Group Nodes that contain a *Types* Attribute are commonly referred to as Gray Boxes. Additional Processes that are not included in *Types* MAY be added during expansion of a Gray Box. A *ResourceLink/@CombinedProcessIndex* is used to map *ResourceLink* Elements to *JDF/@Types* in the *ProcessGroup*. Process Group Nodes with a non-empty *Types* Attribute MUST NOT be executed. A Device that receives the Process Group Node MUST define the exact structure of the Process Group Node by executing the following steps until the *Types* list referenced by the Process Group Node is empty:

Step 1 — Select at least one of the Process types defined in *Types* and remove these values from the *Types* list of values referenced by the Process Group Node.

Step 2 — Create one new JDF child Node within the *ProcessGroup* that either:

- Has a *Type* Attribute matching the removed *Types* entry value, or
- Is a JDF Node with a *Type* Attribute Value of “*Combined*” or “*ProcessGroup*” that contains the removed *Types* value or values.

Step 3 — Link the appropriate Resources that were predefined in the original Process Group Node to the newly created subordinate JDF Node(s). The *ResourceLink* MUST either be retained or deleted from the Process Group Node. If it is retained, the Process Group Node MUST NOT be executed before the Resource that is linked by that *ResourceLink* is available. Otherwise, the Process Group Node MAY be executed, even if the Resource is not available.

Step 4 — Add missing *Types* to the subordinate JDF Node where appropriate. For instance, the original *Types* Attribute list referenced by Process Group Node might have specified “*Interpreting Rendering*” or simply “*RIPing*”, but the newly created RIP Node would specify “*Interpreting Rendering Trapping Screening*”.

Step 5 — Finalize the newly created subordinate JDF Node by adding any missing Resources and Resource parameters. Note that newly created Resources MUST NOT be linked to the Process Group 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 Process Group Nodes that contain both a non-empty *Types* Attribute and child JDF Nodes are *not* supported, although a Process Group Node MAY contain child Process Group Nodes that have non-empty *Types* Attribute.

3.3.2.2 Use of the NamedFeatures Attribute in Product and Process Group Nodes

[New in JDF 1.2](#)

ProcessGroup and Product Intent 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 776.), 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 Process Group Nodes as described in “Use of the Types Attribute in Process Group Nodes – Gray Boxes” on page 55. *Types* abstractly specifies the set of Processes to execute, whereas *NamedFeatures* abstractly specifies the set of Resources for the Processes specified in *Types*.

3.3.2.3 ResourceLink Structure in Process Group Nodes

[New in JDF 1.2](#)

The contents of the ResourceLinkPool of a Process Group Node define the Resources that MUST be available for the Process Group Node itself to be executed.

Example 3-1: ResourceLink Structure for a ProcessGroup

The following example shows the ResourceLink structure for a *ProcessGroup* digital printing with near-line finishing Node. The input **Media** is Available and the Output **Component** is of interest to the submitting Controller. The Parameter Resources are assumed to be supplied by the sub-Controller that executes the Process Group Node. Note the presence of intermediate component links that link the individual Processes. The corresponding ResourcePool Elements and Resource Elements have been omitted for brevity.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <!--the ResourceLink Elements in the ProcessGroup define the Input
    Resources that are to be available for the ProcessGroup to be
    submitted and the Output Resources that are produced by the ProcessGroup
  -->
  <ResourcePool>
    <DigitalPrintingParams ID="L1" Class="Parameter" Status="Available"/>
    <Media ID="L2" Class="Consumable" Status="Available"/>
    <RunList ID="L8" Class="Parameter" Status="Available"/>
    <Component ID="L3" Class="Quantity" Status="Unavailable"
      ComponentType="Sheet"/>
    <GatheringParams ID="L4" Class="Parameter" Status="Available"/>
    <Component ID="L5" Class="Quantity" Status="Unavailable"
      ComponentType="Sheet"/>
    <StitchingParams ID="L6" Class="Parameter" Status="Available"/>
    <Component ID="L7" Class="Quantity" Status="Unavailable"
      ComponentType="Sheet"/>
  </ResourcePool>
</ResourceLinkPool>
```

```

<!-- print input media -->
<MediaLink Usage="Input" rRef="L2"/>
<!-- gathered output components -->
<ComponentLink Usage="Output" rRef="L7"/>
</ResourceLinkPool>
<JDF ID="J2" Status="Waiting" JobPartID="ID301" Type="DigitalPrinting">
  <ResourceLinkPool>
    <!-- digital printing parameters -->
    <DigitalPrintingParamsLink Usage="Input" rRef="L1"/>
    <!-- input sheets -->
    <MediaLink Usage="Input" rRef="L2"/>
    <RunListLink Usage="Input" rRef="L8"/>
    <!-- printed output components -->
    <ComponentLink Usage="Output" rRef="L3"/>
  </ResourceLinkPool>
</JDF>
<JDF ID="J3" Status="Waiting" JobPartID="ID302" Type="Gathering">
  <ResourceLinkPool>
    <!-- gathering parameters -->
    <GatheringParamsLink Usage="Input" rRef="L4"/>
    <!-- printed output components -->
    <ComponentLink Usage="Input" rRef="L3"/>
    <!-- gathered output components -->
    <ComponentLink Usage="Output" rRef="L5"/>
  </ResourceLinkPool>
</JDF>
<JDF ID="J4" Status="Waiting" JobPartID="ID303" Type="Stitching">
  <ResourceLinkPool>
    <!-- Stitching parameters -->
    <StitchingParamsLink Usage="Input" rRef="L6"/>
    <!-- gathered output components -->
    <ComponentLink Usage="Input" rRef="L5"/>
    <!-- stitched output components -->
    <ComponentLink Usage="Output" rRef="L7"/>
  </ResourceLinkPool>
</JDF>
</JDF>

```

3.3.3 Combined Process Nodes

The Processes described in "Processes" on page 269 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 Process 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 Process 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.

3.3.3.1 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.

Example 3-2: Combined Process Node

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

Example 3-3: ResourceLinkPool for Combined Process Node

The following example of the ResourceLinkPool of a JDF Node describes digital printing with in-line finishing and includes the same Processes as the previous ProcessGroup example. The Node requires the Parameter 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" JobPartID="ID200"
  Version="1.4">
  <ResourceLinkPool>
    <!-- digital printing input RunList -->
    <RunListLink CombinedProcessIndex="0" Usage="Input" rRef="L1"/>
    <!-- digital printing parameters -->
    <DigitalPrintingParamsLink CombinedProcessIndex="0" Usage="Input" rRef="L2"/>
```

```

<!-- gathering parameters -->
<GatheringParamsLink CombinedProcessIndex="1" Usage="Input" rRef="L3"/>
<!-- Stitching parameters -->
<StitchingParamsLink CombinedProcessIndex="2" Usage="Input" rRef="L4"/>
<!-- input sheets -->
<MediaLink CombinedProcessIndex="0" Usage="Input" rRef="L5"/>
<!-- stitched output components -->
<ComponentLink CombinedProcessIndex="2" Usage="Output" rRef="L6"/>
</ResourceLinkPool>
<ResourcePool>
  <RunList ID="L1" Class="Parameter" Status="Available"/>
  <DigitalPrintingParams ID="L2" Class="Parameter" Status="Available"/>
  <GatheringParams ID="L3" Class="Parameter" Status="Available"/>
  <StitchingParams ID="L4" Class="Parameter" Status="Available"/>
  <Media ID="L5" Class="Consumable" Status="Available"/>
  <Component ID="L6" Class="Quantity" Status="Unavailable"
    ComponentType="Sheet"/>
</ResourcePool>
</JDF>

```

3.3.3.2 Specifying non-linear dependencies in a Combined Process Node

A Combined Process Node typically specified a linear execution chain of the individual Process steps defined in JDF/ @Types. A Device that executes a Combined Process Node MAY execute a more complex network of individual work steps. For instance, a Cover might be printed from one tray, the insert from another tray and both be bound to produce a bound component. This behavior is modeled by explicitly declaring the exchange Resource and by defining it as a pipe by specifying Resource/@PipeID and Resource/@PipeProtocol="Internal". The exchange Resource linking it to the 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.

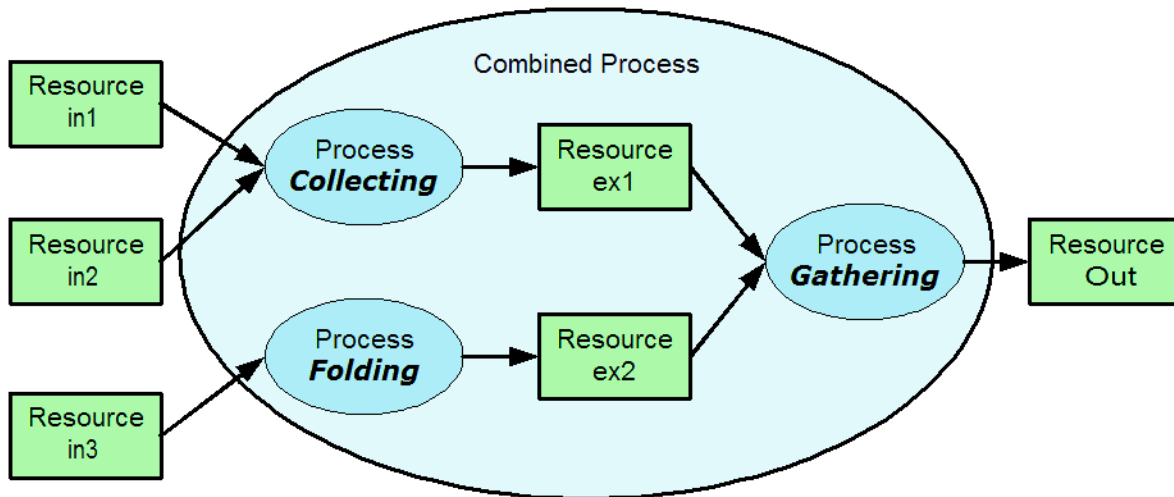


Figure 3-4: Combined Process Node dependencies

Example 3-4: Complex Combined Process Node

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

```

<JDF ID="ID" xmlns="http://www.CIP4.org/JDFSchema_1_1" Status="Waiting"
  Type="Combined" Types="Collecting Gathering Folding" JobPartID="ID345"
  Version="1.4">
  <ResourcePool>
    <GatheringParams ID="gpl" Class="Parameter" Status="Available"/>

```

```

<FoldingParams ID="fpl" Class="Parameter" Status="Available"/>
<Component ID="in1" Class="Quantity" Status="Available" ComponentType="Sheet"/>
<Component ID="in2" Class="Quantity" Status="Available" ComponentType="Sheet"/>
<Component ID="in3" Class="Quantity" Status="Available" ComponentType="Sheet"/>
<Component ID="ex1" Class="Quantity" Status="Unavailable" ComponentType="Sheet"
  PipeProtocol="Internal" PipeID="ex1"/>
<Component ID="ex2" Class="Quantity" Status="Unavailable" ComponentType="Sheet"
  PipeProtocol="Internal" PipeID="ex2"/>
<Component ID="Out" Class="Quantity" Status="Unavailable"
  ComponentType="Sheet"/>
</ResourcePool>
<ResourceLinkPool>
  <GatheringParamsLink Usage="Input" rRef="gp1"/>
  <FoldingParamsLink Usage="Input" rRef="fpl"/>
  <ComponentLink CombinedProcessIndex="0" Usage="Input" rRef="in1"/>
  <ComponentLink CombinedProcessIndex="0" Usage="Input" rRef="in2"/>
  <ComponentLink CombinedProcessIndex="2" Usage="Input" rRef="in3"/>
  <ComponentLink CombinedProcessIndex="0" Usage="Output" rRef="ex1"/>
  <ComponentLink CombinedProcessIndex="2" Usage="Output" rRef="ex2"/>
  <ComponentLink CombinedProcessIndex="1" Usage="Input" rRef="ex1"/>
  <ComponentLink CombinedProcessIndex="1" Usage="Input" rRef="ex2"/>
  <ComponentLink CombinedProcessIndex="1" Usage="Output" rRef="Out"/>
</ResourceLinkPool>
</JDF>

```

3.3.4 Process Nodes

Process Nodes represent the very lowest level in a Job hierarchy. They 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.

3.4 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 3-7: AncestorPool Element

Name	Data Type	Description
Ancestor +	element	Ordered list of one or more Ancestor Elements, which reflect the family tree of a spawned JDF.
Part * New in JDF 1.1	element	List of parts that this Node was spawned with. Used in case of parallel spawning of a Node. This defines the aggregated Part(s) in the case of nested spawns, (i.e., a logical AND of all spawned Part(s)). For instance, the JDF that was spawned with a <i>SheetName</i> Partition and subsequently spawned with a <i>Separation</i> would contain both <i>SheetName</i> and <i>Separation</i> within Part.

3.4.1 Element: Ancestor

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 3-8: Ancestor Element (Section 1 of 2)

Name	Data Type	Description
<i>Activation?</i>	enumeration	Copy of the <i>Activation</i> Attribute from the ancestor Node. For details, see Table 3-5, “JDF Node,” on page 47.). Values are from: JDF/@ <i>Activation</i> .
<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 3-5, “JDF Node,” on page 47. Values are from: JDF/@ <i>Category</i> .
<i>FileName?</i>	URL	The URL of the JDF file where the ancestor Node resided prior to spawning. Note that despite of Attribute name, <i>URL</i> NEED NOT refer to a file. <i>URL</i> MAY refer to any url scheme.
<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 3-5, “JDF Node,” on page 47. Values are from: JDF/@ <i>ICSVersions</i> .
<i>JobID?</i>	string	Copy of the <i>JobID</i> Attribute from the original ancestor Node. For details, see Table 3-5, “JDF Node,” on page 47.
<i>JobPartID?</i>	string	Copy of the <i>JobPartID</i> Attribute from the original ancestor Node. For details, see Table 3-5, “JDF Node,” on page 47.
<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 3-5, “JDF Node,” on page 47.
<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 3-5, “JDF Node,” on page 47.
<i>NodeID</i>	NMTOKEN	Copy of the <i>ID</i> Attribute of the ancestor Node. Note: the data type is NMTOKEN and not ID because the ID does not reside in the spawned JDF. The corresponding <i>ID</i> Attribute resides in the original JDF.
<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	string	Copy of the <i>RelatedJobID</i> Attribute from the original ancestor Node. For details, see Table 3-5, “JDF Node,” on page 47.
<i>RelatedJobPartID?</i> New in JDF 1.2	string	Copy of the <i>RelatedJobPartID</i> Attribute from the original ancestor Node. For details, see Table 3-5, “JDF Node,” on page 47.
<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 3-5, “JDF Node,” on page 47. Values are from: JDF/@ <i>Status</i> .
<i>StatusDetails?</i> New in JDF 1.2	string	Copy of the <i>StatusDetails</i> Attribute from the original ancestor Node. For values and details, see Table 3-5, “JDF Node,” on page 47. Values include those from: JDF/@ <i>StatusDetails</i>

Table 3-8: Ancestor Element (Section 2 of 2)

Name	Data Type	Description
<i>Type</i> ?	NMTOKEN	Copy of the <i>Type</i> Attribute from the original ancestor Node. For details, see Table 3-5, “JDF Node,” on page 47. Values are from: JDF/@ <i>Type</i> .
<i>Types</i> ?	NMTOKENS	Copy of the <i>Types</i> Attribute from the original ancestor Node. For details, see Table 3-5, “JDF Node,” on page 47. Values are from: JDF/@ <i>Types</i> .
<i>Version</i> ?	JDFJMFVersion	Copy of the <i>Version</i> Attribute from the original ancestor Node. For details, see Table 3-5, “JDF Node,” on page 47.
CustomerInfo ? New in JDF 1.1 Modified in JDF 1.3	refelement	Reference to or copy of the CustomerInfo Element or Resource from the original Node. In JDF 1.3 and beyond, CustomerInfo MAY be a Resource reference. For details, see Table 3-5, “JDF Node,” on page 47.
NodeInfo ? New in JDF 1.1 Modified in JDF 1.3	refelement	Reference to or copy of the NodeInfo Element or Resource from the original Node. In JDF 1.3 and beyond, NodeInfo MAY be a Resource reference. For details, see Table 3-5, “JDF Node,” on page 47.

3.5 CustomerInfo

[Deprecated in JDF 1.3](#)

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

3.6 NodeInfo

[Deprecated in JDF 1.3](#)

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

3.7 StatusPool

[Deprecated in JDF 1.3](#)

Starting with JDF 1.3, StatusPool is deprecated and replaced by a Partitioned **NodeInfo** Resource. For details, see “StatusPool and PartStatus” on page 1005.

3.8 ResourcePool and its Resource Children

3.8.1 ResourcePool

All Resources are contained in the ResourcePool Element of some Node. The ResourcePool Element is described in the following table.

Table 3-9: ResourcePool Element

Name	Data Type	Description
Resource *	element	List of Resource Elements. The Resource Elements are abstract and serve as placeholders for any Resource type.

3.8.2 Resource

Resources represent the “things” that are produced or consumed by Processes. They might be physical items such as inks, plates or glue; electronic items such as files or images; or conceptual items such as parameters and Device set-

tings. Processes describe what Resources they input or output through ResourceLink Elements, discussed in Section 3.9, 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.

3.8.3 Abstract Resource

Like the *Type* Attribute in abstract JDF Nodes, the *Class* Attribute in Resource Elements helps to identify how particular Resources are to be used. These values are listed in Table 3-10, “Abstract Resource Element,” on page 63, below, and are described in greater detail in the sections that follow.

Modification note: GeneralID has moved to Table 3-1, “Any Element (generic content),” on page 40

Table 3-10: 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.
<i>CatalogID</i> ? Deprecated in JDF 1.4	string	Identification of the Resource, (e.g., in a catalog environment). Defaults to the value of <i>ProductID</i> . Deprecation note: starting with JDF 1.4, use <i>GeneralID</i> .
<i>CatalogDetails</i> ? Deprecated in JDF 1.4	string	Additional details of a Resource in a catalog environment. Deprecation note: starting with JDF 1.4, use <i>GeneralID</i> .
<i>Class</i>	enumeration	Defines the Abstract Resource type. For details, see the sections that follow. <i>Class</i> MUST be specified in the Resource root, MUST NOT be specified in a Resource leaf and SHOULD be specified in an inline Resource Subelement. Values are: <i>Consumable</i> <i>Handling</i> <i>Implementation</i> <i>Intent</i> <i>Parameter</i> <i>Placeholder</i> <i>Quantity</i>
<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> .

Table 3-10: Abstract Resource Element (Section 2 of 4)

Name	Data Type	Description
<p><i>PartUsage</i> = "Explicit"</p> <p>New in JDF 1.1</p> <p>Modified in JDF 1.3</p>	enumeration	<p>Description of the interpretation of Partitions.</p> <p><i>PartUsage</i> MUST NOT be specified outside of the root of a Resource. For details on <i>PartUsage</i> and Partitioning, see Section 3.10.7.4, Implicit, Sparse and Explicit PartUsage in Partitioned Resources.</p> <p>Values are:</p> <p><i>Explicit</i> – Require explicit Partition matches. All referenced Partitions referenced in <i>Part</i> MUST exist, otherwise it is an error.</p> <p><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.</p> <p><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</p> <p>Modification note: <i>PartUsage</i> was moved to this table from Table 3-27 on page 103 in JDF 1.2.</p>
<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 151.
<p><i>PipeProtocol</i>?</p> <p>New in JDF 1.2</p>	NMTOKEN	<p>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.</p> <p>Values include:</p> <p><i>Internal</i> – Internal or virtual pipe used within a Combined Process.</p> <p><i>JMF</i> – JMF-based PipePush / PipePull Messages.</p> <p><i>None</i> – No pipe support.</p>
<p><i>PipeURL</i>?</p> <p>New in JDF 1.2</p>	URL	<p>Pipe request URL. Dynamic pipe requests to this Resource SHOULD be made to this URL. Note that this URL is only used for initiating pipe requests. Responses to a pipe request are issued to the URL that is defined in the PipePush or PipePull Message. For details on using <i>PipeURL</i>, see Section 4.3.3, Overlapping Processing Using Pipes.</p> <p>Note: in most cases this is the URL of the Controller of the <i>other end</i> 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.</p>
<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.

Table 3-10: Abstract Resource Element (Section 3 of 4)

Name	Data Type	Description
<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 <i>SpawnID</i> values. This is used as a reference count for how often the Resource has been spawned.
<i>SpawnStatus</i> = “ <i>NotSpawned</i> ”	enumeration	<p>The spawn status of a Resource indicates whether or not a Resource has been spawned, and under what circumstances. The <i>SpawnStatus</i> of a Resource that has <i>ResourceRef</i> Elements is defined as the maximum <i>SpawnStatus</i> (whose values are ordered) of all recursively linked Resources.</p> <p>Value are ordered from lowest to highest</p> <p>Values are:</p> <p><i>NotSpawned</i> — Indicates that the Resource has not been copied to another Process.</p> <p><i>SpawnedRO</i> – Indicates that the Resource has been copied to another Process where it cannot be modified. The “RO” stands for read-only.</p> <p><i>SpawnedRW</i> – Indicates that the Resource has been copied to another Process where it can be modified. The “RW” stands for read/write.</p>

Table 3-10: Abstract Resource Element (Section 4 of 4)

Name	Data Type	Description
<p><i>Status</i> Modified in JDF 1.2</p>	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 Sub-element 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.</p> <p>The values are ordered from lowest to highest</p> <p>Values 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 <i>Approval</i> 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 3.8.7, 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 ResourceRef from a Parameter Resource to commence execution.</p> <p><i>Available</i> – Indicates that the whole Resource is available for usage.</p>
<p><i>UpdateID?</i> New in JDF 1.1 Deprecated in JDF 1.3</p>	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>
<p>SourceResource * New in JDF 1.3</p>	element	<p>List of Resources that were or SHOULD be taken into account to populate this Resource.</p>
<p>QualityControlResult * New in JDF 1.2</p>	refelement	<p>Results of quality measurements which were performed during or after the production of this Resource.</p>

3.8.3.1 Element: SourceResource

Table 3-11: SourceResource Element

Name	Data Type	Description
Resource	refelement	Reference to Resources that were or SHOULD be taken into account to populate this Resource. Resource is an Abstract Element that MAY reference either Process Resources or Intent 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.

3.8.4 Structure Diagram

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.

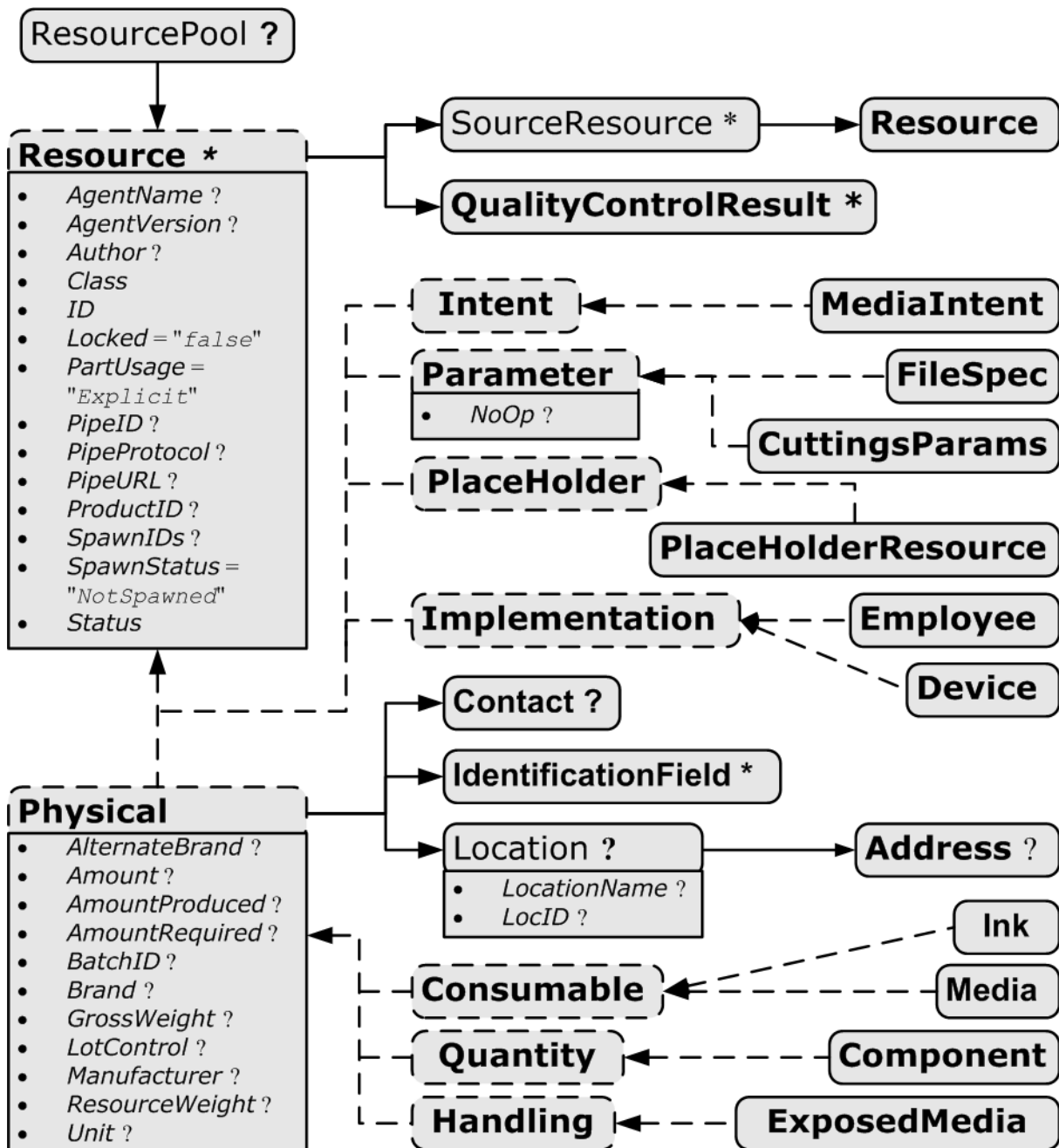



Figure 3-5: Abstract Resource Element – a diagram of its structure

3.8.5 Resource Classes

The following sections describe the functions of each of the seven values of the *Class* Attribute. All Resources fall into one of these Classes. In Section 7, Resources, the Class of each Resource is indicated in the Resource Properties subheading.



Parameter & Intent Resources

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

3.8.5.1 Parameter Resource

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

3.8.5.1.1 Abstract Parameter Resource

Table 3-12: 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).

3.8.5.2 Intent Resource

Intent Resources define the details of products to be produced without defining the Process to produce them. In addition, they provide structures to define sets of allowable options and to match these selections with prices. The details of all 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.

3.8.5.3 Implementation Resource

Implementation Resources define the Devices and operators that execute a given Node. Only two Implementation Resource types are defined: **Employee** (see Section 7.2.71, 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 3.9.6, ImplementationLink.

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

3.8.5.3.1 ImplementationResource

When an Implementation Resource is a Subelement, it is called an ImplementationResource.

3.8.5.4 Consumable Resource

A Consumable Resource is consumed during a Process. Examples include **Ink** and **Media**. Consumable Resources are the unmodified inputs in a Process chain. A Consumable Resource is a Physical Resource and inherits the contents of the Abstract Physical Resource Element.

3.8.5.5 Quantity Resource

A Quantity Resource has been created by a Process from either a Consumable Resource or an earlier Quantity Resource. For example, printed Sheets are cut and a pile of cut blocks is created. A **Component Resource** is an example of a Quantity Resource. A Quantity Resource is a Physical Resource and inherits the contents of the Abstract Physical Resource Element.

3.8.5.6 Handling Resource

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

3.8.5.7 Physical Resource

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

3.8.5.7.1 PhysicalResource

When a Physical Resource is a Subelement, it is called a PhysicalResource.

3.8.5.7.2 Abstract Physical Resource

Table 3-13, "Abstract Physical Resource Element," on page 70, 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.



**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 3-13: 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 <i>ResourceLink</i> Element. For details on amount handling, see Section 3.10.4, <i>Resource Amount</i> . For the unit of measurement, see the <i>Unit</i> Attribute below.
<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 <i>ResourceLink</i> Elements of leaf JDF Nodes with <i>Status</i> = " <i>Completed</i> " that reference this Resource. For the unit of measurement, see the <i>Unit</i> Attribute below.
<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 <i>ResourceLink</i> 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 <i>ResourceLink</i> Elements that produce this Resource. For the unit of measurement, see the <i>Unit</i> Attribute below.
<i>BatchID?</i>	string	ID of a specific batch of the Physical Resource

Table 3-13: Abstract Physical Resource Element (Section 2 of 2)

Name	Data Type	Description
<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> .
<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> , <i>AmountProduced</i> and <i>AmountRequired</i> . Values include those from: Table 1-7, “Units used in JDF”. Note: that it is strongly discouraged to specify units other than those that are defined in Table 1-7, “Units used in JDF”.
<i>Contact</i> ?	refelement	If this Element is specified, it describes the owner of the Resource.
<i>IdentificationField</i> * New in JDF 1.1	refelement	If this Element is specified, a bar code or label is associated with this Physical Resource.
<i>Location</i> ?	element	Description of details of the location of this Resource. Note, in order to describe multiple locations, Resources MAY be Partitioned by the <i>Location</i> Partition Key as described in Section 3.10.5, Description of Partitioned Resources.

3.8.5.7.3 Element: Location**Table 3-14: 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. Values include those from: Table C-21, “Input Tray and Output Bin Names,” on page 889. Note: the specified values are for printer locations.
<i>LocID</i> ?	string	Location identifier, (e.g., within a warehouse system).
<i>Address</i> ?	refelement	Address of the storage facility. For more information, see Section 7.2.2, Address.

3.8.5.8 Placeholder Resource

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.

3.8.6 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 Press. For further details on independent Job merging, see Section 4.4.5, Case 5: Spawning and Merging of Independent Jobs.

It is good practice to put Resources into the closest Node that references the Resource. For example, the **RenderingParams** Resource SHOULD be located in the *Rendering* Node, unless it is used by multiple *Rendering* Processes, in which case it SHOULD be located in the Process Group Node that contains the *Rendering* Process Nodes. Resources that link more than one Node SHOULD be placed in the parent Node of the siblings that are linked by the Resource.

A Process that needs additional detailed Process information specifying the creation of a Resource MUST infer this information by explicitly linking to the appropriate Parameter Resource.

3.8.7 Pipe Resources

A Pipe describes the Resource dependency in which a Process begins to consume a Resource while it is being produced by another Process (e.g., stacking components while they are being printed) or consuming a data stream while it is being written by an upstream Process. Note that defining a Pipe Resource does not automatically set up communication between Processes. The Controllers/Agents that execute the Process 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.

3.8.8 ResourceUpdate

[New in JDF 1.1.](#)

[Deprecated in JDF 1.3](#)

For details of the deprecated ResourceUpdate Element, see “ResourceUpdate” on page 1005.

3.9 ResourceLinkPool and ResourceLink

3.9.1 ResourceLinkPool

Each JDF Node contains a `ResourceLinkPool` Element that in turn contains all of the `ResourceLink` Elements that link the Node to the Resources it uses. The following table shows the contents of a `ResourceLinkPool` Element.

Table 3-15: ResourceLinkPool Element

Name	Data Type	Description
<code>ResourceLink *</code>	element	List of <code>ResourceLink</code> Elements. A <code>ResourceLink</code> Element is Abstract and is a placeholder for a concrete <code>ResourceLink</code> Element, such as <code>MediaLink</code> .

3.9.2 ResourceLink

`ResourceLink` Elements describe what Resources a Node uses, and how it uses them. They also define whether the Resources are inputs or outputs. These inputs and outputs provide conceptual links between the execution Elements of JDF Nodes. Outputs of one Node can in turn become inputs in another Node, and a given Node **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.

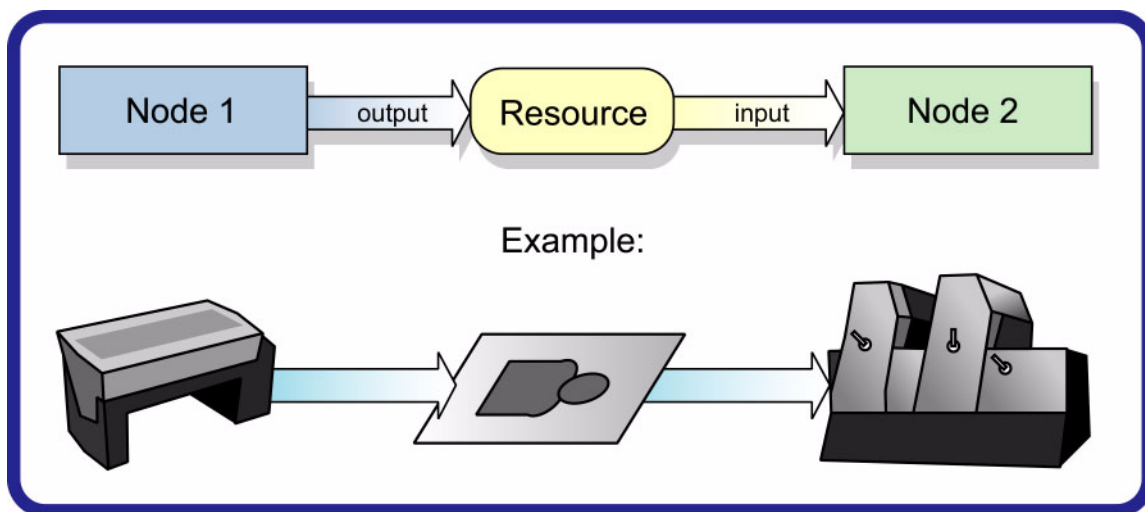


Figure 3-6: Nodes linked by a Resource

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

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

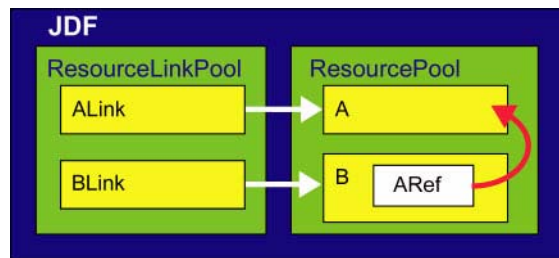


Figure 3-7: ResourceLink Elements and ResourceRef Elements

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

ResourceLink Elements MAY also contain Attributes to select a part of a Resource, such as a single separation. A detailed description of Resource Partitioning is given in Section 3.10.5, Description of Partitioned Resources.

Because implementation ResourceLink Elements define the usage of a specific Device during the course of a Job, situations can arise where that Resource is not needed during the whole processing time. For instance, a forklift that only has to transport the completed components need not be available during the entire Process run, only during the times when it is needed. This means that, contrary to the general rule that all Resources 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*.

ProcessGroup and Product Intent Nodes can be defined without the knowledge of the individual Process Nodes that define a specific workflow. In this case, these intermediate Nodes will contain ResourceLink Elements that link the appropriate Resources. For example, a prepress Node might be defined that produces a set of plates. When the Processes for creating the plates are defined in detail, the Agent that writes the Nodes might remove the ResourceLink Elements from the intermediate Node. Removing the ResourceLink specifies that the intermediate Node can execute; (i.e., it can be sent to the appropriate Controller or department), even though the specific Resources are not yet available. If the ResourceLink Elements are not removed, the intermediate Node cannot execute until the linked Input Resources become available.

ResourceLink Elements MAY be used for Process control. For example, if a proof Input Resource is needed for a print Process, a print run can commence only when the proof is signed. The JDF format specification also includes a complete specification of how Resources are managed when JDF tickets are spawned and merged.

In some cases, determining whether to store information in an input or an Output Resource can be difficult, as the distinction can be ambiguous. For example, is the definition of the color of a separation in the RIP Process a property of the output separation or a parameter that describes the RIP Process? In order to reduce this ambiguity, the following rules have been defined for input and Output Resources of Processes (see 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 Resources and Handling Resources are used both as Input Resources and Output Resources. Their usage is defined by the “natural” Process usage. For example, a printing plate is described as a Resource that is the output of a Process and the input of a Process.
- Processed material is exchanged from Node to Node using the **Component** Resource. Product Intent Nodes also create **Component** Output Resources.

- Every detailed Process description **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.

3.9.3 Structure Diagram

Figure Section 3-8, Abstract ResourceLink Element – a diagram of its structure shows the Abstract types derived from the ResourceLink type.

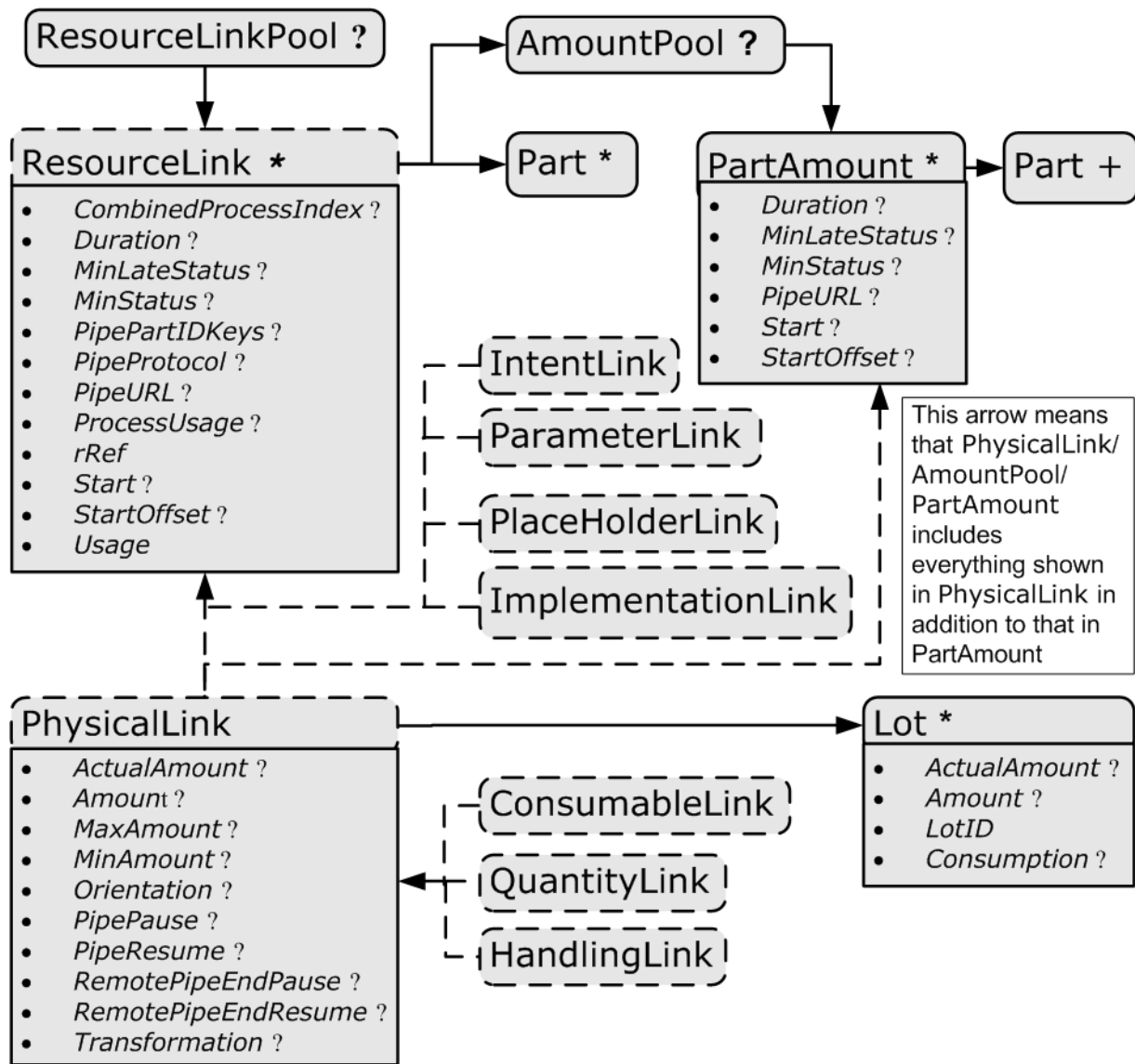


Figure 3-8: Abstract ResourceLink Element – a diagram of its structure

3.9.4 Abstract ResourceLink

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

Table 3-16: Abstract ResourceLink Element (Section 1 of 3)

Name	Data Type	Description
<i>CombinedProcessIndex</i> ? New in JDF 1.1	IntegerList	Combined Process Nodes and Process Group 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 ResourceLink in a Combined Process Nodes or Process Group Node belongs. Multiple entries in <i>CombinedProcessIndex</i> specify that the ResourceLink is used by the respective multiple Processes in the Combined Process Node. It MUST be specified when multiple Resources of the same name and ResourceLink/@Usage are specified in one JDF Node. If <i>CombinedProcessIndex</i> is not specified, even though multiple Processes in the Combined Process Node or Process Group Node MAY link to the Resource, the ResourceLink applies to all of these Processes.
<i>CombinedProcessType</i> ? Deprecated in JDF 1.1	NMTOKEN	Combined Process Nodes contain Input Resources from multiple Process Nodes. The <i>CombinedProcessType</i> Attribute specifies the name individual Process to which a ResourceLink in a Combined Process Node belongs. It MUST match one of the entries in the <i>Types</i> Attribute of the Node. Deprecation note: 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>Duration</i> ? Modified in JDF 1.4	duration	Estimated duration during which the Resource will be used. Modification note: starting with JDF 1.4, this Attribute is moved from Table 3-20, "Abstract ImplementationLink or //AmountPool/PartAmount Element".
<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 <i>NodeInfo/@LastStart</i> or implied by <i>NodeInfo/@LastEnd</i> is approaching. Default value is from: <i>@MinStatus</i> . Values are from: Resource/@Status.
<i>MinStatus</i> ? New in JDF 1.3	enumeration	Minimum value of Resource/@Status for the execution of this Node to commence. Default value is: " <i>Available</i> " if <i>@Usage</i> = " <i>Input</i> " and " <i>Unavailable</i> " if <i>@Usage</i> = " <i>Output</i> ". Values are from: Resource/@Status.

Table 3-16: Abstract ResourceLink Element (Section 2 of 3)

Name	Data Type	Description
<i>PipePartIDKeys</i> ?	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. <i>Resource/@PartIDKeys</i> = “<i>SheetName Side Separation</i>”) and if the <i>ResourceLink/@PipePartIDKeys</i> = “<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>.</p> <p>Default value is from: the implied or explicit value of <i>@PipePartIDKeys</i> of the referenced Resource.</p> <p>Values are from: <i>Resource/@PartIDKeys</i>.</p>
<i>PipeProtocol</i> ? New in JDF 1.1 Modified in JDF 1.2	NMTOKEN	<p>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.</p> <p>Default value is: <i>JMF</i> (if <i>PipeURL</i> is specified); otherwise referenced <i>Resource/@PipeProtocol</i>.</p> <p>Values include:</p> <p><i>Internal</i> – Internal or virtual pipe used within a Combined Process. New in JDF 1.2</p> <p><i>JMF</i> – JMF-based PipePush / PipePull Messages.</p> <p><i>None</i> – No pipe support.</p>
<i>PipeURL</i> ? Modified in JDF 1.2	URL	<p>Pipe request URL. Dynamic pipe requests from this end of a pipe SHOULD be made <i>to</i> this URL. In most cases this URL is the URL of the Controller of the <i>other end</i> 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.</p> <p>Default value is: referenced <i>Resource/@PipeURL</i></p> <p>Note: 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.</p>
<i>ProcessUsage</i> ? Modified in JDF 1.4	string	<p>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 Book-Block—are used in <i>CoverApplication</i>.</p> <p>Values include those from: Table 3-17, “ProcessUsage Attribute Values,” on page 79</p> <p>Values include those from: ICS documents. New in JDF 1.4</p> <p>Note: the values of <i>ProcessUsage</i> can be derived from the appropriate Process descriptions in Section 6, Processes. Section 6.1, Process Template defines the parenthesized notation for denoting the value of <i>ProcessUsage</i> (e.g., Component (Cover)).</p>
<i>rRef</i>	IDREF	Link to the target Resource.

Table 3-16: Abstract ResourceLink Element (Section 3 of 3)

Name	Data Type	Description
<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>Start?</i> Modified in JDF 1.4	dateTime	Time and date when the usage of the Resource starts. Modification note: starting with JDF 1.4, this Attribute is moved from Table 3-20, “Abstract ImplementationLink or //AmountPool/PartAmount Element”.
<i>StartOffset?</i> Modified in JDF 1.4	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. Modification note: starting with JDF 1.4, this Attribute is moved from Table 3-20, “Abstract ImplementationLink or //AmountPool/PartAmount Element”.
<i>Usage</i>	enumeration	Resource usage within this JDF Node. Values are: <i>Input</i> – The Resource is an input. <i>Output</i> – The Resource is an output.
AmountPool ? New in JDF 1.1 Modified in JDF 1.2	element	Definition of partial amounts and pipe parameters for this ResourceLink. The allowed contents of the AmountPool are described for the various Subclasses of ResourceLink in the sections below. If AmountPool is specified, ResourceLink MUST NOT contain any of <i>Amount</i> , <i>ActualAmount</i> , <i>MaxAmount</i> or <i>MinAmount</i>
Part *	element	The Part Elements identify the parts of a Partitioned Resource that are referenced by the ResourceLink. The structure of the Part Element is defined in Table 3-28, “Part Element,” on page 104. For details on Partitioned Resources, see Section 3.10.5, Description of Partitioned Resources.

— Attribute: ProcessUsage

Table 3-17: ProcessUsage Attribute Values (Section 1 of 2)

Value	Description
<i>Accepted</i>	Used for Resource in an Output Resource of <i>Approval</i>
<i>Application</i>	Used for Component in an Input Resource of <i>BoxFolding</i>
<i>BackEndSheet</i>	Used for Component in an Input Resource of <i>EndSheetGluing</i>
<i>Book</i>	Used for Component in an Input Resource of <i>Jacketing</i>
<i>BookBlock</i>	Used for Component in an Input Resource of <i>ChannelBinding</i> , <i>EndSheetGluing</i> and <i>RingBinding</i>
<i>Box</i>	Used for Component in an Input Resource of <i>BoxPacking</i>
<i>Case</i>	Used for Component in an Input Resource of <i>CasingIn</i>
<i>Child</i>	Used for Component in an Input Resource of <i>Inserting</i>
<i>Cover</i>	Used for Component in an Input Resource of <i>ChannelBinding</i> and <i>CoverApplication</i>
<i>CoverBoard</i>	Used for Media in an Input Resource of <i>CaseMaking</i>

Table 3-17: ProcessUsage Attribute Values (Section 2 of 2)

Value	Description
<i>CoverMaterial</i>	Used for Component and Media in an Input Resource of <i>CaseMaking</i>
<i>Cylinder</i>	Used for ExposedMedia in an Input Resource of <i>ConventionalPrinting</i>
<i>Document</i>	Used for RunList in an Input Resource of <i>Imposition</i> , <i>LayoutPreparation</i> and <i>Stripping</i> and used for RunList in an Output Resource of <i>Stripping</i>
<i>FrontEndSheet</i>	Used for Component in an Input Resource of <i>EndSheetGluing</i>
<i>Good</i>	Used for Component in an Output Resource of <i>ConventionalPrinting</i> and <i>DigitalPrinting</i>
<i>Input</i>	Used for Component in an Input Resource of <i>ConventionalPrinting</i> and <i>DigitalPrinting</i>
<i>Jacket</i>	Used for Component in an Input Resource of <i>Jacketing</i>
<i>Label</i>	Used for Component in an Input Resource of <i>Labeling</i>
<i>Marks</i>	Used for RunList in an Input Resource of <i>Imposition</i> , <i>LayoutPreparation</i> and <i>Tiling</i> , and used for RunList in an Output Resource of <i>LayoutPreparation</i> and <i>Stripping</i>
<i>Mother</i>	Used for Component in an Input Resource of <i>Inserting</i>
<i>Plate</i>	Used for ExposedMedia in an Input Resource of <i>ConventionalPrinting</i>
<i>Proof</i>	Used for Component in an Input Resource of <i>ConventionalPrinting</i> and <i>DigitalPrinting</i> , and used for ExposedMedia in an Input Resource of <i>ConventionalPrinting</i>
<i>Rejected</i>	Used for Resource in an Output Resource of <i>Approval</i>
<i>RingBinder</i>	Used for Component in an Input Resource of <i>RingBinding</i>
<i>SpineBoard</i>	Used for Media in an Input Resource of <i>CaseMaking</i>
<i>Surface</i>	Used for RunList in an Input Resource of <i>Tiling</i>
<i>Tie</i>	Used for Media in an Input Resource of <i>BoxPacking</i>
<i>Underlay</i>	Used for Media in an Input Resource of <i>BoxPacking</i>
<i>Waste</i>	Used for Component in an Output Resource of <i>ConventionalPrinting</i> and <i>DigitalPrinting</i>

3.9.4.1 AmountPool and PartAmount

[New in JDF 1.1](#)

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

3.9.4.1.1 AmountPool

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

Table 3-18: AmountPool Element

Name	Data Type	Description
PartAmount *	element	Element that defines the amounts and pipe parameters for a Partitioned Resource. The contents of a PartAmount depends on the type of the ResourceLink.

3.9.4.1.2 PartAmount

The following table lists the generic contents of a PartAmount Element. Further parameters of the PartAmount are described in the respective sections below (Table 3-20, “Abstract ImplementationLink or //AmountPool/PartAmount Element,” on page 82 and Table 3-21, “Abstract PhysicalLink or //AmountPool/PartAmount Element,” on page 83). Note that PartAmount inherits values from its parent ResourceLink.

Table 3-19: PartAmount Element (Section 1 of 2)

Name	Data Type	Description
DraftOK? 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.
Duration? Modified in JDF 1.4	duration	Estimated duration during which the Resource will be used. Modification note: starting with JDF 1.4, this Attribute is moved from Table 3-20, “Abstract ImplementationLink or //AmountPool/PartAmount Element”.
MinLateStatus? New in JDF 1.3	enumeration	Minimum value of Resource/@Status for the execution of this Node to commence when deadlines are endangered, i.e., when the time defined by NodeInfo/@LastStart or implied by NodeInfo/@LastEnd is approaching. Default value is from: @MinStatus. Values are from: Resource/@Status.
MinStatus? New in JDF 1.3	enumeration	Minimum value of Resource/@Status for the execution of this Node to commence. Default value is: “Available” if @Usage = “Input” and “Unavailable” if @Usage = “Output”. Values are from: Resource/@Status.
PipeURL?	URL	Pipe request URL for this Partition. Dynamic pipe requests from this end of a pipe SHOULD be made to this URL. Note that this URL is only used for initiating pipe requests. Responses to a pipe request are issued to the URL that is defined in the PipePush or PipePull Message. For details on using <i>PipeURL</i> , see Section 4.3.3, Overlapping Processing Using Pipes. Note: in most cases this is the URL of the Controller of the <i>other end</i> 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.
Start? Modified in JDF 1.4	dateTime	Time and date when the usage of the Resource starts. Modification note: starting with JDF 1.4, this Attribute is moved from Table 3-20, “Abstract ImplementationLink or //AmountPool/PartAmount Element”.
StartOffset? Modified in JDF 1.4	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. Modification note: starting with JDF 1.4, this Attribute is moved from Table 3-20, “Abstract ImplementationLink or //AmountPool/PartAmount Element”.

Table 3-19: PartAmount Element (Section 2 of 2)

Name	Data Type	Description
Part + Modified in JDF 1.3	element	Specifies the selected parts that the PartAmount is valid for. The granularity of Part MUST be at least that of a leaf Partition of the Resource. For instance, a Component MAY be Partitioned by <i>SheetName</i> and PartAmount could refer to <i>SheetName</i> and <i>Condition</i> . Multiple Part Elements specify that the referenced Elements have been Processed in one step, for instance two separations on a press run of a two color press.

3.9.5 ParameterLink

A Parameter Resource is linked by an instance of a ParameterLink Element. This Element contains no further Attributes or Elements besides those found in the Abstract ResourceLink Element.

3.9.6 ImplementationLink

An Implementation Resource is linked by an instance of an ImplementationLink Element.

3.9.6.1 Abstract ImplementationLink

The Attributes in Table 3-20 can occur in either an Abstract ImplementationLink or //AmountPool/PartAmount Element that references an Implementation Resource.

Modification note: starting with JDF 1.4, all existing Attributes from Table 3-20, “Abstract ImplementationLink or //AmountPool/PartAmount Element” are moved to Table 3-16, “Abstract ResourceLink Element” and Table 3-19, “PartAmount Element”.

Table 3-20: Abstract ImplementationLink or //AmountPool/PartAmount Element

Name	Data Type	Description
<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.

Example 3-5: EmployeeLink As ImplementationLink

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

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Employee Class="Implementation" ID="L1" Status="Available"
      PersonalID="007">
      <Person FamilyName="Smith" JobTitle="Press Operator"/>
    </Employee>
  </ResourcePool>
  <ResourceLinkPool>
    <EmployeeLink Usage="Input" rRef="L1"/>
  </ResourceLinkPool>
</JDF>
```

3.9.7 ConsumableLink

A Consumable Resource is linked by an instance of an ConsumableLink Element, which inherits the contents of the Abstract PhysicalLink Element.

3.9.8 HandlingLink

A Handling Resource is linked by an instance of an HandlingLink Element, which inherits the contents of the Abstract PhysicalLink Element.

3.9.9 QuantityLink

A Quantity Resource is linked by an instance of an QuantityLink Element, which inherits the contents of the Abstract PhysicalLink Element.

3.9.10 PhysicalLink

Just as a Physical Resource inherits the contents of the Abstract Resource Element, a PhysicalLink inherits the contents of the Abstract ResourceLink Element.

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**.

3.9.10.1 Abstract PhysicalLink

The Table 3-21 describes additional Attributes for a PhysicalLink Element. The Attributes in Table 3-21 can occur in either an Abstract PhysicalLink or //AmountPool/PartAmount Element that references a Physical Resource.

Table 3-21: 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 3.10.4, 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 3.10.4, 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> .
<i>MinAmount</i> ? 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> .
<i>Orientation</i> ? 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.

Table 3-21: Abstract PhysicalLink or //AmountPool/PartAmount Element (Section 2 of 2)

Name	Data Type	Description
<i>PipePause</i> ?	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.
<i>PipeResume</i> ?	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.
<i>RemotePipeEndPause</i> ?	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.
<i>RemotePipeEndResume</i> ?	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.
<i>Transformation</i> ? 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 3.9.10.2, Identification of Physical Resources for details. For Resources that are solely identified by <i>ProductID</i> , Lot Element(s) NEED NOT be specified.

Example 3-6: PartAmount

The following example shows an InkLink with an AmountPool/.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <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>
      </AmountPool>
    </InkLink>
  </ResourceLinkPool>
</JDF>
```

```

    </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>
</JDF>

```

3.9.10.2 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.

3.9.10.2.1 Element: Lot

In the case of identifying Resources that are planned to be consumed, Lot Elements for each unique Resource lot are placed in the associated ResourceLink or a PartAmount Element within the ConsumableLink, See Table 3-21 on page 83.

Table 3-22: 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.
<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.

Table 3-22: Lot Element (Section 2 of 2)

Name	Data Type	Description
<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. 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. Values are: <i>Full</i> <i>Partial</i>

In the case of identifying Resources after they have been consumed, Lot Elements are specified within the first ResourceLink in the ResourceAudit, or in the AmountPool that can appear inside the ResourceLink. See Section 3.11.4.8, ResourceAudit for the structure of the ResourceAudit Element.

Example 3-7: MediaLink with Lot

The following is an example of a ResourceLink used to report that a substitute Resource was used:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" Type="ConventionalPrinting"
  Status="Completed" ID="ID100" JobPartID="ID345" Version="1.4">
  <ResourcePool>
    <Media ID="RI007" Class="Consumable" ProductID="3002" Status="Unavailable"
      Brand="Coated Roll Stock" Dimension="2520 8640000"
      MediaType="Paper" Thickness="36"/>
    <ConventionalPrintingParams ID="RI008" Class="Parameter" Status="Available"/>
    <Component ID="RI009" Class="Quantity" Status="Unavailable"
      ComponentType="Sheet"/>
  </ResourcePool>
  <ResourceLinkPool>
    <MediaLink rRef="RI007" Amount="9800" ActualAmount="9703" Usage="Input">
      <Lot ActualAmount="5250" Consumption="Full"
        LotID="LN1040788312RN2005091-04"/>
      <Lot ActualAmount="4453" Consumption="Partial"
        LotID="LN1040788339RN2005091-01"/>
    </MediaLink>
    <ConventionalPrintingParamsLink rRef="RI008" Usage="Input"/>
    <ComponentLink rRef="RI009" Usage="Output"/>
  </ResourceLinkPool>
</JDF>
```

3.9.11 PlaceholderLink

A Placeholder Resource is linked by a PlaceholderLink Element. A PlaceholderLink Element, 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.

3.9.12 IntentLink

An Intent Resource is linked by an instance of an IntentLink Element. It has no additional parameters.

3.10 ResourcePool and ResourceLinkPool – Deep Structure

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

3.10.1 ResourceElement – Subelement of a Resource

3.10.1.1 ResourceElement

A `ResourceElement` is always a Subelement of a `Resource` or Subelement of a `JMF Message`

3.10.1.2 Abstract ResourceElement

An Abstract `ResourceElement` is defined in the Table 3-23 below. A `ResourceElement` does not inherit from the Abstract `Resource`. Examples of `ResourceElement` Resources are `SeparationSpec` and `MISDetails`.

Table 3-23: Abstract ResourceElement

Name	Data Type	Description
<i>ID?</i> Deprecated in JDF 1.2	ID	Unique identifier of a <code>Resource Element</code> . In JDF 1.2 and beyond, an Element that is not a direct child of a <code>ResourcePool</code> SHOULD NOT contain an <i>ID</i> . This <i>ID</i> MUST NOT be referenced by <code>ResourceRef/@rRef</code> or <code>ResourceLink/@rRef</code> because a <code>ResourceRef</code> or <code>ResourceLink</code> Element is able to reference a <code>Resource</code> only if it is a direct child of a <code>ResourcePool</code> .

3.10.2 ResourceRef – Element for Inter-Resource Linking and refElement

3.10.2.1 ResourceRef

In some cases, it is necessary to reference a `Resource Element` directly from another Element in order to reuse information. Such a reference is a `ResourceRef` Element. A `ResourceRef` Element's name is generated by appending the string "Ref" to the Element's name. Candidate Elements for inter-Resource linking have a data type of **refelement** in the content description tables of this chapter and Section 7, `Resources`. A data type of **refelement** allows either a `ResourceRef` or an inline `Resource Element`. In the latter case, the `Resource Element` inherits Attributes and Elements from the Abstract `Resource` and (where appropriate) from the Abstract `Parameter Resource` or the Abstract `Physical Resource`. Note that some Attributes and Elements in these Abstract Elements have rules for inline `Resource` Subelements that differ from the rules for a `Resource` root.

3.10.2.2 Abstract ResourceRef

The following table defines the Attributes of the Abstract `ResourceRef` Element (see also Figure 3-5 and `ResourceElement` in Table 3-10, "Abstract Resource Element," on page 63).

The `Part` Element in a `ResourceRef` defines the part of the target that this `ResourceRef` references. If both the `Resource` that contains `ResourceRef` Element and the target `Resource` are Partitioned, the `ResourceRef` does *not* implicitly reference the part of the target with the same Partitioning Attributes, but rather the parts of the target `Resource` that are explicitly specified by the `Part` Element within the `ResourceRef`.

When a `ResourceRef` references a Partitioned `Resource Node` that is not a `Resource` leaf, the children of the referenced `Resource` are ignored. See Example 3-8 and Example 3-9 for an illustration of this equivalence. Otherwise, the referenced structure would be a Partitioned Element and thus invalid when inlined.; see Example 3-10

Table 3-24: Abstract ResourceRef Element (Section 1 of 2)

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

Table 3-24: Abstract ResourceRef Element (Section 2 of 2)

Name	Data Type	Description
rSubRef? Deprecated in JDF 1.2	IDREF	Reference to a Subelement of the Resource. In JDF 1.2 and beyond, a ResourceRef Element is able to reference a Resource only if it is a direct child of a ResourcePool
Part ? New in JDF 1.1	element	Definition of the Partition that this ResourceRef references.

Example 3-8: MediaRef to Partitioned Media

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

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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>
  </ResourcePool>
  <ResourceLinkPool>
    <LayoutLink Usage="Input" rRef="Sheet"/>
  </ResourceLinkPool>
</JDF>
```

Example 3-9: Equivalent Inline Media

Media is inline in **Layout**. This is equivalent to the preceding Example 3-8 with MediaRef:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Layout Class="Parameter" ID="Sheet" Status="Available">
      <Media Dimension="72 72">
        <Comment Name="foo">bar</Comment>
      </Media>
    </Layout>
  </ResourcePool>
  <ResourceLinkPool>
    <LayoutLink Usage="Input" rRef="Sheet"/>
  </ResourceLinkPool>
</JDF>
```

Example 3-10: Invalid Inline Partitioned Media

This example takes the **Media** from Example 3-8 and make it be inline in **Layout**. The result is an invalid Partition:

```
<ResourcePool>
  <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>
</ResourcePool>
```

3.10.2.3 ResourceRef Elements in the AncestorPool/Ancestor Element

ResourceRef Elements MAY also occur in the AmountPool/PartAmount Element of a JDF Node. Resources that are referenced MUST reside in a ResourcePool. The restrictions on locations of Resource Elements described in Section 3.8.6, Position of Resources within JDF Nodes that apply to ResourceLink Elements similarly apply to ResourceRef Elements.

3.10.2.4 Status of a Resource that Contains an rRef Reference

The *Status* of a Resource that contains an *rRef* Attribute is defined by the lowest *Status* of all recursively referenced Resources. The ordering is defined in Table 3-10, “Abstract Resource Element,” on page 63:

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

3.10.2.5 Alignment of ResourceLink and ResourceRef

[New in JDF 1.1A](#)

ResourceRef Elements MUST NOT contain any of the Attributes and Elements that are specified in the ResourceLink as defined in Section 3.9, ResourceLinkPool and ResourceLink. The value of these properties is implied from the value of the properties for the appropriate part in the AmountPool of the ResourceLink.

Example 3-11: MediaLink and MediaRef

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

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

```

</ResourcePool>
<ResourceLinkPool>
  <MediaLink Usage="Input" rRef="r0006">
    <!-- the AmountPool contains the ResourceLink partition details -->
    <AmountPool>
      <PartAmount Orientation="Flip180">
        <Part RunIndex="0 -1"/>
      </PartAmount>
      <PartAmount Orientation="Rotate0">
        <Part RunIndex="1 ~ -2"/>
      </PartAmount>
    </AmountPool>
  </MediaLink>
  <DigitalPrintingParamsLink Usage="Input" rRef="r0007"/>
  <RunListLink Usage="Input" rRef="r0008"/>
  <ComponentLink Usage="Output" rRef="c0008"/>
</ResourceLinkPool>
</JDF>

```

3.10.3 Set of Resources and Partitioned Subsets Thereof

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

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

Resource Elements and ResourceLink Elements have OPTIONAL Attributes that enable an Agent to specify an explicit part of a structured Resource. There are two ways to reference a subset of a Resource. The first is by quantity, (i.e., by specifying an Amount in a ResourceLink that is less than the Resource's Amount.) The second is to select certain parts of a Partitioned Resource by supplying a filtering Part Element in the ResourceLink.

3.10.4 Resource Amount

Yet another flexible feature of Resources is that they can be only partially consumed. For example, in a scenario in which various versions of a product share identical parts—such as versioned books that all have the same cover—each version will only use as many copies of the cover as it needs to fulfill its Job requirement, even though all of the covers can be printed in one step for all versions. This feature is specified in the *Amount* Attribute of the ResourceLink Elements and allows multiple JDF Nodes to share Resources. It allows both the sharing of Output Resources (when a binding Process consumes identical Sheets from multiple press lines) and the sharing of Input Resources (when the covers for multiple Jobs are identical and are all printed in one press run).

The *Amount* Attribute of a 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 3.10.5, Description of Partitioned Resources.

Note that for Resources which are the output of Processes, the *Amount* Attribute on the *ResourceLink* determines the quantity of the Resource to be produced. For example, in a *DigitalPrinting* Process that included a *RunList* as its input with 16 pages to be printed and a *ComponentLink* to its output, the *Amount* and *AmountProduced* Attributes would indicate the number of copies of those 16 pages that the Process would produce.

3.10.4.1 Evaluating and Updating Amount-Related Attributes in a Device

ResourceLink/@Amount specifies the planned amount whereas *ResourceLink/@ActualAmount* specifies the actual production amount. When a Device executes a JDF Node that consumes and produces 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 and additional production amount.

3.10.4.2 Specifying Amount for a Partially-Completed Process

[New in JDF 1.2](#)

A Process can be interrupted before the requested amount of output has been produced. When the Job is resent from the Controller to the Device, it MUST produce only the remaining *Amount*. The following figure shows the various Processes, Resources and *ResourceLink* Elements and their corresponding entries in Table 3-25 on page 92 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:

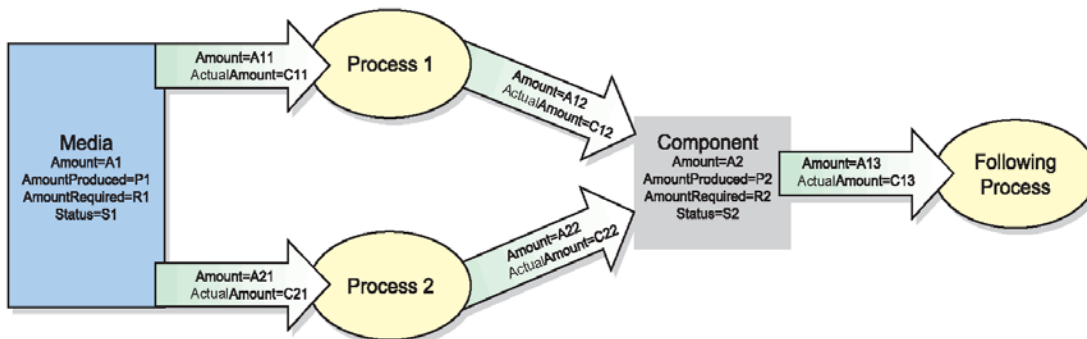


Figure 3-9: Amount handling

Table 3-25: 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					

Table 3-25: 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 following Process consumes 50,010 copies	390677 — 110000 Available	110000 109323 — —	100000 101234 — —	51224 101234 50000 Available	50000 50010
Additional Copy Request					
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

3.10.5 Description of Partitioned Resources

Printing workflows contain a number of Processes that are repeated over a potentially large number of individual files, Sheets, surfaces or separations. In order to define a Partitioned Resource in a concise manner without having to create a large number of individual Nodes and Resources, a set of Resources might be Partitioned by factoring them by one or more Attributes. The common Elements and defaults are placed in the parent Element while Partition-specific Attributes and overrides are placed in the child Elements. This saves space. Also, by providing a single parent ID for the Resources, it allows easy access to the entire Resource or iteration over each part.

To reference part of a Resource, a `ResourceLink` references the parent Resource and supplies a `Part` Element that contains an actual value for a Partition. The result is all the child Elements with matching Partition values, including common values and defaults from the parent Resource. If `PartUsage = "Implicit"`, the parent Attributes are returned if there is no matching Partition.

A Partitioned Resource MAY contain nested Elements, each with the same name as the Partitioned Resource root. The part-independent Resource Elements and Attributes are located in the root of the Resource, while the Partition-dependent Elements are located in the nested Elements. Thus one individual part is defined by the convolution of the Partition-independent Elements and Attributes with the Elements and Attributes contained in the appropriate nested Elements. The Attributes of nested part Elements are overwritten by the equivalent Attributes in descendant Elements.

Some Processes will enumerate a Resource in XML order and use its Partition Key values and actually set the values of those Partition Keys during its processing. Other Processes will treat the Resource as a random access Resource and look up leaf Nodes based on the current settings of `PartIDKeys` values. For example, the `RunList` Resource can be used by the `Imposition` Process to define key values (such as the `Run` Partition Key during consumption of the `RunList`, and the `Layout` Resource uses Partitioning to define a set of templates chosen based on the current content from the `RunList` being processed.

3.10.5.1 Subelements in Partitioned Resources

Subelements of a Partitioned Resource are inherited by a descendant Element if and only if no equivalent Subelements exist in the descendant Element. Subelements are completely replaced by those in descendant Elements even if cardinality of the Subelement allows multiple occurrences.

Example 3-12: Inheritance for Subelements of a Partitioned Resource

For example, the following `SeparationSpec` is two color duo-tone (only `Black` and `SpotGreen`) in the part with `PageNumber = "1"`. For additional examples and restrictions, see also "Position of PlacedObject Elements in Layout" on page 608 which contains Example 7-25 and Example 7-26.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <LayoutElement Class="Parameter" ID="ID1" PartIDKeys="PageNumber"
      Status="Available">
      <SeparationSpec Name="Cyan"/>
      <SeparationSpec Name="Magenta"/>
      <SeparationSpec Name="Yellow"/>
      <SeparationSpec Name="Black"/>
      <FileSpec/>
      <LayoutElement PageNumber="0"/>
      <LayoutElement PageNumber="1">
        <!-- These two SeparationSpec Elements completely replace the
          CMYK in the root
        -->
        <SeparationSpec Name="Black"/>
        <SeparationSpec Name="SpotGreen"/>
      </LayoutElement>
    </LayoutElement>
  </ResourcePool>
  <ResourceLinkPool>
    <LayoutElementLink Usage="Input" rRef="ID1"/>
  </ResourceLinkPool>
</JDF>
```



```
</ResourceLinkPool>
</JDF>
```

3.10.5.2 Amount in Partitioned Resources

[New in JDF 1.2](#)

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

Example 3-13: Partitioned ExposedMedia

The following example defines two sets of 12 plates for two Sheets with three surfaces. Each has a common brand Attribute called “Goopy”. 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*.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <ExposedMedia Amount="2" Brand="Goopy" Class="Handling" ID="L1"
      PartIDKeys="SheetName Side Separation" Status="Available">
      <Media Dimension="500 600" MediaType="Plate"/>
      <ExposedMedia SheetName="S1">
        <ExposedMedia Side="Front">
          <ExposedMedia ProductID="S1FCPlateJ42" Separation="Cyan"/>
          <ExposedMedia ProductID="S1FMPlateJ42" Separation="Magenta"/>
          <ExposedMedia ProductID="S1FYPlateJ42" Separation="Yellow"
            Status="Unavailable"/>
          <ExposedMedia ProductID="S1FKPlateJ42" Separation="Black"/>
        </ExposedMedia>
        <ExposedMedia Side="Back">
          <ExposedMedia ProductID="S1BCPlateJ42" Separation="Cyan"/>
          <ExposedMedia ProductID="S1BMPlateJ42" Separation="Magenta"/>
          <ExposedMedia ProductID="S1BYPlateJ42" Separation="Yellow"/>
          <ExposedMedia ProductID="S1BKPlateJ42" Separation="Black"/>
        </ExposedMedia>
      </ExposedMedia>
      <ExposedMedia SheetName="S2">
        <ExposedMedia Side="Front">
          <ExposedMedia ProductID="S2FCPlateJ42" Separation="Cyan"/>
          <ExposedMedia ProductID="S2FMPlateJ42" Separation="Magenta"/>
          <ExposedMedia ProductID="S2FYPlateJ42" Separation="Yellow"/>
          <ExposedMedia ProductID="S2FKPlateJ42" Separation="Black"/>
        </ExposedMedia>
      </ExposedMedia>
    </ResourcePool>
    <ResourceLinkPool>
      <ExposedMediaLink Usage="Input" rRef="L1"/>
    </ResourceLinkPool>
  </JDF>
```

3.10.5.3 Relating PartIDKeys and Partitions

[New in JDF 1.2](#)

The *PartIDKeys* Attribute (see Section 3.10.6, PartIDKeys Attribute and Partition Keys) describes the Partition Keys that occur in a Partitioned Resource. The sequence and number of keys is restricted in order and cardinality to ensure interoperability. The first entry in the *PartIDKeys* list defines the Partition closest to the root, the next entry defines the next intermediate Partition Node and so forth until the last entry, which defines the Partition leaves. Each

Partition Key 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.

3.10.5.3.1 Incomplete Partitions

[New in JDF 1.2](#)

Partitioned Resources MAY be Partitioned by a restricted subset of keys in the *PartIDKeys* list. Keys from the back of the list MAY be omitted in individual Partitions. If a key is omitted, all following keys MUST also be omitted.

Example 3-14: Legal Incomplete Partition

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

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Preview Class="Parameter" ID="P1" PartIDKeys="PreviewType Separation"
      URL="File:///aaa.pdf" Status="Available">
      <Preview PreviewType="Separation">
        <Preview Separation="Cyan" />
        <Preview Separation="Magenta" />
      </Preview>
      <Preview PreviewType="ThumbNail" />
    </Preview>
  </ResourcePool>
  <ResourceLinkPool>
    <PreviewLink Usage="Input" rRef="P1" />
  </ResourceLinkPool>
</JDF>
```

Example 3-15: Illegal Incomplete Partition

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

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

3.10.5.3.2 Number of Partition Keys per Partitioned Leaf or Node

[New in JDF 1.2](#)

Exactly one Partition Key MUST be specified per leaf or Node, excluding the Root Node. This allows XPath-type searches on Partitioned leaves.

Example 3-16: Legal Complete Partition

The following example demonstrates a legal Partition:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Preview Class="Parameter" ID="P3" PartIDKeys="PreviewType Separation"
      URL="File:///aaa.pdf" Status="Available">
      <Preview PreviewType="Separation">
        <Preview Separation="Cyan" />
      </Preview>
    </Preview>
  </ResourcePool>
  <ResourceLinkPool>
    <PreviewLink Usage="Input" rRef="P3" />
  </ResourceLinkPool>
</JDF>
```

```

    </ResourceLinkPool>
  </JDF>

```

Example 3-17: Illegal Partition

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

```

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

```

3.10.5.3.3 Degenerate Partitions

[New in JDF 1.2](#)

A Partitioned Resource 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.

Example 3-18: Degenerate Partition

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

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="Folding" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Component Class="Quantity" ID="c1" PartIDKeys="SheetName" Status="Available"
      ComponentType="Sheet">
      <Component SheetName="Sheet 1"/>
    </Component>
    <Component Class="Quantity" ID="c2" PartIDKeys="SheetName" Status="Available"
      ComponentType="Sheet">
      <Component SheetName="Sheet 2"/>
    </Component>
    <FoldingParams Class="Parameter" ID="fold" NoOp="true" PartIDKeys="SheetName"
      Status="Available">
      <FoldingParams NoOp="false" SheetName="Sheet 1"/>
    </FoldingParams>
  </ResourcePool>
  <ResourceLinkPool>
    <FoldingParamsLink Usage="Input" rRef="fold"/>
    <ComponentLink Usage="Input" rRef="c1"/>
    <ComponentLink Usage="Output" rRef="c2"/>
  </ResourceLinkPool>
</JDF>

```

Example 3-19: Invalid Degenerate Partition

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

```

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

```

3.10.5.4 Partitioning of Resource Subelements

[New in JDF 1.2](#)

Only Resources can be Partitioned. If a Resource contains Subelements, the Subelements 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.

Example 3-20: Partitioned ExposedMedia with Media Subelements

In the first example, the **ExposedMedia** Resource is Partitioned.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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>
  </ResourcePool>
  <ResourceLinkPool>
    <ExposedMediaLink Usage="Input" rRef="L1"/>
  </ResourceLinkPool>
</JDF>
```

Example 3-21: Partitioned ExposedMedia with Incomplete Media Subelements

In this incomplete example, the **Media** in the leaves is not complete because it does not contain the *MediaType* Attribute. *MediaType* is *not* inherited from the **Media** Element in the root Resource because in this case **Media** is not the Partitioned Resource.

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

Example 3-22: Partitioned ExposedMedia with Invalid Partitioning of Media Subelements

In this invalid example, **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>
```

Example 3-23: Partitioned ExposedMedia with MediaRef Subelements

Partitioning MAY be combined with inter-Resource links, (i.e., *ResourceRef* Elements.) In the following valid example, each *MediaRef* is equivalent to an in-lined leaf with the explicit *Part* Elements to define the Partition, (i.e., it is equivalent to the valid Example 3-20.)

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
```

```

<ResourcePool>
  <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>
</ResourcePool>
<ResourceLinkPool>
  <ExposedMediaLink Usage="Input" rRef="L41" />
</ResourceLinkPool>
</JDF>

```

Example 3-24: Partitioned ExposedMedia with Invalid MediaRef Subelements

In this invalid example, MediaRef does not reference the leaves of Media but, rather, to the root of Media. It is equivalent to the invalid Example 3-22.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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>
  </ResourcePool>
</ResourceLinkPool>
  <ExposedMediaLink Usage="Input" rRef="L51" />
</ResourceLinkPool>
</JDF>

```

3.10.5.5 Logical Partitions and the Identical Element

[New in JDF 1.3](#)

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

3.10.5.5.1 Element: Identical

Any Partitioned Resource MAY contain an Identical Subelement. The Resource Partition containing the Identical Element is called the logical Partition or slave Partition. Linking a logical Partition using a ResourceLink or referencing a logical Partition using a ResourceRef is semantically the same as linking/referencing the master Partition.

All Attributes except for the Attributes specified in *PartIDKeys* and all Subelements of the Resource (See “Abstract Resource Element” on page 63.) specified or inherited in the logical Partition MUST be ignored and replaced by the Attributes and Subelements of the master Partition.

Table 3-26: Identical Element

Name	Data Type	Description
Part	element	Identifies the physical Partition which will be used instead of the logical Partition. The logical Partition is defined by the Resource Partition containing the Identical Element.

Example 3-25: Partitioning with the Identical Element

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

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <ExposedMedia Class="Handling" ID="L1" PartIDKeys="SheetName Side Separation"
      Status="Available">
      <Media Class="Consumable" MediaType="Film"/>
      <ExposedMedia SheetName="S1">
        <ExposedMedia Side="Front">
          <ExposedMedia ProductID="1" Separation="Cyan"/>
          <ExposedMedia ProductID="2" Separation="Magenta"/>
          <ExposedMedia ProductID="3" Separation="Yellow"/>
          <ExposedMedia ProductID="4" Separation="Black"/>
        </ExposedMedia>
        <!-- Master partition that is referenced by an Identical Element -->
        <ExposedMedia Side="Back">
          <ExposedMedia ProductID="5" Separation="Cyan"/>
          <ExposedMedia ProductID="6" Separation="Magenta"/>
          <ExposedMedia ProductID="7" Separation="Yellow"/>
          <ExposedMedia ProductID="8" Separation="Black"/>
        </ExposedMedia>
      </ExposedMedia>
      <ExposedMedia SheetName="S2">
        <ExposedMedia Side="Front">
          <ExposedMedia ProductID="9" Separation="Cyan"/>
          <ExposedMedia ProductID="10" Separation="Magenta"/>
          <ExposedMedia ProductID="11" Separation="Yellow"/>
          <ExposedMedia ProductID="12" Separation="Black"/>
        </ExposedMedia>
        <!-- Logical partition with an Identical Element -->
        <ExposedMedia Side="Back">
          <Identical>
            <Part SheetName="S1" Side="Back"/>
          </Identical>
        </ExposedMedia>
      </ExposedMedia>
    </ExposedMedia>
  </ResourcePool>
  <ResourceLinkPool>
    <ExposedMediaLink Usage="Input" rRef="L1">
      <Part SheetName="S2" Side="Back" Separation="Black"/>
    </ExposedMediaLink>
  </ResourceLinkPool>
</JDF>
```

3.10.5.5.2 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. Such a master Partition MUST NOT contain an **Identical** element. In this way, the logical Partition obeys the rules described in Section 3.10.5.3, *Relating PartIDKeys and Partitions*.

Example 3-26: ResourceLink with Part Element

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

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <ExposedMedia Class="Handling" ID="L1" PartIDKeys="SheetName Side Separation"
      Status="Available">
      <Media Class="Consumable" MediaType="Film"/>
      <ExposedMedia SheetName="S1">
        <ExposedMedia Side="Front">
          <ExposedMedia ProductID="1" Separation="Cyan"/>
          <ExposedMedia ProductID="2" Separation="Magenta"/>
          <ExposedMedia ProductID="3" Separation="Yellow"/>
          <ExposedMedia ProductID="4" Separation="Black"/>
        </ExposedMedia>
        <!-- Master partition that is referenced by an Identical Element -->
        <ExposedMedia Side="Back">
          <ExposedMedia ProductID="5" Separation="Cyan"/>
          <ExposedMedia ProductID="6" Separation="Magenta"/>
          <ExposedMedia ProductID="7" Separation="Yellow"/>
          <ExposedMedia ProductID="8" Separation="Black"/>
        </ExposedMedia>
      </ExposedMedia>
      <ExposedMedia SheetName="S2">
        <ExposedMedia Side="Front">
          <ExposedMedia ProductID="9" Separation="Cyan"/>
          <ExposedMedia ProductID="10" Separation="Magenta"/>
          <ExposedMedia ProductID="11" Separation="Yellow"/>
          <ExposedMedia ProductID="12" Separation="Black"/>
        </ExposedMedia>
        <!-- Logical partition with an Identical Element -->
        <ExposedMedia Side="Back">
          <Identical>
            <Part SheetName="S1" Side="Back"/>
          </Identical>
        </ExposedMedia>
      </ExposedMedia>
    </ExposedMedia>
  </ResourcePool>
  <ResourceLinkPool>
    <ExposedMediaLink Usage="Input" rRef="L1">
      <Part SheetName="S2" Side="Back" Separation="Black"/>
    </ExposedMediaLink>
  </ResourceLinkPool>
</JDF>
```

Example 3-27: Partitioning with an Invalid Identical Element

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

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

3.10.6 PartIDKeys Attribute and Partition Keys[New in JDF 1.2](#)**3.10.6.1 Partitionable Resource**

In addition to the usual Resource Attributes and Elements, the Partitionable Resource Element has Partition-specific Attributes and Elements in its root. Specifying *PartIDKeys* in the root defines a Partitioned Resource. Throughout this document, the term “Partition Key” (depending on the context) refers to either

- an enumeration value of the *PartIDKeys* Attribute, e.g. *Side*

```
<ExposedMedia PartIDKeys = "Side"...>
```
- an Attribute that with two specialized functions:
 - can identify a Partition, e.g. *Side*.


```
<ExposedMedia ID="XM"...>
  <ExposedMedia Side="Front"...>
</ExposedMedia>
```
 - can reference a Partition from within a *Part* Element, e.g. *Side*

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

Further Attributes that apply to Partitioned Resources are listed in the following table.

Table 3-27: Partitionable Resource Element (Section 1 of 2)

Name	Data Type	Description																																																																					
<p><i>PartIDKeys</i> ? Modified in JDF 1.4</p>	<p>enumerations</p>	<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.</p> <p>For details, see Table 3-28, “Part Element,” on page 104.</p> <p>Modification note: before JDF 1.4, Part/<i>@Sorting</i> and Part/<i>@SortAmount</i> were not valid values of <i>PartIDKeys</i>. Now they have been deprecated so all values of <i>PartIDKeys</i> are also Elements of Part.</p> <p>Values are:</p> <table border="1" data-bbox="618 636 1421 1430"> <tr> <td><i>Bindery-</i></td> <td><i>FountainNumber</i></td> <td><i>Run</i></td> </tr> <tr> <td><i>SignatureName</i></td> <td><i>ItemNames</i></td> <td><i>RunIndex</i></td> </tr> <tr> <td><i>BlockName</i></td> <td><i>LayerIDs</i></td> <td><i>RunPage</i></td> </tr> <tr> <td><i>BundleItemIndex</i></td> <td><i>Location</i></td> <td><i>RunPageRange</i></td> </tr> <tr> <td><i>CellIndex</i></td> <td><i>Metadata0</i></td> <td><i>RunSet</i></td> </tr> <tr> <td><i>Condition</i></td> <td><i>Metadata1</i></td> <td><i>RunTags</i></td> </tr> <tr> <td><i>DeliveryUnit0</i></td> <td><i>Metadata2</i></td> <td><i>SectionIndex</i></td> </tr> <tr> <td><i>DeliveryUnit1</i></td> <td><i>Metadata3</i></td> <td><i>Separation</i></td> </tr> <tr> <td><i>DeliveryUnit2</i></td> <td><i>Metadata4</i></td> <td><i>SetDocIndex</i></td> </tr> <tr> <td><i>DeliveryUnit3</i></td> <td><i>Metadata5</i></td> <td><i>SetIndex</i></td> </tr> <tr> <td><i>DeliveryUnit4</i></td> <td><i>Metadata6</i></td> <td><i>SetRunIndex</i></td> </tr> <tr> <td><i>DeliveryUnit5</i></td> <td><i>Metadata7</i></td> <td><i>SetSheetIndex</i></td> </tr> <tr> <td><i>DeliveryUnit6</i></td> <td><i>Metadata8</i></td> <td><i>SetTags</i></td> </tr> <tr> <td><i>DeliveryUnit7</i></td> <td><i>Metadata9</i></td> <td><i>SheetIndex</i></td> </tr> <tr> <td><i>DeliveryUnit8</i></td> <td><i>Option</i></td> <td><i>SheetName</i></td> </tr> <tr> <td><i>DeliveryUnit9</i></td> <td><i>PageNumber</i></td> <td><i>Side</i></td> </tr> <tr> <td><i>DocCopies</i></td> <td><i>PageTags</i></td> <td><i>SignatureName</i></td> </tr> <tr> <td><i>DocIndex</i></td> <td><i>PartVersion</i></td> <td><i>StationName</i></td> </tr> <tr> <td><i>DocRunIndex</i></td> <td><i>PlateLayout</i></td> <td><i>SubRun</i></td> </tr> <tr> <td><i>DocSheetIndex</i></td> <td><i>PreflightRule</i></td> <td><i>TileID</i></td> </tr> <tr> <td><i>DocTags</i></td> <td><i>PreviewType</i></td> <td><i>WebName</i></td> </tr> <tr> <td><i>Edition</i></td> <td><i>RibbonName</i></td> <td><i>WebProduct</i></td> </tr> <tr> <td><i>EditionVersion</i></td> <td></td> <td><i>WebSetup</i></td> </tr> </table>	<i>Bindery-</i>	<i>FountainNumber</i>	<i>Run</i>	<i>SignatureName</i>	<i>ItemNames</i>	<i>RunIndex</i>	<i>BlockName</i>	<i>LayerIDs</i>	<i>RunPage</i>	<i>BundleItemIndex</i>	<i>Location</i>	<i>RunPageRange</i>	<i>CellIndex</i>	<i>Metadata0</i>	<i>RunSet</i>	<i>Condition</i>	<i>Metadata1</i>	<i>RunTags</i>	<i>DeliveryUnit0</i>	<i>Metadata2</i>	<i>SectionIndex</i>	<i>DeliveryUnit1</i>	<i>Metadata3</i>	<i>Separation</i>	<i>DeliveryUnit2</i>	<i>Metadata4</i>	<i>SetDocIndex</i>	<i>DeliveryUnit3</i>	<i>Metadata5</i>	<i>SetIndex</i>	<i>DeliveryUnit4</i>	<i>Metadata6</i>	<i>SetRunIndex</i>	<i>DeliveryUnit5</i>	<i>Metadata7</i>	<i>SetSheetIndex</i>	<i>DeliveryUnit6</i>	<i>Metadata8</i>	<i>SetTags</i>	<i>DeliveryUnit7</i>	<i>Metadata9</i>	<i>SheetIndex</i>	<i>DeliveryUnit8</i>	<i>Option</i>	<i>SheetName</i>	<i>DeliveryUnit9</i>	<i>PageNumber</i>	<i>Side</i>	<i>DocCopies</i>	<i>PageTags</i>	<i>SignatureName</i>	<i>DocIndex</i>	<i>PartVersion</i>	<i>StationName</i>	<i>DocRunIndex</i>	<i>PlateLayout</i>	<i>SubRun</i>	<i>DocSheetIndex</i>	<i>PreflightRule</i>	<i>TileID</i>	<i>DocTags</i>	<i>PreviewType</i>	<i>WebName</i>	<i>Edition</i>	<i>RibbonName</i>	<i>WebProduct</i>	<i>EditionVersion</i>		<i>WebSetup</i>
<i>Bindery-</i>	<i>FountainNumber</i>	<i>Run</i>																																																																					
<i>SignatureName</i>	<i>ItemNames</i>	<i>RunIndex</i>																																																																					
<i>BlockName</i>	<i>LayerIDs</i>	<i>RunPage</i>																																																																					
<i>BundleItemIndex</i>	<i>Location</i>	<i>RunPageRange</i>																																																																					
<i>CellIndex</i>	<i>Metadata0</i>	<i>RunSet</i>																																																																					
<i>Condition</i>	<i>Metadata1</i>	<i>RunTags</i>																																																																					
<i>DeliveryUnit0</i>	<i>Metadata2</i>	<i>SectionIndex</i>																																																																					
<i>DeliveryUnit1</i>	<i>Metadata3</i>	<i>Separation</i>																																																																					
<i>DeliveryUnit2</i>	<i>Metadata4</i>	<i>SetDocIndex</i>																																																																					
<i>DeliveryUnit3</i>	<i>Metadata5</i>	<i>SetIndex</i>																																																																					
<i>DeliveryUnit4</i>	<i>Metadata6</i>	<i>SetRunIndex</i>																																																																					
<i>DeliveryUnit5</i>	<i>Metadata7</i>	<i>SetSheetIndex</i>																																																																					
<i>DeliveryUnit6</i>	<i>Metadata8</i>	<i>SetTags</i>																																																																					
<i>DeliveryUnit7</i>	<i>Metadata9</i>	<i>SheetIndex</i>																																																																					
<i>DeliveryUnit8</i>	<i>Option</i>	<i>SheetName</i>																																																																					
<i>DeliveryUnit9</i>	<i>PageNumber</i>	<i>Side</i>																																																																					
<i>DocCopies</i>	<i>PageTags</i>	<i>SignatureName</i>																																																																					
<i>DocIndex</i>	<i>PartVersion</i>	<i>StationName</i>																																																																					
<i>DocRunIndex</i>	<i>PlateLayout</i>	<i>SubRun</i>																																																																					
<i>DocSheetIndex</i>	<i>PreflightRule</i>	<i>TileID</i>																																																																					
<i>DocTags</i>	<i>PreviewType</i>	<i>WebName</i>																																																																					
<i>Edition</i>	<i>RibbonName</i>	<i>WebProduct</i>																																																																					
<i>EditionVersion</i>		<i>WebSetup</i>																																																																					
<p><i>PipePartIDKeys</i> ? New in JDF 1.2</p>	<p>enumerations</p>	<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/<i>@PartIDKeys</i>= “<i>SheetName Side Separation</i>”) and if the ResourceLink/<i>@PipePartIDKeys</i>= “<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.</p> <p>Default value is from: <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 93.</p> <p>Values are from: <i>@PartIDKeys</i>.</p>																																																																					

Table 3-27: Partitionable Resource Element (Section 2 of 2)

Name	Data Type	Description
Identical ?	element	Cross reference to a logical Partition. For details on logical Partitions and the Identical Element, see Section 3.10.5.5, Logical Partitions and the Identical Element.
Resource*	element	Nested Resource Elements that contain the appropriate Partition Keys as specified in <i>PartIDKeys</i> . These Elements MUST be of the same name and type as the root Resource Element. They represent the individual parts or groups of parts.

3.10.6.2 Element: Part

Partitionable Resources are uniquely identified by the Attribute Values listed in *PartIDKeys* Attributes. The choice of which Attributes to use depends on how the Agent organizes the Job.

The following table lists the content of a Part Element, which contains a set of Attributes that have a well described meaning. Each of the Attributes, except *Sorting*, MAY be used in the nested Resource Elements of Partitioned Resources as the Partition Key (see example above).

Part Elements match a given Partition when all of the Attributes of a Part Element match the Attributes of the referenced Resource. This corresponds to Boolean AND operation. Note that a Part Element MAY specify a subset of the Partition Keys (e.g., only lower level Partition Keys) and thus implicitly select multiple Partitions leaves or Nodes from a Partitioned Resource (see Section 3.10.7.4, Implicit, Sparse and Explicit PartUsage in Partitioned Resources). If multiple Part Elements are specified, the result is a Boolean OR of the multiple parts. A Part Element with no Attributes explicitly references the root Resource.

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

Table 3-28: Part Element (Section 1 of 9)

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 <i>Cutting</i> 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 <i>BundleItemIndex</i> 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. Values include those from: Table 3-29, “Condition Attribute Values,” on page 113

Table 3-28: Part Element (Section 2 of 9)

Name	Data Type	Description
DeliveryUnit0? 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 <i>PartIDKeys</i> . Note that <i>N</i> is a placeholder for the values 0 through 9.
DeliveryUnit1? New in JDF 1.3	NMTOKEN	See <i>DeliveryUnit0</i> .
DeliveryUnit2? New in JDF 1.3	NMTOKEN	See <i>DeliveryUnit0</i> .
DeliveryUnit3? New in JDF 1.3	NMTOKEN	See <i>DeliveryUnit0</i> .
DeliveryUnit4? New in JDF 1.3	NMTOKEN	See <i>DeliveryUnit0</i> .
DeliveryUnit5? New in JDF 1.3	NMTOKEN	See <i>DeliveryUnit0</i> .
DeliveryUnit6? New in JDF 1.3	NMTOKEN	See <i>DeliveryUnit0</i> .
DeliveryUnit7? New in JDF 1.3	NMTOKEN	See <i>DeliveryUnit0</i> .
DeliveryUnit8? New in JDF 1.3	NMTOKEN	See <i>DeliveryUnit0</i> .
DeliveryUnit9? New in JDF 1.3	NMTOKEN	See <i>DeliveryUnit0</i> .
DocCopies?	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.
DocIndex?	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.
DocRunIndex?	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 .

Table 3-28: Part Element (Section 3 of 9)

Name	Data Type	Description
<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 Modified in JDF 1.4	NameRangeList	List of tags of documents in a multi-document RunList . Used to Partition Resources that are linked from Processes that also have a RunList as input. The Partition is selected if the implied, 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. Modification note: starting with JDF 1.4, the data type was expanded from NMTOKENS to NameRangeList.
<i>Edition</i> ? New in JDF 1.3	NMTOKEN	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	NMTOKEN	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 . Default behavior: 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 <i>LayerID</i> . Default behavior: all layers are processed.
<i>Location</i> ? Modified in JDF 1.3	NMTOKEN	Name of the location, (e.g., in MIS). This part key allows to describe distributed Resources. Note that this name does not define the location by itself. See Section 3.10.6.4, Locations of Physical Resources for details on specifying locations. Values include those from: Table C-21, "Input Tray and Output Bin Names," on page 889. Note: the specified values are for printer locations.
<i>Option</i> ? Modified in JDF 1.3	NMTOKEN	Option of an RFQ. Used mainly in Intent Resources.

Table 3-28: Part Element (Section 4 of 9)

Name	Data Type	Description
Metadata0? New in JDF 1.4	NameRangeList	Metadata extracted from a PDL using RunList /MetadataMap Elements. See “MetadataMap” on page 717.
Metadata1? New in JDF 1.4	NameRangeList	See <i>Metadata0</i> .
Metadata2? New in JDF 1.4	NameRangeList	See <i>Metadata0</i> .
Metadata3? New in JDF 1.4	NameRangeList	See <i>Metadata0</i> .
Metadata4? New in JDF 1.4	NameRangeList	See <i>Metadata0</i> .
Metadata5? New in JDF 1.4	NameRangeList	See <i>Metadata0</i> .
Metadata6? New in JDF 1.4	NameRangeList	See <i>Metadata0</i> .
Metadata7? New in JDF 1.4	NameRangeList	See <i>Metadata0</i> .
Metadata8? New in JDF 1.4	NameRangeList	See <i>Metadata0</i> .
Metadata9? New in JDF 1.4	NameRangeList	See <i>Metadata0</i> .
PageNumber?	IntegerRangeList	Page number in a Component or document, (e.g., FileSpec that is not described as a RunList). References an index in a PageList .
PageTags? New in JDF 1.3 Modified in JDF 1.4	NameRangeList	List of tags of pages in a multi-page RunList . Used to Partition Resources that are linked from Processes that also have a RunList as input. The Partition is selected if the implied, i.e., from the PDL, value of the page in the RunList matches any of the entries in <i>PageTags</i> . Modification note: starting with JDF 1.4, the data type was expanded from NMTOKENS to NameRangeList.
PartVersion? Modified in JDF 1.3	NMTOKENS	Version identifier, (e.g., the language version of a catalog). Compatibility note: The data type of <i>PartVersion</i> was changed from string to NMTOKENS in JDF 1.3 in order to accommodate Resources that contain Elements from multiple versions, e.g. Sheets with two language versions.
PlateLayout? New in JDF 1.3	NMTOKEN	Identifier of a single plate layout (mainly used for newspaper processes, where multiple plates are needed for one cylinder)
PreflightRule? New in JDF 1.2 Modified in JDF 1.3	NMTOKEN	Definition of the specific parts of a PreflightReportRulePool /PRRule used in preflight applications.

Table 3-28: Part Element (Section 5 of 9)

Name	Data Type	Description
<p><i>PreviewType</i> ? New in JDF 1.1 Modified in JDF 1.2</p>	enumeration	<p>Type of the preview.</p> <p>Constraint: If both <i>PreviewType</i> and Preview/@PreviewUsage or PreviewGenerationParams/@PreviewUsage are specified, they MUST match.</p> <p>Values are:</p> <p><i>SeparatedThumbNail</i> – Very low resolution separated preview.</p> <p><i>Separation</i> – Separated preview in medium resolution.</p> <p><i>SeparationRaw</i> – Separated preview in medium resolution with no compensation. New in JDF 1.2</p> <p><i>ThumbNail</i> – Very low resolution RGB preview.</p> <p><i>Viewable</i> – RGB preview in medium resolution.</p>
<p><i>RibbonName</i> ? Modified in JDF 1.3</p>	NMTOKEN	A string that uniquely identifies each ribbon. Multiple ribbons are created out of one Web after dividing in case of Web Printing.
<p><i>Run</i> ? Modified in JDF 1.3</p>	NMTOKEN	The <i>Run</i> Attribute selects an individual RunList Partition from a RunList Resource.
<p><i>RunIndex</i> ?</p>	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 (~). 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 .
<p><i>RunPage</i> ? New in JDF 1.1</p>	integer	<p>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:</p> <pre><RunList URL="doc.pdf" (...)/></pre> <p>is split into:</p> <pre><RunList PartIDKeys="RunPage" (...)> <RunList URL="doc_page0.pdf" RunPage="0" (...)/> <RunList URL="doc_page1.pdf" RunPage="1" (...)/> </RunList></pre>
<p><i>RunPageRange</i> ? New in JDF 1.4</p>	IntegerRangeList	Used when splitting RunList Resources into larger chunks that are not yet based on PageList indices.
<p><i>RunSet</i> ? New in JDF 1.3</p>	NMTOKEN	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.

Table 3-28: Part Element (Section 6 of 9)

Name	Data Type	Description
<p><i>RunTags</i> ?</p> <p>New in JDF 1.1</p> <p>Modified in JDF 1.4</p>	NameRangeList	<p>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>.</p> <p>Note: 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 <i>RunList</i> implied by their names.</p> <p>Modification note: starting with JDF 1.4, the data type was expanded from NMTOKENS to NameRangeList.</p>
<p><i>SectionIndex</i> ?</p> <p>New in JDF 1.2</p>	IntegerRangeList	List of sections in a StrippingParams .

Table 3-28: Part Element (Section 7 of 9)

Name	Data Type	Description
<i>Separation</i> ?	string	<p>Identifies the separation name.</p> <p>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>Note: other values include any separation name defined in the <i>Name</i> Attribute of a Color Element in the ColorPool.</p> <p>Note: 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 445.</p>
<i>SetDocIndex</i> ? 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>
<i>SetIndex</i> ? 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 3-28: Part Element (Section 8 of 9)

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 Modified in JDF 1.4	NameRangeList	List of tags of pages in a multi-set RunList . Used to Partition Resources that are linked from Processes that also have a RunList as input. The Partition is selected if the implied, i.e., from the PDL, value of the set in the RunList matches any of the entries in <i>SetTags</i> . Modification note: starting with JDF 1.4, the data type was expanded from NMTOKENS to NameRangeList.
<i>SheetIndex</i> ? Modified in JDF 1.4	IntegerRangeList	The <i>SheetIndex</i> Attribute selects a set of logical Sheets from a RunList Resource either implicitly or explicitly Partitioned by <i>SheetIndex</i> . <i>SheetIndex</i> is only valid when a RunList is describing sheet/surfaces.
<i>SheetName</i> ?	string	A string that uniquely identifies each Sheet.
<i>Side</i> ?	enumeration	Denotes the side of the Sheet. 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. Values are: <i>Front</i> <i>Back</i>
<i>SignatureName</i> ?	string	A string that uniquely identifies the Signature within the Partitioned Resource.

Table 3-28: Part Element (Section 9 of 9)

Name	Data Type	Description
<i>Sorting?</i> Deprecated in JDF 1.4	IntegerRangeList	Mapping from the implied Partitioned Resource order to a Process order. The indices refer to the Elements of the complete Partitioned Resource, not to the index in the selection of parts defined by the Part Element. If not specified the part order is the same as the sorting order. <i>Sorting</i> MUST NOT be used as a Partition Key. Note: <i>Sorting</i> and <i>SortAmount</i> are semantically different from the other Attributes in this table as they define the ordering of parts, whereas the other Attributes define the selection of parts. Deprecation note: the order of the Part Elements contained in a ResourceLink is significant, and selects each specified subset of the Resource in the XML order of the Part Elements.
<i>SortAmount?</i> Deprecated in JDF 1.4	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. Deprecation note: see <i>Sorting</i> .
<i>StationName?</i> New in JDF 1.3	string	The name of the 1-up design in a DieLayout .
<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 7.2.186, 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> Modified in JDF 1.3	NMTOKEN	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. Multiple WebProducts MAY be produced simultaneously on one Web Press.
<i>WebSetup?</i> New in JDF 1.3	NMTOKEN	Defines one setup of a Web Press that MAY produce multiple WebProducts.

— Attribute: Condition

Table 3-29: Condition Attribute Values

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>AuxiliarySheet</i> New in JDF 1.4	This Partition identifies Media that was consumed as specified by InsertSheet/@SheetType = “AccountingSheet”, “ErrorSheet”, “JobSheet” or “SeparatorSheet”.
<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.
<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>Reusable</i> New in JDF 1.4	Waste to be used for setup in the next process.
<i>WhitePaperWaste</i>	White paper waste.

3.10.6.3 Options in Intent Resources

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

3.10.6.4 Locations of 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.

Example 3-28: ExposedMedia with Location Elements

The following example describes a set of plates that are distributed over two locations. (Note: See "Input Tray and Output Bin Names" on page 889 for additional detail on locating Physical Resources.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <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>
</JDF>
```

Example 3-29: Media with Location Elements

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

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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>
  </ResourcePool>
  <ResourceLinkPool>
    <LayoutPreparationParamsLink Usage="Input" rRef="L1"/>
  </ResourceLinkPool>
</JDF>
```

3.10.7 Linking to Resources

Modification note: starting with JDF 1.4, all text up to Section 3.10.7.3 is new and replaces now-deleted text that was present in JDF 1.3

A JDF Node can specify a reordering or subset of a Resource by including one or more Part Elements in the ResourceLink Element that links to that Resource. For details of the Part Element, please refer to Table 3-28, “Part Element,” on page 104.

3.10.7.1 Linking to Subsets of Resources

Each ResourceLink/Part Element selects a subset of the Resource, where the aggregation of each selected subset (in the case of multiple ResourceLink/Part Elements) creates a "virtual" Resource that will then be used during Node processing. This feature is often useful to reproduce part of the Job described by a Node, as the default interpretation of the Part Elements maintains the context as if the Node had been executed without any ResourceLink Partitioning.

Example 3-30: Linking to Subsets of Resources

For instance, if an *Imposition* Process outputs multiple sheets, and each sheet has dynamic marks placed on the sheet based on the value of *SheetIndex*, selecting a single sheet to be processed by *Imposition* would produce that sheet using the original *SheetIndex* value. This example would generate the imposed sheet #5 followed by the imposed sheet #1, where all dynamic marks on both sheets retain the context in which *SheetIndex* would have been defined when processing the full *RunList* Resource.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <RunList Class="Parameter" ID="SheetSurfacesGeneratedByImposition"
      PartIDKeys="SheetIndex" Status="Available">
      <RunList SheetIndex="1"/>
      <RunList SheetIndex="3"/>
      <RunList SheetIndex="5"/>
    </RunList>
  </ResourcePool>
  <ResourceLinkPool>
    <RunListLink rRef="SheetSurfacesGeneratedByImposition" Usage="Output">
      <!-- output of imposition -->
      <Part SheetIndex="5"/>
      <Part SheetIndex="1"/>
    </RunListLink>
  </ResourceLinkPool>
</JDF>
```

3.10.7.2 Reordering the Processing of Resources

ResourceLink Partitioning may also be used to reorder the processing order of content described by a *RunList*. This is done by using the *RunList/@IgnoreContext* Attribute, which specifies which Part Element Partition Keys' Job context should be ignored during processing. For more information and an example of this, see *RunList/@IgnoreContext* in Section 7.2.160, “RunList” on page 710 and following the *RunList* table, see Example 7-50, “RunList/MetadataMap” on page 721.

3.10.7.3 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.

Example 3-31: @Amount in an ExposedMediaLink to a Partitioned ExposedMedia

The following example shows the ResourceLink that refers to Example 3-13, “Partitioned ExposedMedia” on page 94 for a total of five plates.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <ExposedMedia Class="Handling" ID="E1" PartIDKeys="SheetName Side Separation"
      Status="Available">
      <Media Class="Consumable" MediaType="Film"/>
    </ExposedMedia>
  </ResourcePool>
</JDF>
```

```

<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>
</ResourcePool>
<ResourceLinkPool>
  <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>
</ResourceLinkPool>
</JDF>

```

3.10.7.4 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.

Example 3-32: PartUsage in a Partitioned Resource

Table 3-30 below describes the behavior of the JDF example that follows. Table 3-30 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 3-30: PartUsage Attribute examples

SheetName	Side	Separation	PartVersion	Implicit	Explicit	Sparse
—	—	—	—	Root	Root	Root
S1	—	—	—	S1	S1	S1
S2	—	—	—	S2	S2	S2
S3	—	—	—	Root	—	—
S2	Back	Cyan	—	S1BC	S1BC	S1BC
S1	Back	Cyan	—	S1BC	S1BC	S1BC
S1	Back	Orange	—	S1B	—	—
S2	Back	Orange	—	S1B	—	—
S1	—	Cyan	—	S1BC, S1FC	S1BC, S1FC	S1BC, S1FC
S1	Back	Cyan	Deutsch	S1BC	—	S1BC
S2	Back	Cyan	Deutsch	S1BC	—	S1BC
S2	Front	Cyan	Deutsch	S2FC	—	S2FC
S1	Back	Black	Deutsch	S1BKD	S1BKD	S1BKD

Note: the example below has *PartUsage* = "Implicit" and explicit values for ExposedMediaLink/Part Attributes, but Table 3-30 above describes the behavior for all values of *PartUsage* and all values of ExposedMediaLink/Part. The example

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourceLinkPool>
    <ExposedMediaLink Usage="Input" rRef="XM_ID">
      <Part SheetName="S1" Side="Front" Separation="Black" PartVersion="Deutsch"/>
    </ExposedMediaLink>
  </ResourceLinkPool>
  <ResourcePool>
    <ExposedMedia Brand="Gooley" 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>
    </ExposedMedia>
  </ResourcePool>
</JDF>
```

```

    </ExposedMedia>
  </ExposedMedia>
  <ExposedMedia ProductID="S1B" Side="Back">
    <ExposedMedia ProductID="S1BC" Separation="Cyan"/>
    <ExposedMedia ProductID="S1BM" Separation="Magenta"/>
    <ExposedMedia ProductID="S1BY" Separation="Yellow"/>
    <ExposedMedia ProductID="S1BK" Separation="Black">
      <ExposedMedia ProductID="S1BKD" PartVersion="Deutsch"/>
      <ExposedMedia ProductID="S1BKE" PartVersion="English"/>
    </ExposedMedia>
  </ExposedMedia>
</ExposedMedia>
<ExposedMedia ProductID="S2" SheetName="S2">
  <ExposedMedia ProductID="S2F" Side="Front">
    <ExposedMedia ProductID="S2FC" Separation="Cyan"/>
    <ExposedMedia ProductID="S2FM" Separation="Magenta"/>
    <ExposedMedia ProductID="S2FY" Separation="Yellow"/>
    <ExposedMedia ProductID="S2FK" Separation="Black"/>
  </ExposedMedia>
  <ExposedMedia Side="Back">
    <Identical>
      <Part SheetName="S1" Side="Back"/>
    </Identical>
  </ExposedMedia>
</ExposedMedia>
</ExposedMedia>
</ResourcePool>
</JDF>

```

3.10.7.5 Referencing Multiple Resources of the Same Type

Some Processes (e.g., *Collecting*, *Gathering*) allow multiple Input Resources of the same type. These multiple Input Resources MAY be represented by multiple individual Resources or by Partitioned Resources or by a mixture of both. If ordering is significant, the order of the leaves in a Partitioned Resource defines said ordering. Example 3-33 and Example 3-34 illustrate equivalent ways of gathering three input Sheets.

For *Gathering*, *Collecting*, *Inserting* and similar Processes that have multiple physical Input Resources, explicit links should be used to define how the output component is ordered. Implicit references of ordered Partitioned Resources are strongly discouraged since there is ambiguity if input components have multiple Partition level.

Example 3-33: Explicit Reference of Ordered Partitioned Resources

```

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

```



```

    <ComponentLink Usage="Input" rRef="Sheets01">
      <Part SheetName="Sheet2"/>
    </ComponentLink>
    <ComponentLink Usage="Input" rRef="Sheets01">
      <Part SheetName="Sheet3"/>
    </ComponentLink>
    <ComponentLink Usage="Output" rRef="SheetsOut"/>
  </ResourceLinkPool>
</JDF>

```

Example 3-34: Implicit Reference of Ordered Partitioned Resources

```

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

```

3.10.8 Splitting and Combining Resources

Depending on the circumstances, it MAY be appropriate either to split a Resource into multiple new Nodes or to specify multiple locations or parts for an individual Resource. There are four possible methods for splitting and combining Resources. Two methods are shown in Figure 3-10 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.

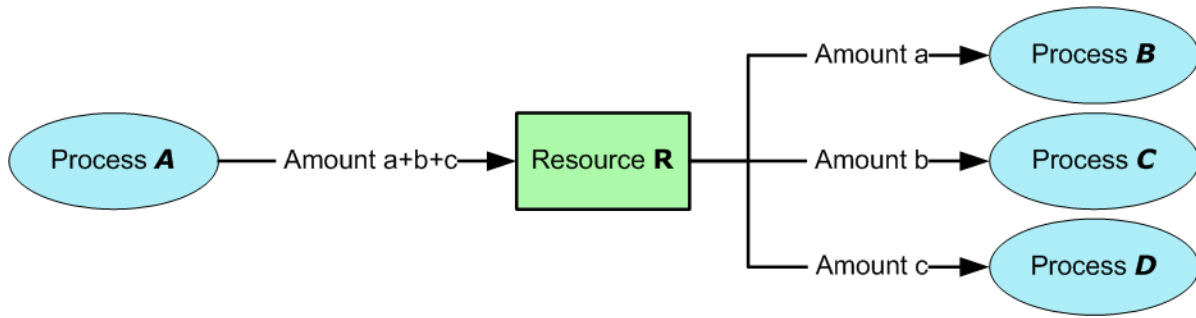


Figure 3-10: Workflow for splitting shared Input Resources

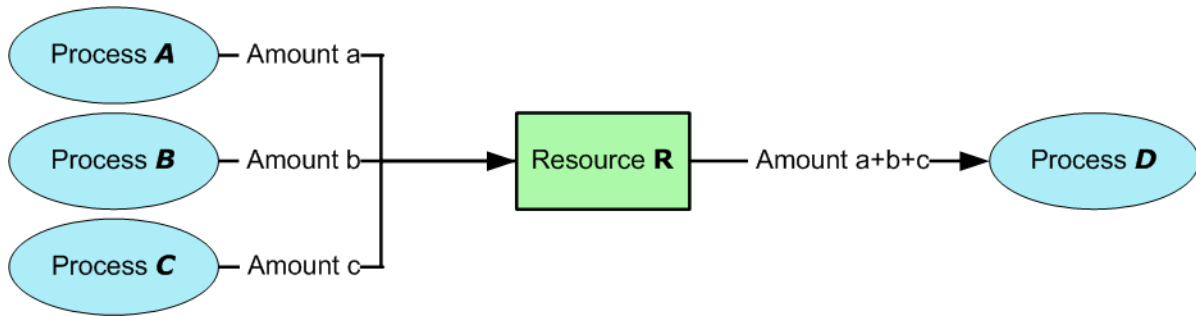


Figure 3-11: Workflow for combining shared Output Resources

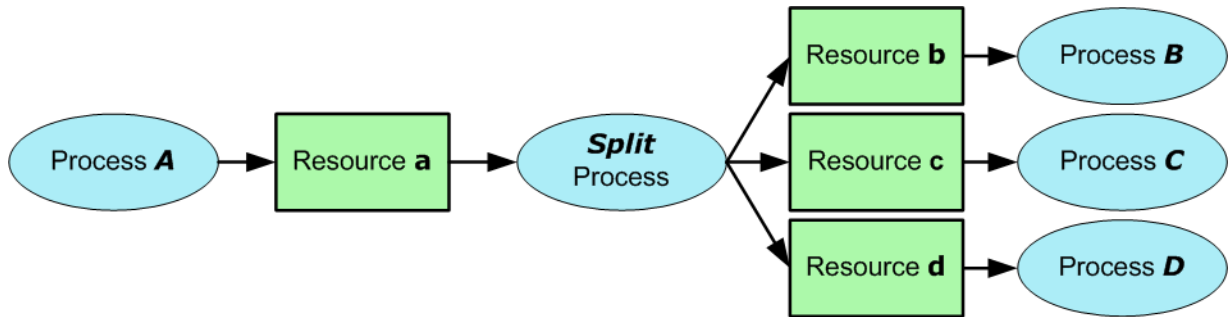


Figure 3-12: Workflow for splitting independent Input Resources

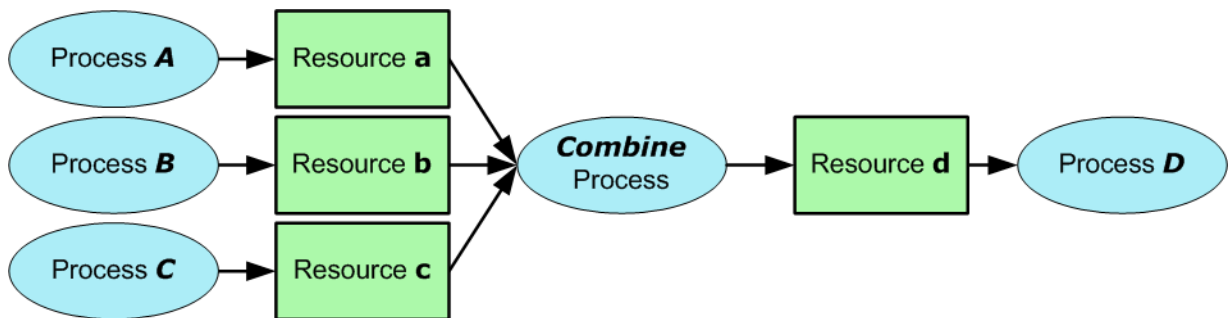


Figure 3-13: Workflow for combining independent Output Resources

3.11 AuditPool and Audit

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.
- 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.

3.11.1 AuditPool

The following table defines the contents of the `AuditPool` Element.

Table 3-31: AuditPool Element

Name	Data Type	Description
<code>rRefs?</code> Deprecated in JDF 1.2	IDREFS	List of all Resources that are referenced from within the <code>AuditPool</code> . In JDF 1.2 and beyond, it is up to the implementation to maintain references.
<code>Audit *</code>	element	Chronologically ordered list of <code>Audit Elements</code> . The <code>Audit Elements</code> are <code>Abstract</code> and serve as placeholders for any concrete <code>Element</code> derived from the <code>Abstract Audit Element</code> . <code>Audit Elements</code> are described in the sections that follow.

3.11.2 Structure Diagram

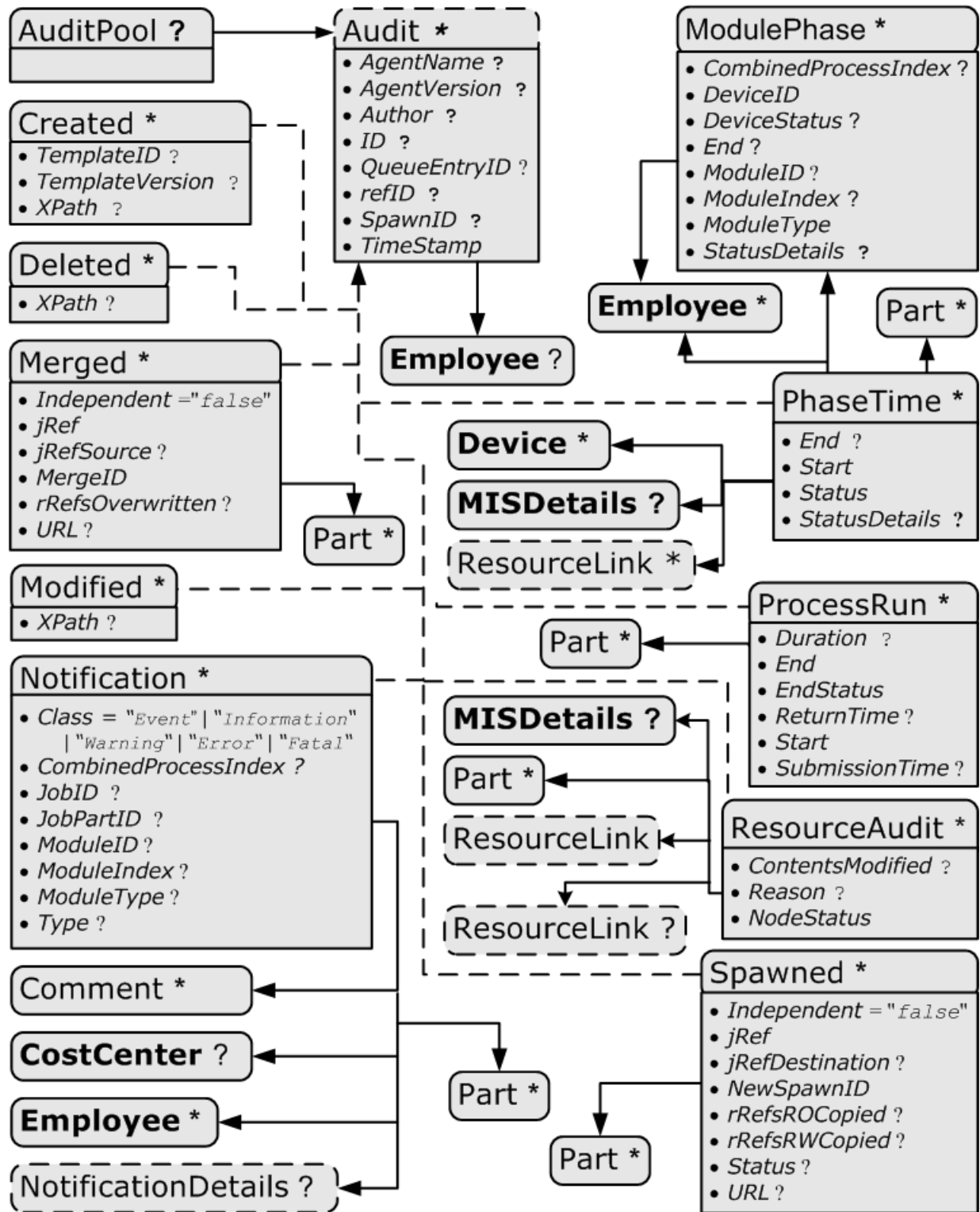


Figure 3-14: Abstract Audit Element – a diagram of its structure

3.11.3 Abstract Audit

All Audit Elements inherit the content from the Abstract Audit Element, described in the following table.

Table 3-32: Abstract Audit Element

Name	Data Type	Description
AgentName ? New in JDF 1.2	string	The name of the Agent application that added the Audit Element to the AuditPool (and was responsible for the creation or modification). Both the company name and the product name MAY appear, and SHOULD be consistent between versions of the application.
AgentVersion ? New in JDF 1.2	string	The version of the Agent application that added the Audit Element to the AuditPool (and was responsible for the creation or modification). The format of the version string MAY vary from one application to another, but SHOULD be consistent for an individual application.
Author ? Modified in JDF 1.2 Deprecated in JDF 1.4	string	Text that identifies the person who made the entry. Prior to JDF 1.2, <i>Author</i> also contained information that is now encoded in <i>AgentName</i> and <i>AgentVersion</i> . Deprecation note: starting with JDF 1.4, use Employee .
ID ? 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.
QueueEntryID ? New in JDF 1.4	string	<i>QueueEntryID</i> of the <i>QueueEntry</i> during which this Audit was generated.
refID ? New in JDF 1.2	IDREF	Reference to a previous Audit that this Audit corrects. The referenced Audit MUST reside in the same AuditPool.
SpawnID ? New in JDF 1.1	NMTOKEN	Text that identifies the spawned processing step when the entry was generated. This is a copy of the <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.
Employee ? New in JDF 1.4	element	Employee who created this Audit Element.

3.11.4 Audit

The following Elements are derived from the Abstract Audit Element:

Table 3-33: List of Audit Elements (Section 1 of 2)

Name	Page	Description
Created	page 124	Logs creation of JDF Node or Resource
Deleted	page 124	Logs deletion of JDF Node or Resource
Merged	page 124	Logs the merging of a spawned Node
Modified	page 125	Logs modifications affecting a JDF Node or its Subelements when the modification is not covered by other Audit Elements
Notification	page 125	Logs individual events that occurred during processing
PhaseTime	page 127	Logs start and end times of any Process states and substates, denoted as phases. Phases can reflect any arbitrary subdivisions of a Process.
ProcessRun	page 130	Summarizes one complete execution run of a Node or delimits a group of Audit Elements for each individual Process run.

Table 3-33: List of Audit Elements (Section 2 of 2)

Name	Page	Description
ResourceAudit	page 131	Describes the usage of Resources during execution of a Node or the modification of the intended usage of a Resource
Spawned	page 134	Logs the spawning of a Node.

3.11.4.1 Created

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

Table 3-34: 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 <code>Created</code> Element.

3.11.4.2 Deleted

[New in JDF 1.2](#)

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

Table 3-35: 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 <code>Deleted</code> Element.

3.11.4.3 Merged

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

Table 3-36: Merged Audit Element (Section 1 of 2)

Name	Data Type	Description
<i>Independent = "false"</i>	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. Note: the data type is NMTOKEN and not IDREF because the Attribute refers to an external ID.

Table 3-36: Merged Audit Element (Section 2 of 2)

Name	Data Type	Description
<i>MergeID</i> New in JDF 1.1	NMTOKEN	Copy of the <i>SpawnID</i> of the merged Node. Note that a <i>Merged Element</i> MAY also contain a <i>SpawnID</i> Attribute, which is the <i>SpawnID</i> of the Node that this <i>Audit</i> 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.
<i>URL?</i> New in JDF 1.1	URL	Locator that specifies the location of the merged Node prior to merging by the merging Process.
Part *	element	Specifies the selected parts of the Resource that were merged in case of parallel spawning and merging of Partitionable Resources. See Section 3.10.5, Description of Partitioned Resources.

3.11.4.4 Modified

This Element allows any modifications affecting a JDF Node or its Subelements to be logged. Changes that can be logged by a more specialized *Audit Element* (e.g., *ResourceAudit* for Resource changes) 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 3-37: Modified Audit Element

Name	Data Type	Description
<i>JRef?</i> Deprecated in JDF 1.2	IDREF	The ID of the modified Node. The <i>Modified Element</i> 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 <i>Modified Element</i> .

3.11.4.5 Notification

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

Table 3-38: Notification Audit Element (Section 1 of 2)

Name	Data Type	Description
<i>Class</i>	enumeration	<p>Class of the notification.</p> <p>Values are (in order of severity from lowest to highest):</p> <p><i>Event</i> – Indicates that a pure event due to certain operation-related activity has occurred, (e.g., Machine events, operator activities, etc.). This Class is used for the transfer of conventional event Messages. In case of <i>Class</i> = "Event", further event information is to be provided by the <i>Type</i> Attribute and NotificationDetails Element. See Section C.3.2, "NotificationDetails" on page 883.</p> <p><i>Information</i> – Any information about a Process which cannot be expressed by the other Classes (e.g., the beginning of execution).. No user interaction is needed.</p> <p><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</p> <p><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.</p> <p><i>Fatal</i> – Indicates that a fatal error led to abortion of the Process. 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.</p>
<i>CombinedProcessIndex?</i> New in JDF 1.4	IntegerList	<i>CombinedProcessIndex</i> Attribute specifies the indices of individual Processes in the <i>Types</i> Attribute to which a Notification in a Combined Process Node or Process Group Node belongs. Multiple entries in <i>CombinedProcessIndex</i> specify that the Module specified by Notification is executing the respective multiple Processes in the Combined Process Node.
<i>JobID?</i> New in JDF 1.3	string	<i>JobID</i> that this Notification applies to. <i>JobID</i> MUST NOT be specified when Notification is used as an Audit Element. Notification/@ <i>JobID</i> MAY be specified within a JMF Message.
<i>JobPartID?</i> New in JDF 1.3	string	<i>JobPartID</i> that this Notification applies to. <i>JobPartID</i> MUST NOT be specified when Notification is used as an Audit Element. Notification/@ <i>JobPartID</i> MAY be specified within a JMF Message.
<i>ModuleID?</i> New in JDF 1.4	string	<i>ModuleID</i> of the Module that this Notification relates to.
<i>ModuleIndex?</i> New in JDF 1.4	Integer-RangeList	<p>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.</p> <p>Constraint: At least one of <i>ModuleID</i> or <i>ModuleIndex</i> MUST be specified.</p>

Table 3-38: Notification Audit Element (Section 2 of 2)

Name	Data Type	Description
<i>ModuleType</i> ? New in JDF 1.4	NMTOKEN	Module description. Values include those from: Section C.2, "ModuleType Supported Strings" on page 879.. Note: the allowed values depend on the type of Device. Each type of Device has a separate table of values.
<i>Type</i> ?	NMTOKEN	Identifies the type of notification. Also defines the name of the Abstract NotificationDetails Element. Note: <i>Type</i> allows parsers that do not have access to the schema to find the instance of NotificationDetails. Values include those from: Table C-11, "List of NotificationDetail Elements," on page 883.
Comment *	element	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> = " <i>Event</i> "), Comment Elements are OPTIONAL.
CostCenter ?	element	The cost center to which this event is related to.
Employee *	refelement	The employee associated with this event.
NotificationDetails ?	element	NotificationDetails is an Abstract Element that 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 derived Elements see Table C-11, "List of NotificationDetail Elements," on page 883.
Part * New in JDF 1.1	element	Describes which parts of a Process this Notification belongs to. If Part is not specified for a Notification, it refers to all parts. For example, imagine a print Job that is to produce three different Sheets. All Sheets are described by one Partitioned Resource. The Part Elements define, unambiguously, the Sheet to which the Audit refers.

3.11.4.6 PhaseTime

This Element contains audit information about the start and end times of any Process states and substates, denoted as phases. Phases can reflect any arbitrary subdivisions of a Process, such as maintenance, washing, plate changing, failures and breaks.

PhaseTime Elements 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 and contain at least one identical ModulePhase MUST NOT overlap in time. PhaseTime Elements that apply to different Partitions MAY overlap in time in order to indicate parallel processing. PhaseTime Elements that apply to different modules MAY overlap in time in order to indicate independent processing with individual modules.

Table 3-39: PhaseTime Audit Element

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 <i>PhaseTime</i> is ongoing and the end of the phase has not yet occurred. This will generally be the case in the last <i>PhaseTime</i> of a snapshot JDF in a Status JMF. See Section 5.9.9, Status for details.,
<i>Start</i>	dateTime	Date and time of the beginning of the phase.
<i>Status</i> Modified in JDF 1.3	enumeration	Status of the phase. Values are (a subset of JDF/@Status): <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> Note: The values of this <i>Status</i> Attribute are a subset of the possible state values JDF/@ <i>Status</i> . For all possible states of a JDF Node see Table 3-5, “JDF Node,” on page 47. 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> .
<i>StatusDetails?</i>	string	Description of the status phase that provides details beyond the enumerative values given by the <i>Status</i> Attribute. Values include those from: "StatusDetails Supported Strings" on page 875
<i>Device</i> *	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)
<i>Employee</i> *	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).
<i>MISDetails?</i> New in JDF 1.2	element	Definition how the costs for the execution of this <i>PhaseTime</i> are to be charged.
<i>ModulePhase</i> *	element	Additional phase information of individual Device modules, such as print units.
<i>Part</i> *	element	Describes which parts of a Job is currently being logged. If a <i>Part</i> is not specified for a Node that modifies Partitioned Resources, <i>PhaseTime</i> 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 <i>Part</i> Elements define, unambiguously, the Sheet to which the <i>Audit</i> refers.
<i>ResourceLink</i> * New in JDF 1.1	element	These <i>ResourceLink</i> Elements specify the actual consumption/usage or production of Resources during this production phase. All Attributes apply to production and consumption within this <i>PhaseTime</i> only, thus <i>ResourceLink/@ActualAmount</i> specifies the actual amount produced or consumed.

3.11.4.6.1 ModulePhase

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

Table 3-40: ModulePhase 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 ModulePhase in a Combined Process Node or Process Group Node belongs. Multiple entries in <i>CombinedProcessIndex</i> specify that the Module specified by ModulePhase is executing the respective multiple Processes in the Combined Process 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 Device Elements specified in the PhaseTime.
<i>DeviceStatus</i> ? Modified in JDF 1.3	enumeration	Status of the Device module. 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. Note: these states are analog to the Device states of Table 5-72, “ModuleStatus Element,” on page 234.
<i>End</i> ? Modified in JDF 1.3 Deprecated in JDF 1.4	dateTime	Date and time of the end of the module phase. If not specified, the ModulePhase is ongoing and the end of the phase has not yet occurred. Deprecation note: starting with JDF 1.4, all Status information is recorded in PhaseTime. ModulePhase selects only the set of modules that a particular PhaseTime applies to.
<i>ModuleID</i> ? New in JDF 1.3	string	<i>ModuleID</i> of the Module that this ModulePhase refers to. If not specified, the module is specified in <i>ModuleIndex</i> . Constraint: at least one of <i>ModuleID</i> or <i>ModuleIndex</i> MUST be specified.

Table 3-40: ModulePhase Element (Section 2 of 2)

Name	Data Type	Description
<i>ModuleIndex</i> ? Modified in JDF 1.3	IntegerRangeList	0-based indices of the module or modules. The list is based on all modules of the Device. If multiple module types are available on one Device, each MUST be unique in the scope of the Device. Constraint: At least one of <i>ModuleID</i> or <i>ModuleIndex</i> MUST be specified.
<i>ModuleType</i>	NMTOKEN	Module description. Values include those from: Section C.2, "ModuleType Supported Strings" on page 879.. Note: the allowed values depend on the type of Device. Each type of Device has a separate table of values.
<i>Start</i> Modified in JDF 1.3 Deprecated in JDF 1.4	dateTime	Date and time of the beginning of the module phase. Deprecation note: starting with JDF 1.4, all Status information is recorded in <i>PhaseTime</i> . <i>ModulePhase</i> selects only the set of modules that a particular <i>PhaseTime</i> applies to.
<i>StatusDetails</i> ?	string	Description of the module status phase that provides details beyond the enumerative values given by the <i>DeviceStatus</i> Attribute.. Values include those from: "StatusDetails Supported Strings" on page 875
<i>Employee</i> *	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>).

3.11.4.7 ProcessRun

This Element serves two related functions.

The first function is to summarize one complete execution run of a Node. It contains Attributes that record the date and time of the start, the end time, the final Process state when the run is finished and, possibly, the Process duration of the Process run. These Attributes are described in Table 3-41.

The second function is to delimit a group of Audit Elements for each individual Process run. Every group of Audit Elements terminates with a *ProcessRun* Element, which contains the information described in Table 3-41. 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.

Table 3-41: 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 3-41: ProcessRun Audit Element (Section 2 of 2)

Name	Data Type	Description
<i>EndStatus</i> Modified in JDF 1.3	enumeration	The <i>Status</i> of the Process at the end of the run. For a description of Process states, see Table 3-5, “JDF Node,” on page 47. Values are: <i>Aborted</i> <i>Completed</i> <i>FailedTestRun</i> <i>Ready</i> <i>Stopped</i> .– 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. Deprecated in JDF 1.3
<i>ReturnTime ?</i> New in JDF 1.4	dateTime	Date and Time of the <i>ReturnQueueEntry</i> submission. If the JDF was returned via a Hot Folder, this corresponds to the time when the JDF was placed into the Hot Folder.
<i>Start</i>	dateTime	Date and time at which the Process started.
<i>SubmissionTime ?</i> New in JDF 1.4	dateTime	Date and Time of the <i>SubmitQueueEntry</i> submission. This value SHOULD be identical with <i>QueueEntry/@SubmissionTime</i> . If the JDF was submitted via a Hot Folder, this time corresponds to the time when the JDF was extracted from the Hot Folder.
<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 <i>Partitioned Resource</i> . The <i>Part</i> Elements define, unambiguously, the processing of the Sheet to which the <i>ProcessRun</i> refers.

3.11.4.8 ResourceAudit

The *ResourceAudit* Element describes the usage of Resources during execution of a Node or the modification of the intended usage of a Resource, (i.e., the modification of a *ResourceLink*) It logs consumption and production amounts of any quantifiable Resources, accumulated over one Process run or one part of a Process run. It contains one or two Abstract *ResourceLink* Elements. The first is REQUIRED and specifies the actual consumption/usage or production of the Resource. The second *ResourceLink* is OPTIONAL and used to store information about the original *ResourceLink*, which also refers to the original Resource. If the original Resource does not need to be saved, a Boolean *ContentsModified* Attribute in the *ResourceAudit* SHOULD be specified as “*true*” to indicate that a change has been made.

Table 3-42: ResourceAudit Audit Element (Section 1 of 2)

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. Values are: <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.

Table 3-42: ResourceAudit Audit Element (Section 2 of 2)

Name	Data Type	Description
<i>NodeStatus</i> New in JDF 1.3	enumeration	Status of the node that was executed during production or consumption of the resource. Values are (a subset of JDF/@Status): <i>TestRunInProgress</i> <i>Setup</i> <i>InProgress</i> <i>Cleanup</i> <i>Suspended</i> <i>Stopped</i> Note: The values of this <i>Status</i> Attribute are a subset of the possible state values JDF/@ <i>Status</i> . For all possible states of a JDF Node see Table 3-5, “JDF Node,” on page 47. 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 ProcessRun/@ <i>EndStatus</i> .
<i>MISDetails</i> ? New in JDF 1.3	element	Specifies how the costs associated with this ResourceAudit are to be charged.
Part *	element	Describes which parts of a job is currently being logged. If a Part is not specified for a node that modifies partitioned resources, ResourceAudit refers to all parts.
ResourceLink	element	The first ResourceLink specifies the actual consumption/usage or production of a Resource. This current Resource after modification NEED NOT be set to @ <i>Locked</i> =“true”.
ResourceLink ?	element	The second ResourceLink, which is OPTIONAL, logs the modification of a ResourceLink and the modification of the Resource it refers to. It holds the planned ResourceLink which also refers to the planned Resource. The planned and actual Resource MAY be the same.

For details on ResourceLink Elements and ResourceLink Subclasses, see Section 3.9, ResourceLinkPool and ResourceLink. The Partitioning of Resources using Part Elements is defined in Section 3.10.5, Description of Partitioned Resources.

3.11.4.8.1 Logging Machine Data by Using the ResourceAudit

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

- 1 Create a copy of the original Resource with a new ID.
- 2 Modify the original Resource to reflect the changes.
- 3 Insert a ResourceAudit Element that references the modified original Resource with the first ResourceLink and the copied Resource with the second ResourceLink Attribute.

Example 3-35: ResourceAudit: Before Logging

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

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
  Type="ConventionalPrinting" JobPartID="ID234" Version="1.4">
  <ResourceLinkPool>
    <MediaLink Amount="400" Usage="Input" rRef="RLink"/>
    <ConventionalPrintingParamsLink Usage="Input" rRef="R01"/>
    <ComponentLink Usage="Output" rRef="R02"/>
  </ResourceLinkPool>
```

```

<ResourcePool>
  <Media ID="RLink" Class="Consumable" Status="Available"
    Amount="400" Weight="80"/>
  <ConventionalPrintingParams ID="R01" Class="Parameter" Status="Available"/>
  <Component ID="R02" Class="Quantity" Status="Unavailable"
    ComponentType="Sheet"/>
</ResourcePool>
</JDF>

```

Example 3-36: ResourceAudit: Logging of Consumption

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

```

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

```

3.11.4.8.2 Logging Changes in Product Descriptions by Using the ResourceAudit

ResourceAudit Elements MAY also be used to store the original Intent Resources of a product specification in a change order or request for quote. The mechanism is the same as above.

Example 3-37: ResourceAudit: Logging Changes

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

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
  Type="Product" JobPartID="ID234" Version="1.4">
  <ResourceLinkPool>
    <MediaIntentLink Usage="Input" rRef="id">
      <Part Option="Option2"/>
    </MediaIntentLink>
    <ComponentLink Usage="Output" rRef="R02"/>
  </ResourceLinkPool>
  <ResourcePool>
    <MediaIntent ID="id" PartIDKeys="Option">
      <!-- the common MediaIntent resource details -->
      <MediaIntent Option="Option1">
        <Weight Preferred="80" DataType="NumberSpan"/>
      </MediaIntent>
      <MediaIntent Option="Option2">

```

```

        <Weight Preferred="90" DataType="NumberSpan" />
    </MediaIntent>
</MediaIntent>
<Component ID="R02" Class="Quantity" Status="Unavailable"
    ComponentType="Sheet" />
</ResourcePool>
<AuditPool>
    <ResourceAudit>
        <!-- the actual MediaIntent ResourceLink -->
        <MediaIntentLink Usage="Input" rRef="id">
            <Part Option="Option2" />
        </MediaIntentLink>
        <!-- the original MediaIntent ResourceLink -->
        <MediaIntentLink Usage="Input" rRef="id">
            <Part Option="Option1" />
        </MediaIntentLink>
    </ResourceAudit>
</AuditPool>
</JDF>

```

3.11.4.9 Spawned

This Element allows a Node that has been spawned to be logged in the `AuditPool` of the parent Node of the spawned Node or in the `AuditPool` of the Node that has been spawned in case of spawning of individual Partitions. For details about spawning and merging, see Section 4.4, Spawning and Merging.

Table 3-43: Spawned Audit Element (Section 1 of 2)

Name	Data Type	Description
<code>Independent = "false" ?</code>	boolean	Declares that independent Jobs that have previously been merged into a Big Job are spawned. If it is set to <code>true</code> , the Attributes <code>jRefDestination</code> , <code>rRefsROCopied</code> and <code>rRefsRWCopied</code> have no meaning and SHOULD be omitted.
<code>jRef</code>	IDREF	ID of the JDF Node that has been spawned.
<code>jRefDestination ?</code>	NMTOKEN	ID of the JDF Node to which the Job has been spawned. This Attribute MUST be specified in the parent of the original Node if independent Jobs are spawned. Note: the data type is NMTOKEN and not IDREF because the Attribute refers to an external ID.
<code>NewSpawnID</code> New in JDF 1.1	NMTOKEN	Copy of the <code>SpawnID</code> of the newly spawned Node. Note that a Spawned Audit MAY also contain a <code>SpawnID</code> Attribute, which is the <code>SpawnID</code> of the Node that this Audit is being placed into prior to spawning.
<code>rRefsROCopied ?</code>	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.
<code>rRefsRWCopied ?</code>	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.
<code>Status ?</code> New in JDF 1.1	enumeration	<code>Status</code> of the spawned Node at the time of spawning. . Values are from: JDF/@ <code>Status</code> (Table 3-5, “JDF Node,” on page 47)
<code>URL ?</code> New in JDF 1.1	URL	Locator that specifies the location where the spawned Node was stored by the spawning Process.

Table 3-43: Spawned Audit Element (Section 2 of 2)

Name	Data Type	Description
Part *	element	Identifies the parts that were selected for spawning in case of parallel spawning of Partitionable Resources. See Section 3.10.5, Description of Partitioned Resources.

3.12 JDF Extensibility

JDF is meant to be flexible and therefore useful to any vendor, as each vendor 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.

3.12.1 Namespaces in XML

JDF Extensibility is implemented using XML Namespaces [XMLNS]. XML namespaces are defined by *xmlns* Attributes. A general example is provided below.

Namespaces are inserted in front of Attribute and Element names. The associated namespace of Element names with no prefix is the default namespace defined by the *xmlns* Attribute. The associated namespace of Attributes with no prefix is that one of the Element (see Section A.3.3, Defined JDF enumeration Data Types). All namespace prefixes MUST be declared using the standard *xmlns:prefix* Attribute declarations.



Using Namespaces in JDF

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

Example 3-38: Namespaces in XML

The example illustrates how private namespaces are declared and used to extend an existing JDF Resource by adding private Attributes and a private Element.

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

3.12.1.1 JDF Namespace

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

3.12.1.2 JDF Extension Namespace

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

3.12.2 Extending Process Types

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

Example 3-39: Extending Process Types

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

3.12.2.1 Rules about Process Extension

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 Process Node. If a Process is simply an extension of an existing Process, it is possible to describe the private data by extending the existing Resource types. This is described in greater detail in the sections below.

**EXTENSIBILITY CAUTION**

JDF “Extensibility” simply means that you can add your own XML elements, Attributes and enumerations to a JDF application. Although JDF is quite extensive, odds are you’ll find that your current databases and workflow systems use information elements that are unique to your client market or company ... *they might have even been defined by your internal MIS staff.* CIP4 acknowledges that it can’t define everything, nor ought it prevent innovation by codifying everything in a static manner, and JDF’s extensibility provides both printers and technology providers with the flexibility they need to make JDF a success.

However, if you or your technology vendors extend JDF, please do so with caution. JDF’s success depends on the ability of MIS systems and JDF-enabled Devices to write, read, parse and use JDF. Extensions are *custom* integration applications and great care needs to be made to ensure that extensions made for one systems or Device will not *jam* the JDF workflow or other JDF enabled systems and Devices. If they use extensions to JDF, your technology providers need to be able to provide you with a fully validated JDF schema and documentation that includes the use of their extensions. Extensions that are not documented, or that are not to be disclosed to third parties for integration purposes, ought to be viewed skeptically.

3.12.3 Extending the NodeInfo and CustomerInfo Nodes

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

3.12.4 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 bits per colorant with a *foo:ColorantDepth* or *foo:ColDepth* Attribute that overrides the JDF defined parameter for bits per colorant (see **RenderingParams/@ColorantDepth** in Section 7.2.156, **RenderingParams**).

3.12.5 Extending NMTOKEN Lists

Many Resources contain Attributes of type NMTOKEN and some of these have a set of predefined, suggested enumerative values. These lists **MAY** be extended with private keywords. In order to identify private keywords, it is strongly **RECOMMENDED** to prefix these keywords with a namespace-like syntax, (i.e., a namespace prefix separated by a single colon “:”). Such a namespace prefix **SHOULD** be defined in the JDF ticket with the standard

xmlns:Prefix="someURI" notation, even if no extension Elements or Attributes from that namespace occur in the JDF ticket. Implementations that find an unknown NMTOKEN prefixed by a namespace prefix MAY then attempt to use the default value of that Attribute if the value of *SettingsPolicy* in effect is *BestEffort*.

Example 3-40: Extending NMTOKEN Lists

For instance, if an implementation encounters `TrappingParams/@TrapEndStyle` (see below in Table 3-44) in the JDF snippet shown below, and if the implementation does not support the "HDM" extension, the best assumption is to use `TrapEndStyle = "Miter"`, which is the default for `TrapEndStyle`.

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

Table 3-44: Excerpt from TrappingParams

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

3.12.6 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.

3.12.7 Future JDF Extensions

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

3.12.8 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.

3.12.9 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.

3.12.10 Derivation of Types in XML Schema

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

3.13 JDF Versioning

New in JDF 1.2

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

3.13.1 JDF Versioning Requirements

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

- JDF Documents with mixed versions 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.

3.13.2 JDF Version Definition

The version of a JDF Node is defined as the highest version of all Attributes or Elements and linked Resources. The version of a Resource is defined as the highest version of all Elements, Attributes or Resources that are referenced via refelements.

3.13.3 JDF Version Policies

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

3.13.3.1 JDF Specification Version Policies

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

- Changes to the JDF specification are always backwards compatible.
 - Extension Elements or Attributes are never required.
 - New Attributes in existing Elements 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 NOT be altered.
 - The default behavior that is specified when an Attribute or Element is missing MUST NOT be altered.

3.13.3.2 JDF Schema Version Policies

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

- Changes to the JDF schema 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.

3.13.3.3 JDF Application Version Policies

This section specifies the policies that implementations SHOULD follow in order to support multiple versions of JDF. The policies are specified for Agents and Controllers/Devices separately.

3.13.3.3.1 JDF Agent Version Policies

JDF Agents 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 3.13.2, JDF Version Definition.
- The *JDF/@Version* of a spawned JDF Node is identical to the *JDF/@Version* of that Node in a complete JDF.

3.13.3.3.2 JDF Device/Controller Version Policies

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

- Implementations **SHOULD** handle deprecated Elements and Attributes gracefully.

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

- Schema validation errors that find an unknown Attribute, Element or Attribute Value in a JDF with a version that is higher than the schema **SHOULD NOT** lead to an abort.
 - A Device or Controller that reads a JDF with an Element or Attribute or Attribute Value with a version that is higher than the version that it was developed for **SHOULD** attempt to execute the JDF if *SettingsPolicy* = "*BestEffort*".
 - A Device or Controller that reads a JDF with an Element or Attribute or Attribute Value with a version that is higher than the version that it was developed for **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 Intent Node that specifies the Product Intent of a series of pages. This Product Intent Node might include information about the number of pages and the paper properties. Another Node might then be inserted that includes a Resource describing how the pages are to be RIPed. Later, another Agent might provide more detail about the RIPing Process by appending optional information to the RIP 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 3.3.1, Section 3.3.2, and Section 3.3.3. 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 Intent Nodes, and the bottom-most level of the product hierarchy represents portions of the product that are each homogeneous in terms of their materials and formats. All Nodes below these Product Intent Nodes begin specifying the Processes needed to produce the products.

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

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

4.1.1.2 Representation of Product Binding

Some Intent 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 3.3.1, 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.

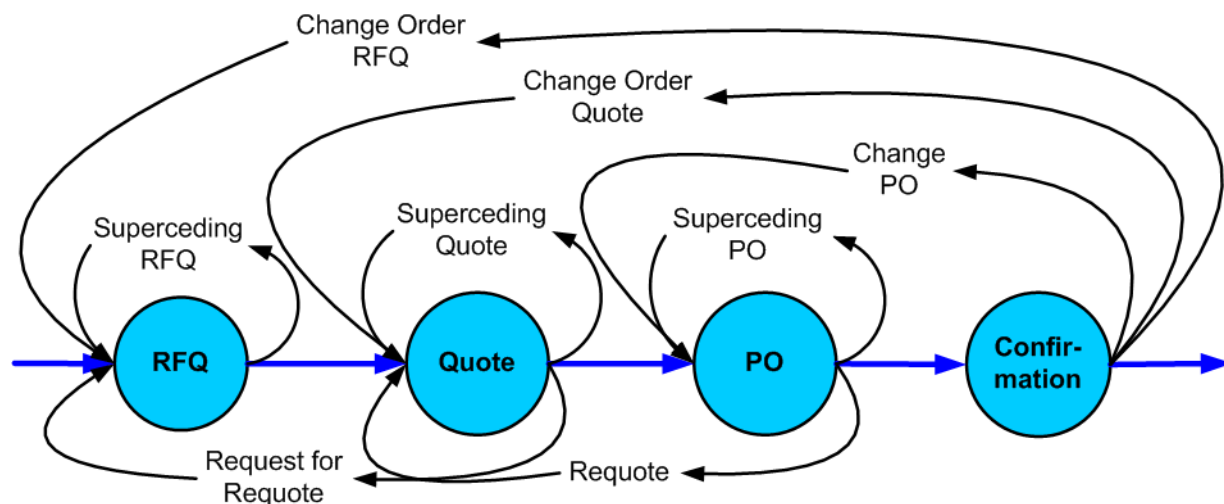


Figure 4-1: Simplified PrintTalk workflow (negotiation phase)

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.

Example 4-1: PrintTalk

This example and Example 4-2 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. This example shows PrintTalk. The **highlighted tags** and **highlighted attributes** show the respective position of an RFQ.

```
<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>
```

Example 4-2: Equivalent Pure JDF

This example shows pure JDF document text that is equivalent to the PrintTalk shown in Example 4-1. The **high-
lighted tags** and **highlighted attributes** show the respective position of an RFQ.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="ScreenTest"
  JobID="ScreenJob" JobPartID="ScreenJob100" Status="Waiting"
  Type="Product" Version="1.4">
  <ResourcePool>
    <NodeInfo ID="ID122" Class="Parameter" Status="Available"
      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>
<Component ID="ID123" Class="Quantity" Status="Unavailable"
  ComponentType="Sheet" />
</ResourcePool>
<ResourceLinkPool>
  <ComponentLink Usage="Output" rRef="ID123" />
  <NodeInfoLink Usage="Input" rRef="ID122" />
</ResourceLinkPool>
</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.

Example 4-3: Product Intent Node

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

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Job1" JobID="J1"
  JobPartID="P1"
  Status="Waiting" Type="Product" Version="1.4">
<ResourcePool>
  <Component Amount="10000" Class="Quantity"
    DescriptiveName="Complete 16-page Brochure" ID="Link0003"
    Status="Unavailable" ComponentType="Sheet" />
  <LayoutIntent Class="Intent" ID="Link0004" Status="Available">
    <Dimensions DataType="XYPairSpan" Preferred="612 792"
      Range="576 720 ~ 648 864" />
    <Pages DataType="IntegerSpan" Preferred="16" />
  </LayoutIntent>
  <MediaIntent Class="Intent" ID="Link0005" PartIDKeys="Option"
    Status="Available">
    <FrontCoatings DataType="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** Resource of a Node contains OPTIONAL *Route* and *TargetRoute* Attributes that let an Agent define a static Process route on a Node-by-Node basis. *JMF/QueueSubmissionParams/@ReturnURL* takes precedence over **NodeInfo/@TargetRoute** of the JDF Node that is processed. If no *Route* or *TargetRoute* Attribute is specified and if a Controller has multiple options where to route a Job, it is up to the implementation to decide which route to use.

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

4.2.1 Determining Executable Nodes

In order to determine which Node to execute, the Controller/Device 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 or the respective **NodeInfo/@NodeStatus** 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** Resource, 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.

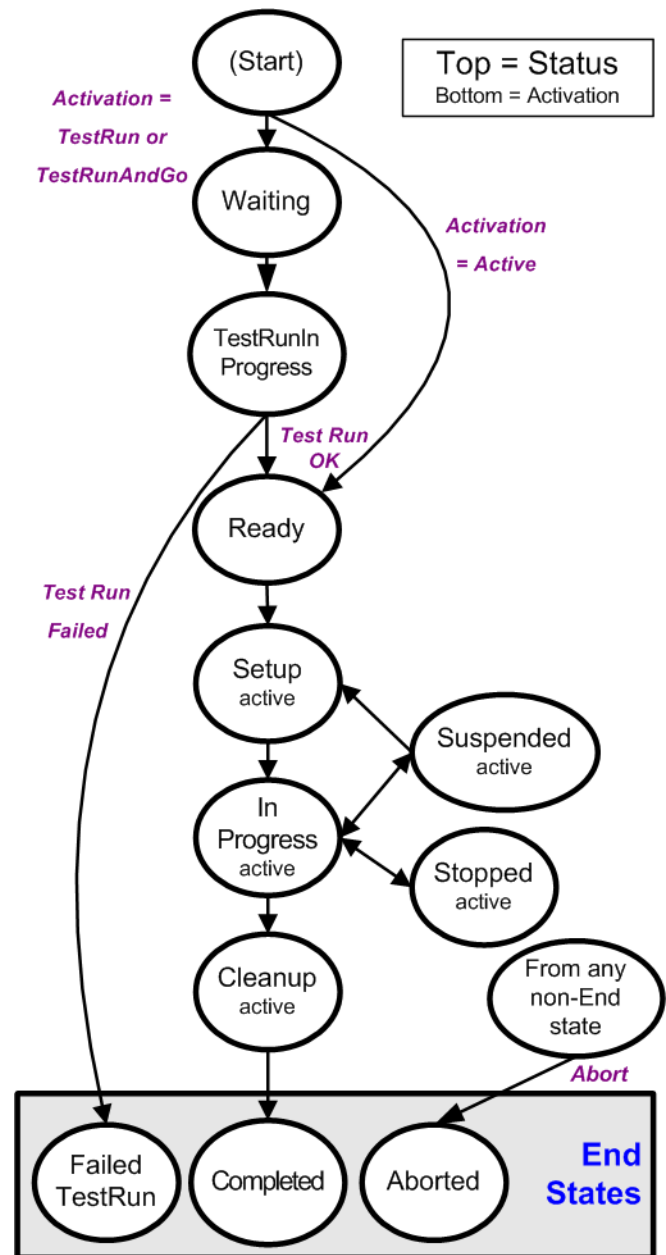


Figure 4-2: Life Cycle of a JDF Node

The Node will go through various states during its life time as is described in Figure 4-2.

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** Resource of a Node, it does not prescribe any specific routing methods. However, some of the tools available to figure out alternative workflows are described below.

Knowledge of the capabilities of lower level Controllers/Devices either MAY be hard-wired into the system or gained using the *KnownDevices* Message. Since JDF does not yet provide mechanisms to determine if a given Device is capable of processing a Node without actually performing a test run, a Controller 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** Resource of a Node, JDF MAY contain implicit routing information in the form of **Device Implementation Resources**.

JMF defines the *KnownControllers* Query Message to find Controllers and the *KnownDevices* Query Message 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**.

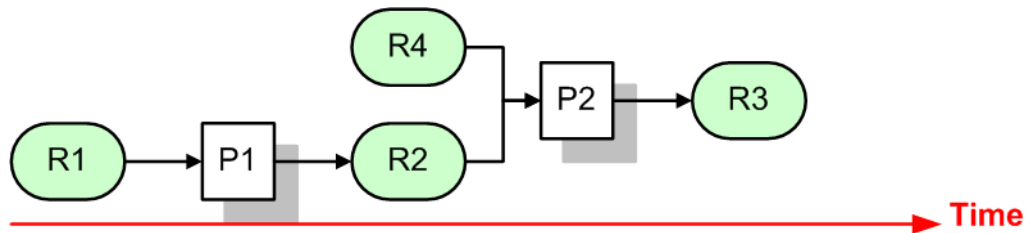


Figure 4-3: Example of a simple Process chain linked by Resources

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 3-10, “Abstract Resource Element,” on page 63. For more information about the values of the *Status* Attribute of Processes, see Table 3-5, “JDF Node,” on page 47.

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 *Amount* Attribute 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 90. 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
<code>SheetName = "S1"</code>	—	—	<code>SheetName = "S1"</code>	If only the input is Partitioned, the Node Partition is defined by the input.
<code>SheetName = "S1"</code> <code>Separation = "Cyan"</code>	—	—	<code>SheetName = "S1"</code> <code>Separation = "Cyan"</code>	If only the input is Partitioned, the Node Partition is defined by the input.
<code>SheetName = "S1"</code> <code>Separation = "Cyan"</code>	<code>Separation = "Cyan" + "Black"</code> <code>(PartUsage = "Implicit")</code>	—	<code>SheetName = "S1"</code> <code>Separation = "Cyan" + "S1"</code> <code>Separation = "Black"</code>	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.
<code>SheetName = "S1"</code>	—	<code>SheetName = "S1"</code> <code>Separation = "Cyan"</code>	<code>SheetName = "S1"</code> <code>Separation = "Cyan"</code>	The input and output base Partitions are identical. The output further restricts the Partition.
<code>SheetName = "S1"</code>	—	<code>SheetName = "S2"</code> <code>Separation = "Cyan"</code>	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 <i>RunList</i> 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 3-21, "Abstract *PhysicalLink* or *//AmountPool/PartAmount* Element," on



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.

page 83). If none of them are specified, any produced *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 3-10, “Abstract Resource Element,” on page 63).

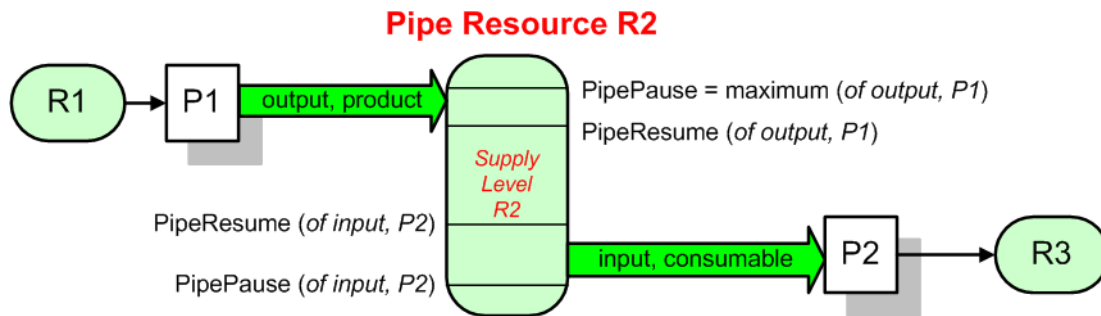


Figure 4-4: Example of a Pipe Resource linking two Processes

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.

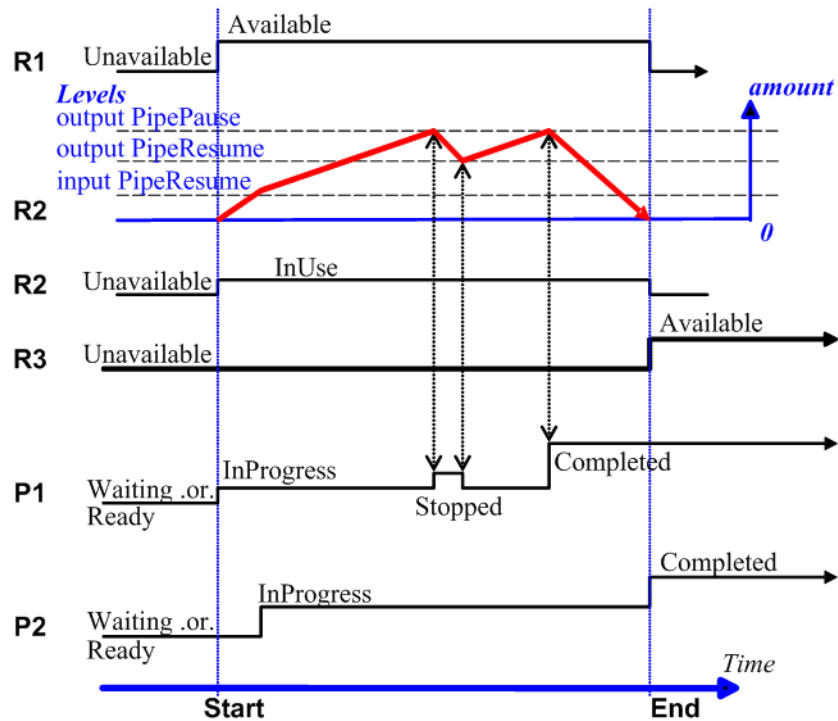


Figure 4-5: Example of status transitions in case of overlapping Processing

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 5.10, Messages for Pipe Control). If the Attributes *PipeResume* or *PipePause* of *ResourceLink* Elements to pipe Resources are not specified, the Controller is responsible when the linked Processes start and stop independent of the level.

4.3.3.1 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 5.10, Messages for Pipe Control) sent to another Controller whose URL address is specified by the `PipeURL` Attribute of the respective `ResourceLink`. Depending on the values of the `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 5.10, Messages for Pipe Control are designed to establish communication between Processes at both ends of dynamic pipe, even if the corresponding Processes are spawned separately.

The following table summarizes the actions to be taken when the buffer in a dynamic pipe reaches a certain level "L".

Table 4-4: 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` Resource 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 Resources` 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.

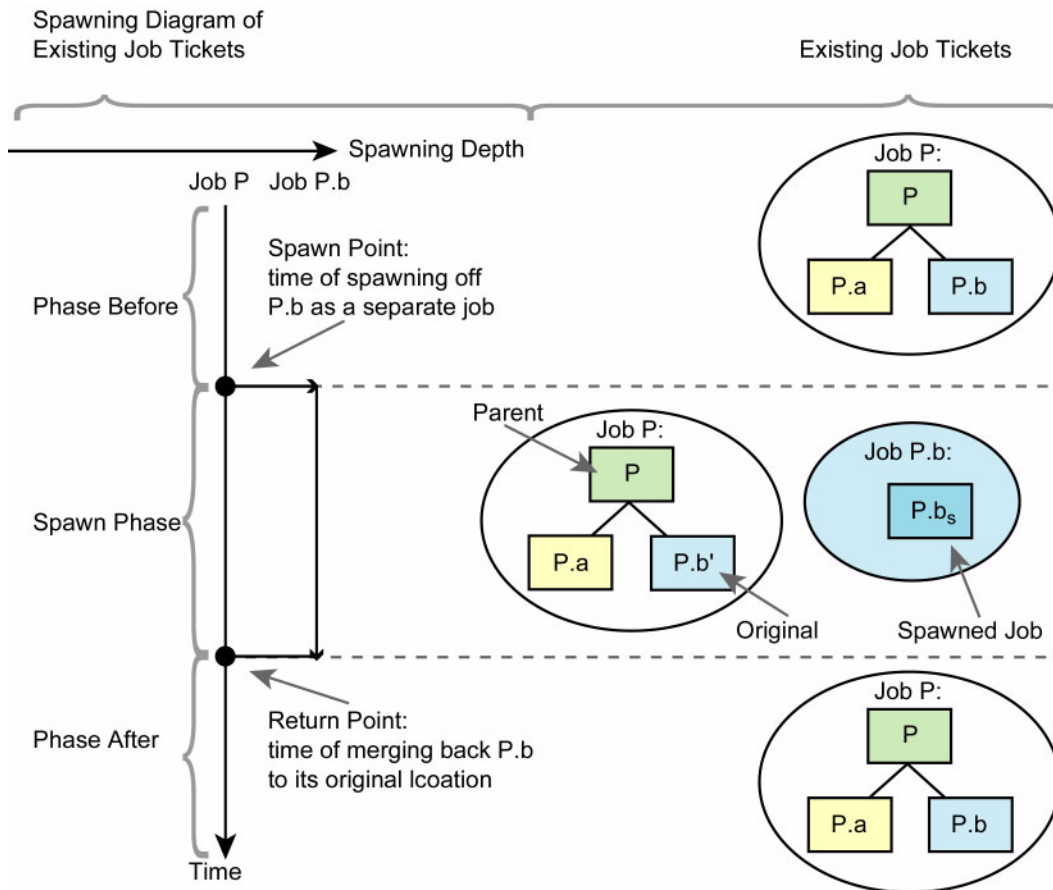


Figure 4-6: The spawning and merging mechanism and its phases

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 3-5, “JDF Node,” on page 47). 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 or refelements **MAY** be copied into the respective *Ancestor* Elements.

The complete ancestor information is **REQUIRED** in order to merge back semi-finished Jobs with nested spawns. If the last spawn is always merged first (“LIFO”—Last In, First Out), then knowing the direct parent is sufficient as each parent will in turn know its own parent back to the original and a complete ancestor line can be inferred.

When a Job is spawned, the action **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.

Example 4-4: Family Tree of Spawned Nodes

The following code is an example of a family tree:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <!-- ... -->
  <AncestorPool>
    <Ancestor FileName="file:///grandparent.jdf" NodeID="p_01"/>
    <Ancestor FileName="file:///parent.jdf" NodeID="p_02"/>
  </AncestorPool>
</JDF>
```


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 `Comment`) 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 3.11, `AuditPool` and `Audit`). The *MergeID* of the `Merged` audit Element SHOULD be set to the value of the *SpawnID* Attribute of the merged Node. Furthermore, the `AncestorPool` container with all child Elements 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.

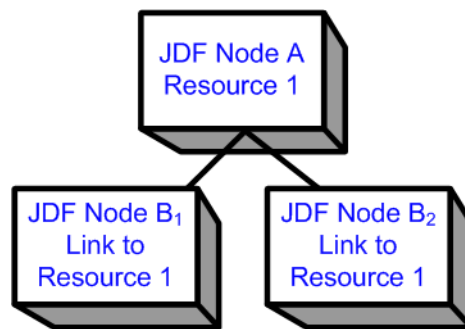


Figure 4-7: JDF Node structure that requires Resource copying during spawning and merging

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` 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 Message is used to inform the Device or Controller that the Resource was spawned to. The *Locked* Attribute of spawned Resources that are copied read-only MUST also be set *true*. Furthermore, the value of 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 "Part" and a NodeInfo Partition for the Partition of this spawn MUST be created. NodeInfo/@NodeStatus of the Partition that describes the newly spawned Node is set to "Spawned".

Exactly one Part Element that contains the Partition Keys of this spawn and all Partition Keys of previous spawns 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 NodeInfo Partition per part. The Spawned and Merged audit Elements are not placed in the parent Node of the Node to be spawned, but rather in the Node itself.

An Agent that receives a JDF Node that has been spawned in parallel and thus has a Part Element in the AncestorPool MUST NOT modify any Elements except for:

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

Synchronizing newly inserted JDF Subnode in spawned JDF Nodes is OPTIONAL.

Merging

After an individual Partitioned spawned Node has been processed, it is merged back to the parent as described in Case 1. In addition, a copy of the Part Elements of the corresponding Spawned Audit Element is appended to the Merged Element and any read/write Resources are merged into their appropriate parts. The *Status* of the spawned Node is copied into the appropriate PartStatus in the StatusPool.

1. Usually Resources become locked (*Locked = true*) if they are referenced by Audit Elements (see also Section 3.11, AuditPool and Audit).

An example of Partitioned Spawning and Merging can be found in "Spawning and Merging" on page 956.

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.

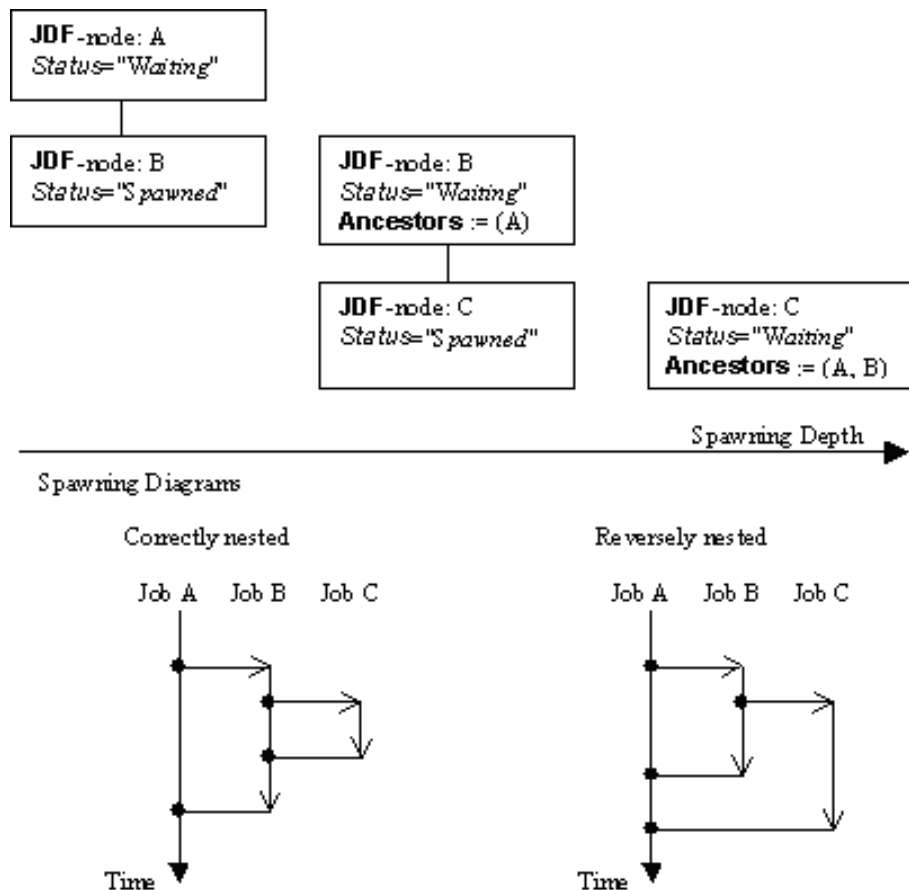


Figure 4-8: Example for a JDF Node structure with nested spawning

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.

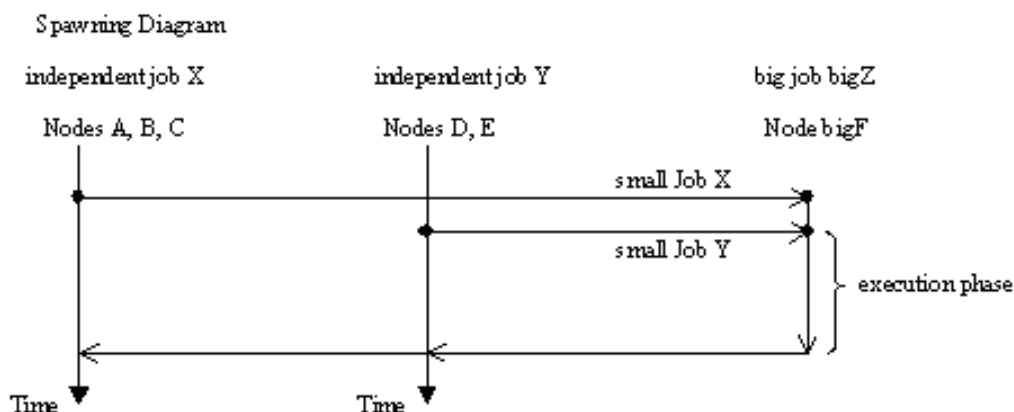
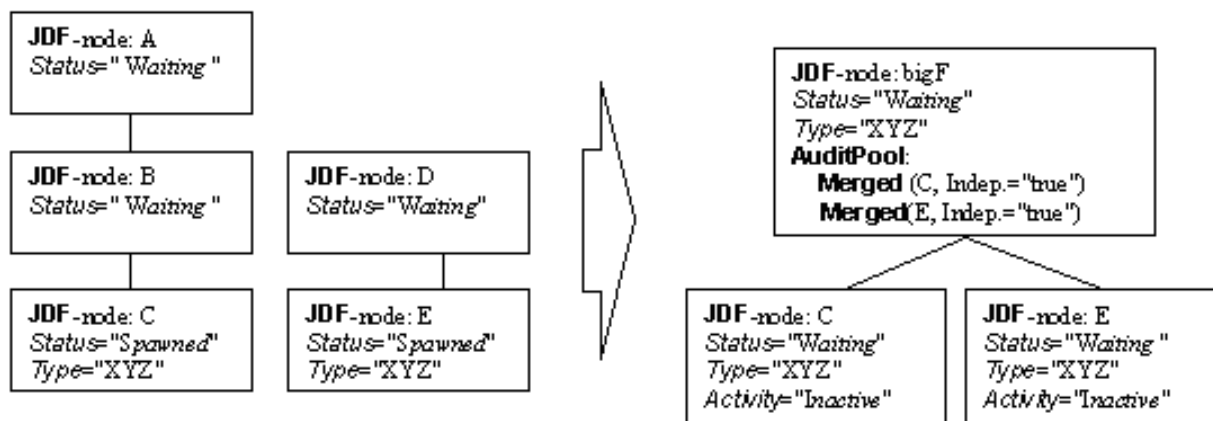


Figure 4-9: Example of the spawning and merging of independent Jobs

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 *jRefDestination* Attribute specified is added to the parent of the original Job. The *jRefDestination* Attribute contains the ID of the Big Job beneath which the spawned Process has been placed. The changes in the parent

are the equivalent of those described in Case 1 except for the specification of the *jRefDestination* Attribute in the *Spawned* Element.

Where necessary, Resource instances **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 *Activation* Attribute 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 *jRefSource* Attribute **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 into one JDF job ticket. The Nodes **MUST** be grouped into a single Process Group Node. This Node can then be spawned and merged as described in the previous sections.

4.5 Node and Resource IDs

All Nodes and Resources **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 A_1 become: $ID = "A1.A"$ $ID = "A1.A.A"$ $ID = "A1.A.B"$, and the IDs of the Small Job A_2 become: $ID = "A2.A"$ $ID = "A2.A.A"$ $ID = "A2.anything"$. All IDs in the Big Job are unique.
- The Agent creates a new Element added to the Small Job A_1 with ID: $ID = "A1.A.C"$. Here the Agent 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 s. The content of the `Notification` Element is described in Table 3-38, "Notification Audit Element," on page 126. For a list of predefined error codes, see "Supported Error Codes in JMF and Notification Elements" on page 891. Further details about error handling are provided in the next four sections.

4.6.1 Classification of Notifications

`Notification` Audit Elements are classified by the `Class` Attribute. Every workflow implementation MUST associate a `Class` with all events on an event-by-event basis. For values, see `Notification/@Class` in Table 3.11.4.5, "Notification," on page 125

4.6.2 Event Description

A description of the event is given by a generic `Comment` Element, which is REQUIRED for the `Notification` Classes `Information`, `Warning`, `Error` or `Fatal`. For example, after a `Process` is aborted, error information describing a `Device` error MAY be logged in the `Comment` Element of the `Notification` Element. If phase times are logged, the `PhaseTime` Element that logged the transition to the `Aborted` state MAY also contain a local `Comment` Element that describes the cause of the `Process` abortion. `PhaseTime` and `Notification` Elements are OPTIONAL Subelements of the `AuditPool`, which is described in Section 3.11, `AuditPool` and `Audit`.

4.6.3 Error Logging in the JDF File

A JDF-compliant `Controller/Agent` SHOULD log an error by inserting a `Notification` Element in the `AuditPool` of the `Node` that generated the error. The `NodeInfo` Resource MAY contain `NotificationFilter` Elements to define the notification events (or, more specifically, errors) that SHOULD be logged.

4.6.4 Error Handling via Messaging (JMF)

A JMF with a `Notification` Element in the `Message` body SHOULD be sent through all persistent channels that subscribed events of `Class Error`. How to subscribe error events via JMF, see Section 5.4.3, `Persistent Channels` and Section 5.8.1, `Events`. Note that this is different from the `NotificationFilter` Elements of the `NodeInfo` Resource, which is defined for logging events by `Notification` Elements to the `AuditPool`.

4.7 Test Running

In JDF, the notion of a test run is similar to the press notion of preflight. The goal is to detect JDF content errors and inconsistencies in a `Job` before the `Job` is executed.

The ability to perform a test run MAY be built into individual `Devices` or `Controllers`. Alternatively, a `Controller` implementation MAY perform test runs on behalf of its `Devices`. A test run MAY be routed through all of the different `Devices` and `Controllers` in a workflow, just as if the test run were a standard execution run. For the routing of `Jobs` and `Nodes` through different `Devices` and `Controllers` for a test, the spawning and merging mechanism MAY also be applied. The `Devices/Controllers` receiving a `Job` read and analyze it WITHOUT initiating execution. Rather, they investigate the content of the `Node` they would execute. A `Device/Controller` with `Agent` capabilities MAY record results into the `AuditPool` associated with a given `Process`.

During test running, the requirements of the `Processes` specified are compared to the capabilities of the `Devices` targeted. A `Device` or `Controller` explicitly tests if the REQUIRED inputs are actually present, valid and without errors. For example, an input requirement might be a `URL` that, when a test run is performed, is found to point to an item that no longer exists in

that location. Test running is meant to prevent errors as a result of that kind of misinformation. It is particularly useful when running expensive or time-consuming Jobs.

It is also possible to test run specific parts of a workflow, or even individual Nodes. An Agent might request a test of certain Nodes by setting the JDF *Activation* Attribute to *TestRun* (see Table 3-5, “JDF Node,” on page 47), 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 5.6, *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 3.11, *AuditPool* and *Audit*.

In principle, execution and test runs might be run simultaneously. For example, one Job Part could be executed while another part requests only a test. JDF also defines an *Activation* value of *TestRunAndGo* that requests a test run and, upon successful completion, automatically initiates processing.

4.7.1 Resource Status During a Test Run

In order to test run a complete set of Nodes, it is sometimes necessary to imply the *Status* of Resources that are produced by prior Nodes. Successful test running does *not* set the *Status* Attribute of a Resource to *Available* unless the Resource actually is available. Nodes that require an Output Resource from a Node that has completed test running for purposes of test running itself can assume that these Resources have a *Status* of *Available* for the purpose of test running as long as the producing Node has a *Status* of *Ready*.

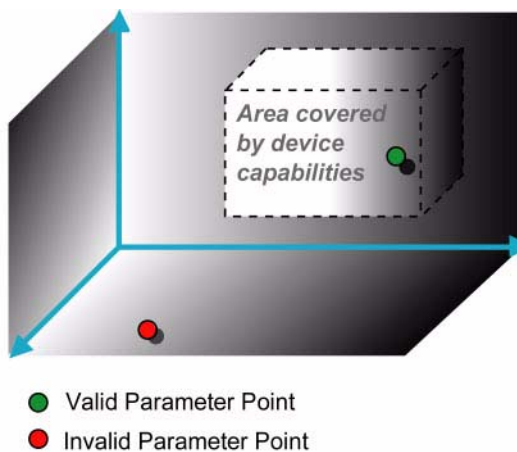


Figure 4-10: Parameter space in Device capabilities^a

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

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).

1. Note that only Devices and Controllers with Agent capabilities can write in a JDF document.

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 776.

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 it 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 851.

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 (Section 1 of 2)

Name	Data Type	Description
<i>AgentName</i> ? New in JDF 1.4	string	The name of the Agent application that generated the JMF. Both the company name and the product name MAY appear, and SHOULD be consistent between versions of the application.
<i>AgentVersion</i> ? New in JDF 1.4	string	The version of the Agent application that generated the JMF. The format of the version string MAY vary from one application to another, but SHOULD be consistent for an individual application.
<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>Message/@ReturnCode</i> = 121, "Unknown DeviceID". 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/ <i>ICSVersions</i> . Values include those from: JDF/ <i>@ICSVersions</i> (Table 3-5, "JDF Node," on page 47).
<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 3.13, 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 Message. 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: <i>"file://master/JMFResponseFolder/Rip1/r12345.jmF"</i>
<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> MAY be modified when a JMF is passed through a proxy.
<i>TimeStamp</i>	dateTime	Time stamp that identifies when the Message was created.

Table 5-1: JMF Element (Section 2 of 2)

Name	Data Type	Description
<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", "1.3" and "1.4". The version of a JMF Message is defined by the highest version of the JMF Message itself or any child Element. For details on JDF versioning see "JDF Versioning" on page 138. 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>Employee</i> ? New in JDF 1.4	element	Employee who created this Message.
Message +	element	Abstract Message Element(s). Note that while a JMF instance MAY include multiple Messages, the order of execution of the Message Elements within a JMF is not deterministic.

5.1.1 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 (Section 1 of 2)

Name	Data Type	Description
<i>AgentName</i> ? New in JDF 1.4	string	The name of the Agent application that generated the JMF. Both the company name and the product name MAY appear, and SHOULD be consistent between versions of the application. If not specified, defaults to the value of JMF/@ <i>AgentName</i> .
<i>AgentVersion</i> ? New in JDF 1.4	string	The version of the Agent application that generated the JMF. The format of the version string MAY vary from one application to another, but SHOULD be consistent for an individual application. If not specified, defaults to the value of JMF/@ <i>AgentVersion</i> .
<i>ICSVersions</i> ? New in JDF 1.4	NMTOKENS	CIP4 Interoperability Conformance Specification (ICS) Versions that this JMF Message complies with. The semantics are identical to JDF/ <i>ICSVersions</i> . If not specified, defaults to the value of JMF/@ <i>ICSVersions</i> . Values include those from: JDF/@ <i>ICSVersions</i> (Table 3-5, "JDF Node," on page 47).
<i>ID</i>	ID	Identifies the Message.
<i>SenderID</i> ? New in JDF 1.4	string	<i>SenderID</i> of the original sender of this Message Element. If not specified, defaults to the <i>SenderID</i> of the parent JMF. <i>SenderID</i> MUST NOT be modified when a JMF is passed through a proxy.
<i>Time</i> ?	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 <i>TimeStamp</i> Attribute of the JMF Element.
<i>Type</i>	NMTOKEN	Name that identifies the Message type. Message types are described in the remainder of this chapter. Values include those from: Table 5-3, "List of JMF Messages," on page 172.

Table 5-2: Abstract Message Element (Section 2 of 2)

Name	Data Type	Description
<i>Version ?</i> New in JDF 1.4	JDFJMFVersion	Text that identifies the version of the JMF Message. The current version of this specification are "1.1", "1.2", "1.3" and "1.4". The version of a JMF Message is defined by the highest version of the JMF Message itself or any child Element. For details on JDF versioning see "JDF Versioning" on page 138. If not specified, defaults to the value of JMF/@Version.
<i>xsi:type ?</i> 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 5.7, Message Template. If omitted then a general definition for the JMF Message will be used. See "JDF Node" on page 45.
Employee ? New in JDF 1.4	element	Employee who created this Message.If not specified, defaults to the value of JMF/Employee.

5.1.2 Structure Diagram

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

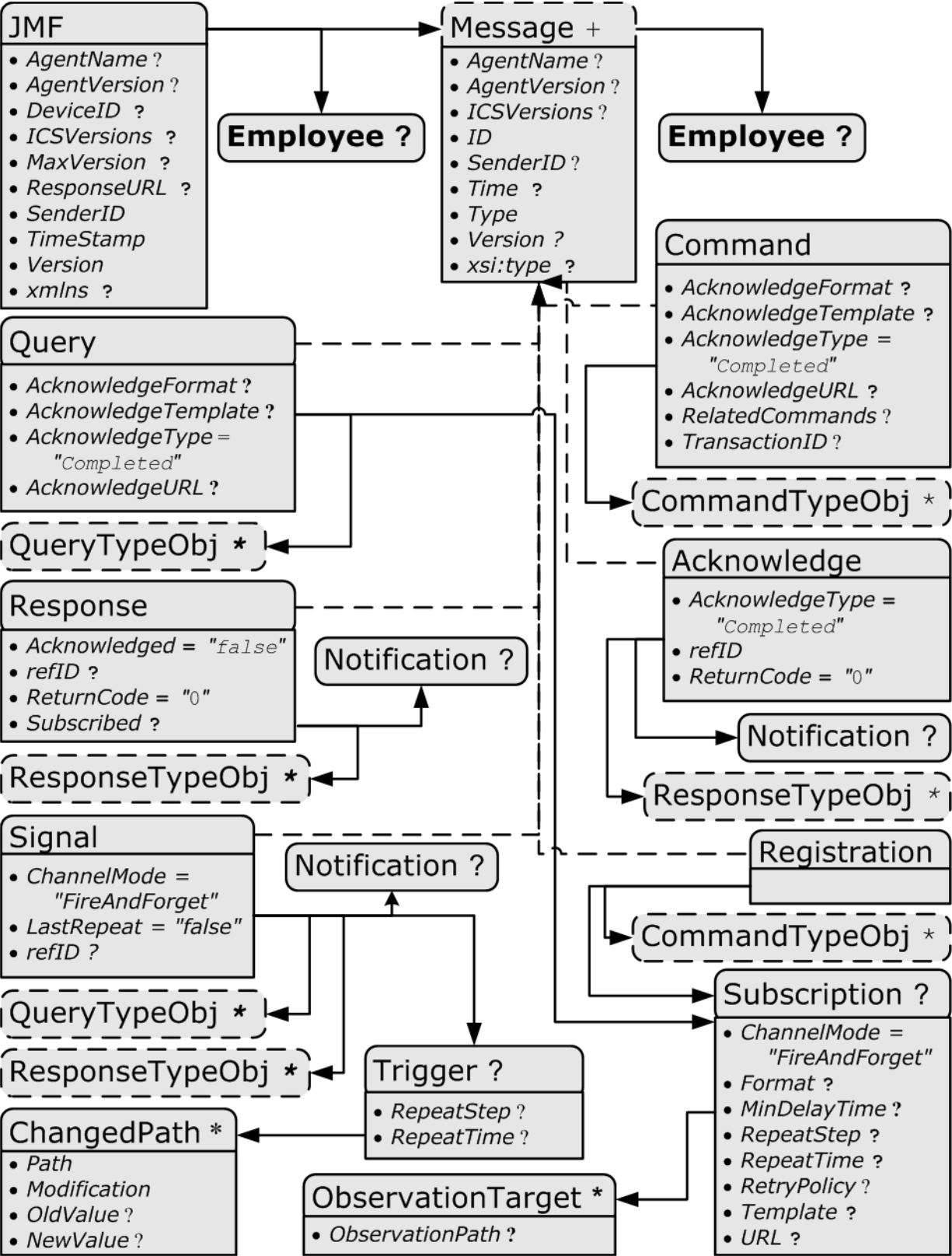


Figure 5-1: JMF Root Element – a diagram of its structure

5.2 List of All JMF Messages

Table 5-3: List of JMF Messages (Section 1 of 3)

Message Type	Page	Family	Description
AbortQueueEntry Modified in JDF 1.2	page 248	CR	The QueueEntry is aborted and remains in the Queue with QueueEntry/@Status = "Aborted".
CloseQueue	page 257	CR	The queue is closed. No Jobs are to be accepted by the queue.
Events	page 189	QRS	Used to subscribe pure events occurring randomly like scanning of a bar code, activation of function keys at a console, error Messages, etc.
FlushQueue	page 257	CQRS	All entries in the queue are removed.
FlushResources New in JDF 1.2	page 208	CRS	Remove temporary Resource from a Device.
ForceGang New in JDF 1.3	page 266	CR	A gang is forced to execute.
GangStatus New in JDF 1.3	page 266	CR	The status of a gang is queried.
HoldQueue	page 259	CR	The queue is held. No Jobs within the queue are to be executed.
HoldQueueEntry	page 248	CR	The entry remains in queue but is not executed until a ResumeQueueEntry Command Message is received.
KnownControllers	page 192	QRS	Returns a list of JMF-capable Controllers.
KnownDevices	page 194	QRS	Returns information about the Devices that are controlled by a Controller.
KnownJDFServices Deprecated in JDF 1.2	page 196	QRS	Returns a list of services (JDF Node Types) that are defined in the JDF specification.
KnownMessages	page 196	QRS	Returns a list of all Messages that are supported by the Controller.
KnownSubscriptions New in JDF 1.4	page 198	QRS	Returns a list of active persistent channels.
ModifyNode New in JDF 1.3	page 209	CRS	modifies details of JDF Nodes.
NewJDF New in JDF 1.2	page 210	CQRS	Initiates or reports modifications of new JDF Nodes.
NodeInfo New in JDF 1.2 Deprecated in JDF 1.3	page 212	CQRS	Initiates or reports modifications of JDF Node information, (e.g., scheduling).
Notification	page 200	S	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.
Occupation	page 212	QRS	Queries the occupation of an employee.
OpenQueue	page 259	CR	The queue is opened. Jobs are to be accepted.

Table 5-3: List of JMF Messages (Section 2 of 3)

Message Type	Page	Family	Description
PipeClose	page 241	CR	Closes a pipe because no further Resources are needed. This is typically used to terminate the producing Process.
PipePause	page 244	CR	Pauses a Process if no further Resources can be consumed or produced.
PipePull	page 242	CR	Requests a new Resource from a pipe.
PipePush	page 243	CR	Notifies that a new Resource is available in a pipe.
QueueEntryStatus Deprecated in JDF 1.2	page 259	QRS	Returns a QueueEntry Element.
QueueStatus	page 259	QRS	Returns the Queue Elements that describe a queue or set of queues.
RemoveQueueEntry	page 249	CR	A Job is removed from the queue.
RequestForAuthentication New in JDF 1.4	page 202	CQRS	Used as a Command to exchange certificates or as a Query to obtain the authentication status of previously exchanged certificates.
RepeatMessages	page 200	QR	Returns a set of previously sent Messages that have been stored by the Controller.
RequestQueueEntry New in JDF 1.2	page 249	CR	A new Job is requested by the Device. This Message is used to signal that a Device has processing Resources available.
Resource	page 214	CGQRS	Queries and/or modifies JDF Resources that are used by a Device, such as Device settings, or by a Job. This Message can also be used to query the level of consumable resources in a Device.
ResourcePull New in JDF 1.2	page 225	CGR	Creates a new QueueEntry from an already existing QueueEntry and submits it to the queue in order to be executed.
ResubmitQueueEntry	page 250	CR	Replaces a queue entry without affecting the entry's parameters. The command is used, for example, for late changes to a submitted JDF.
ResumeQueue	page 259	CR	The queue is activated and queue entries are to be executed.
ResumeQueueEntry	page 251	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	page 251	CR	Returns a Job that had been submitted with a SubmitQueueEntry to the queue that represents the Controller that originally submitted the Job.
SetQueueEntryPosition	page 252	CR	Queues a Job behind a given position n, where n represents a numerical value. "0" = pole position. Priority is set to the priority of the Job at position n.
SetQueueEntryPriority	page 253	CR	Sets the priority of a queued Job to a new value. This does not apply to Jobs that are already running.
ShutDown New in JDF 1.2	page 227	CR	Shuts down a Device.
Status	page 228	QRS	Queries the general status of a Device, Controller or Job.

Table 5-3: List of JMF Messages (Section 3 of 3)

Message Type	Page	Family	Description
StopPersistentChannel	page 206	CR	Closes a persistent channel.
SubmissionMethods	page 260	QR	Queries a list of supported submission methods to the queue.
SubmitQueueEntry	page 253	CR	A Job is submitted to a queue in order to be executed.
SuspendQueueEntry New in JDF 1.2	page 256	CR	The entry is suspended if it is already running. It remains suspended until a ResumeQueueEntry Command Message is received.
Track	page 236	QRS	Queries the location of a given Job or Job Part.
UpdateJDF New in JDF 1.3	page 237	CRS	Synchronizes and relinks modified JDF Nodes.
WakeUp New in JDF 1.2	page 241	CR	Wakes up a Device that is in standby mode.

5.3 JMF Message Families

A Message contains one or more of the following six high level Elements, referred to as **Message Families**, in the Root Node. These families are Query, Response, Signal, Command, Acknowledge and Registration. An explanation of each family is provided in the following sections, along with an encoding example.



Response & Acknowledgement

The terminology used for Message Families contradicts common usage but will be retained for backwards compatibility. The *Response* actually functions as an *Acknowledgement* that a *Command* will be acted upon, while the *Acknowledge* could more properly be named *Completion* or *Result*. The naming was defined to be consistent with HTTP naming conventions so that a *Response* is always transported on an HTTP response in case HTTP is used as the JMF transport protocol layer.

5.3.1 Query

A Query Element is used as a Message that retrieves information from a Controller without changing the state of that Controller. A query is sent to a Controller. After a Query Message is sent, a Response Message is returned. If the Query Message included a Subscription, Signal Messages are sent to the designated URL until a StopPersistentChannel Command Message is sent.

Query with Subscription

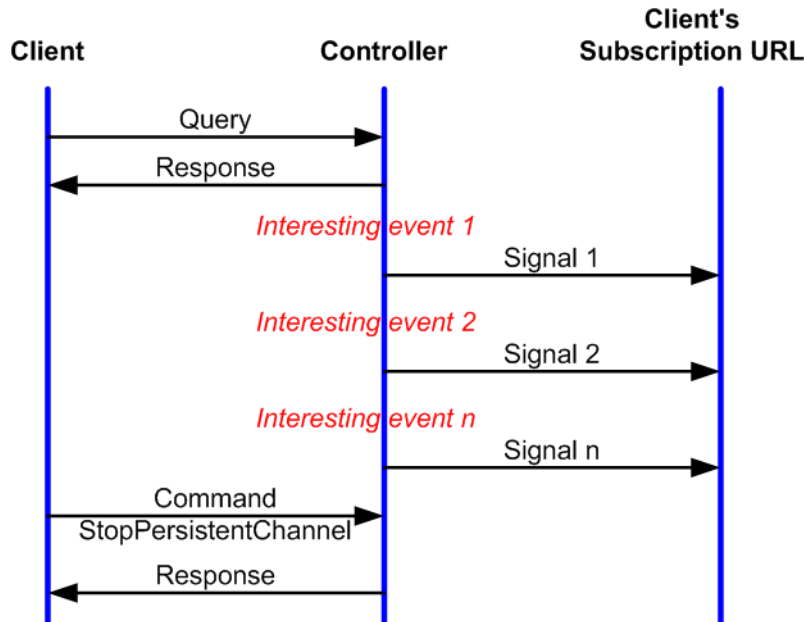


Figure 5-2: Interaction of Messages with a Subscription

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 169. JMF supports a number of well defined query types, and each query type can contain additional descriptive Elements, which are described in Section 5.11 and Section 5.16. The following table shows the content of a Query Message Element:

Table 5-4: Query Message Element (Section 1 of 2)

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. If <i>AcknowledgeFormat</i> is specified, then <i>AcknowledgeTemplate</i> MUST also be specified and <i>AcknowledgeURL</i> MUST NOT be specified. Values are from: "Generating strings with Format and Template" on page 909.
<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. 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. Values are from: "Generating strings with Format and Template" on page 909.

Table 5-4: Query Message Element (Section 2 of 2)

Name	Data Type	Description
<i>AcknowledgeType</i> = "Completed" New in JDF 1.3	enumerations	Defines the actions to be acknowledged. This is necessary mainly for Device-Machine pairs where the Machine is not accessible online. Values are: <i>Received</i> – The Query has been received and understood, (e.g., by an operator). <i>Applied</i> – The Query has been applied to the Machine, (e.g., by an operator). <i>Completed</i> – The Query has been completely responded to.
<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>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 Message Element.
<i>Subscription</i> ?	element	If specified creates a persistent channel. For the structure of a <i>Subscription</i> Element, see Section 5.4.3, Persistent Channels.

Example 5-1: Query Message

The following is an example of a Query Message:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Controller-1"
  Timestamp="2005-07-25T11:38:23+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Query ID="M007" Type="KnownDevices" xsi:type="QueryKnownDevices" />
</JMF>
```

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

A command's Response Message 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 5.3.5, 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 5-5: Response Message Element (Section 1 of 2)

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.

Table 5-5: Response Message Element (Section 2 of 2)

Name	Data Type	Description
<u>refID?</u> <u>Modified in JDF 1.2</u>	NMTOKEN	Copy of the <i>ID</i> Attribute of the initiating Query Message or Command Message to which the Response Message refers. If not specified, the Response Message refers to the entire JMF Message, (e.g., if the JMF was not parseable). <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 891.
<u>Subscribed?</u> <u>Modified in JDF 1.2</u>	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 5.4.3, Persistent Channels.
Notification ?	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 3.11.4.5, Notification.
ResponseTypeObj *	element	Abstract Element that is a placeholder for any descriptive Elements that provide details queried for or details about command execution. If <i>Response/@Acknowledged</i> = "true", ResponseTypeObj Element(s) MAY be missing or incomplete in a Response.

Example 5-2: Response Message for Query

An example of a Response Message to a Command Message is provided in the Section 5.3.4, Command. The encoding example for the Query Message, shown above, might generate the following Response Message:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="RIP-1"
  Timestamp="2000-07-25T11:38:25+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M107" Type="KnownDevices" xsi:type="ResponseKnownDevices"
    refID="M007">
    <DeviceList>
      <DeviceInfo DeviceStatus="Unknown">
        <Device DeviceID="Rip1"/>
      </DeviceInfo>
      <DeviceInfo DeviceStatus="Unknown">
        <Device DeviceID="Rip2"/>
      </DeviceInfo>
    </DeviceList>
  </Response>
</JMF>
```

5.3.3 Signal

A Signal Element is used as a Message, which is equivalent to a combination of a Query Message and a Response Message. It is a unidirectional Message sent on any event to other Controllers. This kind of Message can be used to automatically broadcast status changes.

Controllers can get Signal Messages in one of three ways. The first way is to subscribe for them with an initiating Query Message transmitted via a Message channel that includes a Subscription Element. The second way is to subscribe for them with an initiating Query Message defined in the **NodeInfo** Element of a JDF Node that also includes a Subscription Element (see JMF Elements in See "NodeInfo" on page 657.). The first Query Message is transmitted separately via a mechanism such as HTTP, whereas the second is read together with the corresponding JDF Node. Once the subscription has been established, signals are sent to the subscribing Controllers via persistent

channels. In both cases, however, the Signal Message contains a *refID* Attribute that refers to the persistent channel. The value of the *refID* Attribute identifies the persistent channel that initiated the Signal.

The third way in which a Controller can receive a signal is to have the signal channels hard-wired, for example, by a tool such as a list of Controller-URLs read from an initialization file. For example, signals MAY be generated independently when a service is started, or when sub-Controllers that are newly connected to a network want to inform other 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 5-6: Signal Message Element

Name	Data Type	Description
<i>ChannelMode</i> = "FireAndForget" New in JDF 1.4	enumeration	Specifies reliability of the signal. Values are: <i>FireAndForget</i> – the receiver of the Signal MAY respond using a JMF Response Message. <i>Reliable</i> – Indicates that the Signal is the result of a subscription where reliable signaling was specified in the Subscription Element. The receiver of the Signal MUST respond using a JMF Response Message.
<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 5.4.3, 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 3.11.4.5, Notification. For details about subscribing pure events see Section 5.8.1, Events.
<i>QueryTypeObj</i> * Modified in JDF 1.4	element	This Element is an Abstract Element and a placeholder for any descriptive Elements that provide details for the virtual Query, which, if sent, would convey the same <i>ResponseTypeObj</i> Elements. These Element types are the same as in the Query Message Element. If the <i>QueryTypeObj</i> is required in the corresponding Query, it MUST also be specified in the Signal, even if the <i>QueryTypeObj</i> in the Subscription Message referred to by <i>refID</i> completely defines the context. The Element type of <i>QueryTypeObj</i> is defined by the <i>Type</i> Attribute of the Abstract Message Element.
<i>ResponseTypeObj</i> *	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 5.4.3, Persistent Channels.

5.3.3.1 Element: Trigger

The following table describes the structure of the Trigger Element.

Table 5-7: Trigger Element

Name	Data Type	Description
<i>RepeatStep</i> ?	integer	Recalls the <i>RepeatStep</i> Attribute specified during Subscription of the signal. For details see Table 5-12.
<i>RepeatTime</i> ?	double	Recalls the <i>RepeatTime</i> Attribute specified during Subscription of the signal. For details see Table 5-12.
<i>Added</i> ? Deprecated in JDF 1.2	element	A pool that contains the description of trigger events caused by the adding of Elements like services, Controllers, Devices or Messages. Replaced by <i>ChangedPath</i> in JDF 1.2 and above. See "Signal" on page 1006 for details.
<i>ChangedAttribute</i> * Deprecated in JDF 1.2	element	If a change of an Attribute triggered this signal, this Element describes the Attribute that changed. Replaced by <i>ChangedPath</i> in JDF 1.2 and above. See "Signal" on page 1006 for details.
<i>ChangedPath</i> * 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.
<i>Removed</i> ? 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 <i>ChangedPath</i> in JDF 1.2 and above. See "Signal" on page 1006 for details.

5.3.3.2 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 5-8: 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> . 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.

Example 5-3: Signal Message

The following is an example of a Signal Message:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Press 45"
  Timestamp="2005-07-25T12:28:01+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Signal ID="s123" Type="Status" xsi:type="SignalStatus">
    <StatusQuParams JobID="42" JobPartID="66"/>
  </Signal>
</JMF>
```

```

    <DeviceInfo DeviceStatus="Setup" />
  </Signal>
</JMF>

```

5.3.4 Command

A Comment Element is syntactically equivalent to a Query, but rather than simply retrieving information, it also causes a state change in the target Device. The following table contains the contents of a Command Message. A Response Message is returned immediately after a Command. If the Command included an *AcknowledgeURL*, and the Command was going to take a while, the Device Controller MAY select to return the Response Message with *Acknowledge = "true"*, and send an Acknowledge Message to the *AcknowledgeURL* when the Command completes.

Table 5-9: Command Message Element (Section 1 of 2)

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. If <i>AcknowledgeFormat</i> is specified, then <i>AcknowledgeTemplate</i> MUST also be specified and <i>AcknowledgeURL</i> MUST NOT be specified. Values are from: "Generating strings with Format and Template" on page 909.
<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. 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. Values are from: "Generating strings with Format and Template" on page 909.
<i>AcknowledgeType</i> = "Completed" New in JDF 1.1	enumerations	Defines the actions to be acknowledged. This is necessary mainly for Device-Machine pairs where the Machine is not accessible online. Values are: <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>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 Acknowledge 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>RelatedCommands</i> ? New in JDF 1.4	NMTOKENS	A list of <i>Command/@ID</i> values that need to be processed as a single transaction (in other words all Commands needs to succeed or all need to be rejected). The Commands MUST be processed in the order specified by this attribute. This attribute MUST only appear in the last Command of a transaction. An application SHOULD wait for a reasonable amount of time to collect all related Commands prior to failing a transaction.

Table 5-9: Command Message Element (Section 2 of 2)

Name	Data Type	Description
TransactionID? New in JDF 1.4	string	The ID on the transaction the Command belongs to. All Commands with the same <i>TransactionID</i> MUST either all succeed or all fail
CommandTypeObj *	element	Abstract Element that is a placeholder for any descriptive Elements that provide details of the command.

Example 5-4: ResumeQueueEntry Command Message

The following example demonstrates how a `ResumeQueueEntry` Command Message can cause a Job in a queue to begin executing:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="A3 Printer"
  SenderID="MIS master A" TimeStamp="2000-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Command ID="M009" Type="ResumeQueueEntry" xsi:type="CommandResumeQueueEntry">
    <QueueEntryDef QueueEntryID="job-0032"/>
  </Command>
</JMF>
```

Example 5-5: ResumeQueueEntry Response Message

The following example shows a possible Response Message to the Command Message example above:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
  TimeStamp="2000-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M109" Type="ResumeQueueEntry" xsi:type="ResponseResumeQueueEntry"
    refID="M009">
    <Queue DeviceID="A3 Printer" Status="Full">
      <QueueEntry JobID="job-0032" QueueEntryID="job-0032" Status="Running"/>
    </Queue>
  </Response>
</JMF>
```

5.3.5 Acknowledge

An `Acknowledge` Element is a Message that is an asynchronous answer to a `Command Message` or `Query Message` issued by a Controller. Each `Acknowledge Message` is unidirectional and similar to a `Response Message`, and the *refID* Attribute of each refers to the initiating command. `Acknowledge Messages` are generated if commands with long latency have been executed in order to inform the `Command Message` sender about the results. `Acknowledge Messages` are only generated if the initiating `Command Message` has specified the *AcknowledgeURL* Attribute or a pair of *AcknowledgeFormat* and *AcknowledgeTemplate* Attributes.

Command with Acknowledge

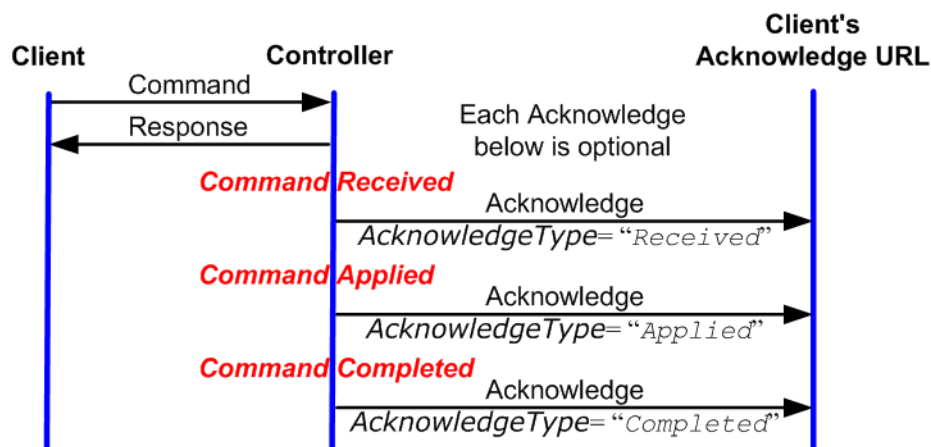


Figure 5-3: Interaction of Command and Acknowledge Messages

They are announced in the *Response Message* to the *Command Message* by the setting the Attribute *Acknowledged = true*.

Table 5-10: 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. Values are: <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 891.
Notification ? Modified in JDF 1.1A	element	Textual description of the command execution. See Section 3.11.4.5, 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 <i>ResponseTypeObj</i> Elements as direct Response Messages.

Example 5-6: Acknowledge Message

The following is an example of an Acknowledge Message:

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
  Timestamp="2000-07-25T12:32:48+02:00" MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Acknowledge ID="M109" Type="PipePush" xsi:type="AcknowledgePipePush" refID="M010">

```



```

    <JobPhase JobID="J1" JobPartID="1" Status="InProgress" />
  </Acknowledge>
</JMF>

```

5.3.6 Registration

[New in JDF 1.3](#)

A Registration Message is a request to the recipient of the JMF to send Command Messages to a Command recipient who is specified in Subscription. See Section 5.4.3.2, Persistent Channels for Commands for details on persistent channels for Commands.

Table 5-11: Registration Message Element

Name	Data Type	Description
CommandTypeObj *	element	Abstract Elements that provide details of the Command that is setup by this Registration Message.
Subscription	element	Creates a persistent channel for a Command. For the structure of a Subscription Element, see Section 5.4.3, Persistent Channels.

5.4 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.

5.4.1 Single Query/Command Response Communication

The handshaking mechanisms for queries and commands are equivalent. The initiating Controller sends a Query Message or Command Message to the target Controller. The target parses the Query or Command and immediately issues an appropriate Response Message. If a Command with long latency is issued, an additional Acknowledge Message MAY be sent to acknowledge when the command has been executed.

5.4.2 Signal and Acknowledge Handshaking

By default, JMF Signal Messages and Acknowledge Messages are “fire and forget.” In case of success, no Response Message is sent by the receiver besides the standard protocol HTTP response with an empty body. If an error occurred at the receiver's end, the Signal or Acknowledge receiver SHOULD return an error Response Message as defined in "Error and Event Messages" on page 187.

If reliable signaling has been specified when the persistent channel is set up (see Table 5-12, “Subscription Element,” on page 185), then the receiver of the JMF Signal MUST respond to the Message using a JMF Response that indicates the appropriate value for the *ReturnCode* Attribute. If the receiver does not respond to the reliable signal, the sender MUST retry the reliable signal, based on the *RetryPolicy* specified in the original Subscription Element. If a Response is received with a *ReturnCode* value other than zero, then the Signal Message MAY have to be retried, depending on the *Error/@ResendPolicy* attribute in the Response.

Any Response related to a Signal or Acknowledge Message MUST NOT specify that an Acknowledge will be sent (the *Acknowledged* attribute MUST be set to false). This is due to the fact that Signal and Acknowledge messages inherently forbid the use of an Acknowledge in Response, since they do not have an *AcknowledgeURL* to indicate where these Acknowledge messages should be sent.

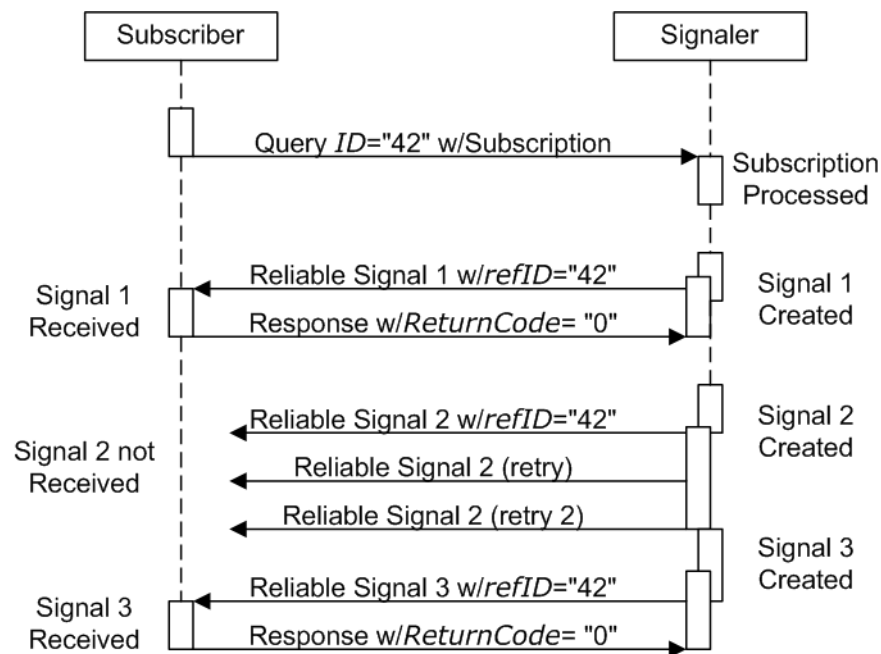


Figure 5-4: Example of Reliable Signaling

5.4.3 Persistent Channels

Query and Command Messages are subscribed for using Subscription Elements.

5.4.3.1 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 Message to the subscribing Controller. Then the responding Controller SHOULD send Signal Messages whenever the condition specified by one of the Attributes in the following table is true. This is referred to as a **persistent channel**. The *refID* Attribute of the Signal is defined by the *ID* Attribute of the Query. In other words, the *refID* of the signal identifies the persistent channel. Any Query can be set up as a persistent channel, although in some cases this might not make sense.

5.4.3.2 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 Message receiver to subsequently send Command Messages to the recipient defined in Subscription/@URL or Subscription/@Format + Subscription/@Template. For instance, an MIS might send a Registration to a prepress workflow system that directs the prepress workflow system to send Command Messages to a press Controller whenever a plate or preview has been produced.

5.4.4 Subscription

Whether or not a responding Controller implements a JDF Persistent Channel as an HTTP/1.1 [RFC2616] persistent connection depends on implementation.

Table 5-12: Subscription Element (Section 1 of 2)

Name	Data Type	Description
<i>ChannelMode</i> = "FireAndForget" New in JDF 1.4	enumerations	Specifies reliability of persistent channel, and whether it is required or just preferred. Ordered list, with most preferred channel mode first. If none of the provided values of <i>ChannelMode</i> are supported by the consumer of the subscription, the Response should indicate <i>ReturnCode</i> 111, which is "Subscription request denied". Values are from: <i>Signal/@ChannelMode</i> . See Table 5-6, "Signal Message Element"
<i>Format</i> ? 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. Values are from: "Generating strings with Format and Template" on page 909. Constraint: if <i>Format</i> is specified, then <i>Template</i> MUST also be specified and <i>URL</i> MUST NOT be specified
<i>MinDelayTime</i> ? New in JDF 1.3	duration	Minimum delay between two subsequent Signal Messages that are triggered by this Subscription. If not specified a Signal SHOULD be fired when any of the conditions described in Subscription is met. Note that Signal Messages that would be fired before <i>MinDelayTime</i> are lost. <i>MinDelayTime</i> SHOULD NOT be applied to Signal Messages that affect costing. Reliable Signal Messages MUST NOT be retried more frequently than the interval specified by <i>MinDelayTime</i> .
<i>RepeatStep</i> ?	integer	Requests an update signal whenever the <i>ActualAmount</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.
<i>RepeatTime</i> ?	double	Requests an update signal every <i>RepeatTime</i> seconds. If defined, the signal is generated periodically independent of any other trigger conditions.
<i>RetryPolicy</i> ? New in JDF 1.4	enumeration	For reliable subscriptions. Indicates whether or not signals should be retried indefinitely, or only until the next Signal from the same Subscription (i.e. has the same <i>@refID</i>) would be sent. <i>RetryPolicy</i> is ignored for non-reliable subscriptions. Values are: <i>DiscardAtNextSignal</i> – if a Signal has not been received, and it is time to send the next Signal related to this Subscription (the next Signal specifies the same <i>@refID</i> value), then discard the current Signal. <i>RetryForever</i> – Continue retrying every Signal indefinitely.
<i>Template</i> ? New in JDF 1.2 <i>Unidirectional</i>	string	A template, used with <i>Format</i> , to define a sequence of generated URLs. Values are from: "Generating strings with Format and Template" on page 909. Constraint: if <i>Template</i> is specified, then <i>Format</i> MUST also be specified and <i>URL</i> MUST NOT be specified.
<i>URL</i> ? 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

Table 5-12: Subscription Element (Section 2 of 2)

Name	Data Type	Description
ObservationTarget *	element	Requests an updating Signal Message whenever the value of one of the Attributes specified in ObservationTarget changes.

5.4.4.1 Element: ObservationTarget

Table 5-13: ObservationTarget Element

Name	Data Type	Description
<i>Attributes?</i> 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. Deprecation note: replaced with <i>ObservationPath</i> in JDF 1.2 and above.
<i>ElementType?</i> 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. Deprecation note: replaced with <i>ObservationPath</i> in JDF 1.2 and above.
<i>ElementIDs?</i> Deprecated in JDF 1.2	NMTOKENS	IDs of the Elements that contain Attributes that can change. Used only in conjunction with a query of the state change of a certain Resource or Node which cannot uniquely be addressed by the other Attributes of this Element. Deprecation note: replaced with <i>ObservationPath</i> in JDF 1.2 and above.
<i>ObservationPath?</i> New in JDF 1.2	XPath	XPath of the Elements or Attributes that are observed. The XPath is in the context of the resulting JMF. If not specified, a Signal is emitted on any change in the abstract <i>ResponseTypeObj</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 Message, 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 Message with a Response Message, set *Subscribed* to "*false*", and set the *ReturnCode* to "111".

Multiple Attributes of a Subscription Element are combined as a Boolean OR operation of these Attributes. For instance, if *RepeatStep* and *ObservationTarget* are both specified, Messages fulfilling either of the requirements are requested. If the Subscription Element contains only a URL, it is up to the emitting Controller to define when to emit Messages.

5.4.5 Creating Persistent Channels in a JDF Node

The *NodeInfo* Element of a JDF Node MAY contain JMF Elements that contains a set of Query or Registration Elements (not Command Elements) 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 Message with *ReturnCode* = "111" and *Subscribed* = "*false*", accompanied by a Notification Element describing the rejection. It is OPTIONAL to emit the Response Message, (e.g., if the Agent parses a Resource request but has no access to the Device information).

5.4.6 Deleting Persistent Channels

A persistent channel is to be deleted by sending a *StopPersistentChannel* Command Message, as described in Section 5.8.10, *StopPersistentChannel*.

5.5 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 5.13.8, *SubmissionMethods*.

Each messaging level encompasses all of the lower messaging levels. Note that the Message levels are provided for information and are not normative.



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.

5.6 Error and Event Messages

If an Acknowledge Message, Command Message, Query Message, Signal Message, or a Registration Message is not successfully handled, a processor must reply with a standardized error response that may contain a Notification Element. Notification Elements, described in detail in Section 3.11.4.5, *Notification*, convey a textual description. The information contained in the Notification Element can be used by a user interface to visualize errors.

The Response Messages and Acknowledge Messages contain a *ReturnCode* Attribute. *ReturnCode* defaults to 0, which indicates that the response is successful. In case of success and in responses to commands an informational Notification Element (*Class* = "*Information*") MAY be provided. In case of a warning, error or fatal error, the *ReturnCode* is greater than 0 and indicates the kind of error committed. In this case, a Notification Element SHOULD be provided. Error codes are defined in "Supported Error Codes in JMF and Notification Elements" on page 891. The responding application SHOULD fill additional Notification/Error Elements that describe the details of the error.

Example 5-7: Response with Notification Element

The following example uses a Notification Element to describe an error:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
  Timestamp="2005-03-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M109" ReturnCode="5" Type="ResumeQueueEntry"
    xsi:type="ResponseResumeQueueEntry" refID="M009">
    <Notification Class="Error" Timestamp="2005-03-25T12:32:48+02:00" Type="Error">
      <Comment>StartJob unsuccessful - Device does not handle commands</Comment>
      <Error ErrorID="1234" Resend="Prohibited">
```

```

        <ErrorData Path="/JMF/Command" ErrorType="Unsupported" />
      </Error>
    </Notification>
  </Response>
</JMF>

```

5.7 Message Template

The previous sections in this chapter provide a description of the overall structure of JMF Messages. This section contains a list of the standard Messages that are defined within the JDF framework. It is OPTIONAL for a JDF-compliant application to support each Signal Message or Query Message described in this list. It is, however, possible to discover which Messages are supported in a workflow. A Controller responds to the KnownMessages Query Message by publishing a list of all the Messages it supports (see Section 5.8.5, KnownMessages, below).

At the beginning of each section there is a table that lists all of the Message types in that category. These tables contain three columns. The first is entitled “Message Type,” and it lists the names of each Message type. The second column is entitled “Family.” The values in this (family) column describe the kind of Message Element that is applicable in the circumstance being illustrated. The following abbreviations are used to describe the values used in the tables below to describe these major Message Element types. (Note: That these are XML Elements that are direct children of the JMF Element.)

- C:** Command
- G:** Registration (“G” is the third letter)
- Q:** Query
- R:** Response and Acknowledge
- S:** Signal

More than one of these values can be valid simultaneously. If that is the case, then all applicable letters are included in the column. Additionally, there are a few special circumstances indicated by particular combinations of these letters. The letters “QR” or “CR” indicate that all Query Messages and Command Messages cause a Response Message to be returned. If the Message can occur as a Signal Message, either from a Subscription or independently, the “Family” field in the table also contains the letter “S”. Finally, the third column provides a description of each Element.

At the beginning of each section describing the contents and function of the Message types listed in the tables described above is a table containing the instantiation (i.e., the type) of all of the abstract Subelements applicable to the Message being described. Each table contains an entry that describes the details of the Query Message or Command Message as well as an additional entry that describes the details of the corresponding response. The tables resemble the following template:

Table 5-14: Template for Message tables

Object Type	Element Name	Description
Abstract Subelement type of the Query or Command.	Name and type of the Subelement that defines specifics of the Query Message or Command Message, followed by a cardinality symbol.	Short description of the Subelement(s) if applicable.
Abstract Subelement type of the Response or Acknowledge.	Name and type of Subelement that contains specific information about the Response Message or Acknowledge Message to a Query Message or Command Message followed by cardinality symbol.	Short description of the Subelement(s) if applicable.

5.7.1 Object Type Column

Each Message in the remainder of this chapter has two cells in the Object Type column. The first is either QueryTypeObj or CommandTypeObj. The second is always a ResponseTypeObj.

5.7.1.1 QueryTypeObj

A QueryTypeObj is an abstract Element that is a placeholder for Subelements of a Query or Signal Message. See Query/QueryTypeObj (Table 5-4) and Signal/QueryTypeObj (Table 5-6). QueryTypeObj also appears in the first row of the Object Type column for each Query Message below. For each such Query Message, the corresponding Elements in the Element Name column are intended to replace the QueryTypeObj in Query/QueryTypeObj or Signal/QueryTypeObj.

5.7.1.2 CommandTypeObj

A CommandTypeObj is an abstract Element that is a placeholder for Subelements of a Command or Registration Message. See Command/CommandTypeObj (Table 5-9) and Registration/CommandTypeObj (Table 5-11). CommandTypeObj also appears in the first row of the Object Type column for each Command Message below. For each such Command Message, the corresponding Elements in the Element Name column are intended to replace the CommandTypeObj in Command/CommandTypeObj or Registration/CommandTypeObj.

5.7.1.3 ResponseTypeObj

A ResponseTypeObj is an abstract Element that is a placeholder for Subelements of a Response, Signal or Acknowledge Message. See Response/ResponseTypeObj (Table 5-5), Signal/ResponseTypeObj (Table 5-6) and Acknowledge/ResponseTypeObj (Table 5-10). CommandTypeObj also appears in the second row of the Object Type column for each Message below. For each such Message, the corresponding Elements in the Element Name column are intended to replace the ResponseTypeObj in the Response/ResponseTypeObj, Signal/ResponseTypeObj or Acknowledge/ResponseTypeObj.

5.8 Messages for Events and Capabilities

The Message types of the following table are defined in order to exchange metadata about Controller or Device abilities and for general communication.

Table 5-15: Messages for events and capabilities

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 active persistent channels.
KnownMessages	QRS	Returns a list of all Messages that are supported by the Controller.
KnownSubscriptions New in JDF 1.4	QRS	Returns a list of active persistent channels.
Notification	QRS	Generally sent as Signals. A Query allows Subscriptions for Notification Messages.
RepeatMessages	QR	Returns a set of previously sent Messages that have been stored by the Controller.
RequestForAuthentication New in JDF 1.4	CQR	Used as a Command to exchange certificates or as a Query to obtain the authentication status of previously exchanged certificates.
StopPersistentChannel	CR	Closes a persistent channel.

5.8.1 Events

The Events Message type is intended to be used to query for supported Signal Messages and to subscribe for asynchronous, randomly occurring Signal Messages of a Device or Controller. These events are described in Section 4.6.1, Classification of Notifications and can only be transmitted via Signal Messages. If the Query Message contains a Subscription Element, a NotificationFilter Element is combined by a logical AND operation with the Subscription Element for selective subscriptions. An empty Events Message (without a Subscription and NotificationFilter Element) can be used to query for all events as described in "Notification" on page 200, which are supported by a Device or Controller. If all signals are requested, a NotificationFilter with *SignalTypes* = "all" MUST be included in the Query Message.

The Controller that subscribes for Events Messages receives Signal Messages. In JDF 1.2, the Events Message was enhanced to subscribe for all types of Signal Messages, not only Notification Signals. The event type and values of Notification Messages are provided by specifying a *Type* Attribute and an Abstract NotificationDetails Element in the Notification Element, as described in Section 3.11.4.5, Notification. Possible NotificationDetails Elements are defined in "NotificationDetails" on page 883.

Table 5-16: Events Message

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

Example 5-8: Query with Subscription to All Events

Example of a Subscription of all Events and the Response Message, including the feature of subscribing for all Messages by setting NotificationFilter/@*SignalTypes* = "all":

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
  Timestamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Query ID="M170" Type="Events" xsi:type="QueryEvents">
    <Subscription URL="http://www.anycompany.com/MIS/JMF/JobTracker"/>
    <NotificationFilter Classes="Event Warning Error Fatal" SignalTypes="All"/>
  </Query>
</JMF>
```

Example 5-9: Response for Subscription to All Events

The Response Message to the previous Query Message:

```
<JMF SenderID="A3 Printer" Timestamp="2005-07-25T12:32:48+02:00"
  xmlns="http://www.CIP4.org/JDFSchema_1_1"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M1001" refID="M170" Type="Events" xsi:type="ResponseEvents"
    xmlns:anycompany="http://www.anycompany.com">
    <NotificationDef Classes="Warning Error Fatal" Type="Error"/>
    <NotificationDef Classes="Event" Type="FCNKey"/>
    <NotificationDef Classes="Event Error" Type="Barcode"/>
    <NotificationDef Classes="Event" Type="SystemTimeSet"/>
    <NotificationDef Classes="Event" Type="anycompany:PrivateEvent_1"/>
    <NotificationDef Classes="Event" Type="anycompany:PrivateEvent_2"/>
    <NotificationDef Classes="Event" Type="anycompany:PrivateEvent_2"/>
    <NotificationDef SignalType="Status"/>
    <NotificationDef SignalType="Resource"/>
  </Response>
</JMF>
```


5.8.1.1 Element: NotificationFilter

Table 5-17: NotificationFilter Element

Name	Data Type	Description
<i>Classes</i> ?	enumerations	Defines the set of Notification/@ <i>Class</i> to be queried/subscribed for. Default behavior: all Notification Classes are subscribed to. Values are: <i>Event</i> <i>Information</i> <i>Warning</i> <i>Error</i> <i>Fatal</i> Constraint note: If the values both <i>Classes</i> and <i>Types</i> are lists of values, the NotificationFilter defines an OR of all combinations.
<i>DeviceID</i> ? 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.
<i>JobID</i> ?	string	JobID of the Job whose Messages are queried/subscribed.
<i>JobPartID</i> ?	string	JobPartID of the Job whose Messages are queried/subscribed.
<i>MilestoneTypes</i> New in JDF 1.4	NMTOKENS	Matching Milestone types are returned and/or subscribed to. Default value is: all supported <i>MilestoneType</i> values. Values include those from: Table C-20, "MessageEvents and MilestoneType Values," on page 886.
<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, NotificationFilter applies to all Jobs.
<i>SignalTypes</i> = "Notification" New in JDF 1.2	NMTOKENS	Possible Signal/@ <i>Type</i> values of the subscribed Messages. Values include: <i>all</i> – specifies that all Signals, regardless of <i>Type</i> are queried/subscribed. Values include those from: Message/@ <i>Type</i> . Note: the values are limited to Signal Messages.
<i>Types</i> ?	NMTOKENS	Matching notification types are returned/subscribed. Default value is: all supported notification types. Values include those from: Table C-11, "List of NotificationDetail Elements," on page 883.
Part * New in JDF 1.2	element	Part 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 150.

5.8.1.2 Element: NotificationDef

Table 5-18: NotificationDef Element

Name	Data Type	Description
<i>Classes</i> ? Modified in JDF 1.2	enumerations	Notification/@ <i>Class</i> of the Notification in a Signal. Values are: <i>Event</i> <i>Information</i> <i>Warning</i> <i>Error</i> <i>Fatal</i> Constraint: <i>Classes</i> MUST NOT be specified unless <i>SignalType</i> = "Notification". For details, see Section 4.6.1, Classification of Notifications.
<i>SignalType</i> = "Notification" New in JDF 1.2	NMTOKEN	Signal/@ <i>Type</i> value of the subscribed Message. Values include those from: Message/@ <i>Type</i> . Note: the values are limited to Signal Messages
<i>Type</i> ? Modified in JDF 1.2	NMTOKEN	Notification type, that is the name of the Element derived from the Abstract NotificationDetails Element. Constraint: <i>Type</i> MUST NOT be specified unless <i>SignalType</i> = "Notification". Values include those from: Table C-11, "List of NotificationDetail Elements," on page 883.

5.8.2 KnownControllers

The *KnownControllers* Query Message requests information about the Controllers and/or Devices that are known to the Controller that is queried and can be directly accessed by JMF messaging. *KnownControllers* is intended to be used with a "registration" server. A processor that needs information about its system environment can query a registration server for a list of known Controllers and/or Devices. A single Controller or Device that supports multiple URLs or protocols is defined using multiple *JDFController* Elements with the same *ControllerID* Attribute. This list can subsequently be iterated using the other Process registration queries in this section. The URL of the master registration server MUST be defined using a method outside of JDF.

Table 5-19: KnownControllers Message

Object Type	Element Name	Description
QueryTypeObj	ControllerFilter New in JDF 1.4	Allows filtering the Response.
ResponseTypeObj	JDFController *	Known Controllers.

5.8.2.1 Element: ControllerFilter

[New in JDF 1.4](#)

Table 5-20: ControllerFilter Element

Name	Data Type	Description
<i>ControllerID?</i>	string	Only Controllers whose <i>ControllerID</i> or Devices whose <i>DeviceID</i> matches this <i>ControllerID</i> should be returned in the Response. If this Attribute is not specified, the Response should contain JDFController Elements for all known Controllers and/or Devices.
<i>URLTypes?</i>	enumerations	Only URL's whose JDFController/@ <i>URLType</i> Attribute in the Response match one of the values in this @ <i>URLTypes</i> Attribute should be returned in the Response. If this Attribute is not specified, the Response should contain JDFController Elements with URLs of any type. Values are from: JDFController/@ <i>URLType</i> .

5.8.2.2 Element: JDFController

Table 5-21: JDFController Element

Name	Data Type	Description
<i>ControllerID?</i> New in JDF 1.2	string	String that identifies the Controller or Device. The <i>ControllerID</i> is used as the <i>SenderID</i> of JMF Messages that are produced by this Controller or Device. A JMF Message that is intended for a specific Controller SHOULD specify the Controller's <i>ControllerID</i> value or the Device's <i>DeviceID</i> in JMF/@ <i>DeviceID</i> .
<i>URL</i>	URL	URL of the Controller or Device. If the URL scheme is " <i>file:</i> ", <i>URL</i> MUST specify the directory where the JMF Messages are to be deposited.
<i>URLType?</i> New in JDF 1.4	enumeration	Identifies the purpose of this URL. Values are: <i>JDFError</i> <i>JDFInput</i> <i>JDFOutput</i> <i>JMF</i> <i>SecureJMF</i>

Example 5-10: KnownControllers Query

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="A3 Printer"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Query ID="Q1" Type="KnownControllers" SenderID="MIS"
    xsi:type="QueryKnownControllers">
    <ControllerFilter ControllerID="PrintController1" URLTypes="JMF SecureJMF"/>
  </Query>
</JMF>
```

Example 5-11: KnownControllers Response

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M1" Type="KnownControllers" xsi:type="ResponseKnownControllers">
```

```

    refID="Q1" SenderID="RegistrationServer">
  <JDFController ControllerID="PrintController1"
    DescriptiveName="Printer Controller"
    URL="http://www.anycompany.com/controller"
    URLType="JMF" />
  <JDFController ControllerID="PrintController1"
    DescriptiveName="Printer Controller"
    URL="https://www.anycompany.com/controller/secure"
    URLType="SecureJMF" />
</Response>
</JMF>

```

5.8.3 KnownDevices

The KnownDevices Query Message requests information about the Devices that are controlled by a Controller. If a high level Controller controls lower level Controllers, it SHOULD also list the Devices that are controlled by these. The response is a DeviceList which is list of DeviceInfo Elements controlled by the Controller that receives the query, as demonstrated in Example 5-12.

Table 5-22: 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 Modified in JDF 1.1A	DeviceList ?	The list of known Devices. Modification note: before JDF 1.1A this was "Device*". It was changed due to inconsistencies of the inheritance model in the JDF schema.

Example 5-12: KnownDevices Response

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M1" refID="Q1" Type="KnownDevices" xsi:type="ResponseKnownDevices">
    <DeviceList>
      <DeviceInfo DeviceStatus="Unknown">
        <Device DeviceID="Joe SpeedMaster"
          DeviceType="Heidelberg SM102/6 rev. 47"/>
      </DeviceInfo>
    </DeviceList>
  </Response>
</JMF>

```

5.8.3.1 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 5-23: 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. Values are: <i>None</i> – Provide only DeviceInfo/@DeviceID and DeviceInfo/@DeviceStatus. <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 Device Resources 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 87.

5.8.3.2 Element: DeviceList

[New in JDF 1.1A](#)

The DeviceList Element contains a list of information about Devices that are returned.

Table 5-24: 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 5-70, "DeviceInfo Element," on page 230 in the Message description Section 5.9.9, Status.

5.8.4 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 1010 for the details of this deprecated Message.

5.8.5 KnownMessages

The `KnownMessages` Query Message returns a list of all Message types that are supported by the Controller.

Table 5-25: 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 <code>MessageService</code> Elements MAY be specified for a Message with a given <code>JMF/@Type</code> .

5.8.5.1 Element: KnownMsgQuParams

The flags of the `KnownMsgQuParams` Element specify the Message Families to include in the response list. Multiple flags are allowed.

Table 5-26: KnownMsgQuParams Element

Name	Data Type	Description
ChannelMode ? New in JDF 1.4	enumerations	Limits the list based on supported channel modes for the Message. Values are from: <code>Signal/@ChannelMode</code> . See Table 5-6, "Signal Message Element"
<code>Exact = "false"</code> New in JDF 1.1	boolean	Requests an exact description of the known Messages. If <code>true</code> , the response also contains the requested <code>DevCaps</code> of the Messages.
<code>ListCommands = "true"</code>	boolean	Lists all supported Command types.
<code>ListQueries = "true"</code>	boolean	Lists all supported Query types.
<code>ListRegistrations = "true"</code> New in JDF 1.3	boolean	Lists all supported Registration Message types.
<code>ListSignals = "true"</code>	boolean	Lists all supported Signal types.
<code>Persistent = "false"</code>	boolean	If <code>true</code> , only lists Messages that can use persistent channels. If <code>false</code> , ignores the ability to use persistent channels.

5.8.5.2 Element: MessageService

The response is a list of `MessageService` Elements, one for each supported Message type. The flags of the `MessageService` Response Message Element are set in each `MessageService` entry. They define the supported usage of the Message by the Controller. Note that no `Response` Attribute is included in the list, since the capability to process one of the other Message Families implies the capability to generate an appropriate `Response` Message. Multiple flags are allowed.

Table 5-27: MessageService Element (Section 1 of 3)

Name	Data Type	Description
<code>Acknowledge = "false"</code> New in JDF 1.1	boolean	If <code>true</code> , the Device supports asynchronous <code>Acknowledge</code> answers to this Message.

Table 5-27: MessageService Element (Section 2 of 3)

Name	Data Type	Description
ChannelMode? New in JDF 1.4	enumerations	Specifies the supported channel modes for the Message. Values are from: Signal/@ChannelMode. See Table 5-6, "Signal Message Element"
Command = "false"	boolean	If "true", the Message is supported as a Command.
GenericAttributes? New in JDF 1.3	NMTOKENS	List of generic Attributes that are supported and unrestricted by the Device implementation. Descriptions of Attributes that appear in State Elements (see the following Section 7.3.7, State) overwrite the description in <i>GenericAttributes</i> , which MUST NOT be specified if KnownMsgQuParams/@Exact = "false"
JMFRole? New in JDF 1.3	enumeration	The role of the Device that responds with the MessageService. Values are: <i>Receiver</i> – The Device that responds to KnownMessages responds to the Message specified in <i>Type</i> . This MessageService specifies Query Messages, Signal Messages, Command Messages and Registration Messages that the Device understands. <i>Sender</i> – The Device that responds to KnownMessages is the originator of the Message specified in <i>Type</i> . This MessageService specifies Response Elements and Acknowledge Elements that the Device understands as a Response to these Messages. – I understand these responses to Messages that I send.
Persistent = "false"	boolean	If true the Message is supported as a persistent channel.
Query = "false"	boolean	If true the Message is supported as a Query.
Registration = "false" New in JDF 1.3	boolean	If true the Message is supported as a Registration Message.
Signal = "false"	boolean	If true the Message is supported as a Signal.
Type	NMTOKEN	Type of the supported Message. Extension types are specified by stating the namespace prefix in <i>Type</i> Values include those from: Message/@Type.
URLSchemes? New in JDF 1.3	NMTOKENS	List of schemes supported for the Message defined by this MessageService. Values include: <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 Action Elements for use as constraints. For details on Action Elements, see "ActionPool" on page 780. ActionPool MUST NOT be specified if KnownMsgQuParams/@Exact = "false".
DevCapPool? New in JDF 1.3	element	Pool of DevCap Elements that can be referenced from multiple Elements within the DeviceCap structure. DevCapPool MUST NOT be specified if KnownMsgQuParams/@Exact = "false".

Table 5-27: MessageService Element (Section 3 of 3)

Name	Data Type	Description
DevCaps * New in JDF 1.1	element	Specifies the restrictions of the parameter space of the supported Messages. For details on using DevCaps, see Section 7.3.5, DevCaps. DevCaps MUST NOT be specified if KnownMsgQuParams/@Exact = "false".
ModulePool ? New in JDF 1.3	element	Pool of ModuleCap Elements that specify the availability of a given Module. See Table 7-399, "ModuleCap Element," on page 782 for details of ModuleCap. ModulePool MUST NOT be specified if KnownMsgQuParams/@Exact = "false".
State * New in JDF 1.4	element	State Elements that define the parameter space that is covered by the Device. One State Element 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> .
TestPool ? New in JDF 1.3	element	Container for zero or more Test Elements that are referenced from ActionPool/Action Elements. TestPool MUST NOT be specified if KnownMsgQuParams/@Exact = "false".

Example 5-13: KnownMessages Response

The following is an example of a Response Message to a KnownMessages Query Message:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M1" Type="KnownMessages" xsi:type="ResponseKnownMessages" refID="Q1">
    <MessageService JMFRole="Receiver" Query="true" Type="KnownMessages"/>
    <MessageService JMFRole="Receiver" Persistent="true" Query="true" Signal="true"
      Type="Status"/>
  </Response>
</JMF>
```

5.8.6 KnownSubscriptions

[New in JDF 1.4](#)

The KnownSubscriptions JMF enables Controllers to query Devices for a list of active persistent channels.

Table 5-28: KnownSubscriptions Message

Object Type	Element Name	Description
QueryTypeObj	SubscriptionFilter ?	Refines the query for known Messages. If not specified, list all supported Message types.
ResponseTypeObj	SubscriptionInfo *	List of active persistent channels.

5.8.6.1 Element: SubscriptionFilter

[New in JDF 1.4](#)

The SubscriptionFilter Element is a filter to limit the list of SubscriptionInfo Elements that are returned in the KnownSubscriptions Response.

Table 5-29: SubscriptionFilter Element (Section 1 of 2)

Name	Data Type	Description
ChannelID ?	NMTOKEN	ChannelID of the persistent channel to be queried. If the channel has been created with a Query Message, the ChannelID specifies the ID of the Query Message (identical to the refID of the Response Message)

Table 5-29: SubscriptionFilter Element (Section 2 of 2)

Name	Data Type	Description
<i>DeviceID</i> ?	string	Only subscription from Devices or Controllers with a matching <i>DeviceID</i> Attribute are queried
<i>Families</i> ?	enumerations	Only Subscriptions with the Family (Signal or Command) listed are queried
<i>JobID</i> ?	string	<i>JobID</i> of the JDF Node that Messages are subscribed for. If not specified, Subscriptions are returned for all <i>JobID</i> values.
<i>JobPartID</i> ?	string	<i>JobPartID</i> of the JDF node that Messages are subscribed for. If not specified, Subscriptions are returned for all <i>JobPartID</i> values.
<i>MessageTypes</i> ?	NMTOKENS	List of <i>Message/@Type</i> values of the subscribed messages. If not specified, Subscriptions are returned for all message types.
<i>QueueEntryID</i> ?	string	<i>QueueEntryID</i> of the Job whose Subscriptions are 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, <i>KnownSubscriptions</i> applies to all persistent channels that were established.
<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 <i>URL</i> are queried.
<i>Part</i> *	element	<i>Part</i> Elements that describe the Partition of the Job whose Subscriptions are queried. For details on Node Partitions, see Section 4.3.2 “Partial Processing of Nodes with Partitioned Resources” on page 150.

5.8.6.2 Element: SubscriptionInfo

[New in JDF 1.4](#)

A SubscriptionInfo Element describes the Subscription details of a persistent channel.

Table 5-30: SubscriptionInfo Element (Section 1 of 2)

Name	Data Type	Description
<i>ChannelID</i>	NMTOKEN	<i>ChannelID</i> specifies the ID of the Query message (identical to the <i>refID</i> of the Signal or Response Message).
<i>SenderID</i>	string	Device or Controller <i>SenderID</i> .
<i>Family</i>	enumeration	Specifies whether the persistent channel is a Signal or Command. Values are: <i>Signal</i> <i>Command</i>
<i>JobID</i> ?	string	<i>JobID</i> of the JDF Node that this PersistentChannel applies to. If not specified, this PersistentChannel applies to all <i>JobID</i> values
<i>JobPartID</i> ?	string	<i>JobPartID</i> of the JDF Node that this PersistentChannel applies to. If not specified, this PersistentChannel applies to all <i>JobPartID</i> values.
<i>MessageType</i>	NMTOKEN	<i>Message/@Type</i> value of the subscribed Messages.
<i>QueueEntryID</i> ?	string	<i>QueueEntryID</i> of the <i>QueueEntry</i> that this PersistentChannel applies to. If not, specified, this PersistentChannel applies to all <i>QueueEntryID</i> values.

Table 5-30: SubscriptionInfo Element (Section 2 of 2)

Name	Data Type	Description
Part *	element	Part Elements that describe the Partition of the JDF Node that this Persistent Channel applies to. For details on Node Partitions, see Section 4.3.2 “Partial Processing of Nodes with Partitioned Resources” on page 150.
Subscription	element	The Subscription Element that describes the persistent channel.

5.8.7 Notification

Notification Messages are generally sent as Signals. The Query is defined to allow subscriptions for Notification Messages.

Notification Elements are also used to signal usual events due to any activities of a Device, operator, etc., (e.g., scanning a bar code). Such pure events can be subscribed to by the Events Message described in Section 5.8.1, Events. Such a Signal always has a *Type* = "Notification":

Table 5-31: Notification Signal

Object Type	Element Name	Description
QueryTypeObj	NotificationFilter ? New in JDF 1.4	Defines the types of Notification Elements that should be returned
ResponseTypeObj	Notification	Notification that describes the error. See Section 3.11.4.5, "Notification" on page 125.

Example 5-14: Notification Signal

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
  Timestamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Signal ID="S1" Type="Notification" xsi:type="SignalNotification">
    <Notification Class="Event" Timestamp="2005-07-25T12:32:48+02:00"
      Type="Barcode">
      <Comment>Palette completed</Comment>
      <Barcode Code="99923AAA123"/>
    </Notification>
  </Signal>
</JMF>
```

5.8.8 RepeatMessages

The RepeatMessages Query Message returns a list of Messages that have been previously sent by the Controller. The OPTIONAL MsgFilter Element allows the list to be filtered. The list of JMF Messages that fulfill the filter criteria can be sorted by time, with the most recent listed first. This specification places no requirements on the size of the Message buffer of a Controller that supports RepeatMessages.

Table 5-32: RepeatMessages Message

Object Type	Element Name	Description
QueryTypeObj	MsgFilter ?	A filter for the Messages to be repeated. For details, see Section 5.8.1, Events.
ResponseTypeObj	Message *	The recent Messages queried.

5.8.8.1 Element: MsgFilter

If the returned list is incomplete because the parameters supplied in the `MsgFilter` Element cannot be fulfilled by the application, the `ReturnCode` is either 108 (empty list) or 109 (incomplete list) and SHOULD be flagged as a warning with `Notification[@Class = "Warning" and @Type = "Error"]`.

Table 5-33: MsgFilter Element

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. Values are: <i>Acknowledge</i> <i>Command</i> – Repeat Command Messages that are triggered by a Registration Message. New in JDF 1.3 <i>Response</i> <i>Signal</i> <i>All</i> – Response, Signal and Acknowledge 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. Values include those from: <i>Message/@Type</i> .
<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, <code>MsgFilter</code> applies to all Jobs that will be processed by the receiver.
<i>ReceiverURL?</i>	URL	URL for which the Messages are intended.
<i>Part</i> * 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 150.

Example 5-15: RepeatMessages Response

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

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
  Timestamp="2005-07-25T12:32:48+02:00" MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="RepMsg" Type="RepeatMessages" xsi:type="ResponseRepeatMessages">
    <Response ID="R1" Time="2000-06-14T11:00:01+02:00" Type="Status"
      xsi:type="ResponseStatus"/>
    <Response ID="R2" Time="2000-06-14T10:50:22+02:00" Type="Occupation"
      xsi:type="ResponseOccupation"/>
  </Response>
</JMF>
```

```

<Signal ID="R3" Time="2000-06-14T08:20:23+02:00" Type="Resource"
  xsi:type="SignalResource"/>
<Signal ID="R4" Time="2000-06-14T03:01:22+02:00" Type="Notification"
  xsi:type="SignalNotification"/>
</Response>
</JMF>

```

5.8.9 RequestForAuthentication

[New in JDF 1.4](#)

The RequestForAuthentication Message can be used as a Command to exchange certificates or as a Query to obtain the authentication status of previously exchanged certificates. Acknowledge Messages MUST NOT be used to respond to a RequestForAuthentication Command or RequestForAuthentication Query. In other words, the Response element MUST NOT specify *Acknowledged* = "true". If it is not possible to confirm authentication before the HTTP channel times out, the *ReturnCode* MUST be 304, which means "Authentication pending".

5.8.9.1 RequestForAuthentication Command

[New in JDF 1.4](#)

The RequestForAuthentication Command Command is used to request authentication and trust of a certificate that is provided in the RequestForAuthentication Command. The sender of the Command is identified by the *SenderID* attribute in the JMF Element that contains the RequestForAuthentication Command. The sender MAY be authenticated as both a client and as a server, and a separate certificate MUST be provided by the sender for each role that the sender wishes to use

If a RequestForAuthentication Command is received over a secure channel, and a previous RequestForAuthentication Command has already been received, the previous RequestForAuthentication Command SHOULD be ignored, and any certificates associated with the prior Command SHOULD be considered untrusted. This allows for a party that is currently trusted to update its certificate as needed (such as when the previous certificate is about to expire),

Once authentication has been established between two parties, any RequestForAuthentication Command that is sent over a non-secure channel MUST result in error 305, which is "Authentication already established". Other *Reason* values MAY be supported over secure channels.

Table 5-34: RequestForAuthentication Command Message

Object Type	Element Name	Description
CommandTypeObj	AuthenticationCmdParams	Details of the certificate of the sender.
ResponseTypeObj	AuthenticationResp ?	<i>ReturnCode</i> = 0 indicates "I trust you". The initial response to a RequestForAuthentication Command MUST include a fully specified AuthenticationResp Element.

5.8.9.1.1 Element: AuthenticationCmdParams

Table 5-35: AuthenticationCmdParams Element

Name	Data Type	Description
<i>AuthenticationType</i>	enumeration	<p>Values are:</p> <p><i>AsClient</i> – Sender of the Message wishes to be authenticated as a client that initiates HTTP requests. Command includes the sender's client certificate, the Response will include the responders server certificate.</p> <p><i>AsServer</i> – Sender of Message wishes to be authenticated as a server that responds to HTTP requests. Command includes the sender's server certificate, the response will include the responders client certificate.</p>
<i>Reason</i>	enumeration	<p>Used to indicate the reason for sending this Message.</p> <p>Values are:</p> <p><i>InitiateConnection</i> – the client wishes to exchange certificates with the server.</p> <p><i>ClientCertificateExpired</i> – the previously-sent client certificate has expired.</p> <p><i>ServerCertificateExpired</i> – the previously-received server certificate has expired.</p> <p><i>ClientHostnameMismatch</i> – the client certificate's Common Name couldn't be resolved to match the IP address or domain name from which the request came.</p> <p><i>ServerHostnameMismatch</i> – the server certificate's Common Name couldn't be resolved to match the IP address or domain name from which the response came.</p> <p><i>ClientCertificateRevoked</i> – the previously-sent client certificate has been revoked.</p> <p><i>ServerCertificateRevoked</i> – the previously-received server certificate has been revoked.</p> <p><i>Other</i> – some other reason. Use <i>ReasonDetails</i> for further explanation.</p>
<i>ReasonDetails ?</i>	string	Further details on the reason for this Message.
<i>SecureURL ?</i>	URL	URL of the port of the Command sender that will accept JMF Messages via the HTTPS protocol. This Attribute MUST be specified when the sender of the RequestForAuthentication Command has specified <i>AuthenticationCmdParams/@AuthenticationType = "AsServer"</i> .
<i>Certificate ?</i>	telem	<p>The requester's certificate.</p> <p>If <i>AuthenticationType = "AsClient"</i>, this certificate MUST be the requester's client certificate. If <i>AuthenticationType = "AsServer"</i>, this certificate MUST be the requester's server certificate.</p>

5.8.9.1.2 Element: Certificate

Table 5-36: Certificate Element

Name	Data Type	Description
	text	The certificate in PEM MD5 format. Implementation Note: there MUST NOT be any whitespace between the end of the tag and the start of the certificate, or between the end of the certificate and the start of the end tag. See example below. Note: The certificate should only include the public key.

5.8.9.1.3 Element: AuthenticationResp

Table 5-37: AuthenticationResp Element

Name	Data Type	Description
<i>SecureURL ?</i>	URL	URL of the port of the command recipient that will accept JMF Messages via the HTTPS protocol. This Attribute MUST be specified when the sender of the RequestForAuthentication Command has specified <i>AuthenticationCmdParams/@AuthenticationType = "AsClient"</i> .
<i>Certificate ?</i>	telem	The Command recipient's certificate. If <i>AuthenticationCmdParams/@AuthenticationType = "AsClient"</i> , this certificate MUST be the Command recipient's server certificate. If <i>AuthenticationCmdParams/@AuthenticationType = "AsServer"</i> , this certificate MUST be the Command recipient's client certificate. When responding to a RequestForAuthentication Command over a non-secure channel with <i>Reason = "InitiateConnection"</i> , this Element MUST be specified. When responding to a RequestForAuthentication Query, the Certificate Element MUST NOT be specified. See <i>AuthenticationCmdParams/Certificate</i> .

5.8.9.2 RequestForAuthentication Query

[New in JDF 1.4](#)

The RequestForAuthentication Query is used to determine the authentication status of a certificate that was provided in an earlier RequestForAuthentication Command or the Response to the Command. The sender of the Query is identified by the *SenderID* Attribute in the JMF Element that contains the RequestForAuthentication Query. The sender MAY be authenticated as both a client and as a server, and a separate certificate MUST be provided by the sender for each role that the sender wishes to use.

If a RequestForAuthentication Query is received, and no previous RequestForAuthentication Command has been received, the Response MUST specify a *ReturnCode* of 306, which is "No authentication request in process".

Table 5-38: RequestForAuthentication Query Message

Object Type	Element Name	Description
QueryTypeObj	AuthenticationQuParam	Specifies the type of authentication being queried.
ResponseTypeObj	AuthenticationResp ?	<i>ReturnCode</i> = 0 indicates "I trust you".

5.8.9.2.1 Element: AuthenticationQuParam

Table 5-39: AuthenticationCmdParams Element

Name	Data Type	Description
<i>AuthenticationType</i>	enumeration	Values are: <i>AsClient</i> – Sender of the Message wishes to check the authentication status of the client certificate associated with it. <i>AsServer</i> – Sender of Message wishes to check the authentication status of the server certificate associated with it..

Example 5-16: RequestForAuthentication Command

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="A3 Printer"
  SenderID="MIS master A" TimeStamp="2000-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Command ID="M001" Type="RequestForAuthentication"
    xsi:type="CommandRequestForAuthentication">
    <AuthenticationCmdParams AuthenticationType="AsClient"
      Reason="InitiateConnection">
      <Certificate>=====BEGIN CERTIFICATE=====
MIIC3jCCApwCBEIWy6YwCwYHKoZiZjgEAWAMFUx CzAJBgNVBAYTAkNIMQ8wDQYDVQQHEwZadXJp
Y2gxDTALBgNVBAoTBENJUDQxZDzANBgNVBAsTBkpnNRiBXRzEVMBMGAlUEAxMMd3d3LmNpcDQub3Jn
MB4XDTA1MDIxODIxNTIzOFoXDTA1MDUyOTIxNTIzOFowVTElMAkGAlUEBhMCQ0gxZDzANBgNVBAcT
Blplcm1jaDENMASGAlUEChMEQ0lQNDEPMA0GAlUECXMGSk1GIFdHMRUwEwYDVQQDEwz3d3cuY2lw
NC5vcmcwgG3MIIBLAYHKoZiZjgEATCCAR8CgYEA/X9TgR11EiLS30qcLuzk5/YRt1I870QAwx4/
gLZRJm1FXUAiUftZPY1Y+r/F9bow9subVwZxgTuAHTRv8mZgt2uZUKWkn5/oBHsQIsJPu6nX/rfG
G/g7V+fgGqKYVDwT7g/bTxr7DAjVUE1oWkTL2dfOuK2HXKu/yIgmZndFIaccCFQCXYFCPFMSLzLKS
uYKi64QL8Fgc9QKBgQD34aCF1ps93su8q1w2uFe5eZSvu/o66oL5V0wLPQeCZ1FZV4661F1P5nEH
EIGAtEkWcSPoTCgWE7fPCTKMyKbhPBZ6i1R8jSjgo64eK7OmdZFuo38L+iE1YvH7YnoBJDvMpg+
qFGQiaid3+Fa5Z8GkotmXoB7VSVkAUw7/s9JKgOBhAACgYArHi/BVNF3OG0JIIIdzWraVrx1wg9RM
do+tYRjY4bXue7LRDCvVaSX1Ddy9kTyeTTntwUrJOyx/8qEi/WmraGXhK8wGSrte/q3S/A16DwEB
CiyehMh1Crd4QiAhp5Wtr4KIMIBjq2Xn8+0Mnnt1qDnmesNaSwdZ/01E0azSPTy5XnDALBgcqhkjO
OAQDBQADLwAwLAIUFZHoJjvsO3+UYMBZk6yDzhdejzMCfHC0WbkDwfImQCa+dTebXZ1e1G1Q
=====END CERTIFICATE=====</Certificate>
    </AuthenticationCmdParams>
  </Command>
</JMF>
```

Example 5-17: RequestForAuthentication Response

The form of Response that would most likely follow the above Command appears below:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="MIS master A"
  SenderID="A3 Printer" TimeStamp="2000-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M101" Type="RequestForAuthentication" refID="M001"
    xsi:type="ResponseRequestForAuthentication" ReturnCode="304">
    <AuthenticationResp SecureURL="https://printserver.mycompany.com/A3Printer">
      <Certificate>=====BEGIN CERTIFICATE=====
uYKi64QL8Fgc9QKBgQD34aCF1ps93su8q1w2uFe5eZSvu/o66oL5V0wLPQeCZ1FZV4661F1P5nEH
EIGAtEkWcSPoTCgWE7fPCTKMyKbhPBZ6i1R8jSjgo64eK7OmdZFuo38L+iE1YvH7YnoBJDvMpg+
qFGQiaid3+Fa5Z8GkotmXoB7VSVkAUw7/s9JKgOBhAACgYArHi/BVNF3OG0JIIIdzWraVrx1wg9RM
do+tYRjY4bXue7LRDCvVaSX1Ddy9kTyeTTntwUrJOyx/8qEi/WmraGXhK8wGSrte/q3S/A16DwEB
CiyehMh1Crd4QiAhp5Wtr4KIMIBjq2Xn8+0Mnnt1qDnmesNaSwdZ/01E0azSPTy5XnDALBgcqhkjO
OAQDBQADLwAwLAIUFZHoJjvsO3+UYMBZk6yDzhdejzMCfHC0WbkDwfImQCa+dTebXZ1e1G1Q
MIIC3jCCApwCBEIWy6YwCwYHKoZiZjgEAWAMFUx CzAJBgNVBAYTAkNIMQ8wDQYDVQQHEwZadXJp
Y2gxDTALBgNVBAoTBENJUDQxZDzANBgNVBAsTBkpnNRiBXRzEVMBMGAlUEAxMMd3d3LmNpcDQub3Jn
MB4XDTA1MDIxODIxNTIzOFoXDTA1MDUyOTIxNTIzOFowVTElMAkGAlUEBhMCQ0gxZDzANBgNVBAcT
Blplcm1jaDENMASGAlUEChMEQ0lQNDEPMA0GAlUECXMGSk1GIFdHMRUwEwYDVQQDEwz3d3cuY2lw
NC5vcmcwgG3MIIBLAYHKoZiZjgEATCCAR8CgYEA/X9TgR11EiLS30qcLuzk5/YRt1I870QAwx4/
```

```

gLZRJmlFXUaiUftZPY1Y+r/F9bow9subVWzXgTuAHTRv8mZgt2uZUKWkn5/oBHsQIsJPu6nX/rfG
G/g7V+fGqKYVDwT7g/bTxR7DAjVUE1oWkTL2dfOuK2HXKu/yIgmZndFIAccCFQCXYFCPFSMLzLKS
=====END CERTIFICATE=====</Certificate>
  </AuthenticationResp>
</Response>
</JMF>

```

Example 5-18: RequestForAuthentication Query Subsequently

Next, the original command sender would send a follow up RequestForAuthentication Query:

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="A3 Printer"
  SenderID="MIS master A" TimeStamp="2000-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Query ID="M004" Type="RequestForAuthentication"
    xsi:type="QueryRequestForAuthentication">
    <AuthenticationQuParams AuthenticationType="AsClient"/>
  </Query>
</JMF>

```

Example 5-19: RequestForAuthentication Response from Subsequent Query

If authentication has been confirmed, the following Response would be sent to the RequestForAuthentication Query::

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="MIS master A"
  SenderID="A3 Printer" TimeStamp="2000-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M102" Type="RequestForAuthentication" refID="M004"
    xsi:type="ResponseRequestForAuthentication" ReturnCode="0">
  </Response>
</JMF>

```

5.8.10 StopPersistentChannel

The StopPersistentChannel Command Message 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.

Table 5-40: StopPersistentChannel Message

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

5.8.10.1 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 5-41: StopPersChParams Element (Section 1 of 2)

Name	Data Type	Description
ChannelID?	NMTOKEN	ChannelID of the persistent channel to be deleted. If the channel has been created with a Query Message, the ChannelID specifies the ID of the Query Message (identical to the refID of the Response Message).

Table 5-41: StopPersChParams Element (Section 2 of 2)

Name	Data Type	Description
<i>MessageType</i> ?	NMTOKEN	Only Messages with a matching Message type are suppressed. Default value is: all Message types Values include those from: <i>Message/@Type</i> .
<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.
<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, <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.
<i>Part</i> * New in JDF 1.2	element	Part 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 150.

5.9 Messages to Query/Command a Job, Device or Controller

JDF Messaging provides methods to trace the status of individual Devices and Resources and additional Job-dependent Job-tracking data. The status of a Job is described by the *Status* Elements of that Job.

Devices are uniquely identified by a *name* — that is, by the Attribute *DeviceID* of the **Device** Resource (see Section 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 5-42: Messages to query/affect a Job, Device or Controller (Section 1 of 2)

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).
Occupation	QRS	Queries the occupation of an employee.
Resource	CGQRS	Queries and/or modifies JDF Resources that are used by a Device, such as Device settings, or by a Job. This Message can also be used to query the level of consumables in a Device.
ResourcePull New in JDF 1.2	CGR	Creates a new <i>QueueEntry</i> from an already existing <i>QueueEntry</i> and submits it to the queue in order to be executed.

Table 5-42: Messages to query/affect a Job, Device or Controller (Section 2 of 2)

Message type	Family	Description
ShutDown New in JDF 1.2	CR	Shuts down a Device.
Status	QRS	Queries the general status of a Device, Controller or Job.
Track	QRS	Queries the location of a given Job or Job Part.
UpdateJDF New in JDF 1.3	CRS	Synchronizes and relinks modified JDF Nodes.
WakeUp New in JDF 1.2	CR	Wakes up a Device that is in standby mode.

5.9.1 FlushResources

[New in JDF 1.2](#)

The `FlushResources` Message is relates to removing temporary Resources from a Device. `FlushResourceParams` specifies the 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.

5.9.1.1 FlushResources Command

The `FlushResources` Command is used to remove temporary Resources from a Device. `FlushResourceParams` allows the specification of which Resources to remove.

Table 5-43: FlushResources Command

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 261.		

5.9.1.2 FlushResources Signal

The `FlushResources` Signal is used to signal that temporary Resources have been removed by a Device. `FlushResourceParams` allows the specification of which Resources were removed.

Table 5-44: FlushResources Command

Object Type	Element Name	Description
QueryTypeObj	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 261.		

5.9.1.2.1 Element: FlushResourceParams

Table 5-45: FlushResourceParams Element

Name	Data Type	Description
QueueFilter ?	element	Defines a <i>QueueFilter</i> that specifies the <i>QueueEntry</i> 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 <i>QueueEntry</i> Resources is requested to be flushed. Values are: <i>Complete</i> – Remove the entire temporary Resources belonging to the <i>QueueEntry</i> . <i>QueueEntry</i> – The Resources belonging to <i>QueueEntry</i> are completely re-moved and no longer available — the default. <i>Intermediate</i> – Remove any intermediate Resources that belong to the <i>QueueEntry</i> (e.g., intermediate raster files in a combined RIP and Image-Setting Process) and retain the original Input Resources. A <i>ResourcePull</i> Message is possible.

5.9.1.2.2 Element: FlushedResources

Table 5-46: FlushedResources Element

Name	Data Type	Description

5.9.2 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.

5.9.2.1 ModifyNode Command

The *ModifyNode* Command is sent by a Controller to a Device to modify the JDF Node on the Device.

Table 5-47: ModifyNode Command

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

5.9.2.2 ModifyNode Signal

The *ModifyNode* Signal is sent by a Device to a Control to Signal that the JDF Node on the Device has been modified.

Table 5-48: ModifyNode Signal

Object Type	Element Name	Description
QueryTypeObj	ModifyNodeCmdParams ?	Defines the details of the <i>ModifyNode</i> Message.
ResponseTypeObj	-	-

5.9.2.2.1 Element: ModifyNodeCmdParams

The *ModifyNodeCmdParams* specifies the details of the JDF Node to be modified.

Table 5-49: ModifyNodeCmdParams Element

Name	Data Type	Description
<i>Activation?</i>	enumeration	The new value for <i>Activation</i> . Values are from: JDF/@ <i>Activation</i> (Table 3-5, “JDF Node” .)
<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 40 and that the semantics are overridden by the definition in this table.
NewComment *	element	Details of modifications of Comment Elements.

5.9.2.2.2 Element: NewComment**Table 5-50: NewComment Element**

Name	Data Type	Description
<i>Action</i>	enumeration	Values are: <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</i> = <i>refID</i> . <i>Concat</i> – Comment is concatenated to the Comment with Comment/@ <i>ID</i> = <i>CommentID</i> . <i>Replace</i> – Comment replaces the Comment with Comment/@ <i>ID</i> = <i>CommentID</i> . <i>Remove</i> – The Comment with Comment/@ <i>ID</i> = <i>CommentID</i> is removed.
<i>CommentID?</i>	NMTOKEN	<i>ID</i> of the existing Comment. MUST be specified if <i>Action</i> = <i>Concat</i> , <i>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</i> = “ <i>Add</i> ”.
<i>Comment?</i>	element	The Comment to <i>Add</i> , <i>Concat</i> or <i>Replace</i> . Comment MUST NOT be specified if <i>Action</i> = “ <i>Remove</i> ”. Note that Comment * is specified in Table 3-1, “Any Element (generic content),” on page 40 and that the cardinality and semantics are overridden by the definition in this table.
<i>Part?</i> New in JDF 1.4	element	Partition of the Resource where the Comment MUST be added. Part MUST NOT be specified unless <i>refID</i> references a Resource and @ <i>Action</i> = “ <i>Add</i> ”

5.9.3 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 Message or Signal Message informs a Controller or MIS system that a JDF Node has been created. A command initiates a modification.

5.9.3.1 NewJDF Query

The NewJDF Query Message is sent to a Device or Controller in order to extract information about previously unknown JDF Nodes. For instance, an MIS that has received a JMF with an unknown *JobPartID* MAY query the JMF sender about details of the JDF with that *JobPartID*. When used as a Signal, the Signaling Device specifies that it has created a new JDF with the properties defined by IDInfo, for instance when a Workflow Controller has instantiated an abstract Process Group Node with new Subnodes. NewJDF is made selective by specifying a NewJDFQuParams Element.

The query's Response Message returns a list of IDInfo Elements that contains the queried information concerning the newly created Nodes.

Table 5-51: NewJDF Query Message

Object Type	Element Name	Description
QueryTypeObj	NewJDFQuParams	Specifies the details of the Nodes that information is requested about.
ResponseTypeObj	IDInfo *	Contains the information about the newly created Nodes.

5.9.3.1.1 Element: NewJDFQuParams

The NewJDF Command Message is sent to an MIS, Device or Controller to initiate creation of new JDF Nodes by that Device or Controller. For instance, a Workflow Controller might have received content data and now requires a JDF Job from an MIS to which work on the content can be booked. The NewJDF Command Message does not imply any Job submission or request for Job submission. Job queue submission MUST still be requested with a RequestQueueEntry Message, and the MIS MUST still subsequently submit the Job to the requesting Controller or Device.

Table 5-52: 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.

5.9.3.2 NewJDF Command

Table 5-53: NewJDF Command Message

Object Type	Element Name	Description
CommandTypeObj	NewJDFCmdParams	Specifies the details of the Nodes that are to be created
ResponseTypeObj	IDInfo ?	Contains the information about the newly created Node.

5.9.3.2.1 Element: NewJDFCmdParams

Table 5-54: NewJDFCmdParams Element

Name	Data Type	Description
<i>JDFDetails</i> ="Brief"	string	Level of detail requested for the returned IDInfo Elements. Values include: <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.

5.9.3.2.2 Element: IDInfo

Table 5-55: IDInfo Element

Name	Data Type	Description
<i>Category?</i>	NMTOKEN	JDF/@ <i>Category</i> of the JDF Node. Values include those from: JDF/@ <i>Category</i> .
<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/@ <i>Type</i> of the JDF Node. Values include those from: JDF/@ <i>Type</i> .
<i>Types?</i>	NMTOKENS	JDF/@ <i>Types</i> of the JDF Node. Values include those from: JDF/@ <i>Types</i> .
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 87.
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.

5.9.4 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 1007.

5.9.5 Occupation

Occupation queries the occupation status of an employee. No Job context is needed to issue an Occupation Message.

Table 5-56: Occupation Message

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

5.9.5.1 Element: EmployeeDef

The Occupation Query Message might be focused to certain employees specifying a EmployeeDef Element. If no EmployeeDef Element is specified, a list of all known employees is returned.

Table 5-57: EmployeeDef Element

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

5.9.5.2 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 5-58: 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 <i>Devices</i> 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.
<i>Device</i> *	element	Devices that the employee is currently assigned to. The data type of <i>Device</i> is <i>ResourceElement</i> . See "ResourceElement – Subelement of a Resource" on page 87.
<i>Employee</i>	element	Description of the employee being tracked. The data type of <i>Employee</i> is <i>ResourceElement</i> . See "ResourceElement – Subelement of a Resource" on page 87.
<i>Part</i> * 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 150

Example 5-20: Occupation Response

The following is an example of Response Message to an *Occupation* Query Message:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M1" Type="Occupation" xsi:type="ResponseOccupation" refID="Q1">
    <!--Two Jobs on one Device with one operator-->
    <Occupation Busy="30" JobID="J1">
      <Employee PersonalID="P1234"/>
      <Device DeviceID="Press1"/>
    </Occupation>
    <Occupation Busy="70" JobID="J2">
      <Employee PersonalID="P1234"/>
      <Device DeviceID="Press1"/>
    </Occupation>
    <!--Another operator on Job j2 -->
    <Occupation Busy="50" JobID="J2">
      <Employee PersonalID="P4321"/>
      <Device DeviceID="Press1"/>
    </Occupation>
    <!--No Job context -->
    <Occupation Busy="0">
      <Device DeviceID="Press2"/>
      <Employee PersonalID="P5678"/>
    </Occupation>
  </Response>
</JMF>
```

5.9.6 Resource

The **Resource** Message can be a **Command** Message or a **Query** Message to modify or to query JDF Resources. In both cases (query and command), it is possible to address either global Device Resources, such as Device settings or Job-specific Resources. The **Query** Message retrieves information about the Resources without modifying them, while the **Command** Message modifies those settings within the Resource that are specified. Settings that are not specified remain unchanged.

5.9.6.1 Resource Query

The **Resource Query** Message can be made selective by specifying a **ResourceQuParams** Element. The presence of the **JobID** Attribute determines whether global Device Resources or Job-related Resources are returned. If no **ResourceQuParams** Element is specified, only the global Device Resources are returned.

The query's Response Message returns a list of **ResourceInfo** Elements that contains the queried information concerning the Resources. If the list is empty because the selective query parameters of the **ResourceQuParams** lead to a null selection of the known Device/Job Resources, then the **ReturnCode** is 103 (*JobID* unknown), 104 (*JobPartID* unknown) or 108 (empty list) and SHOULD be flagged as a warning with **Notification**[@*Class* = "Warning" and @*Type* = "Error"].

Table 5-59: Resource Query Message

Object Type	Element Name	Description
QueryTypeObj	ResourceQuParams ?	Specifies the Resources queried.
ResponseTypeObj	ResourceInfo *	Contains the amount data of Resources and if requested, the Resources itself.

5.9.6.1.1 Element: ResourceQuParams

Table 5-60: ResourceQuParams Element (Section 1 of 3)

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 <i>JobID</i> Attribute. Default value is: all Classes (if <i>Classes</i> is empty or not specified). Values are from: Resource/@ <i>Classes</i> (Table 3-10, "Abstract Resource Element," on page 63.).
<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> ? Modified in JDF 1.4	string	<i>JobID</i> of the JDF Node that is being queried. If no <i>JobID</i> is specified, global resources are queried.
<i>JobPartID</i> ?	string	<i>JobPartID</i> of the JDF Node that is being queried.. If no <i>JobPartID</i> is specified, all resources related to <i>JobID</i> are queried.

Table 5-60: ResourceQuParams Element (Section 2 of 3)

Name	Data Type	Description
<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 3.10.6.4, Locations of Physical Resources). Default value is: the location will be selected by the Device Values include those from: Table C-21, "Input Tray and Output Bin Names," on page 889. Note: the specified values are for printer locations.
<i>LotDetails = "Brief"</i> New in JDF 1.4	enumeration	Refines the level of information provided about individual lots of the Resources. This attribute is most useful when querying an MIS, and SHOULD NOT be specified when querying a device. Values are: <i>Brief</i> – Provides only the <i>LotControlled</i> Attribute in the Response indicating whether or not the Resources are lot controlled. <i>Full</i> – Provides Lot Elements related to the Resources. <i>Amount</i> – Same as "Full", but with the addition of the <i>Amount</i> Attribute so the MIS can indicate what the current "on hand" balance for the Lot is in the MIS.
<i>LotID?</i> New in JDF 1.4	string	<i>LotID</i> of the individual lot of the Resource that is queried.
<i>ProcessUsage?</i>	string	Selects a Resource in which <i>ResourceLink/@ProcessUsage</i> matches the token specified. Only necessary if a Resource name is used more than once by one Node. For example, the Component input ExposedMedia of a <i>ConventionalPrinting</i> Process MUST be distinguished by specifying <i>ProcessUsage = "Plate"</i> and <i>ProcessUsage = "Proof"</i> , 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. Values include those from: <i>ResourceLink/@ProcessUsage</i> (Table 3-16, "Abstract ResourceLink Element," on page 77)
<i>ProductID?</i> New in JDF 1.2	string	<i>ProductID</i> of the Resource that is queried.
<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> MUST be 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.
<i>ResourceDetails = "Full"</i> New in JDF 1.4	enumeration	Refines the level of information provided about the Resources. Values are: <i>Brief</i> – Provides appropriate ID information specific to the type of Resource and <i>DescriptiveName</i> Attributes only. For example, <i>ProductID</i> would be included for Consumable Resources, <i>PersonalID</i> for Employee Resources. <i>Full</i> – Provides all of the attributes of the resources.

Table 5-60: ResourceQuParams Element (Section 3 of 3)

Name	Data Type	Description
ResourceID? New in JDF 1.3	NMTOKEN	Resource/@ID of the Resource that is queried. Note: The data type is NMTOKEN and not IDREF because the referenced ID need not be present in the JMF.
ResourceName? Modified in JDF 1.4	NMTOKENS	Name of the Resource(s) being queried. Values include those from: Section 7, Resources Modification note: starting with JDF 1.4, the data type was expanded from NMTOKEN to NMTOKENS.
Scope? New in JDF 1.4	enumeration	Specifies whether the query refers to a complete list of all potential Resources or to the currently loaded Resources. Values are: <i>Allowed</i> – All known Resources MUST be returned. <i>Present</i> – Currently available Resources MUST be returned.
Usage?	enumeration	Selects a Resource in which the value of the ResourceLink/@Usage Attribute matches the token specified here in this Attribute. Only necessary if a Resource name is used both as input and output by one Node. Values are from: ResourceLink/@Usage (Table 3-16, “Abstract ResourceLink Element,” on page 77).
Part* New in JDF 1.2	element	Part Elements that describe the Resource whose Messages are queried.

Example 5-21: Resource Query about Paper

The following is an example of a press system sending a Resource Query to an MIS to get information on all paper known by the MIS:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="PressSystem"
  DeviceID="MIS" TimeStamp="2000-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Query ID="M170" Type="Resource" xsi:type="QueryResource" >
    <ResourceQuParams ResourceDetails="Brief" LotDetails="Full" Scope="Allowed" />
  </Query>
</JMF>
```

Example 5-22: Resource Response about Paper

The following is an example of a Resource Response to the previous Resource Query

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="MIS"
  DeviceID="PressSystem" TimeStamp="2000-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M1001" Type="Resource" xsi:type="ResponseResource" refID="M170">
    <ResourceInfo LotControlled="false">
      <Media ID="R01" Class="Consumable" Status="Available"
        ProductID="9902-1" DescriptiveName="60 lb #3 Gloss Book"/>
    </ResourceInfo>
    <ResourceInfo LotControlled="true">
      <Media ID="R01" Class="Consumable" Status="Available"
        ProductID="9903-1" DescriptiveName="80 lb #3 Cls Cover"/>
      <Lot LotID="LN8845739CN7787399-03" />
      <Lot LotID="LN8845739CN7787399-04" />
      <Lot LotID="LN8845739CN7787399-06" />
      <Lot LotID="LN8845739CN7787399-10" />
    </ResourceInfo>
  </Response>
</JMF>
```

```

<!-- ... -->
<ResourceInfo LotControlled="false">
  <Media ID="R01" Class="Consumable" Status="Available"
    ProductID="9989-5" DescriptiveName="110 lb #1 Coated Cover"/>
</ResourceInfo>
</Response>
</JMF>

```

Example 5-23: Resource Query about Employees

The following is an example of a press system sending a Resource Query to an MIS to get a list of all known employees in the MIS:

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="PressSystem"
  DeviceID="MIS" TimeStamp="2000-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Query ID="M170" Type="Resource" xsi:type="QueryResource">
    <ResourceQuParams ResourceName="Employee" ResourceDetails="Brief"/>
  </Query>
</JMF>

```

Example 5-24: Resource Response about Employees

The following is an example of a Resource Response to the previous Resource Query

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="MIS"
  DeviceID="PressSystem" TimeStamp="2000-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M1001" Type="Resource" xsi:type="ResponseResource"
    refID="M170">
    <ResourceInfo>
      <Employee ID="E01" Class="Implementation" Status="Available"
        PersonalID="1034" DescriptiveName="John Allen"/>
    </ResourceInfo>
    <ResourceInfo>
      <Employee ID="E02" Class="Implementation" Status="Available"
        PersonalID="1057" DescriptiveName="Sally Brown"/>
    </ResourceInfo>
    <ResourceInfo>
      <Employee ID="E03" Class="Implementation" Status="Available"
        PersonalID="2105" DescriptiveName="Mike Davison"/>
    </ResourceInfo>
    <!-- ... -->
    <ResourceInfo>
      <Employee ID="E04" Class="Implementation" Status="Available"
        PersonalID="6410" DescriptiveName="Will Smith"/>
    </ResourceInfo>
  </Response>
</JMF>

```

Example 5-25: Resource Signal about Consumed Resources

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

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="ProductionSystem"
  TimeStamp="2005-09-23T17:47:18-05:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Signal ID="P172" Type="Resource" xsi:type="SignalResource">
    <ResourceQuParams JobID="34028" JobPartID="_F05A84BD"/>
    <ResourceInfo>

```

```

<Media PartIDKeys="SheetName" ID="RI007" Class="Consumable"
  ProductID="3002" Brand="Roll Stock"
  Dimension="2520 8640000" MediaType="Paper">
  <Media SheetName="1"/>
  <Media SheetName="2"/>
</Media>
<AmountPool>
  <PartAmount ActualAmount="9700">
    <Part SheetName="1"/>
    <Lot ActualAmount="4850" Consumption="Full"
      LotID="LN1040788312RN20050917-04"/>
    <Lot ActualAmount="4850" Consumption="Partial"
      LotID="LN1040788339RN20050919-01"/>
  </PartAmount>
  <PartAmount ActualAmount="5027">
    <Part SheetName="2"/>
    <Lot ActualAmount="5027" Consumption="Partial"
      LotID="LN1040788319RN20050917-04"/>
  </PartAmount>
</AmountPool>
</ResourceInfo>
</Signal>
</JMF>

```

5.9.6.2 Resource Command

The Resource Command Message 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 Resource Message contains a list of ResourceInfo Elements with all Resources and private extensions of the Device after the changes have been applied. The type of the Resource that is given as a response depends on the type of the Resource given in the command.

If the Resource Command Message was successful, the value of 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.

Table 5-61: Resource Command Message

Object Type	Element Name	Description
CommandTypeObj	ResourceCmdParams	Specifies the Resources to be modified.
ResponseTypeObj	ResourceInfo *	Contains information about the Resources after modification.

5.9.6.2.1 Element: ResourceCmdParams

Table 5-62: 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. Values are in order of involvement from least to most active. Values 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 <i>JDF/@Activation</i> , but that it does not explicitly set <i>JDF/@Activation</i> in the JDF on the Device. The <i>Inactive</i> value defined in <i>JDF/@Activation</i> 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 Message 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.
<i>ProcessUsage</i> ?	NMTOKEN	Selects a Resource in which the value of the <i>ResourceLink/@ProcessUsage</i> Attribute matches the token specified here in this Attribute. Only necessary if a Resource name is used more than once by one Node. For example, the ExposedMedia Input Resources of a ConventionalPrinting Process can be distinguished by specifying <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 modified. Values include those from: <i>ResourceLink/@ProcessUsage</i> (Table 3-16, "Abstract ResourceLink Element," on page 77).
<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 <i>ResourceLink/@Amount</i> of the selected Resource.
<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>ResourceCmdParams</i> applies to all Jobs.

Table 5-62: ResourceCmdParams Element (Section 2 of 3)

Name	Data Type	Description
<i>ResourceID</i> ? New in JDF 1.3	NMTOKEN	Resource/@ID of the Resource that is modified. If both <i>ResourceID</i> and <i>Resource</i> specified, Resources with a non-matching Resource/@ID MUST NOT be updated. Note: The data type is NMTOKEN and not IDREF because the referenced ID NEED NOT be present in the JMF.
<i>ResourceName</i> ?	NMTOKEN	Name of the Resource whose production amount will be modified. Values include those from: Section 7, Resources.
<i>Status</i> ? New in JDF 1.2	enumeration	Updated <i>Status</i> of the selected Resource. Values are from: Resource/@ <i>Status</i> (Table 3-10, “Abstract Resource Element,” on page 63).
<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 Modified in JDF 1.4	enumeration	Method how the Resource is updated. Attributes that are required to correctly identify the Resource MUST be specified, even if <i>UpdateMethod</i> = "Remove" or <i>UpdateMethod</i> = "Incremental". These Attributes include <i>ID</i> , <i>Class</i> , <i>PartIDKeys</i> , and any Partition Keys. 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. <i>Remove</i> – The Resources or Resource Partitions are removed. New in JDF 1.4
<i>Usage</i> ? New in JDF 1.4	enumeration	Selects a Resource in which the value of the ResourceLink/@ <i>Usage</i> Attribute matches the token specified here in this Attribute. Only neces-sary if a Resource name is used both as input and output by one Node. Values are from: ResourceLink/@ <i>Usage</i> (Table 3-16, “Abstract ResourceLink Element,” on page 77).
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 proper-ties of the Partition that represents the set are modified in addition to the properties of the instance, then the Part that represents the set SHOULD also be specified explicitly. For example, if the fourth plate of a four color process set is now avail-able, and thus the entire surface is now available, Part Elements for both the fourth plate and for the entire surface SHOULD be specified. If the other surface is also available, then a Part Element for the sheet SHOULD be specified as well.

Table 5-62: ResourceCmdParams Element (Section 3 of 3)

Name	Data Type	Description
Resource *	element	Resources to be uploaded to the Device. They replace the original Resources with the same Resource/@ID according to the policy specified in <i>UpdateMethod</i> . 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 Message. The data type and <i>Class</i> of Resource is derived from the Abstract Resource. See “Abstract Resource” on page 63.

Example 5-26: Resource Command: Single Resource is Available

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

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Command ID="C1" Type="Resource" xsi:type="CommandResource">
    <ResourceCmdParams JobID="MakeBrochure 012" ResourceID="ExposedMediaID">
      <Part SignatureName="Sig1" SheetName="Sheet2" Side="Front"
        Separation="Cyan"/>
    </ResourceCmdParams>
  </Command>
</JMF>
```

Example 5-27: Resource Command: Multiple Resources are Available

The following is an example for specifying that the Black, Front plate of *Sheet2*, Signature 1 has become available and is also the last plate of Sheet 2.

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Command ID="C2" Type="Resource" xsi:type="CommandResource">
    <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>
</JMF>
```

5.9.6.2.2 Element: ResourceInfo

Table 5-63: ResourceInfo Element (Section 1 of 4)

Name	Data Type	Description
<i>ActualAmount</i> ? New in JDF 1.2	double	When querying a Device, this Attribute reflects the current accumulated amount of the Resource that has been consumed (input) or produced (output) by the Process. This corresponds to the current value of ResourceLink/@ <i>ActualAmount</i> if it would be written now. When querying an MIS, this Attribute SHOULD NOT be specified.

Table 5-63: ResourceInfo Element (Section 2 of 4)

Name	Data Type	Description
<i>Amount?</i>	double	When querying a Device, this Attribute reflects the intended accumulated amount of the Resource that will be consumed (input) or produced (output) by the Process. This corresponds to the current value of ResourceLink/@Amount if it would be written now. When querying an MIS, this Attribute specifies the amount of the Consumable Resource that is available in inventory.
<i>AvailableAmount?</i>	double	When querying a Device, this Attribute specifies the Device-specific amount of the Consumable Resource that is available in the Device. When querying an MIS, this Attribute specifies the amount of the Consumable Resource that is available in inventory
<i>CommandResult?</i> New in JDF 1.4	enumeration	Result of a Resource Command. Values are: <i>Rejected</i> – the Resource Command was not applied to this Resource. <i>Removed</i> – An existing Resource was removed completely by a Resource specified in ResourceCmdParams. <i>New</i> – A new Resource with the values specified in ResourceCmdParams was created. <i>Merged</i> – Values from the Resource in ResourceCmdParams were merged into an existing Resource. See the ResourceInfo/Resource for the merged result. <i>Replaced</i> – An existing Resource was replaced completely by a Resource specified in ResourceCmdParams.
<i>Level?</i> Modified in JDF 1.4	enumeration	This Attribute is Device dependent. A Device MAY specify a level status that describes a low or empty consumable level. 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. Modification note: starting with JDF 1.4, the default of "OK" is removed to allow job independent Resource information.
<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 3.10.6.4, Locations of Physical Resources). Default value is: the location will be selected by the Device Values include those from: Table C-21, "Input Tray and Output Bin Names," on page 889. Note: the specified values are for printer locations.
<i>LotControlled?</i> New in JDF 1.4	boolean	Indicates that the Resource is lot controlled.

Table 5-63: ResourceInfo Element (Section 3 of 4)

Name	Data Type	Description
<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>ProcessUsage</i> ?	NMTOKEN	Selects a Resource in which the value of the ResourceLink/@ <i>ProcessUsage</i> Attribute matches the token specified here in this Attribute. Only necessary if a Resource name is used more than once by one Node. For example, the ExposedMedia Input Resources of a <i>ConventionalPrinting</i> 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. Values include those from: ResourceLink/@ <i>ProcessUsage</i> (Table 3-16, “Abstract ResourceLink Element,” on page 77).
<i>ProductID</i> ? New in JDF 1.2	string	<i>ProductID</i> of the Resource.
<i>ResourceID</i> ? New in JDF 1.3	NMTOKEN	Resource/@ <i>ID</i> of the Resource. 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 if <i>Exact</i> = <i>false</i> in the query. <i>ResourceName</i> specifies the primary Resource that this ResourceInfo applies to. Additional Resources MAY be specified to ensure complete references from the primary Resource. Values include those from: Section 7, Resources.
<i>Status</i> ? New in JDF 1.2	enumeration	Updated <i>Status</i> of the selected Resource. Values are from: Resource/@ <i>Status</i> (Table 3-10, “Abstract Resource Element,” on page 63).
<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). Values include those from: Table 1-7, “Units used in JDF” .
<i>Usage</i> ? New in JDF 1.3	enumeration	Specifies a Resource in which the value of the ResourceLink/@ <i>Usage</i> Attribute matches the value of this Attribute. Only required if a Resource name is used both as input and output by one Node. Values are from: ResourceLink/@ <i>Usage</i> (Table 3-16, “Abstract ResourceLink Element,” on page 77).

Table 5-63: ResourceInfo Element (Section 4 of 4)

Name	Data Type	Description
AmountPool ? New in JDF 1.3	element	Definition of partial amounts and pipe parameters for this Resource. The contents of the AmountPool are described for the various types of ResourceLink Elements in Table 3-18, "AmountPool Element," on page 81. If AmountPool is specified, the ResourceInfo MUST NOT contain any of the amount related Attributes defined in AmountPool/PartAmount.
CostCenter ?	element	Cost center to which the Resource consumption is allocated.
Lot * New in JDF 1.4	element	Used when a Device is querying a Controller to determine what lots exist for the Resource being queried. When a Device is the sender of this Message, lot information is specified in the AmountPool, and MUST NOT be specified here.
MISDetails ? New in JDF 1.2	element	Definition how the costs for the production of the Resource are to be charged.
Part * New in JDF 1.4	element	Part Elements that describe the Resource. Creation note: starting with JDF 1.4, Part is back after being deprecated in JDF 1.3.
Resource * Modified in JDF 1.4	element	JDF description of the Resource. If the query or command leading to this Response Message Element contains Part Elements, the Resource 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 63. Modification note: starting with JDF 1.4, there can be multiple occurrences of Resource Elements. See <i>ResourceName</i> for the reason.

Example 5-28: Resource Query for Consumables

The following is an example for retrieving settings:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Query ID="Q1" Type="Resource" xsi:type="QueryResource">
    <ResourceQuParams Classes="Consumable" Exact="true"/>
  </Query>
</JMF>
```

Example 5-29: Resource Response about Consumables

The following is a possible Response Message to the Query Message above:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="A3 Printer"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M1" Type="Resource" xsi:type="ResponseResource" refID="Q1">
    <ResourceInfo AvailableAmount="2120" Location="Paper Tray 1">
      <Media ID="ID123" Class="Consumable" Status="Available">
        <!-- Media resource defined in JDF -->
      </Media>
    </ResourceInfo>
    <ResourceInfo AvailableAmount="0" Level="Empty" Location="Ink1" Unit="l">
      <Ink ID="ID124" Class="Consumable" Status="Available">>
        <!-- Ink description resource defined in JDF -->
      </Ink>
    </ResourceInfo>
  </Response>
</JMF>
```

```

    </ResourceInfo>
  </Response>
</JMF>

```

Example 5-30: Resource Command for Changing Amount

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

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Command ID="C1" Type="Resource" xsi:type="CommandResource">
    <ResourceCmdParams JobID="MakeBrochure 012" ProductionAmount="7500"
      ResourceName="Component" />
  </Command>
</JMF>

```

Example 5-31: Resource Response for Changing Amount

The following is a possible response to the Resource Command Message above:

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M2" Type="Resource" xsi:type="ResponseResource" refID="C1">
    <ResourceInfo Amount="7500" ResourceName="Component" />
  </Response>
</JMF>

```

5.9.7 ResourcePull

[New in JDF 1.2](#)

The ResourcePull Message requests a Resource from a Controller or Device. The Resource is specified as the Output Resource of a JDF Node. The requested Resource MAY be a subset of the Resource specified in the original JDF. The ResourcePullParams Element provides the parameters. The command can be used to regenerate the output of a QueueEntry or JDF Node with any *Status*.

If the ResourcePull is accepted, the respective QueueEntry is requeued with QueueEntry/@Status = "Waiting". After processing, the processing result MUST be sent to the original submitter of the QueueEntry that is being repeated using a ReturnQueueEntry Message. The sender of the ResourcePull Message SHOULD be informed of the completion of the ResourcePull Message with a Resource Command.

Workflow Integration with ResourcePull

When ResourcePull is submitted directly to a Device in a workflow that is monitored by an MIS system, the MIS system 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.

Table 5-64: ResourcePull Message (Section 1 of 2)

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.

Table 5-64: ResourcePull Message (Section 2 of 2)

Object Type	Element	Description
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 261.		

5.9.7.1 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.

Table 5-65: 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 QueueEntry. This <i>Amount</i> is the amount to be produced by the Process that is executed due to the ResourcePull. Thus if 200 copies had been created previously and 100 copies are requested by the ResourcePull, <i>Amount</i> = "100" and not "300".
<i>Hold = "false"</i>	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 QueueEntryID is specified, JobID is ignored. Exactly one of JobID or QueueEntryID MUST be specified.
<i>Priority = "1"</i>	integer	Number from 0 to 100, where 0 is the lowest priority and 100 is the maximum priority.
<i>QueueEntryID?</i>	string	QueueEntryID of the JDF Node that creates the requested Resource. If QueueEntryID is specified, JobID is ignored. Exactly one of JobID or QueueEntryID MUST be specified.
<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). Values are: <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.

Table 5-65: ResourcePullParams Element (Section 2 of 2)

Name	Data Type	Description
ReturnURL ? Deprecated in JDF 1.4	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. If <i>ReturnURL</i> is specified with the “file” scheme, <i>ReturnURL</i> MUST specify an individual file. <i>ReturnURL</i> takes precedence when <i>NodeInfo/@TargetRoute</i> is specified in the previously submitted JDF.
WatchURL ? Deprecated in JDF 1.4	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 <i>Subscription</i> for an <i>Events Message</i> with <i>SignalTypes</i> = “All”.
Part *	element	The Part Elements identify the parts of a Partitioned Output Resource to be created by the JDF Node. The structure of the Part Element is defined in Table 3-28, “Part Element,” on page 104. For details on Partitioned Resources, see Section 3.10.5, Description of Partitioned Resources. For details on Node Partitions, see “Partial Processing of Nodes with Partitioned Resources” on page 150.
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 Resource are to be charged.

Example 5-32: ResourcePull Command

For example, if an *ImageSetting* Process produces a Partitioned set of plates, the following example Message would request only the yellow plate of the *Front Surface* of *Sheet1*.

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Command ID="C3" Type="ResourcePull" xsi:type="CommandResourcePull">
    <ResourcePullParams QueueEntryID="AllPlates" Priority="100" ResourceID="R42">
      <Part SheetName="Sheet1" Side="Front" Separation="Yellow"/>
    </ResourcePullParams>
  </Command>
</JMF>
```

5.9.8 ShutDown[New in JDF 1.2](#)

The ShutDown Command Message shuts down a Controller or Device. A Device MUST use the Status Message if it signals its own shutdown.

Table 5-66: ShutDown Message (Section 1 of 2)

Object Type	Element	Description
CommandTypeObj	ShutDownCmdParams	Defines the details of a shutdown.
	QueueFilter	Defines a filter for the returned Queue Element in the ShutDown Message.

Table 5-66: 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 261s.

5.9.8.1 Element: ShutDownCmdParams

Table 5-67: ShutDownCmdParams Element

Name	Data Type	Description
<i>ShutDownType</i> = "StandBy"	enumeration	Defines the Device shutdown method. 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.

5.9.9 Status

The Status Message queries the general status of a Device or a Controller and the status of Jobs associated with this Device or Controller. No Job context is needed to issue a Status Message. The response contains one or more DeviceInfo Elements, which contain the Device specific information and which MAY contain other JobPhase Elements that in turn contain the Job specific information. The response MAY also provide a Queue Element.

Table 5-68: Status Message

Object Type	Element Name	Description
QueryTypeObj	StatusQuParams	Refines the query to include various aspects of the Device and Job states.
ResponseTypeObj	DeviceInfo +	Describes the actual device Status. If multiple DeviceInfo Elements are specified, these describe multiple Devices. A sequential state change of an individual Device MUST be encoded as 2 separate Signals.
	Queue ?	Provides information about the queue and all its entries. This Element will only be provided if the Device has queue capabilities. The Queue Element is described in Section 5.14, Elements for Queues.

Example 5-33: Status Signal

[New in JDF 1.4](#)

Example of a Status Signal for a phase switch from setup to running

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  SenderID="MIS master A" TimeStamp="2007-08-09T11:35:41+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Signal ID="m18" Type="Status" xsi:type="SignalStatus">
    <DeviceInfo DeviceStatus="Running">
      <JobPhase JobID="jID" JobPartID="jpID">
```

```

        PhaseStartTime="2007-08-09T11:35:40+02:00" Status="Setup" />
    </DeviceInfo>
</Signal>
<Signal ID="m19" Type="Status" xsi:type="SignalStatus">
    <DeviceInfo DeviceStatus="Running">
        <JobPhase JobID="jID" JobPartID="jpID"
            PhaseStartTime="2007-08-09T11:35:41+02:00" Status="InProgress" />
    </DeviceInfo>
</Signal>
</JMF>

```

5.9.9.1 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 891).

Table 5-69: StatusQuParams Element (Section 1 of 2)

Name	Data Type	Description
<i>DeviceDetails</i> = " <i>None</i> "	enumeration	Refines the provided status information about the Device. Values are: <i>None</i> – Provide only <i>DeviceInfo/@DeviceID</i> and <i>DeviceInfo/@DeviceStatus</i> . <i>Brief</i> – Provide all available Device information except for Device Elements. <i>Modules</i> – Provide <i>ModuleStatus</i> Elements with module specific status details. <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 <i>DeviceCap</i> 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> = " <i>false</i> "	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.

Table 5-69: StatusQuParams Element (Section 2 of 2)

Name	Data Type	Description
<i>JobDetails</i> = "None"	enumeration	Refines the provided status information about the Jobs associated with the Device. Each higher entry includes the values specified in the lower entries. Values are: <i>None</i> – Specify only <i>JobID</i> , <i>JobPartID</i> and <i>Amount</i> and/or <i>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 a URL reference to 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, <i>StatusQuParams</i> applies to all Jobs.
<i>QueueInfo</i> = "false"	boolean	If <i>true</i> , a <i>Queue</i> Element is requested to be provided. This is analogous to a <i>QueueStatus</i> Query Message (see Section 5.13.6, <i>QueueStatus</i>).
<i>Part</i> * New in JDF 1.2	element	<i>Part</i> 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 150.

5.9.9.2 Element: DeviceInfo

The Response Message returns a *DeviceInfo* Element for the queried Device.

Table 5-70: DeviceInfo Element (Section 1 of 3)

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-Fed printer, it is the number of Sheets; in case of a Web Printer, it is the length of printed Web in meters. Values include those from: Table 1-7, "Units used in JDF" .
<i>DeviceCondition</i> ? New in JDF 1.2	enumeration	The general condition of a Device. Values are: <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.

Table 5-70: DeviceInfo Element (Section 2 of 3)

Name	Data Type	Description
<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. Values are: <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. 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>IdleStartTime</i> ? New in JDF 1.4	dateTime	Specifies the beginning of the last phase with no JobPhase entries. A Device is idle when no active Jobs are being processed. Multiple phases with different stati and no active Job phases MAY be specified, for instance a maintenance phase followed by an idle phase. <i>IdleStartTime</i> MUST NOT be specified if JobPhase Elements are present in the DeviceInfo or <i>DeviceStatus</i> != " <i>Idle</i> ", " <i>Down</i> " or " <i>Stopped</i> ".
<i>PowerOnTime</i> ?	dateTime	Date and time when the Device was switched on.
<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 883).

Table 5-70: DeviceInfo Element (Section 3 of 3)

Name	Data Type	Description
<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. Values include those from: "StatusDetails Supported Strings" on page 875
<i>TotalProductionCounter?</i>	double	The current total Machine production counter since the Machine was produced.
<i>Device?</i>	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 87.
<i>Employee*</i>	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 87.
<i>JobPhase*</i>	element	Describes the actual status of Jobs in the Device. All Jobs that are active on the Device MUST be specified. Supplying no <i>JobPhase</i> specifies that no Job is currently active on the Device. Active Jobs have JDF/@ <i>Activation</i> = "Active", "TestRun" or "TestRunAndGo" and JDF/@ <i>Status</i> or JDF/StatusPool/PartStatus/@ <i>Status</i> = "TestRunInProgress", "Setup", "InProgress", "Cleanup" or "Stopped". Multiple <i>JobPhase</i> Elements specify that multiple Job phases are active simultaneously on the Device. For details on using <i>JobPhase</i> Elements, see Table 5-71 on page 232.
<i>ModuleStatus*</i>	element	Status of individual modules. <i>ModuleStatus</i> MUST NOT be specified for modules that are specified in <i>JobPhase/ModuleStatus</i> . For details on using <i>ModuleStatus</i> Elements, see Table 5-72 on page 234.

5.9.9.3 Element: JobPhase

A Status Response Message **MAY** provide *JobPhase* Elements. The *JobPhase* Element represents the actual state of a Job. The *JobPhase* Element is an analogue to the *PhaseTime* Audit Element described in Section 3.11.4.6, *PhaseTime*. The main difference between a *JobPhase* Element and a *PhaseTime* Audit Element is that a *JobPhase* Message Element reflects a snapshot of the current Job status whereas the *PhaseTime* Audit Element reflects a time span bordered by two (sub-) status transitions.

For exact information about the Job phase, a *JobPhase* Element **MAY** include a URL reference to a copy of the current state of the Job described as JDF. If *Part* Elements are specified, all Attributes in *JobPhase* apply only to the specified parts. If an actual JDF is not supported by the Controller, the same rules apply for the Status Response Message as those which apply for the Resource Response Message.

Table 5-71: 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. Values are from: JDF/@ <i>Activation</i> (Table 3-5, "JDF Node").
<i>Amount?</i>	double	Sum of actual <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> .

Table 5-71: JobPhase Element (Section 2 of 3)

Name	Data Type	Description
<i>Deadline</i> ?	enumeration	Scheduling state of the Job. 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. Note: 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	Actual 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	Actual 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 <i>JDF/@Status</i> from <i>Waiting</i> or <i>Ready</i> to any active value.
<i>Status</i>	enumeration	The status of the JDF Node. Values are from: <i>JDF/@Status</i> (Table 3-5, "JDF Node," on page 47).
<i>StatusDetails</i> ?	string	String that defines the Job state more specifically. . Values include those from: "StatusDetails Supported Strings" on page 875.
<i>TotalAmount</i> ? New in JDF 1.1	double	Planned amount that will be produced when this Job phase is 100% completed. The unit is specified in the <i>CounterUnit</i> Attribute of the parent Element <i>DeviceInfo</i> .
<i>URL</i> ? New in JDF 1.4	URL	URL of a copy of the complete JDF that represents a snapshot of the Job that is currently being processed. The JDF is for reference only and must not be merged with the main JDF of the Job using spawning and merging methods. <i>JDF/@Activation</i> SHOULD be set to "Informative" in this JDF Element. The URL SHOULD reference a MIME part using a "cid" URL scheme.
<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> .

Table 5-71: JobPhase Element (Section 3 of 3)

Name	Data Type	Description
CostCenter ?	element	The cost center that the Job is currently being charged to. Defaults to the cost center specified in the DeviceInfo Element.
JDF ? Deprecated in JDF 1.4	element	Complete JDF Node that represents a snapshot of the Job that is currently being processed. This Element is for reference only and MUST NOT be merged with the main JDF of the Job using spawning and merging methods. JDF/@Activation MUST be set to "Informative" in this JDF Element. Deprecation note: starting with JDF 1.4, JDF has been replaced by URL. This avoids clashes of identical ID Attributes when multiple JobPhase Elements from the same JDF are specified.
MISDetails ? New in JDF 1.2	element	Definition how the costs for this JobPhase are to be charged.
ModuleStatus * New in JDF 1.3	element	Status of individual modules that are used to execute this JobPhase. ModuleStatus MUST NOT be specified for modules that are specified in DeviceInfo/ModuleStatus. For details on using ModuleStatus Elements, see Table 5-72 on page 234.
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 150.

5.9.9.4 Element: ModuleStatus

The ModuleStatus Element is identical to the ModulePhase Element of the PhaseTime Audit Element (see Table 3-40, "ModulePhase Element," on page 129), except that the Attributes *Start* and *End* are missing. These Attributes specify the time interval in the Audit pendant ModulePhase. ModulePhase/@DeviceID Attribute is also not specified because it is already uniquely identified in DeviceInfo/@DeviceID. The ModuleStatus Element is described in the following table.

Table 5-72: ModuleStatus Element (Section 1 of 2)

Name	Data Type	Description
CombinedProcessIndex ? New in JDF 1.3	IntegerList	CombinedProcessIndex Attribute specifies the indices of individual Processes in the Types Attribute to which a ModuleStatus that describes a Combined Process Node or Process Group Node belongs. Multiple entries in CombinedProcessIndex specify that the Module specified by ModuleStatus is executing the respective multiple Processes in the Combined Process Node.

Table 5-72: ModuleStatus Element (Section 2 of 2)

Name	Data Type	Description
<i>DeviceStatus</i>	enumeration	Status of the module. 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 <i>ModuleStatus</i> 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.
<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 Values include those from: Section C.2, "ModuleType Supported Strings" on page 879.. Note: the allowed values depend on the type of Device. Each type of Device has a separate table of values.
<i>StatusDetails</i> ?	string	Description of the module status phase that provides details beyond the enumerative values given by the <i>DeviceStatus</i> Attribute. Values include those from: "StatusDetails Supported Strings" on page 875.
<i>Employee</i> *	element	Employee Resource(s) that represent the employee(s) 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 87.

Example 5-34: Status Response to Query

The following is an example of a Response Message to a Status Query Message. The Device in this example holds one Job and executes another Job that is currently printed duplex (each side) on four-color modules for the front and three-color modules for the back, with one idle:

```
<JMF SenderID="A3 Printer" TimeStamp="2005-07-25T12:32:48+02:00"
  xmlns="http://www.CIP4.org/JDFSchema_1_1"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M1" refID="Q1" Type="Status" xsi:type="ResponseStatus">
    <DeviceInfo DeviceStatus="Running" StatusDetails="Waste">
      <JobPhase Amount="2560" DeadLine="InTime" JobID="678" JobPartID="01"
```

```

        PercentCompleted="52" QueueEntryID="Job-05" Status="InProgress"
        StatusDetails="Waste" />
    <JobPhase Amount="0" DeadLine="Warning" JobID="679" JobPartID="01"
        PercentCompleted="0" QueueEntryID="Job-06" Status="Ready" />
    <ModuleStatus ModuleIndex="0~3 6~8" ModuleType="PrintModule"
        DeviceStatus="Running" />
    <ModuleStatus ModuleIndex="4" ModuleType="PrintModule" DeviceStatus="Idle" />
    <ModuleStatus ModuleIndex="5" ModuleType="PerfectingModule"
        DeviceStatus="Running" />
</DeviceInfo>
</Response>
</JMF>

```

5.9.10 Track

The Track Query Message requests information about the location of Jobs that are known by a Controller. If a high level Controller controls lower level Controllers, it SHOULD also list the Jobs that are controlled by these. The Response Message is a list of TrackResult Elements.

Table 5-73: Track Message

Object Type	Element Name	Description
QueryTypeObj	TrackFilter ?	Refines the Track Query Message.
ResponseTypeObj	TrackResult *	Details of the tracked Jobs.

5.9.10.1 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 5-74: 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, TrackFilter applies to all Jobs.
<i>Status</i> ?	enumerations	The JDF/@ <i>Status</i> of the Jobs being tracked. The value of this attribute is a list of any combination of values. Default value is: all enumerations, if not known or specified. Values are from: JDF/@ <i>Status</i> (Table 3-5, "JDF Node," on page 47.)
<i>Part</i> * New in JDF 1.2	element	Part Elements that describe the Partition of the Job that is being tracked. For details on Node Partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 150.

5.9.10.2 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 5-75: 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.
Part * New in JDF 1.2	element	Part Elements that describe the Partition of the Job that is being tracked. For details on Node Partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 150.

Example 5-35: Track Response

The following is an example of a Response Message on a Track Message:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M1" Type="Track" xsi:type="ResponseTrack" refID="Q1">
    <TrackResult IsDevice="true" JobID="1" JobPartID="42"
      URL="http://www.anycompany.com/controller"/>
  </Response>
</JMF>
```

5.9.11 UpdateJDF

[New in JDF 1.3](#)

This JMF is used to synchronize a JDF Node that has been submitted by a Controller to a Device.

5.9.11.1 UpdateJDF Command

The UpdateJDF Command will be sent from a Controller (e.g. an MIS) to a Device (e.g. a Workflow System) which received the original Job. The changes 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.

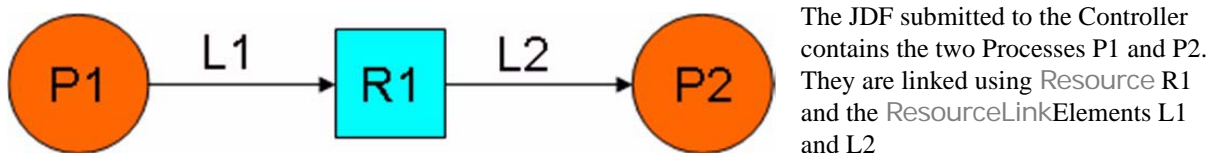
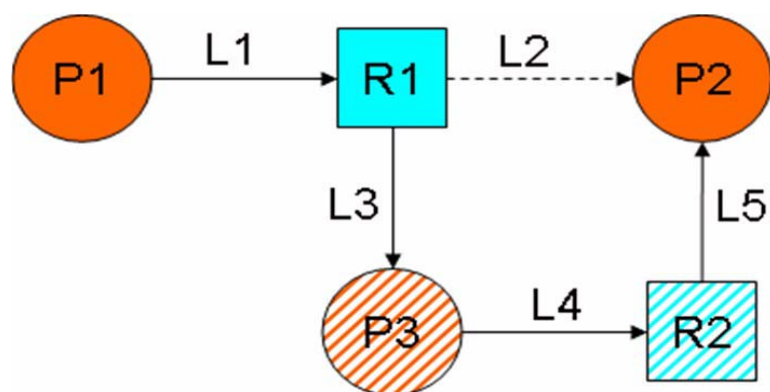


Figure 5-5: Without UpdateJDF Message



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 5-6: With UpdateJDF Message

Table 5-76: UpdateJDF Command

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

5.9.11.2 UpdateJDF Signal

[New in JDF 1.4](#)

The UpdateJDF Signal will be sent from the Device to a Controller. It notifies the Controller about modifications that have occurred on the Device.

Table 5-77: UpdateJDF Signal

Object Type	Element Name	Description
QueryTypeObj	UpdateJDFCmdParams ?	Defines the details of the UpdateJDF Message.
ResponseTypeObj	-	-

5.9.11.2.1 Element: UpdateJDFCmdParams

The UpdateJDFCmdParams specifies a JDF Node, new Resource Elements and new ResourceLink Elements to add to existing Nodes.

Table 5-78: UpdateJDFCmdParams Element (Section 1 of 2)

Name	Data Type	Description
<i>ParentJobID</i>	string	<i>JobID</i> of the Node in which the new Node is to be inserted.
<i>ParentJobPartID</i>	string	<i>JobPartID</i> 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.
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.

Table 5-78: UpdateJDFCmdParams Element (Section 2 of 2)

Name	Data Type	Description
MoveResource *	element	Specifies Resources in previously submitted JDF Nodes that are to be moved to another ResourcePool so that they are accessible for all new JDF Nodes that link to the Resources. Note: MoveResource does not create new Partitions in existing Resources.
RemoveLink *	element	ResourceLink Elements in the previously submitted Job that are no longer in use and are to be removed.
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 <i>JobID</i> and <i>JobPartID</i> that matches an existing JDF Node in the JDF ticket in which the parent Node resides.

5.9.11.2.2 Element: CreateLink**Table 5-79: CreateLink Element**

Name	Data Type	Description
<i>JobID</i>	string	<i>JobID</i> of the Node in which the new ResourceLink is inserted.
<i>JobPartID</i>	string	<i>JobPartID</i> 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.

5.9.11.2.3 Element: CreateResource**Table 5-80: CreateResource Element**

Name	Data Type	Description
<i>JobID</i>	string	<i>JobID</i> of the Node in which the new Resources are to be inserted.
<i>JobPartID</i>	string	<i>JobPartID</i> of the Node in which the new Resources are to be inserted.
Resource +	element	The new Resource Elements. In general, these are created to link the new Node to existing Nodes. The data type and <i>Class</i> of Resource is derived from the Abstract Resource. See “Abstract Resource” on page 63.

5.9.11.2.4 Element: MoveResource**Table 5-81: MoveResource Element**

Name	Data Type	Description
<i>JobID</i>	string	<i>JobID</i> of the Node to which the new Resource is to be moved.
<i>JobPartID</i>	string	<i>JobPartID</i> of the Node in which the new Resource is to be moved.
<i>ResourceID</i>	NMTOKEN	Resource/@ID of the Resource that is moved. Note: If the Resource has been spawned, an error MAY be reported back.

5.9.11.2.5 Element: RemoveLink

Table 5-82: 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 Part Elements specified in this ResourceLink are to be removed.

Note: This Message might not work:

- if one of the Resources or Links have references to a Pipe.
- if the Controller has submitted parts of the Job to a second Controller or a Device.

The JDF after executing the Message is valid

- on a Job which is waiting.
- if all Nodes, to which the new Node is linked are waiting.
- if the link to a running Node is not using a pipe.

Example 5-36: UpdateJDF Command

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" TimeStamp="2005-07-25T12:32:48+02:00"
  DeviceID="AnyDevice" SenderID="JDFMaster"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Command ID="ID1" Type="UpdateJDF" xsi:type="CommandUpdateJDF">
    <UpdateJDFCmdParams ParentJobID="ID100" ParentJobPartID="ID112">
      <CreateLink JobID="ID100" JobPartID="ID111">
        <MediaLink Usage="Input" rRef="link001111"/>
      </CreateLink>
      <CreateResource JobID="100" JobPartID="110">
        <Component rRef="link001112"/>
      </CreateResource>
      <RemoveLink JobID="100" JobPartID="111">
        <MediaLink Usage="Input" rRef="link001113"/>
      </RemoveLink>
      <MoveResource JobID="100" JobPartID="101" ResourceID="link000004"/>
      <JDF JobPartID="200" Type="Cutting">
        <AuditPool>
          <Created Author="MIS" TimeStamp="2005-06-02T09:01:45+01:00"
            AgentName="MIS" AgentVersion="1.0"/>
        </AuditPool>
        <ResourcePool>
          <Component ID="link000002" Class="Quantity" Status="Available"
            ComponentType="Sheet"/>
          <CuttingParams ID="link000007" Class="Parameter" Status="Available"/>
        </ResourcePool>
        <ResourceLinkPool>
          <ComponentLink Usage="Output" rRef="link000002"/>
          <CuttingParamsLink Usage="Input" rRef="link000007"/>
        </ResourceLinkPool>
      </JDF>
    </UpdateJDFCmdParams>
  </Command>
</JMF>
```

5.9.12 WakeUp

[New in JDF 1.2](#)

The WakeUp Command Message activates a Controller or Device that has been in stand-by mode. All queues that belong to the Device are held upon its receiving a WakeUp and 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 ResubmitQueueEntry Message. A Device MUST use the Status Message if it signals its own awakening.

Table 5-83: WakeUp Message

Object Type	Element Name	Description
CommandTypeObj	WakeUpCmdParams ?	Defines the details of the WakeUp Message.
ResponseTypeObj	DeviceInfo	Describes the Device status immediately after the WakeUp Message has been sent. The Device SHOULD also send an Acknowledge/WakeUp Message after its warm up cycle has been completed if applicable.

5.9.12.1 Element: WakeUpCmdParams

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

Table 5-84: WakeUpCmdParams Element

Name	Data Type	Description
—	—	—

5.10 Messages for Pipe Control

JDF Messaging provides methods to control dynamic pipes. Dynamic pipes are described in detail in Section 4.3.3, Overlapping Processing Using Pipes.

Table 5-85: 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	CGR	Requests a new Resource from a pipe.
PipePush	CGR	Notifies that a new Resource is available in a pipe.
PipePause	CR	Pauses a Process if no further Resources can be consumed or produced.

5.10.1 PipeClose

The PipeClose Message notifies the Process at the other end of a dynamic pipe that the sender of this Message needs no further Resources or will produce no further Resources through the pipe. The PipeClose Command Message response is equivalent to the PipePull and PipePush Command Message responses described below.

Table 5-86: PipeClose Message

Object Type	Element Name	Description
CommandTypeObj	PipeParams	Describes the pipe Resource. The PipeParams Element is described in Section 5.10.2, PipePull.
ResponseTypeObj	JobPhase	The status of the responding Process. The JobPhase Element is defined in Table 5-71 on page 232.

5.10.2 PipePull

The PipePull Message requests Resources that are described in a JDF dynamic pipe (see Section 3.8.7, Pipe Resources and Section 4.3.3, Overlapping Processing Using Pipes). PipePull Messages are the JMF equivalent of a dynamic input ResourceLink. Below, depicts the mode of operation of a PipePull Message.

The PipePull Command Message response returns a *ReturnCode* of 0 if the command has been accepted by the receiving Controller. If not successful the *ReturnCode* is one of the codes presented in Section D, Supported Error Codes in JMF and Notification Elements. The Response Message MAY contain a Notification Element. The JobPhase Element (see Section 5.9.9, Status) returned SHOULD provide only the *Status* Attribute that describes the Job status of the responding Process after receiving the command.

Table 5-87: 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 5-71 on page 232.

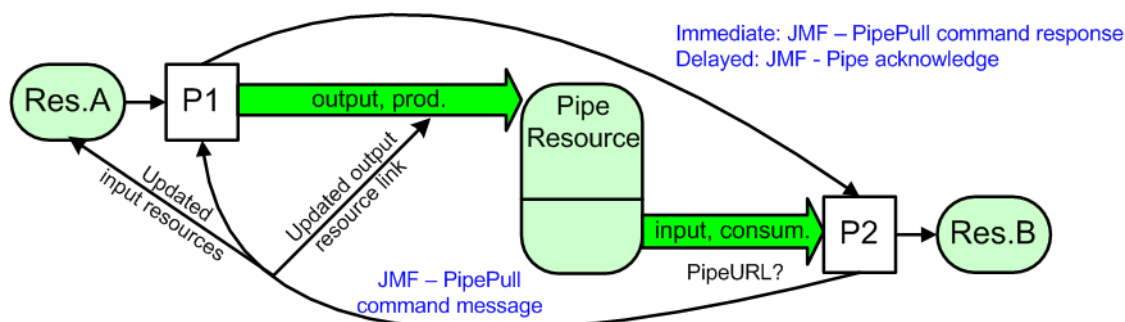


Figure 5-7: Mechanism of a PipePull Message

5.10.2.1 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 5-88: 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. Values are from: JDF/@ <i>Status</i> (Table 3-5, "JDF Node," on page 47.)

Table 5-88: 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.. Values are from: Resource/@ <i>Status</i> (Table 3-10, “Abstract Resource Element,” on page 63.).
Resource *	element	Updated Input Resources to be used by the Process that receives the pipe command: PipePull (the receiver creates the pipe Resource), PipePush (the receiver consumes the pipe Resource) and PipePause (the receiver only updates the inputs). 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: PipePull, PipePush or PipePause. In case of the PipeClose Command Message, the Resources are ignored. The data type and <i>Class</i> of Resource is derived from the Abstract Resource. See “Abstract Resource” on page 63.
ResourceLink ?	element	Updated ResourceLink to the pipe Resource: PipePull (it is an output link), PipePush (it is an input link) and PipePause (depends on the pipe end). This ResourceLink MAY be used by the Process that links to the pipe Resource. The Attributes <i>rRef</i> and <i>Usage</i> of a ResourceLink MUST NOT be modified by the Agent that sends the Pipe Control Message because these Attributes are used by the JMF receiver to identify the ResourceLink that is to be modified. For details see Section 3.8.8, ResourceUpdate. In the context of dynamic pipes these two Attributes have no meaning. In case of the PipeClose Command Message, the ResourceLink is ignored.

5.10.3 PipePush

The PipePush Message notifies the availability of pipe Resources that are described in a JDF dynamic pipe (see Section 3.8.7, Pipe Resources and Section 4.3.3, Overlapping Processing Using Pipes). PipePush Messages are the JMF equivalent of a dynamic output ResourceLink. The Figure 5-8 depicts the mode of operation of a PipePush Message. The PipePush Command Message response is equivalent to the PipePull Command Message Response described above.

Table 5-89: PipePush Message

Object Type	Element Name	Description
CommandTypeObj	PipeParams	Describes the produced pipe Resource. The PipeParams Element is described in Section 5.10.2, PipePull.
ResponseTypeObj	JobPhase	The status of the responding Process. The JobPhase Element is defined in Table 5-71 on page 232.

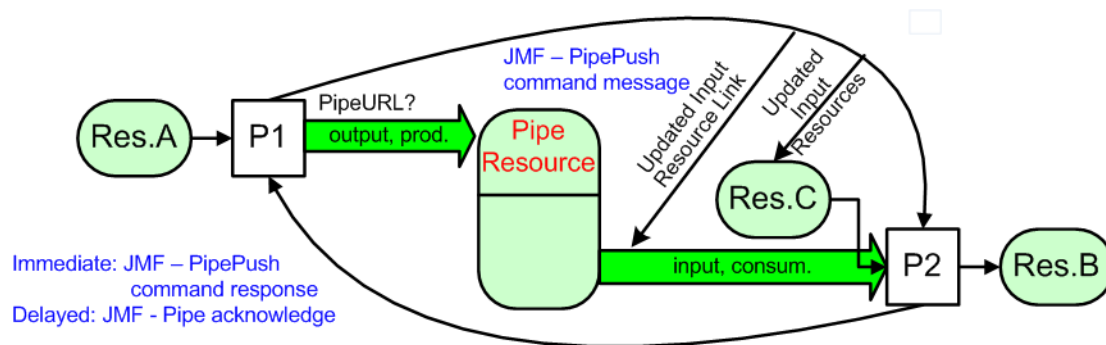


Figure 5-8: Mechanism of a PipePush Message

5.10.4 PipePause

The PipePause Message pauses execution of a Process that is at the other end of a dynamic pipe. The PipePause Command Message response is equivalent to the PipePull Command Message response described above.

Table 5-90: PipePause Message

Object Type	Element Name	Description
CommandTypeObj	PipeParams	Describes the pipe Resource. The PipeParams Element is described in Section 5.10.2, PipePull.
ResponseTypeObj	JobPhase	The status of the responding Process. The JobPhase Element is defined in Table 5-71 on page 232.

5.11 Queue Support

In JMF, a Controller or Device is assumed to have one input queue that accepts submitted Jobs. Controllers which receive submitted Jobs 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 5.7, 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).

5.11.1 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 Message. *QueueEntryID* MUST uniquely identify an entry within the scope of one queue. An implementation is free to choose the algorithm that generates *QueueEntryID* values.

5.11.2 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 264.) 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.

5.12 Messages for Queue Entry Handling

Queue-entry handling is provided so that the state of individual Jobs within a queue can be changed. Job submission, queue-entry grouping, priorities and hold / suspend / resume of entries are all supported. The individual commands are defined in the table and explained in greater detail in the sections that follow.

Table 5-91: 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 Message 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 n, where n represents a numerical value. "0" = pole position. Priority is set to the priority of the Job at position n.
<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 Message is received.

The following table specifies the status transitions for the respective queue entry handling Messages. The error(n) indicates the ReturnCode which is returned on an illegal Status transition and the queue entry Status is unchanged. For details on error codes, see "Supported Error Codes in JMF and Notification Elements" on page 891.

The following are codes for the following table:

A: Aborted

C: Completed

H: Held

PR: PendingReturn [New in JDF 1.4](#)

Rm: Removed

Rn: Running

S: Suspended

W: Waiting

number: Error that specified number, e.g. "105" means "error(105)"

Table 5-92: Status Transitions for QueueEntry Handling Messages

Previous Status Message type	Non-existent	W	H	Rn	S	PR	C	A
AbortQueueEntry	105	A	A	A	A	114	114	113
HoldQueueEntry	105	H	113	106	106	114	114	114
RemoveQueueEntry	105	Rm	Rm	106	106	106	Rm	Rm
RequestQueueEntry	RequestQueueEntry is emitted by the Controller of the queue and not sent to the queue. Therefore it is not applicable in this section.							
ResubmitQueueEntry	105	W	H	Rn + W + 107	S + 107	114	Rn + W + 114	Rn + W + 114
ResumeQueueEntry	105	113	W	113	Rn	114	114	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	105	W	H	107	107	114	114	114
SetQueueEntryPriority	105	W	H	107	107	114	114	114
SubmitQueueEntry	W,H, Rn	A new <i>QueueEntryID</i> is generated by the queue owner on submission. Therefore these states are not applicable.						
SuspendQueueEntry	105	115	115	S	113	114	114	114

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

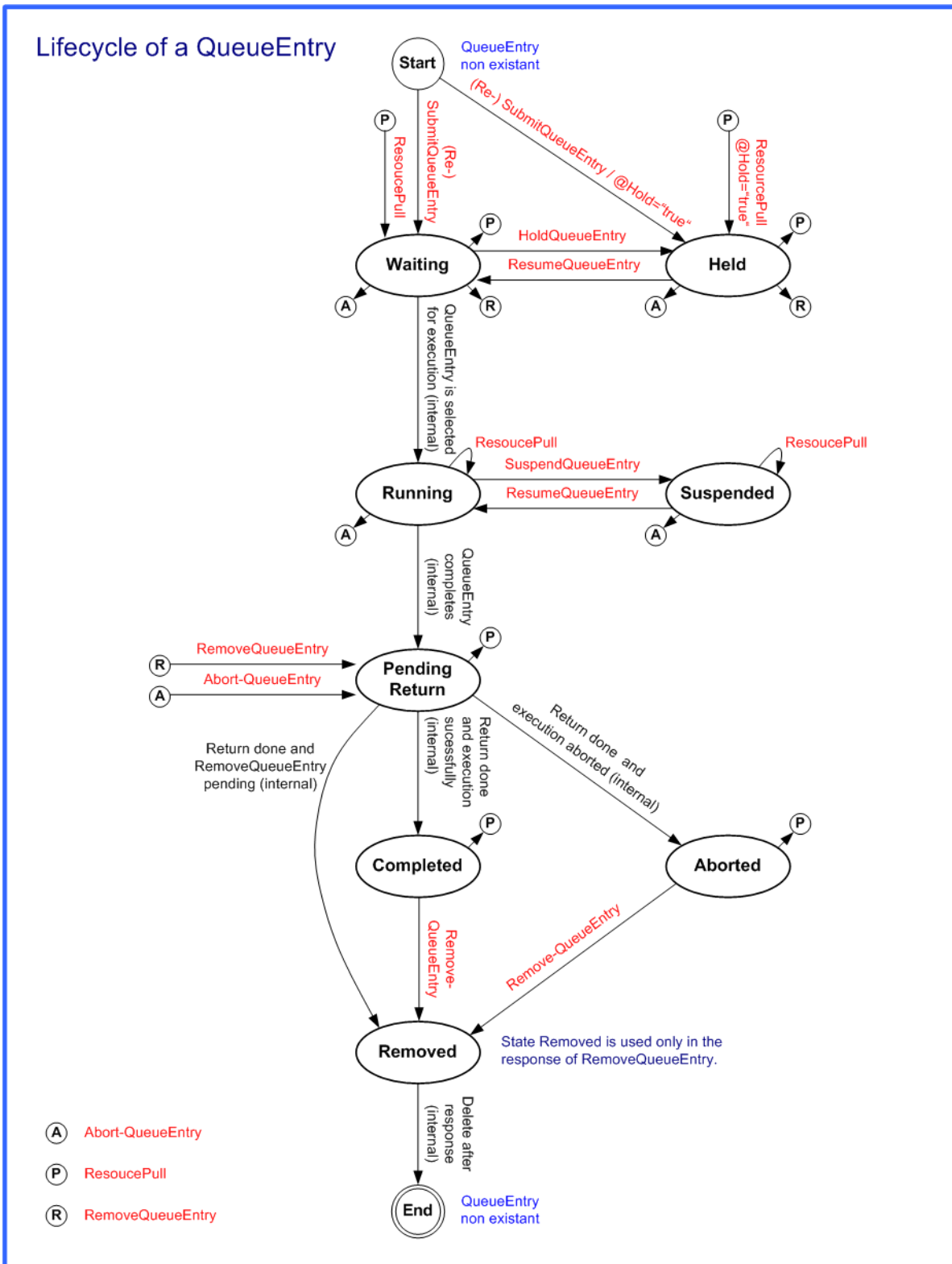


Figure 5-9: JMF QueueEntry Status Transition Diagram

5.12.1 AbortQueueEntry

Once this command is issued, the entry specified by `QueueEntryDef` is aborted and remains in the `Queue` with `QueueEntry/@Status="Aborted"`. The Audit Elements 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 5-93: 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 Queue Element in the AbortQueueEntry 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 5.14, Elements for Queues.		

Example 5-37: AbortQueueEntry Command

The following example demonstrates how an `AbortQueueEntry` Command Message causes a Job in a queue to be aborted and only return the `Status` of the aborted `QueueEntry` in the response, rather than the entire `Queue`:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="A3 Printer"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Command ID="M009" Type="AbortQueueEntry" xsi:type="CommandAbortQueueEntry">
    <QueueEntryDef QueueEntryID="job-0032"/>
    <QueueFilter>
      <QueueEntryDef QueueEntryID="job-0032"/>
    </QueueFilter>
  </Command>
</JMF>
```

Example 5-38: AbortQueueEntry Response

The following example shows a possible Response Message to the Command Message example above:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
  TimeStamp="2005-07-25T12:32:48+02:00" MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M109" Type="AbortQueueEntry" xsi:type="ResponseAbortQueueEntry"
    refID="M009">
    <Queue DeviceID="A3 Printer" Status="Running">
      <QueueEntry JobID="job-0032" QueueEntryID="job-0032" Status="Aborted"/>
    </Queue>
  </Response>
</JMF>
```

5.12.2 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 Message 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 246.

Table 5-94: 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 Queue Element in the HoldQueueEntry 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 5.14, Elements for Queues.		

5.12.3 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 246.

Table 5-95: 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 Queue Element in the RemoveQueueEntry 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 5.14, Elements for Queues.		

5.12.4 RequestQueueEntry

[New in JDF 1.2](#)

This command requests a new queue entry from a potential submitting Agent. The actual submission is still handled by the standard queue entry handling parameters. Note that this command is emitted from the Device that is represented by the queue to a Controller or dispatcher and not to the queue, as is the case with most other queue handling commands.

Table 5-96: RequestQueueEntry Message

Object Type	Element Name	Description
CommandTypeObj	RequestQueueEntryParams	Defines the specifics for the requested Job.
ResponseTypeObj	—	The response to this message contains no <code>ResponseTypeObj</code> , only an empty Response element that specifies the <code>ReturnCode</code> . 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 5.14, Elements for Queues.		

5.12.4.1 Element: RequestQueueEntryParams

Table 5-97: RequestQueueEntryParams Element

Name	Data Type	Description
<i>JobID?</i>	string	<i>JobID</i> of the requested <i>QueueEntry</i> .
<i>JobPartID?</i>	string	<i>JobPartID</i> of the requested <i>QueueEntry</i> .
<i>QueueURL</i>	URL	URL of the <i>Queue</i> Controller that is requesting the <i>QueueEntry</i> and will accept <i>Queue</i> manipulation Messages.
<i>SubmitPolicy?</i> New in JDF 1.3	enumeration	Defines the requested policy for submitting the Node. If not specified, the submission policy is dependent on the Controller implementation. <i>SubmitPolicy</i> allows a Device to request a Node that would otherwise not be submitted by the Controller due to missing Resources. Values are: <i>Standard</i> : All linked Resources MUST have a <i>Resource/@Status</i> as defined by <i>ResourceLink/@MinStatus</i> . <i>Late</i> : All linked Resources MUST have a <i>Resource/@Status</i> as defined by <i>ResourceLink/@MinLateStatus</i> . <i>Force</i> : The Node MUST be submitted regardless of the values of linked <i>Resource/@Status</i> .
Part *	element	Partition parts of the requested <i>QueueEntry</i> .
Queue ?	element	Representation of the current status of the Device's <i>Queue</i> .

5.12.5 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*. If *QueueEntry/@Status* is neither "*waiting*" nor "*held*", resubmitting a queue entry MAY fail because a Device NEED NOT implement *ResubmitQueueEntry* for running queue entries. 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 246.

Table 5-98: ResubmitQueueEntry Message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	<i>ResubmissionParams</i>	Defines the Job resubmission.
	<i>QueueFilter?</i> New in JDF 1.2	Defines a filter for the returned <i>Queue</i> Element in the <i>ResubmitQueueEntry</i> Message.
ResponseTypeObj	<i>Queue</i>	Describes the state of the queue after the command has been executed.
For the definition of the <i>Queue</i> Element, see Section Section 5.14, Elements for Queues.		

5.12.5.1 Element: ResubmissionParams

Table 5-99: ResubmissionParams Element

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

5.12.6 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 246.

Table 5-100: ResumeQueueEntry Message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	<i>QueueEntryDef</i>	Defines the queue entry.
	<i>QueueFilter</i> ? New in JDF 1.2	Defines a filter for the returned Queue Element in the ResumeQueueEntry Message.
ResponseTypeObj	<i>Queue</i>	Describes the state of the queue after the command has been executed.
For the definition of the Elements listed above, see Section 5.14, Elements for Queues.		

5.12.7 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 5-101: ReturnQueueEntry Message

Object Type	Element Name	Description
CommandTypeObj	<i>ReturnQueueEntryParams</i>	Defines the Job being returned from Device to Controller after processing is completed or aborted.
ResponseTypeObj	-	

5.12.7.1 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 5-102: ReturnQueueEntryParams Element (Section 1 of 2)

Name	Data Type	Description
<i>Aborted?</i>	NMTOKENS	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. Note that the data type of this Attribute was erroneously specified as IDFREFS in JDF 1.2. and JDF 1.3.

Table 5-102: ReturnQueueEntryParams Element (Section 2 of 2)

Name	Data Type	Description
<i>Completed?</i>	NMTOKENS	ID of the JDF Nodes that have been executed and completed or succeeded in test run. Note that the data type of this Attribute was erroneously specified as IDFREFS in JDF 1.2. and JDF 1.3.
<i>Priority?</i>	integer	The priority of the <i>QueueEntry</i> when it was executed on the Device. The Controller receiving this Message MAY prioritize this Job for continued processing based on this value.
<i>QueueEntryID</i>	NMTOKEN	<i>QueueEntry/@QueueEntryID</i> of the returned queue entry. Note that this Attribute was erroneously omitted in JDF 1.2. and JDF 1.3.
<i>URL</i>	URL	Location of the JDF to be returned. Note that the <i>URL</i> SHOULD be queried with a <i>SubmissionMethods</i> Query Message to determine whether MIME Multipart/Related is supported

5.12.8 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 246.

Table 5-103: SetQueueEntry Message

Object Type	Element Name	Description
CommandTypeObj Modified in JDF 1.2	<i>QueueEntryPosParams</i>	Defines the queue entry.
	<i>QueueFilter ?</i> New in JDF 1.2	Defines a filter for the returned <i>Queue</i> Element in the <i>SetQueueEntryPosition</i> Message.
ResponseTypeObj	<i>Queue</i>	Describes the state of the queue after the command has been executed.

For the definition of the *Queue* Element, see Section 5.14, Elements for Queues.

5.12.8.1 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 5-104: 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.

5.12.9 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 246.

Table 5-105: 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 5.14, Elements for Queues.		

5.12.9.1 Element: QueueEntryPriParams

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

Table 5-106: QueueEntryPriParams Element

Name	Data Type	Description
<i>Priority</i>	integer	Number from 0 to 100, where "0" = lowest priority and "100" = maximum priority. The priority from <code>QueueSubmissionParams/@Priority</code> and <code>QueueEntryPriParams/@Priority</code> takes precedence over <code>NodeInfo/@JobPriority</code> .
<i>QueueEntryID</i>	string	ID of a queue entry.

5.12.10 SubmitQueueEntry

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

Table 5-107: 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 5.14, Elements for Queues.		

5.12.10.1 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 5-108: 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 <i>QueueEntry</i> SHOULD be executed together with other <i>QueueEntry</i> 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 <i>QueueEntry</i> . 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 <i>QueueEntry</i> 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 <i>QueueEntry</i> 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.
<i>Hold</i> = "false"	boolean	If <i>true</i> , the entry is submitted as with <i>QueueEntry/@Status="Held"</i> . 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>NodeInfo/@Priority</i> . <i>QueueSubmissionParams/@Priority</i> specifies the priority in the context of the Device queue whereas <i>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>NodeInfo/@FirstStart</i>). The priority from <i>QueueSubmissionParams/@Priority</i> and <i>QueueEntryPriParams/@Priority</i> takes precedence over <i>NodeInfo/@JobPriority</i> .
<i>refID</i> ? New in JDF 1.2	NMTOKEN	Copy of the <i>ID</i> Attribute of the initiating <i>RequestQueueEntry</i> Message.

Table 5-108: QueueSubmissionParams Element (Section 2 of 2)

Name	Data Type	Description
ReturnJMF ? New in JDF 1.2	URL	Address of a JMF queue where a ReturnQueueEntry Message is to be sent when the QueueEntry is completed or aborted. Note that the ReturnJMF queue SHOULD be queried with a SubmissionMethods Query Message to determine whether MIME Multipart/Related is supported by the return queue. ReturnJMF MUST NOT be specified if ReturnURL is present.
ReturnURL ? Modified in JDF 1.2	URL	URL where the JDF file is to be written when the QueueEntry is completed or aborted. A Controller MUST write only a JDF document to the URL and MUST NOT write a MIME Multipart package to the URL. If ReturnURL is specified with the "file" scheme, ReturnURL MUST specify an individual file. ReturnURL MUST take precedence when NodeInfo/@TargetRoute is specified in the submitted JDF. Note: A Controller MUST NOT return a JDF file or MIME Multipart/Related file by performing a SubmitQueueEntry or ReturnQueueEntry to the ReturnURL URL. The Controller specified by ReturnURL MUST NOT accept JMF Messages. See instead ReturnJMF. ReturnURL MUST NOT be specified if ReturnJMF is present.
URL Modified in JDF 1.2	URL	Location of the JDF to be submitted. In the case of MIME Multipart/Related, the URL MAY have a "cid" scheme.
WatchURL ? Modified in JDF 1.2	URL	URL of the Controller that is to be notified when the status of the QueueEntry or the underlying Job changes.
Disposition ? New in JDF 1.2	element	Definition how long the QueueEntry SHOULD be retained in the queue. If not specified, the QueueEntry MAY be removed from the queue immediately after Process completion of the QueueEntry.

URL with “file” Scheme

If the URL has a "file" scheme, the Device retrieves the file at the location specified in the URL Attribute. The following example declares a file on the network:

Example 5-39: SubmitQueueEntry Command with “file” Scheme

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="A3 Printer"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Command ID="M1" Type="SubmitQueueEntry" xsi:type="CommandSubmitQueueEntry">
    <QueueSubmissionParams URL="File://MyNetWorkShare/AnyDirectory/job1.jdf"/>
  </Command>
</JMF>
```

URL with “http” Scheme

In this example, the queue Controller retrieves the file with a standard HTTP **get** command from a host that MAY be remote. The Job delivered as a response to the HTTP **get** command MAY be a MIME Multipart/Related entity. The HTTP server MAY retrieve a file or it MAY generate the response dynamically with a CGI script or other such tool.

Example 5-40: SubmitQueueEntry Command with “http” Scheme

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Command ID="M2" Type="SubmitQueueEntry" xsi:type="CommandSubmitQueueEntry" >
    <QueueSubmissionParams URL="http://JobServer.JDF.COM?job1"/>
  </Command>
</JMF>
```

```
</Command>
</JMF>
```

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 851 for details of MIME Multipart/Related packaging.

5.12.11 SuspendQueueEntry

[New in JDF 1.2](#)

The entry specified by `QueueEntryDef` is suspended if its *Status* is "Running". Its *Status* is set to "Suspended". Whether other queue entries can be run while the queue entry remains suspended depends on implementation. The `SuspendQueueEntry` Command Message has no effect on Jobs with a *Status* other than "Running". For details, see "Status Transitions for QueueEntry Handling Messages" on page 246.

Table 5-109: SuspendQueueEntry Message

Object Type	Element Name	Description
CommandTypeObj	QueueEntryDef	Defines the queue entry.
	QueueFilter ?	Defines a filter for the returned Queue Element in the SuspendQueueEntry Message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed. See "Elements for Queues" on page 261. for the definition of the Elements listed above. The entry specified by <code>QueueEntryDef</code> remains in the queue but moved into the <i>Suspended</i> state.
For the definition of the Elements listed above, see Section 5.14, Elements for Queues.		

5.13 Messages for Global Handling of Queues

Whereas the commands in the preceding section change the state of an individual queue entry, the commands in this section modify the state of an entire queue. Note that entries that are executing in a Device are not affected by the global queue-handling commands and 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 5-110: Messages for global handling of queues

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.
OpenQueue	CR	The queue is opened. Jobs are to be accepted.
QueueEntryStatus Deprecated in JDF 1.2	QRS	Returns a QueueEntry Element.
QueueStatus	QRS	Returns the Queue 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 5-111: 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>

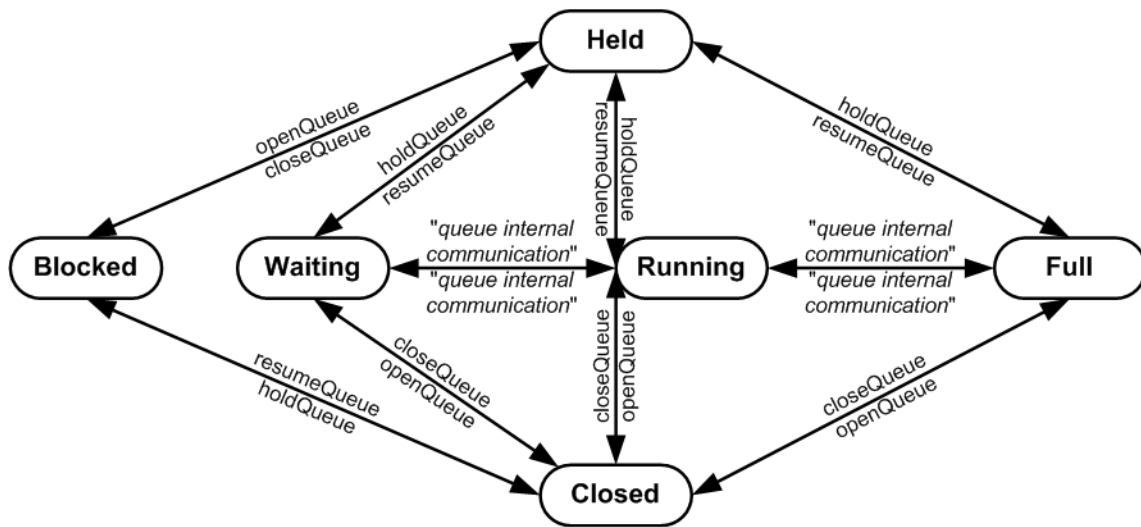


Figure 5-10: Effects of the global queue Messages on the queue Status

5.13.1 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 5-112: 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 5.14, Elements for Queues.		

5.13.2 FlushQueue

5.13.2.1 FlushQueue Command

FlushQueue Command is used to remove QueueEntry Elements from the Queue. Note: A QueueEntry is not automatically deleted when executed or aborted, but rather it remains in the Queue and its Status is changed to "Completed" or "Aborted" accordingly. FlushQueueParams allows the specification of which QueueEntry Elements to remove. The QueueFilter in the FlushQueue Message is applied to the Queue returned after the com-

mand is executed. The `QueueFilter` contained within the `FlushQueueParams` is used to specify which `QueueEntry` Elements to remove.

Table 5-113: 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 = "Waiting"</i> and <i>Status = "Held"</i> 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 5.14, Elements for Queues.		

5.13.2.1.1 Element: FlushQueueParams[New in JDF 1.2](#)**Table 5-114: FlushQueueParams Element**

Name	Data Type	Description
QueueFilter ?	element	Defines a <code>QueueFilter</code> that specifies the <code>QueueEntry</code> Elements to be removed. If not specified, the <code>Queue</code> is completely flushed.

5.13.2.2 FlushQueue Query

When used as a Signal or Query, `FlushQueue Query` allows a Controller to monitor queue flushing that is initiated by the Device, (e.g., due to Resource constraints.) The `QueueFilter` in the `FlushQueue` Message is applied to the `Queue` returned after the command is executed. The `QueueFilter` contained within the `FlushQueueInfo` is used to specify which `QueueEntry` Elements were removed.

Table 5-115: 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 5.14, Elements for Queues.		

5.13.2.2.1 Element: FlushQueueInfo[New in JDF 1.2](#)

The `QueueFilter` in `FlushQueueParams` defines the `QueueEntry` Elements to be removed by `FlushQueue`. Those `QueueEntry` Elements meeting the criteria set in the `QueueFilter` will be removed.

Table 5-116: FlushQueueInfo Element

Name	Data Type	Description
QueueFilter	element	Defines a <code>QueueFilter</code> that specifies the <code>QueueEntry</code> Elements that were removed.

5.13.3 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 Message.

Table 5-117: 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 5.14, Elements for Queues.		

5.13.4 OpenQueue

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

Table 5-118: OpenQueue 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 OpenQueue Message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.
For the definition of the Queue Element, see Section 5.14, Elements for Queues.		

5.13.5 QueueEntryStatus

[Deprecated in JDF 1.2](#)

Deprecation note: starting with JDF 1.2, use `QueueStatus` with an appropriate `QueueFilter` instead of `QueueEntryStatus`. See "QueueEntryStatus" on page 1011 for details of this deprecated JMF Element.

5.13.6 QueueStatus

Returns a queue description.

Table 5-119: QueueStatus Message

Object Type	Element Name	Description
QueryTypeObj Modified in JDF 1.2	QueueFilter ? New in JDF 1.2	Defines a filter for the QueueStatus Message.
ResponseTypeObj	Queue	Describes the status of the queue.
For the definition of the Queue Element, see Section 5.14, Elements for Queues.		

5.13.7 ResumeQueue

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

Table 5-120: ResumeQueue 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 ResumeQueue Message.

Table 5-120: ResumeQueue Message (Section 2 of 2)

Object Type	Element Name	Description
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.
For the definition of the Queue Element, see Section 5.14, Elements for Queues.		

5.13.8 SubmissionMethods

The `SubmissionMethods` Message returns information about the messaging formats that are supported by a Device or Controller. For example, it can be used to determine how a `SubmitQueueEntry` Message can be sent to a Device, or how a `ReturnQueueEntry` Message can be sent to a Controller.

Table 5-121: SubmissionMethods Message

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

5.13.8.1 Element: SubmissionMethods

The Response Message Element MAY contain multiple Attributes, as defined below. If an Attribute is not specified, the corresponding submission method is not supported.

Table 5-122: SubmissionMethods Element

Name	Data Type	Description
<i>File</i> ? Deprecated in JDF 1.2	boolean	Can retrieve a JDF from a File specified in the URL In JDF 1.2 and beyond, include " <i>file</i> " in <i>URLSchemes</i> .
<i>HotFolder</i> ? Deprecated in JDF 1.4	URL	URL specification of a hot folder location. Deprecation note: starting with JDF 1.4, use the <code>KnownDevices</code> Response: <code>/JMF/Response/DeviceInfo/Device/@JDFInputURL</code>
<i>HttpGet</i> ? Deprecated in JDF 1.2	boolean	Can retrieve a JDF via HTTP get commands. In JDF 1.2 and beyond, include " <i>http</i> " in <i>URLSchemes</i> .
<i>Packaging</i> ? New in JDF 1.2 Modified in JDF 1.4	enumerations	List of packaging methods supported. Default behavior: the Controller does not support receiving packaged Messages and MUST retrieve JDF files using a URL with a scheme other than "cid". 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 ".mjm" extension. <i>None</i> – no form of packaging is supported. New in JDF 1.4
<i>MIME</i> ? Deprecated in JDF 1.2	boolean	Accepts MIME Multipart/Related submission Messages via a Message post. In JDF 1.2 and beyond, use <i>Packaging</i> = " <i>MIME</i> ".
<i>URLSchemes</i> ? 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)

Example 5-41: SubmissionMethods Response

The following is an example of a Response Message to a SubmissionMethods Query Message:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="A3 Printer"
  SenderID="MIS master A" TimeStamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M1" Type="SubmissionMethods"
    xsi:type="ResponseSubmissionMethods" refID="Q1">
    <SubmissionMethods HotFolder="file://MyDevice/HotFolder" Packaging="MIME"
      URLSchemes="http file ftp"/>
  </Response>
</JMF>
```

5.14 Elements for Queues

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

5.14.1 Queue

The Attributes in the following table are defined for Queue Message Elements. Queue Elements represent the queue of a Device including QueueEntry Elements that represent both pending and running queue entries.

Table 5-123: Queue Element (Section 1 of 2)

Name	Data Type	Description
<i>DeviceID</i>	string	Identifies the Device that is represented by the queue.
<i>QueueSize</i> ? New in JDF 1.2	integer	The maximum number of QueueEntry Elements that can be in the Queue. Note: QueueEntry[@Status="Completed" or @Status="Aborted"] Elements MUST NOT count towards determining Queue/@Status based on the number of QueueEntry Elements versus the QueueSize.
<i>Status</i>	enumeration	Status of the queue. 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 OpenQueue or ResumeQueue 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 OpenQueue Command Message. <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 ResumeQueue Command Message.
<i>Device</i> *	element	The Devices that execute entries in this queue. Only Device/@DeviceID SHOULD be specified in these Device Elements.

Table 5-123: Queue Element (Section 2 of 2)

Name	Data Type	Description
QueueEntry * Modified in JDF 1.2	element	<p>QueueEntry Elements (see Table 5-124, “QueueEntry Element,” on page 262, 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 5-124, “QueueEntry Element,” on page 262.</p> <p>A QueueEntry is not automatically deleted when executed or aborted, but rather it remains in the Queue and its Status is changed to Completed or Aborted accordingly. QueueEntry[@Status = "Completed" or @Status = "Aborted"] Elements MUST NOT count towards determining Queue/@Status based on the number of QueueEntry Elements versus the QueueSize.</p>

Example 5-42: Queue Element

Example of a Queue Element:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="MIS"
  DeviceID="PressSystem" TimeStamp="2000-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Response ID="M1001" Type="SubmitQueueEntry"
    xsi:type="ResponseSubmitQueueEntry" refID="M170">
    <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>
  </Response>
</JMF>
```

5.14.2 QueueEntry

Table 5-124: 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. GangName MUST be specified, if the QueueEntry is member of a gang Job.
GangPolicy ? New in JDF 1.3	enumeration	Ganging policy for the QueueEntry. Values are from: QueueSubmissionParams/@GangPolicy (Table 5-108, “QueueSubmissionParams Element,” on page 254).

Table 5-124: QueueEntry Element (Section 2 of 2)

Name	Data Type	Description
JobID ? Modified in JDF 1.1	string	The Job ID of the JDF Process.
JobPartID ?	string	The JobPartID of the JDF Process.
Priority = "1"	integer	Priority of the QueueEntry. Values are 0-100."0" is the lowest priority, while "100" is the highest priority.
QueueEntryID	string	ID of a QueueEntry. This ID 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	<p>Status of the individual entry.</p> <p>Values are:</p> <p><i>Running</i> – The queue entry is running on the Device. A QueueEntry is <i>Running</i> when JDF/@Status of any node associated to the QueueEntry is one of <i>Setup</i>, <i>InProgress</i> or <i>Cleanup</i>.</p> <p><i>Waiting</i> – The queue entry is waiting and will be executed when Resources are available.</p> <p><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.</p> <p><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.</p> <p><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 the QueueEntry has been suspended using the SuspendQueueEntry message or a UI equivalent on the device. New in JDF 1.2</p> <p><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</p> <p><i>Completed</i> – Indicates that the queue entry has been executed correctly, and is finished. New in JDF 1.2</p> <p><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</p>
SubmissionTime ?	dateTime	Time when the entry was submitted to the queue.
JobPhase * New in JDF 1.2	element	Description of the current status of the Job that is associated with the QueueEntry. Note that in JDF 1.3 and above, one QueueEntry MAY have multiple active JobPhase Elements.
Part * New in JDF 1.2	element	Describes which parts of a Job were submitted to the queue. The Part Elements are copies of AncestorPool/Part of the JDF Node that is executed by the Device.
Preview * New in JDF 1.2	element	Any number of Preview Elements MAY be associated with a QueueEntry and used for display purposes. Preview/@PreviewUsage SHOULD be Thumbnail or Viewable.

5.14.3 QueueEntryDef

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

Table 5-125: QueueEntryDef Element

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

5.14.4 QueueFilter

[New in JDF 1.2](#)

The *QueueFilter* Element defines a filter for all Messages that return a queue. The supplied Elements of the *QueueFilter* define a matching criteria that is a logical “and”. Only *QueueEntry* Elements that match all restrictions specified by the *QueueFilter* are included in the *Queue* Element that is returned by the queue-handling Message. The *QueueFilter* Element is also used to specify the *QueueEntry* Elements to be removed by the *FlushQueue* Message.

Table 5-126: QueueFilter Element (Section 1 of 2)

Name	Data Type	Description
<i>GangNames</i> ? New in JDF 1.3	NMTOKENS	Gang names of the <i>QueueEntry</i> Elements to be returned. If not specified, there is no filtering on <i>QueueEntry/@GangName</i> .
<i>JobID</i> ? New in JDF 1.4	string	Return only <i>QueueEntry</i> Elements with specified <i>JobID</i> . If not specified, there is no filtering on <i>QueueEntry/@JobId</i> .
<i>JobPartID</i> ? New in JDF 1.4	string	Return only <i>QueueEntry</i> Elements with specified <i>JobPartID</i> . If not specified, there is no filtering on <i>QueueEntry/@JobPartID</i> .
<i>MaxEntries</i> ?	integer	Maximum number of <i>QueueEntry</i> Elements to provide in the <i>Queue</i> Element. If not specified, fill in all matching <i>QueueEntry</i> Elements.
<i>OlderThan</i> ?	dateTime	Only <i>QueueEntry</i> Elements with a <i>SubmissionTime</i> older than or equal to this <i>dateTime</i> are provided in the <i>Queue</i> Element or removed by the <i>FlushQueue</i> Message. If not specified, there is no <i>dateTime</i> lower bound on candidates.
<i>NewerThan</i> ?	dateTime	Only <i>QueueEntry</i> Elements with a <i>SubmissionTime</i> newer than or equal to this <i>dateTime</i> are provided in the <i>Queue</i> Element or removed by the <i>FlushQueue</i> Message. If not specified, there is no <i>dateTime</i> upper bound on candidates.
<i>PreviewUsages</i> ? New in JDF 1.4	enumerations	Specifies the particular kind (or kinds) of Preview Resources to return in <i>QueueEntry/Preview</i> . If <i>PreviewUsages</i> is empty or not supplied, the <i>QueueEntry</i> Element MUST NOT contain any Preview Resources. The Preview Resources returned in a <i>QueueEntry</i> are a subset of those in the actual <i>QueueEntry</i> defined by: <i>QueueEntry/Preview</i> [contains (<i>QueueFilter/@PreviewUsages</i> , <i>@PreviewUsage</i>)] Values are from: <i>Preview/@PreviewUsages</i> (Table 7-313, “Preview Resource,” on page 687).

Table 5-126: QueueFilter Element (Section 2 of 2)

Name	Data Type	Description
<i>QueueEntryDetails</i> = " <i>Brief</i> " Modified in JDF 1.4	enumeration	Refines the level of provided information about the Queue. Values are: <i>None</i> – Do not fill in the QueueEntry Elements into the Queue. <i>Brief</i> – Provide all available QueueEntry information except for the associated JobPhase Element. <i>JobPhase</i> – Provide all available QueueEntry information including the associated JobPhase Elements. <i>JDF</i> – Provide all available QueueEntry information including the associated JobPhase Element and the associated JDF Element in the JobPhase Element. Deprecation note: starting with JDF 1.4, use the Status Query to retrieve status information including information about the current JDF. Deprecated in JDF 1.4
<i>StatusList?</i>	enumerations	Only QueueEntry Elements with a <i>Status</i> matching one of the entries in <i>StatusList</i> are considered. Values are from: QueueEntry/@ <i>Status</i> (Table 5-124, “QueueEntry Element,” on page 262).
<i>UpdateGranularity?</i> New in JDF 1.4	enumeration	Specifies whether all or only the updated QueueEntry Elements should be included in the Queue. Values are: <i>All</i> – The Queue Element describes all QueueEntry Elements. <i>ChangesOnly</i> – The Queue Element describes only those QueueEntry Elements that have new information since the last Queue Element was sent. When used in conjunction with a Signal, the Queue Element describes all Jobs on the first instance of the Signal being sent.
QueueEntryDef *	element	Defines an explicit list of queue entries. If not specified, all entries in the Queue are considered.
Device *	element	Devices that returned queue entries are targeted to. QueueEntry/@ <i>DeviceID</i> MUST match QueueFilter/Device/@ <i>DeviceID</i> for the QueueEntry to be returned in the queue. If not specified, all entries in the Queue are considered.
Part * New in JDF 1.4	element	Return only QueueEntry Elements with all specified Part Elements. If not specified, there is no filtering on QueueEntry/Part.

5.15 Gang Jobs

[New in JDF 1.3](#)

JMF provides a mechanism to specify groups of QueueEntry Elements within a queue that are processed together in a gang. A Job is submitted to a gang by specifying QueueSubmissionParams/@*GangPolicy*.

Table 5-127: Messages for Gang Jobs

Message type	Family	Description
ForceGang New in JDF 1.3	CR	A gang is forced to execute.
GangStatus New in JDF 1.3	CR	The status of a gang is queried.

5.15.1 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 5-128: Contents of the ForceGang Command Message

Object Type	Element Name	Description
CommandTypeObj	GangCmdFilter	Defines the gang(s) to be force executed.
ResponseTypeObj	—	

5.15.1.1 Element: GangCmdFilter

Table 5-129: GangCmdFilter Element

Object Type	Element Name	Description
<i>GangNames</i> ?	NMTOKENS	<i>GangName</i> of the gang(s) being queried.

5.15.2 GangStatus

[New in JDF 1.3](#)

GangStatus returns a description of the gang(s). Details are specified in GangInfo Element.

Table 5-130: GangStatus Message

Object Type	Element Name	Description
QueryTypeObj	GangQuFilter ?	Defines a filter for the gang(s) that are queried. If GangQuFilter is not supplied, all gangs are queried.
ResponseTypeObj	GangInfo *	Describes the status of the gang(s).

5.15.2.1 Element: GangQuFilter

Table 5-131: GangQuFilter Element

Name	Data Type	Description
<i>GangNames</i> ?	NMTOKENS	<i>GangName</i> of the gang(s) being queried.

5.15.2.2 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 5-132: GangInfo Element

Name	Data Type	Description
<i>GangName</i>	NMTOKEN	Name of the gang.

5.16 Extending Messages

This specification defines a set of predefined Messages for general usage. Extensions to existing Messages and additional Message types can be defined using the standard extension rules described in Section 3.12, JDF Extensibility. Note, the generic content of Section 3.1, Generic Contents of All Elements is also valid for JMF Elements. It is not allowed to define Message extensions which duplicate the functionality of messaging types, messaging Elements or Message Attributes that are already defined in this specification.

For example the content of the *Type* Attribute MAY be specified with a prefix that identifies the organization that defined the extension. The prefix and name SHOULD be separated by a single colon (':'). Any additional

Attributes and Elements are allowed, and internal Elements MAY be declared with explicit namespaces. The official namespace of JMF Elements is `xmlns="http://www.CIP4.org/JDFSchema_1_1"`. This namespace is identical to that defined for JDF in Section 3.12, JDF Extensibility. An example is provided:

Example 5-43: Custom Query

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Circus"
  Timestamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:Circus="Circus Schema URI">
  <Query ID="Q1" Type="Circus:IsClownHappy" xsi:type="QueryIsClownHappy">
    <Circus:ClownParams Gender="male"/>
  </Query>
</JMF>
```

Example 5-44: Custom Response

The Response Message will also have the "Circus:" namespace identifier. All Circus Elements are explicitly declared.

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Circus 2"
  Timestamp="2005-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:Circus="Circus Schema URI">
  <Response ID="M1" Type="Circus:IsClownHappy" xsi:type="ResponseIsClownHappy"
    refID="Q1">
    <Circus:Clown happy="true" name="Joe"/>
    <Circus:Clown happy="false" name="John"/>
  </Response>
</JMF>
```

5.16.1 IfraTrack Support

The extending mechanism can be used to implement compatibility with other XML-based messaging standards, for example version 3.0 of IfraTrack. The *Type* Attribute is set to the appropriate namespace, and the foreign Message is included, as demonstrated in the following example:

Note that the application is free to select the appropriate response types in order to fulfill its local (IfraTrack) protocol requirements if it uses its own namespace. In the examples below, the default namespace associated with the JMF Query Message and Response Elements has been overwritten by the Ifra namespace.

Example 5-45: Custom Query for IfraTrack

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="IFRA"
  Timestamp="2003-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:IFRA="IfraTrack URI">
  <Query ID="Q1" Type="IFRA:IMF" xsi:type="QueryIMF">
    <imf:IMF xmlns:imf="IfraTrack URI">
      Whatever you want (might be multiple top level Elements)
    </imf:IMF>
  </Query>
</JMF>
```



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)

Example 5-46: Custom Response for IfraTrack

The legal Response Message would be:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="IFRA"
  Timestamp="2003-07-25T12:32:48+02:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:IFRA="IfraTrack URI">
  <Response ID="M1" Type="IFRA:IMF" xsi:type="ResponseIMF" refID="Q1">
    <imf:IMF xmlns:imf="IfraTrack URI">
      The appropriate IFRA response(s)
    </imf:IMF>
  </Response>
</JMF>
```

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 "someValue" 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 <i>Approval</i> 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.
ImplementationResource *	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.

Table 6-1: Template for Input Resources (Section 2 of 2)

Name	Description
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.
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 <i>Preflight</i> Process, See “Preflight” on page 300.
Preview * New in JDF 1.1A Deprecated in JDF 1.4	Any number of previews MAY be associated with a Process and used for display purposes. Preview/@PreviewUsage SHOULD be <i>Thumbnail</i> or <i>Viewable</i> . Deprecation note: starting with JDF 1.4, a Preview MAY be a member of any Element. See Table 3-1, “Any Element (generic content),” on page 40.
Tool * New in JDF 1.4	Miscellaneous reusable tool required for a Process.
UsageCounter * New in JDF 1.3	Devices MAY use counters, called “usage counters”, to track equipment utilization or work performed, such as impressions produced or documents generated.

Table 6-2: Template for Output Resources

Name	Description
<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 <i>Approval</i> Process is a ByteMap , it is assumed that it will be displayed on a viewing Device

Table 6-4: Approval – Output Resources

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 (<i>Accepted</i>) *	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).
Resource (<i>Rejected</i>) *	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/@ <i>Status</i> 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., **RunListResources** 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 <i>ManualLabor</i> Process.

Table 6-12: ManualLabor – Output Resources

Name	Description
Resource * Modified in JDF 1.4	The Resources that were created by manual work. In general these will be Component Resources, but Handling Resources MAY also be processed manually. If no Output Resource is specified, the <i>ManualLabor</i> Process describes incidental work. Modification note: starting with JDF 1.4, multiple Resources are allowed.

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 1012 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 <i>QualityControl</i> Process.

Table 6-16: QualityControl – Output Resources

Name	Description
QualityControlResult	Results of the Process, (e.g., measurement statistics).
Resource	The Resource after <i>QualityControl</i> 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 <i>Verification</i> . It has been added here to allow modeling of <i>Verification</i> 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.

Table 6-23: Product Intent – Input Resources (Section 2 of 2)

Name	Description
FoldingIntent ?	This Resource specifies the fold intent for a JDF Job using information that identifies the number of folds, the height and width of the folds, and the folding catalog number.
HoleMakingIntent ?	This Resource specifies the holemaking intent for a JDF Job.
InsertingIntent ?	This Resource specifies the placing or inserting of one component within another, using information that identifies page location, position and attachment method.
LaminatingIntent ?	This Resource specifies the laminating intent for a JDF Job using information that identifies whether or not the product is laminated.
LayoutIntent ?	This Resource records the size of the finished pages for the product component.
MediaIntent ?	This Resource describes the media to be used for the product component.
NumberingIntent ?	This Resource describes the parameters of stamping or applying variable marks in order to produce unique components, for items such as lottery notes or currency.
PackingIntent ?	This Resource specifies the packaging intent for a JDF Job, using information that identifies the type of package, the wrapping used and the shape of the package.
ProductionIntent ?	This Resource specifies the manufacturing intent and considerations for a JDF Job using information that identifies the desired result or specified manufacturing path.
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 Process consisting of *ImageSetting* and *Bending* Processes.

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 color space might be.

The *ColorCorrection* Process MAY be part of a Combined Process with the *ColorSpaceConversion* Process, in which case the source and destination profiles used by the *ColorSpaceConversion* Process would be supplied from **ColorSpaceConversionParams**. Either the direct *Adjustment* Attribute or the ICC profile Attribute **ColorCorrectionOp/FileSpec** with *ResourceUsage* = "AbstractProfile" can be used in this scenario to apply color corrections in the Device independent ICC Profile Connection Space interpreted from the ICC source profile before the ICC destination profile is applied.

Alternatively, a *ColorCorrection* Process MAY occur after a *ColorSpaceConversion* Process. In this scenario only the **ColorCorrectionOp/FileSpec** with *ResourceUsage* = "DeviceLinkProfile" supplied in **ColorCorrectionOp** is used.

Table 6-29: ColorCorrection – Input Resources (Section 1 of 2)

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 <i>ColorCorrection</i> Process

Table 6-29: ColorCorrection – Input Resources (Section 2 of 2)

Name	Description
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 color space. There are two ways in which a Controller can use this Process to accomplish the color conversion. It can simply order the colors to be converted by the Device assigned to the task, or it can request that the Process simply tag the input data for eventual conversion. Additionally, the Process can remove all tags from the 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 color spaces will be converted in the file.
RunList	List of pages, Sheets or byte maps on which to perform the selected operation.

Table 6-32: ColorSpaceConversion – Output Resources

Name	Description
ColorantControl ?	Identifies the assumed color model for the Job. The ColorantControl Resource can be modified by a <i>ColorSpaceConversion</i> 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.

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 DeviceLink profiles are CMYK to CMYK print process conversions or RGB to CMYK color separations.

Table 6-33: ContactCopying – Input Resources

Name	Description
ContactCopyParams	The settings of the contact copying task.
DevelopingParams ?	Controls the physical and chemical specifics of the media development process.
ExposedMedia +	The film or films to be copied onto the film or plate.
Media	The unexposed film or plate.
TransferCurvePool ?	Area coverage correction and coordinate transformations of the device.

Table 6-34: 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.

Table 6-35: 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 6-36: 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 6-37: CylinderLayoutPreparation – Input Resources

Name	Description
CylinderLayoutPreparationParams ?	Set of parameters for <i>CylinderLayoutPreparation</i> .
Layout	Definition of the Layout of the individual plates. The resulting CylinderLayout references plate layouts.
RunList	The document RunList .

Table 6-38: 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 6-39: 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 6-40: 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 6-41: 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 6-42: 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 DieDesign

[New in JDF 1.4](#)

This Process describes the design of a die tool set starting from a **DieLayout**.

Table 6-43: DieDesign – Input Resources

Name	Description
DieLayout	A Resource describing the die cutter layout

Table 6-44: DieDesign – Output Resources

Name	Description
DieLayout +	A set of Resources describing the die cutter tool set.

6.4.11 DieLayoutProduction

[New in JDF 1.4](#)

This process describes the layout of one or more structural designs for a given **Media**. The output of this process is a **DieLayout** Resource, describing a tool set for the die cutter machine. The *DieLayoutProduction* Process can be performed by a human operator using a CAD application. In some cases it can be an automated Process. The Process can be run in estimation mode in which case multiple solutions are returned that can then be used as input of a cost estimation module to determine the optimal layout.

Table 6-45: DieLayoutProduction – Input Resources

Name	Description
ShapeDef +	ShapeDef Resources describing the different 1-up structural designs to be stepped and repeated on the Media .
DieLayoutProductionParams	The parameters for the <i>DieLayoutProduction</i> .

Table 6-46: DieLayoutProduction – Output Resources

Name	Description
DieLayout +	A resource describing the die cutter tool set. When the process is run in estimation mode, multiple alternative DieLayout Elements are returned, otherwise a single DieLayout is generated.

Example 6-1: DieLayoutProduction: Single Shape and Two Sheet Sizes

Example of *DieLayoutProduction* of a single shape on 2 stock sheet sizes

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- DieLayoutProduction Sample
      Date:Sept 2007 Version: 1.00
      Single Shape is repeated on a range of alternative sheet sizes.
-->
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n001"
      Type="DieLayoutProduction" Status="Waiting" JobPartID="ID234"
      DescriptiveName="Single shape versus a set of sheet sizes"
      Version="1.4">
  <ResourcePool>
    <ShapeDef Class="Parameter" ID="ShapelUp" Status="Available">
      <FileSpec URL="file://myserver/myshare/olive.dd3"/>
    </ShapeDef>
    <!-- Layout can chose from 2 stock sheet sizes. Nesting with 2nd row
          rotated and secondary gutters. Rotate against grain/flute
          is not allowed.
    -->
    <DieLayoutProductionParams Class="Parameter" ID="LayParam"
      Status="Available">
      <ConvertingConfig SheetWidth="2834.64 ~ 2834.64"
        SheetHeight="2267.72 ~ 2267.72"/>
      <ConvertingConfig SheetWidth="3401.57 ~ 3401.57"
```

```

        SheetHeight="2834.64 ~ 2834.64"/>
        <RepeatDesc GutterY="0.0" GutterY2="14.17" AllowRotate="false"
            LayoutStyle="Reverse2ndRow"/>
    </DieLayoutProductionParams>
    <!-- The layout with minimum waste will be returned as the final result. -->
    <DieLayout Class="Parameter" ID="DieLay" Status="Unavailable"/>
</ResourcePool>
<ResourceLinkPool>
    <ShapeDefLink rRef="ShapelUp" Usage="Input"/>
    <DieLayoutProductionParamsLink rRef="LayParam" Usage="Input"/>
    <DieLayoutLink rRef="DieLay" Usage="Output"/>
</ResourceLinkPool>
</JDF>

```

Example 6-2: DieLayoutProduction: Single Shape and Range of Sheet Sizes

Example of *DieLayoutProduction* of a single shape on a range of sheet sizes. The sheet sizes have defined minimum and maximum width and height. The layout is optimized for a particular order quantity

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- DieLayoutProduction Sample
    Date:Sept 2007 Version: 1.00
    Single Shape is repeated on a continuous range of sheet sizes. -->
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n001"
    Type="DieLayoutProduction" Status="Waiting"
    DescriptiveName="Single shape versus a set of sheet sizes"
    JobPartID="ID400" Version="1.4">
    <ResourcePool>
        <ShapeDef Class="Parameter" ID="ShapelUp" Status="Available">
            <FileSpec URL="file://myserver/myshare/olive.dd3"/>
        </ShapeDef>
        <!-- Layout can choose sheet sizes between 1200mm-1000mm wide and
            1000mm-800mm high. The layout will be optimized for order quantities
            of 1 million boxes. Gutters are 5mm and cross flute/grain rotation
            is not allowed.
        -->
        <DieLayoutProductionParams Class="Parameter" ID="LayParam"
            Status="Available">
            <ConvertingConfig SheetWidth="3401.57 ~ 2834.64"
                SheetHeight="2834.64 ~ 2267.72"/>
            <RepeatDesc OrderQuantity="1000000" GutterX="14.17" GutterY="14.17"
                AllowRotate="false"/>
        </DieLayoutProductionParams>
        <!-- The layout with minimum waste will be returned as the
            final result. -->
        <DieLayout Class="Parameter" ID="DieLay" Status="Unavailable"/>
    </ResourcePool>
    <ResourceLinkPool>
        <ShapeDefLink rRef="ShapelUp" Usage="Input"/>
        <DieLayoutProductionParamsLink rRef="LayParam" Usage="Input"/>
        <DieLayoutLink rRef="DieLay" Usage="Output"/>
    </ResourceLinkPool>
</JDF>

```

Example 6-3: DieLayoutProduction: Two Shapes and Range of Sheet Sizes

Example of *DieLayoutProduction* of 2 shapes on a range of sheet sizes. The sheet sizes have defined minimum and maximum width and height. The layout is optimized for a particular order quantity of 2 boxes.

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- DieLayoutProduction Sample
    Date:Sept 2007 Version: 1.00
    2 Shapes is repeated on a continuous range of sheet sizes.
-->

```

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n001"
  Type="DieLayoutProduction"
  Status="Waiting"
  DescriptiveName="Single shape versus a set of sheet sizes"
  Version="1.4">
  <ResourcePool>
    <ShapeDef Class="Parameter" ID="ShapelUp" Status="Available">
      <FileSpec URL="file://myserver/myshare/beef.dd3"/>
    </ShapeDef>
    <ShapeDef Class="Parameter" ID="ShapelUp2" Status="Available">
      <FileSpec URL="file://myserver/myshare/chicken.dd3"/>
    </ShapeDef>
    <!-- Layout can chose sheetsizes between 1200mm-1000mm wide and
      1000mm-800mm high. Layout is optimized for an order
      quantity of 300k boxes for beef and 700k boxes for chicken.
      Gutters are 5mm and cross flute/grain rotation is not allowed.
    -->
    <DieLayoutProductionParams Class="Parameter" ID="LayParam"
      Status="Available">
      <ConvertingConfig SheetWidth="3401.57 ~ 2834.64"
        SheetHeight="2834.64 ~ 2267.72"/>
      <RepeatDesc OrderQuantity="300000" GutterX="14.17" GutterY="14.17"
        AllowRotate="false"/>
      <RepeatDesc OrderQuantity="700000" GutterX="14.17" GutterY="14.17"
        AllowRotate="false"/>
    </DieLayoutProductionParams>
    <!-- The layout with minimum waste will be returned as the final
      result. -->
    <DieLayout Class="Parameter" ID="DieLay" Status="Unavailable"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ShapeDefLink rRef="ShapelUp" Usage="Input"/>
    <ShapeDefLink rRef="ShapelUp2" Usage="Input"/>
    <DieLayoutProductionParamsLink rRef="LayParam" Usage="Input"/>
    <DieLayoutLink rRef="DieLay" Usage="Output"/>
  </ResourceLinkPool>
</JDF>

```

6.4.12 DigitalDelivery

[New in JDF 1.2](#)

This Process specifies the delivery of digital assets in any stage of the flow. It could be images, documents, layout, text files, ready to print raster files or any other file type. When `ArtDeliveryIntent/ArtDelivery/@ArtDeliveryType` is `"DigitalNetwork"` or `"DigitalFile"`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 953.

Table 6-47: DigitalDelivery – Input Resources (Section 1 of 2)

Name	Description
DigitalDeliveryParams	Parameter specifying the artwork files delivery characteristics.

1. When `ArtDeliveryIntent/ArtDelivery/@ArtDeliveryType="DigitalFile"`, the Process MAY also be `Delivery`, in case the file is delivered on digital media.

Table 6-47: DigitalDelivery – Input Resources (Section 2 of 2)

Name	Description
RunList * Modified in JDF 1.3	The list of digital files to be delivered.

Table 6-48: DigitalDelivery – Output Resources

Name	Description
RunList + Modified in JDF 1.3	The list of digital files which were actually delivered to the destination.

6.4.13 FilmToPlateCopying

[Deprecated in JDF 1.1](#)

FilmToPlateCopying has been replaced by the more generic *ContactCopying*. See "FilmToPlateCopying" on page 1012 for details of this deprecated Process.

6.4.14 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 6-49: 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 6-50: 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.

6.4.15 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 6-51: ImageReplacement – Input Resources (Section 1 of 2) (Section 1 of 2)

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.

Table 6-51: ImageReplacement – Input Resources (Section 2 of 2) (Section 2 of 2)

Name	Description
RunList	List of page contents on which to perform the selected operation.

Table 6-52: ImageReplacement – Output Resources

Name	Description
RunList	List of page contents with images that have been manipulated as indicated by the <code>ImageReplacementParams</code> Resource.

6.4.16 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 270.)

Table 6-53: ImageSetting – Input Resources

Name	Description
<code>ColorantControl</code> ? New in JDF 1.2	The <code>ColorantControl</code> Resources that define the ordering and usage of inks during marking on the imagesetter.
<code>DevelopingParams</code> ? New in JDF 1.1	Controls the physical and chemical specifics of the media development process.
<code>ExposedMedia</code> ? New in JDF 1.3	When imaging to reusable media, <code>ExposedMedia</code> MAY also be used as input to <i>ImageSetting</i> . Constraint: exactly one of <code>Media</code> or <code>ExposedMedia</code> MUST be specified.
<code>ImageSetterParams</code> ? Modified in JDF 1.1	Controls the Device specific features of the imagesetter.
<code>Media</code> ?	The unexposed media. Constraint: exactly one of <code>Media</code> or <code>ExposedMedia</code> MUST be specified.
RunList	Identifies the set of bitmaps to image. MAY contain bytemaps or images.
<code>TransferCurvePool</code> ? New in JDF 1.1	Area coverage correction and coordinate transformations of the Device.

Table 6-54: ImageSetting – Output Resources

Name	Description
<code>ExposedMedia</code>	The exposed media Resource.

6.4.17 Imposition

[Modified in JDF 1.4](#)

Modification note: starting with JDF 1.4, automated imposition is added.

The *Imposition* Process is responsible for combining pages of input graphical content onto surfaces whose dimensions are reflective of the physical output media. Static or dynamic printer's marks can be added to the surface in order to facilitate various aspects of the production process. Among other things, these marks are used for press alignment, color calibration, job identification and as guides for cutting and folding.

Note that the *Imposition* Process specifies the task of combining pages and marks on sheets. The task of setting up the parameters needed for *Imposition* (e.g., creating the `Layout` Resource) is defined either by *LayoutPreparation*, *Stripping* or by the generic *ResourceDefinition* Process.

Table 6-55: 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 (<i>Document</i>)	Structured list of incoming page contents which is transformed to produce the imposed surface images.
RunList (<i>Marks</i>) ?	Structured list of incoming marks. These are typically printer's marks such as fold marks, cut marks, punch marks or color bars.

Table 6-56: Imposition – Output Resources

Name	Description
RunList	Structured list of imposed surfaces. The <i>ElementType</i> of the LayoutElement Resource MUST be <i>Surface</i> . Typically the output RunList will be Partitioned by <i>PartIDKeys = "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 .

There are two mechanisms provided for controlling the flow of page images onto sheet surfaces:

The default mechanism is for non-automated (e.g. fully-specified) **Imposition**, which originally derived from **Layout** in PJTF. Fully-specified imposition explicitly identifies all page content for each sheet imaged and references these pages by means of the order in which they are defined in the input **RunList** (*Document*) Resource. Static printer's marks are referenced in a similar fashion from the input **RunList** (*Marks*) Resource.

Setting the *Automated* attribute of the **Layout** Resource to "*true*" activates a template approach to imposition and relies upon the full hierarchy structure of the document (as specified by the **RunList** (*Document*) and referenced Structured PDL data) to specify the page content to be imposed.

In JDF, there is a single **Layout** Resource definition. Its structure is broad enough to encompass the needs of both fully specified and template-driven imposition. When described fully (*Automated = "false"*), the **Layout** Resource Partition structure defines the imposition to take place. The highest level of each Partition defines a signature. The children of each of the signatures in turn specifies an array of sheets, and each sheet MAY have up to two surfaces (Front and/or Back), on which the page images and any printer's marks are to be placed using **PlacedObject** Elements. A sheet that specifies no surface content MUST be interpreted as blank. Pages that are to be printed MUST be placed onto surfaces using **ContentObject** Subelements which explicitly identify the page (Typically done using the **ContentObject/@Ord** Attribute which specifies an index into the document **RunList**). Thus, the **Layout** Partition hierarchy MUST explicitly specify which pages are to be imaged onto each surface.

For JDF 1.3, automated imposition was originally defined such that **Layout** Resource Partitions specified a single signature of sheet(s) upon which page content was to be imposed. The sequence of pages to be imaged via automated imposition was defined by the Document **RunList**. The pages were pulled from this sequence as needed in order to satisfy the **ContentObject** Elements defined for each sheet surface in the signature of the **Layout** Resource. The signature was repeated as necessary until all pages available in the Document **RunList** had been used.

Note that the XML order in which the Partitions of the **Layout** Resource are defined is significant for both automated and non-automated imposition and defines the order in which the imposition engine processes the **RunList** (*Document*).

6.4.17.1 Glossary for Automated Imposition

This table below introduces terms and concepts necessary for understanding automated imposition processing.

Table 6-57: Glossary for Automated Imposition (Section 1 of 3)

Term	Definition
Base Index	When processing an Imposition Template, the imposition engine maintains an internal Base Index into the Page Pool being processed. That Base Index is added to the ContentObject/@Ord value, resulting in an index into the Page Pool for referencing the page to be placed, and is updated for each Imposition Template iteration. Both positive and negative base indices are maintained for use when ContentObject/@Ord has either a negative or positive value. For an example, see Example N-31, "Algorithm for Processing an Imposition Template" on page 980.
Base Ord	Same as Base Index.
Imposed Sheet Set	Describes a single set of sheet definitions generated by the imposition engine containing imposed content. Note that this may represent a precut set of sheets in a cut-and-stack workflow (where the maximum number of sheets in the Imposed Sheet Set is defined by Layout/LogicalStackParams/@MaxStackDepth), or a collect when no Logical Stacks are defined.
Imposition Template	A first-level branch of a Partitioned Layout Resource having Automated="true" that describes a single set of sheets with a common imposition layout that accommodates very specific production characteristics. A single Layout Resource defines a collection of one or more Imposition Templates.
Instance Document	The imposition engine treats each immediate child Node of a set in a Structured PDL as an Instance Document. This is used as the basis for generating EndOfDocument breaks in the resulting RunList (Surfaces), and for processing RunList/@DocCopies Attributes (see Section 7.2.160, "RunList" on page 710). If a set has only pages as its children, then a single Instance Document is assumed to exist.
Logical Sheet	One or more pages placed onto a sheet definition within a Logical Stack (i.e a sheet definition within a Logical Stack).
Logical Stack	When Layout/LogicalStackParams/@MaxStackDepth is specified in the root of the Layout Resource, then the imposition engine is configured for imposition onto multiple Logical Stacks. These stacks are described through the use of adding Layout/PlacedObject/@LogicalStackOrd to stack-specific descriptions for each placed object. For more information, see Section 6.4.17.4.1, "Using Logical Stacks" on page 292. For example usage, see Example N-34, "Booklet Using Automated Imposition" on page 984.
Logical Stack Set	The set of Logical Stacks described by an Imposed Sheet Set.

Table 6-57: Glossary for Automated Imposition (Section 2 of 3)

Term	Definition
Page Pool	<p>A Page Pool refers to a delimited sequence of pages defined within the RunList (<i>Document</i>) input to the Imposition Process. A Page Pool MAY encompass all pages of the RunList (<i>Document</i>) as in the case of Unstructured PDLs. In the case of Structured PDLs, a Page Pool is defined to be that set of pages represented by a leaf node of the document structure. For example, a brochure which has a sub-structure of Cover and Body has two leaf nodes, Cover and Body, respectively. If Body were further divided into Chapter sections, then the leaf nodes of the Brochure would be the Cover and each Body Chapter. LayoutElement/@ElementType may be used to demote an already Structured PDL to be treated as an Unstructured PDL. Examples of Structured PDL formats include PPML, PPML/VDX, and ISO 16612-2 PDF/VT.</p> <p>Imposition Templates select Page Pools to be processed based on their Partition Keys whose values are derived from metadata present in the PDL data (e.g. Layout Partitioned by <i>DocTags</i> = "Letter" would process all Page Pools of the current Set whose metadata derived Partition Key <i>DocTags</i> matches "Letter"). See below for more detail.</p> <p>It is important to note that the pages in a Page Pool MUST be presented to the imposition engine in a well defined order known to the Layout Resource creator (typically reader order) in order for them to be processed correctly.</p>
Page Pool List	<p>A Page Pool List refers to a sequence of one or more Page Pools (contiguous or disjoint in the RunList (<i>Document</i>)) aggregated together and treated as a single Page Pool for processing by a selected Imposition Template. For example, if a Page Pool List is constructed from the Page Pools: Chapter1, Chapter2, and Chapter4 as defined in an input RunList (<i>Document</i>), then the aggregate result is a single pool of pages consisting of the pages from Chapter1, Chapter2 and Chapter4. The order of the pages of the Page Pool List MUST be processed in the order in which the Page Pools are defined in the RunList (<i>Document</i>). The boundaries between Page Pools in a Page Pool List are implicitly maintained for use by the imposition processor for making page level sheet surface mapping decisions during processing (e.g. specifying a right side facing pages start at the beginning of each chapter). Page Pools are aggregated into Page Pool Lists through the use of the Layout/@BaseOrdReset Attribute. If <i>BaseOrdReset</i> = "PagePoolList" then all Page Pools processed by the Imposition Template are aggregated. If <i>BaseOrdReset</i> = "PagePool1", then each Page Pool is processed separately.</p> <p>It is important to note that the pages in a Page Pool List MUST be presented to the imposition engine in a well defined order known to the Layout Resource creator (typically reader order) in order for them to be processed correctly.</p>
PDL Processor	<p>A PDL interface that hides details of a particular PDL and syntax, etc. from the imposition engine itself. Its role is to present the structure of the PDL and pools of pages within the PDL structure to the imposition engine in a PDL independent way.</p>
Recipient Set	<p>Set of finished pages produced for a single recipient.</p>
Sheet Definition	<p>A branch of an Imposition Template that describes the imposition to be performed for a sheet. Sheet Definitions for automated imposition MUST be partitioned by <i>SheetName</i> and <i>Side</i>.</p>

Table 6-57: Glossary for Automated Imposition (Section 3 of 3)

Term	Definition
Structured PDL	<p>A Structured PDL defines sequences of groupings of pages. These groupings may be as simple as specifying the set of pages belonging to a chapter or cover of a booklet where such a group is a Page Pool. In the case of Variable Document Printing (VDP) Structured PDLs, there are often multiple sets of content where typically a set instance comprises the content to be delivered to a single recipient. Each set has one or more documents, and documents may be further subdivided into subdocuments in hierarchical fashion. The imposition engine processes each set individually in the sequence specified in the interpretation specified by the RunList that references the Structured PDL data file.</p> <p>It is the responsibility of the specification of the Structured PDL data and its PDL Processor to identify the Set-level (e.g. recipient record level) structural context to the imposition engine. A Structured PDL format suitable for VDP printing using JDF must supply its own definition for document structure either in its data or in a specification for the PDL.</p>
Unstructured PDL	<p>An Unstructured PDL is a content file consisting of a single set of one or more pages. Typically such a PDL file is considered to be a single document and a single Layout Imposition Template would be applied to the entire set of pages. When a JDF imposes structure on such a file either using direct <i>@Page</i> indices or a Partitioned RunList pointing to different page ranges of the file using <i>EndOfSet</i>, <i>EndOfDocument</i> Attributes, then the imposition engine will treat the input RunList Resource as a Structured PDL.</p>
PDL Metadata	<p>Various PDL formats provide for the definition of key/value pairs within the PDL that MAY be treated as metadata for the purpose of Process parameterization. For example, the metadata key/value pairs specified in the PDL data may identify the type of finished document using <i>DocumentType = "PostCard"</i> or <i>"Booklet"</i>, which would then affect the selection of which Imposition Template is to be applied.</p> <p>The Imposition Process makes use of metadata to make decisions as to which Page Pools should be processed through an Imposition Template. These decisions are performed by comparing the explicit Partition Key settings for each Imposition Template to the Partition Key/value settings mapped from the PDL for each Page Pool in the current set, and each matching Page Pool is processed through the corresponding Imposition Template(s).</p> <p>Within an Imposition Template, metadata associated with individual pages MAY also be used to parameterize dynamic mark and slug-line content generation (see example below). Refer to the RunList/MetadataMap Element definition for information on how to specify the mapping of PDL specified metadata values for use by JDF (e.g. using Partition Keys or GeneralID keys).</p> <p>The PDL Processor MUST make use of the RunList/MetadataMap to generate Partition Keys, GeneralID and other values during the course of imposition processing. These values must be regenerated as necessary, as the metadata key/value pairs in the PDL change based on which portion of the PDL is being processed.</p>
Document Major Processing Order	<p>Document Major Processing Order refers to the scenario wherein all instances of a given document class (across all sets to be processed) must be produced before starting processing for the next document class.</p> <p>For instance, the production requirements may state that all brochures must be produced for each set, followed by all cover letters and then all postcards. This processing order is an example of Document Major.</p>
Set Major Processing Order	<p>Set Major Processing Order refers to the scenario when all documents of a set instance are produced before starting on the next set instance; this is the typical processing order for most VDP applications.</p>

6.4.17.2 Variables for Automated Imposition

The imposition engine maintains a set of locally scoped variables that may be referenced during imposition processing. The values of these variables reflect the current context of processing during execution of the Imposition process. These variables include those described in Section I, "Generating strings with Format and Template" on page 909, as well as those described in bulleted items below. All variables below are integer variables.

- *CollectIndex* represents a zero based index of the current collect of sheets being generated within an automated Imposition Template. May be greater than zero if **Layout/@MaxCollect** is specified and is greater than 1.
- *CollectSheetIndex* is a zero-based index of the current sheet of the current collect being generated within an automated Imposition Template.
- *ImposedSheetSetIndex* is the 0-based Imposed Sheet Set index.
- *PoolSheetIndex* is a zero-based index of the current sheet generated from the current page pool or page pool list within an automated Imposition Template. The value of this variable is independent of the number of collects generated by the same automated Imposition Template.
- *SheetCount* is the current number of sheets generated during the processing of the automated **Layout** Resource. At the beginning of processing of the **Layout** Resource, the value of this variable is set to zero. The value of this variable may be reset to zero in later **Layout** Partitions using the **Layout/@SheetCountReset** Attribute.
- *SubDocIndex_n* where *n* represents any hierarchical structure levels below the level of the current document present in the Structured PDL data to be processed. For example, *SubDocIndex0* might represent a collection of chapters in a brochure where its containing parent is at the document level (*DocIndex* is used to indicate the position (index) of the document in its containing Set).
- *TotalCollects* is the total number of collects generated by an automated Imposition Template from the current page pool or page pool list being processed.
- *TotalImposedSheetSets* is the total number of Imposed Sheet Sets defined for the job.
- *TotalSets* is the total number of recipient sets generated for the Job. Note that in cases where it is used before the end of content imposition, it is necessary for the imposition processor to count the number of sets in the PDL content.
- *TotalSheetCount* is the total number of sheets generated during the processing of the automated **Layout** Resource. The value of this variable may be recalculated in later **Layout** Partitions using the **Layout/@SheetCountReset** Attribute.
- *TotalSheetsInCollect* is the total number of sheets that make up the current collect.
- *TotalSheetsInPool* is the total number of sheets generated from the current page pool or page pool list within an automated Imposition Template.

The above variables MAY be used for controlling the activation of printer's marks (See **Layout/MarkObject/MarkActivation**). For example:

Example 6-4: Automated Imposition: MarkObject

This example causes a slug line to be imaged on the bottom center of the first sheet of the set of sheets comprising a signature instance. Here are the details. For **MarkActivation/@Context**, its value of "*CollectIndex*" specifies that the value of *CollectIndex* is the index used with **MarkActivation/@Index**. For **MarkActivation/@Index**, its value of 0 specifies that the sheet receive the specified slug line if the value of *CollectIndex* is 0 (i.e. if it is first sheet of the signature instance). **Note:** if **@Index** were "*1 4 6*", then the slug line would go on the second, fifth and seventh sheets.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
```

```

<Layout Class="Parameter" ID="L1" Status="Available">
  <MarkObject Anchor="BottomCenter" CTM="1 0 0 1 0 0">
    <DeviceMark FontSize="8" Font="MySlugLineFont" />
    <!--Result: Gender=male -->
    <JobField JobFormat="Gender=%s" JobTemplate="GeneralID:Gender" />
    <RefAnchor Anchor="BottomCenter" AnchorType="Sibling" rRef="1000006" />
    <MarkActivation Context="CollectIndex" Index="0" />
  </MarkObject>
</Layout>
</ResourcePool>
<ResourceLinkPool>
  <LayoutLink Usage="Input" rRef="L1" />
</ResourceLinkPool>
</JDF>

```

6.4.17.3 Execution Model for Automated Imposition

The *Imposition* Process transforms the sequences of pages contained within a Page Pool or Page Pool List to a specific sequence of imposed sheet surfaces. The Imposition Templates and the order of the Imposition Templates defined by the **Layout** Resource explicitly define the page to sheet surface mapping transformation applied by the imposition engine.

The pseudo-code below describes the processing performed by the imposition engine at a high level:

```

For each Set in the order specified in the input RunList(Document)
  For each Imposition Template
    For each Page Pool in the Set
      If the Partition Key conditions for the Imposition Template are satisfied
        then process the Page Pool through the Imposition Template.

```

Thus, each **Layout** Resource Imposition Template is processed in the XML structure order specified. Every Page Pool belonging to the current set is then evaluated against the Partition Keys specified for that Imposition Template to determine if it is to be processed by that Imposition Template.

Since each Page Pool is evaluated for each Imposition Template, it is possible to reuse the same Page Pool with multiple Imposition Templates. For an example algorithm for processing Page Pools through an Imposition Template, see Example N-31, "Algorithm for Processing an Imposition Template" on page 980.

The **RunList** Resource output from the *Imposition* Process represents a sequence of imposed sheet surfaces where each surface may be represented either by pointing to PDL content where all the input pages are imposed onto single PDL pages, or, when used with a Combined Process may refer to the page set along with imposition instructions to the interpreter using an exchange Resource. The structure of the **Layout** Resource affects the Partition Keys conserved by its output **RunList** (and its referenced content), by conserving all Partition Keys specified in the **Layout** along with generating all of the appropriate Partition Keys, such as *SetIndex*, *DocIndex*, *SheetIndex*. The output **RunList** can be viewed conceptually as a collection of sheet surface pairings (front and back) that conserves information about which **Layout** Imposition Template and Page Pool metadata that was in scope at the time the sheets were generated.

Note: *DocIndex* is always generated even if every set contains only a single document; a set that contains only pages is treated as a set with a single document.

Example 6-5: Imposition Template: Layout

Thus, if the Imposition Template (**Layout**) in this example is applied, then the resulting **RunList** Resource conceptually conserves the following Partition Keys: *SetIndex*, *SheetIndex*, *DocTags*, *DocIndex*, *SheetName* and *Side* along with any other in-scope Partition Keys.

Note that in this example, *SetIndex* and *DocIndex* are conserved by setting *EndOfSet* and *EndOfDocument* respectively in the output **RunList** (*Surfaces*). In a **Layout** that defines Logical Stacks containing multiple documents or sets within Imposed Sheet Sets, *SetIndex* and *DocIndex* would need to be conserved by explicitly setting the value of the *SetIndex* and *DocIndex* Partition Keys. The **RunList** is expected to be partitioned by *Run*,

where each *Run* represents one or more Sheets, each having at least one surface either implied by *RunList/@SheetSides*, or explicitly Partitioned by *Side*.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Layout Class="Parameter" ID="L1" Status="Available"
      PartIDKeys="DocTags SheetName Side" Automated="true">
      <Layout DocTags="CoverLetter">
        <Layout SheetName="CoverLetterSheets">
          <Layout Side="Front">
            <ContentObject Ord="0" CTM="1 0 0 1 0 0"/>
          </Layout>
        </Layout>
      </Layout>
      <Layout DocTags="Booklet">
        <Layout SheetName="BookletSheets">
          <Layout Side="Front">
            <ContentObject Ord="0" CTM="1 0 0 1 0 0"/>
            <ContentObject Ord="-1" CTM="1 0 0 1 0 0"/>
          </Layout>
          <Layout Side="Back">
            <ContentObject Ord="1" CTM="1 0 0 1 0 0"/>
            <ContentObject Ord="-2" CTM="1 0 0 1 0 0"/>
          </Layout>
        </Layout>
      </Layout>
    </ResourcePool>
    <ResourceLinkPool>
      <LayoutLink Usage="Input" rRef="L1"/>
    </ResourceLinkPool>
  </JDF>
```

Example 6-6: Output RunList (Surfaces)

```
<?xml version="1.0" encoding="UTF-8"?>
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n_000000"
  JobID="JobID" JobPartID="n_000000" Status="Waiting" Type="Combined"
  Types="Interpreting Rendering DigitalPrinting Stitching" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Combined">
  <AuditPool>
    <Created ID="a_000001" TimeStamp="2008-10-23T11:14:03+02:00"/>
  </AuditPool>
  <!--Generated by the CIP4 Java open source JDF Library version : CIP4 JDF Writer
  Java 1.3 BLD 52-->
  <ResourcePool>
    <Component Class="Quantity" ID="r_000002" Status="Unavailable"
      ComponentType="Sheet" />
    <DigitalPrintingParams Class="Parameter" ID="r_000003" Status="Available"/>
    <InterpretingParams Class="Parameter" ID="I_000001" Status="Available"/>
    <StitchingParams Class="Parameter" ID="SP_000001" Status="Available"/>
    <Media Class="Consumable" ID="r_000004" Status="Available"/>
    <RunList Class="Parameter" ID="r_000005" PartIDKeys="Run" Status="Unavailable">
      <RunList EndOfSet="true" NPage="1" Pages="0" Run="1" SheetSides="Front">
        <LayoutElement Class="Parameter" ContentDataRefs="l_000007">
          <ContentListRef rRef="r_000006"/>
        </LayoutElement>
      </RunList>
      <RunList EndOfSet="true" NPage="4" Pages="1 ~ 4" Run="2"
        SheetSides="FrontBack">
        <LayoutElement Class="Parameter" ContentDataRefs="l_000008">
          <ContentListRef rRef="r_000006"/>
        </LayoutElement>
```

```

</RunList>
<RunList EndOfSet="true" NPage="1" Pages="5" Run="3" SheetSides="Front">
  <LayoutElement Class="Parameter" ContentDataRefs="l_000007"/>
</RunList>
<RunList EndOfSet="true" NPage="4" Pages="6 ~ 9" Run="4"
  SheetSides="FrontBack">
  <LayoutElement Class="Parameter" ContentDataRefs="l_000008">
    <ContentListRef rRef="r_000006"/>
  </LayoutElement>
</RunList>
</RunList>
<ContentList Class="Parameter" ID="r_000006" Status="Unavailable">
  <ContentData ID="l_000007">
    <ContentMetaData>
      <Part RunTags="CoverLetter" SheetName="CoverLetterSheet"/>
    </ContentMetaData>
  </ContentData>
  <ContentData ID="l_000008">
    <ContentMetaData>
      <Part RunTags="BrochureSheets" SheetName="BrochureSheet"/>
    </ContentMetaData>
  </ContentData>
</ContentList>
</ResourcePool>
<ResourceLinkPool>
  <ComponentLink CombinedProcessIndex="3" Usage="Output" rRef="r_000002"/>
  <DigitalPrintingParamsLink CombinedProcessIndex="2" Usage="Input"
    rRef="r_000003"/>
  <MediaLink CombinedProcessIndex="1 2" Usage="Input" rRef="r_000004"/>
  <RunListLink CombinedProcessIndex="0 2" Usage="Input" rRef="r_000005"/>
  <InterpretingParamsLink Usage="Input" rRef="l_000001"/>
  <StitchingParamsLink Usage="Input" rRef="SP_000001"/>
</ResourceLinkPool>
</JDF>

```

6.4.17.4 Configuration for Various Automated Impositions

6.4.17.4.1 Using Logical Stacks

An Imposed Sheet Set output by the imposition engine can describe multiple Logical Stacks. Each of these Logical Stacks is placed onto a well-defined section of the sheet definitions, and after printing will typically be cut in a post-press finishing operation, generating the representative physical stacks.

Logical Stacks are configured through the use of two mechanisms:

- **Layout/LogicalStackParams** Element specifies the control for each Logical Stack including how Logical Sheets are sequenced onto a Logical Stack, and restrictions on how Logical Sheets of Recipient Sets can span Logical Stacks and Imposed Sheet Sets.
- The abstract PlacedObject/@*LogicalStackOrd* is used to assign individual placed object definitions to a Logical Stack. Each PlacedObject defines the CTM for placing that object onto the Logical Stack. Each of the PlacedObject Elements will have the same *Ord* value across the Logical Stacks.

To define a Logical Stack, the **Layout/LogicalStackParams** Element MUST be present in the root of the **Layout** Resource. This Element configures the imposition engine to place Logical Sheets within Logical Stacks. The maximum number of sheets that can make up an Imposed Sheet Set is specified by **LogicalStackParams/@MaxStackDepth**. Stacks are identified through the use of **LogicalStackParams/Stack/@LogicalStackOrd**; the first Logical Stack is *LogicalStackOrd*="0", the 2nd is "1", etc.

All Logical Stacks defined by **Layout/LogicalStackParams** MUST be used in all Imposition Templates, with the exception of an optional sheet (see **Layout/SheetCondition** in Section 7.2.109.6, "SheetCondition" on page 599) having a *Condition* of "LogicalStackSetBegin" or "LogicalStackSetEnd" – these optional

Logical Sheets are placed into a specific Logical Stack as specified by the PlacedObject/@*LogicalStackOrd* in the optional sheet.

The imposition works by traversing each Logical Stack (in the sequence specified by LogicalStackParams/Stack/@*LogicalStackSequence*). Each Imposition Template is processed where PlacedObject Elements are evaluated for one of two cases:"

- 1 The PlacedObject has no *LogicalStackOrd*. In this case, the PlacedObject is considered to be a physical sheet-level object, and is placed once at the start of processing for a physical sheet. Note that only information relevant to a physical sheet (such as *SheetIndex*) is in scope for use in generating dynamic marks. An example of a physical sheet-level mark is a cut mark for where to cut the stacks.
- 2 The PlacedObject has a *LogicalStackOrd*. In this case, only PlacedObject Elements that have a matching *LogicalStackOrd* for the current Logical Stack being processed are placed. Note that information relevant to documents and pages (such as *CollectIndex* or *TotalSheetsInPool*) is in scope for use in generating dynamic marks.

When insufficient number of pages remain to complete all Logical Stacks in an Imposed Sheet Set, the imposition engine MUST distribute all content evenly across Logical Stacks in order to minimize the number of sheets in that Imposed Sheet Set, while still honoring any restrictions specified in **Layout**/SheetCondition, LogicalStackParams/@*Restrictions* or **Layout**/PageCondition.

6.4.17.4.1.1 Imposition for Cut and Stack

This example shows how to configure for cut and stack imposition. Cut and stack produces a sequence of Imposed Sheet Sets, where each Imposed Sheet Set is cut into separate physical stacks, then each physical stack is restacked into a larger stack. This simple example is configured for 2 Logical Stacks with a *MaxStackDepth* = "3", and is filled with 20 pages. Content on the back of the sheet is placed head-to-head with the front content.

Note: that the 2nd Imposed Sheet Set has distributed the remaining 8 pages onto 2 sheets.

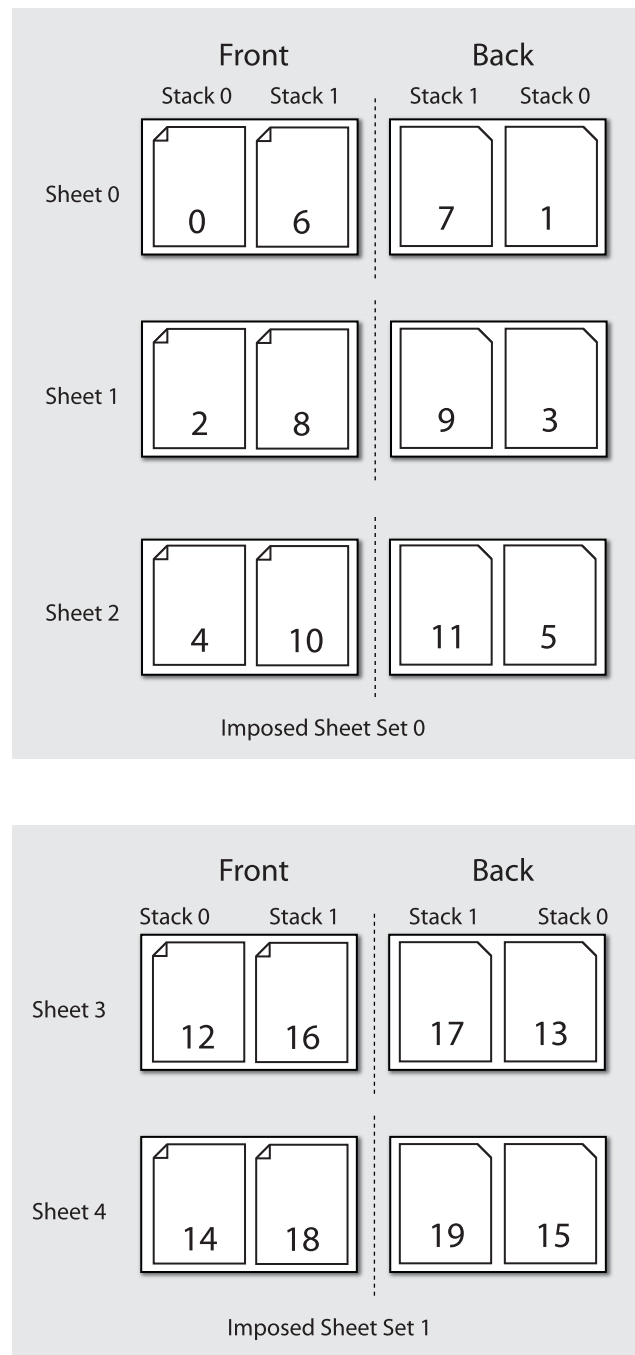


Figure 6-1: Imposition for Cut and Stack

6.4.17.4.2 Imposition for Signatures with Saddle Stitching

Saddle stitched booklets typically contain pages selected from the front of the reader ordered list of pages and pages selected from the back of the reader ordered list of pages on the same sheet. For instance the outside cover of a 16 page booklet will contain the first page ($@Ord = 0$) on the right of the sheet and the last page ($@Ord = 15$) on the left of the sheet. The pagination for the inner sheets is calculated by adding to the page number from the front and by subtracting from the back. The next page inside the cover of a booklet printed in duplex will typically contain the third page ($@Ord = 2$) on the right and the third from last page ($@Ord = 13$) on the left. This behavior is described by

specifying negative *@Ord* values for the *ContentObject* Elements that are filled with pages from the back of the *RunList* in automated imposition. The following code illustrates how absolute *@Ord* values are assigned based on sheet iterations.

Note: *Layout/@MaxCollect* specifies the maximum number of sheets per signature, e.g. in a perfect bound book. *MaxCollect* specifies the maximum number of loops prior to restarting the signature.

Example 6-7: Automated Imposition: Ord Values

```

/*
 * calculates a "real" ord value in an automated layout
 *
 * @param ord the Value of Ord in the layout
 * @param nPages the total number of pages that are consumed by the Layout, if
 *   frontOffset!=0 the pages before frontOffset are NOT counted
 * @param loop which sheet loop are we on?
 * @param maxOrdFront number of pages consumed from the front of the list
 * @param maxOrdBack positive number of pages consumed from the back of the list
 * @param frontOffset page number of the first page to be placed on ord 0 in loop 0
 * @return the pge to assign in this Ord, -1 if no page fits
 */
public static int calcOrd(int ord, int nPages, int loop, int maxOrdFront,
    int maxOrdBack, int frontOffset){
    final int maxOrd = maxOrdFront + maxOrdBack;
    if(maxOrd*loop >= nPages){
        return -1; // we are in a loop that has no remaining pages
    }
    int page;
    if(ord >= 0){ // count from front
        page = ord + loop*maxOrdFront;
    } else { // the page to put on -1
        int end = nPages + maxOrd - 1 - ((nPages +maxOrd - 1)%maxOrd);
        page = end - loop*maxOrdBack+ord;
    }
    // if a page evaluates to e.g. 10 and we only have 9 pages, ciao
    return page< nPages? page+frontOffset : -1;
}

```

6.4.17.4.3 Selecting from Multiple Imposition Templates When Processing an unstructured PDL

In this case, the imposition engine optionally selects between Imposition Templates based on the quantity of pages present in the Page Pool:

Layout/@OrdsConsumed restricts the pages of a Page Pool to which a given Imposition Template of an automated layout is applied. It is designed for use with Unstructured PDLs that only allow access to pages by index. For instance, a wraparound cover might be specified as page 0 and therefore a special cover sheet with only one *ContentObject* can be defined whereas the body sheets might contain 2 *ContentObject* Elements per surface.

OrdsConsumed is only used when you have one Page Pool and you want to restrict the number of pages to be processed for a given Imposition Template.

6.4.17.4.4 Imposition for Start of a Chapter

The *Layout/PageCondition* Element may be used to specify where on a sheet a first page of a chapter (Page Pool) starts. It does this by specifying which *ContentObject* Elements on a sheet may not be used to place the first page of a chapter. An example may be found after [Table 7-244, "PageCondition Element," on page 597](#).

6.4.17.4.5 Imposition for Regenerating Sheet Surfaces

There are two methods to configure the imposition engine for re-imposing sheet surfaces:

- 3 **Re-imposition by sheet or sheet surface:** A specific selection of sheets or surfaces imposed by the imposition engine may be selected using the controls of the RunListLink to the **RunList(Surfaces)** output from the *Imposition* Process.
- 4 **Re-imposition of sheets from content:** Alternatively the RunListLink to the **RunList(Document)** input to the imposition engine may be Partitioned to select specific content to be re-imposed.

For example, if the *Metadata0* Partition Key has been configured to represent a recipient record number in a VDP job, that Partition Keys can be used to select a specific recipient record(s) for which to re-impose sheet surfaces.

Details on how to configure ResourceLink/Part elements for sheet re-imposition including how to correctly regenerate dynamic sheet marks may be found at Section 3.10.7, "Linking to Resources" on page 114 and @ *IgnoreContext* in Table 7-340, "RunList Resource," on page 711

6.4.17.4.6 Imposition for Document-Major Processing of a VDP Structured PDL

To process a Structured PDL in Document Major Processing Order, the **RunList(Document)** input ResourceLink MUST contain Part Elements specifying the order in which documents MUST be processed. This effects a virtual reordering of the content present in the PDL. Details on how to configure ResourceLink/Part Elements for content reordering may be found at Section 3.10.7, "Linking to Resources" on page 114 and @ *IgnoreContext* in Table 7-340, "RunList Resource," on page 711.

6.4.18 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 6-58: 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 <i>ContactCopying</i> , <i>DigitalPrinting</i> and <i>ConventionalPrinting</i> Process characteristics (e.g., press, climate and substrate) under certain standardized circumstances. This function can be used to generate an accurate InkZoneProfile .

Table 6-59: InkZoneCalculation – Output Resources

Name	Description
InkZoneProfile	Contains information about ink coverage along and perpendicular to the ink zones for a specific press geometry.

6.4.19 Interpreting

The interpreting Device consumes page descriptions and instructions for controlling the marking Device, (e.g., imager, digital printers, CTP, digital printing Combined Processes, etc.). The parsing of graphical content in the page descriptions produces a canonical display list of the elements to be drawn on each page.

The interpreter 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 6-60: 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 6-61: 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 Subelements. 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> .

6.4.20 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 6-62: 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 <i>LayoutElementProduction</i> Process.

Table 6-63: LayoutElementProduction – Output Resources (Section 1 of 2)

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.

Table 6-63: LayoutElementProduction – Output Resources (Section 2 of 2)

Name	Description
RunList ?	A RunList of a LayoutElement Resource of <i>ElementType Page</i> or <i>Document</i> is produced if this <i>LayoutElementProduction</i> task is the last Process of type <i>LayoutElementProduction</i> . Exactly one of <i>LayoutElement</i> or <i>RunList</i> MUST be specified.

6.4.21 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 6-64: LayoutPreparation – Input Resources

Name	Description
LayoutPreparationParams	Set of parameters needed to control the <i>LayoutPreparation</i> Process.
RunList (<i>Document</i>) ? Modified in JDF 1.2	List of documents and/or pages that will be input into the layout. Note that this <i>RunList</i> is for information only and not modified by the <i>LayoutPreparation</i> Process.
RunList (<i>Marks</i>) ?	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 6-65: LayoutPreparation – Output Resources

Name	Description
Layout	The layout of the document to be imposed.
RunList (<i>Marks</i>) ?	List of marks that is to be used as input of the following <i>Imposition</i> Process.
TransferCurvePool ?	Definition of the transfer curves and coordinate systems of the Devices.

6.4.22 LayoutShifting

[New in JDF 1.4](#)

Apply separation dependent shifts on a flat or objects on the sheet.

The exact location of the process within the *RIPing* and *ImageSetting* and the Elements referenced by Input and Output *RunList* Elements are not defined by the spec since used within a Gray Box.

Table 6-66: LayoutShifting – Input Resources

Name	Description
LayoutShift	Parameters for the <i>LayoutShifting</i>
RunList	Points to the input objects/flats to apply shifting

Table 6-67: LayoutPreparation – Output Resources

Name	Description
RunList	Points to the adjusted Elements

6.4.23 PageAssigning

[New in JDF 1.4](#)

This Process sorts the possibly-unordered pages from one or more input **RunList** Resources into reader's order and places the result in the output **RunList**.

Table 6-68: PageAssigning – Input Resources

Name	Description
PageAssignParams ?	Container for future or proprietary extensions.
RunList +	One or more RunList Resources with potentially unsorted pages

Table 6-69: PageAssigning – Output Resources

Name	Description
RunList	RunList with pages sorted in reader's order so that it can be input to an <i>Imposition</i> Process – i.e.the sequence of pages in RunList corresponds to <i>Layout/ContentObject/@Ord</i> .

6.4.24 PDFToPSConversion

The *PDFToPSConversion* Process controls the generation of PostScript from a single PDF document. This Process MAY be used at any time in a host-based PDF workflow to exit to PostScript for use of tools that consume such data. Additionally, it MAY be used to actively control the physical printing of data to a Device that consumes PostScript data. The JDF model of this MAY include a *PDFToPSConversion* Process in a Combined Process Node with a *PDFToPSConversion* Process.

It is RECOMMENDED to replace *PDFToPSConversion* with the combination of *Interpreting* and *PDLCreation* Processes.

Table 6-70: 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 6-71: 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. <i>PDFToPSConversionParams/@GeneratePageStreams</i> 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.

6.4.25 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 6-72: PDLCreation – Input Resources (Section 1 of 2)

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.

Table 6-72: PDLCreation – Input Resources (Section 2 of 2)

Name	Description
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 6-73: 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.

6.4.26 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 6-74: 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 6-75: Preflight – Output Resources

Name	Description
PreflightReport	PreflightReport is a container for logging information that is generated by the Preflight Process.

6.4.27 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:

$$error[\%] = \frac{100}{4 * resolution[L/mm] * zone_width[mm]}$$

For zone width broader than 25 mm, a resolution of 2 lines per mm will always result in an error less than 0.5%. Therefore, a resolution of 2 lines per mm (equal to 50.8 dpi) is suggested.

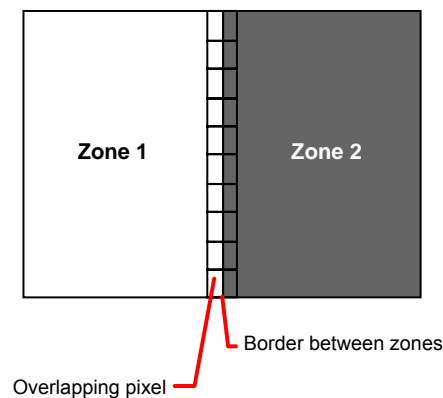


Figure 6-2: Worst case scenario for area coverage calculation

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 RIPed with 406.4 dpi monochrome.
- With anti-aliasing, the image data can be filtered down by a factor of 8 in both directions. This results in an image of 50.8 dpi with 65 color shades.
- High resolution data can also be filtered using anti-aliasing. First, the RIPed data, at 2540 dpi monochrome, are taken and filtered down by a factor of 50 in both directions. This produces an image of 50.8 dpi with 2501 color shades. Finally those shades are mapped to 256 shades, without affecting the spatial resolution.

Rasterizing a Job with 50.8 dpi and 256 shades of gray is not sufficient. The problem in this case is the rendering of thin lines (see Line Art Resolution above).

Recommendations for Implementation

The following three guidelines are strongly RECOMMENDED:

- The resolution of RIPed line art SHOULD be at least 300 dpi.
- The spatial resolution of the preview image SHOULD be approximately 20 pixel/cm (= 50.8 dpi).
- The tonal resolution of the preview image SHOULD be at least 64 shades.

Table 6-76: PreviewGeneration – Input Resources

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 <i>PreviewGeneration</i> 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 <i>PreviewGeneration</i> 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 <i>PreviewGeneration</i> Process.
PreviewGenerationParams	Parameters specifying the size and the type of the preview.
RunList ?	High resolution bitmap data are consumed by the <i>PreviewGeneration</i> 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 <i>PreviewGeneration</i> Process.
TransferCurvePool ? New in JDF 1.1	Area coverage correction and coordinate transformations of the Device.

Table 6-77: 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.

6.4.28 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 1016 for details of this deprecated Process. In JDF 1.2 and beyond, proofing is a Combined Process.

6.4.29 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 6-78: 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 6-79: PStoPDFConversion – Output Resources

Name	Description
RunList	This Resource identifies the location of the resulting PDF pages.

6.4.30 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 6-80: RasterReading – Input Resources

Name	Description
RasterReadingParams ?	Additional parameters for reading raster files.
RunList	This Resource identifies a set of raster pages or surfaces that will be inserted into the display list. This Resource MUST reference ByteMap images.

Table 6-81: RasterReading – Output Resources

Name	Description
RunList	Pipe of streamed data that represents the results of <i>RasterReading</i> 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 <i>RasterReading</i> .

6.4.31 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 6-82: Rendering – Input Resources (Section 1 of 2)

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 <i>Rendering</i> .
InterpretedPDLData ? Deprecated in JDF 1.2	Pipe of streamed data that represents the results of <i>Interpreting</i> the pages in the RunList . In JDF 1.2 and beyond, a RunList/InterpretedPDLData Subelement describes the input content data for <i>Rendering</i> .

Table 6-82: Rendering – Input Resources (Section 2 of 2)

Name	Description
RenderingParams ?	This Resource describes the format of the byte maps to be created and other specifics of the <i>Rendering</i> Process.
RunList ? New in JDF 1.2	Pipe of streamed data that represents the results of <i>Interpreting</i> or <i>RasterReading</i> the pages in the input RunList. The data is specified in <i>InterpretedPDLData</i> Subelements. The format and detail of these is implementation specific. In general, it is assumed that the <i>Interpreting</i> , <i>RasterReading</i> , <i>Rendering</i> and <i>PDLCreation</i> are tightly coupled and that there is no value in attempting to develop a general specification for the format of this data. Modification note: starting with JDF 1.4, all text replaced by text from RunList in Output Resource

Table 6-83: Rendering – Output Resources

Name	Description
RunList	Pipe of streamed data that represents the results of <i>Rendering</i> . This RunList MAY be consumed by any following Process that consumes raster data, including <i>PDLCreation</i> , <i>ImageSetting</i> or <i>DigitalPrinting</i> . The data MAY be specified in <i>ByteMap</i> sub-elements. In general, it is assumed that the <i>Interpreting</i> , <i>RasterReading</i> , <i>Rendering</i> and <i>PDLCreation</i> are tightly coupled and that there is no value in attempting to develop a general specification for the format of this data. Modification note: starting with JDF 1.4, first half of text is modified.

6.4.32 RIPing

RIPing is a Gray Box (See “Use of the Types Attribute in Process Group Nodes – Gray Boxes” on page 55.) that is a combination of at least two Processes. Most often it includes *Interpreting* and *Rendering*, but it MAY also include *ColorSpaceConversion*, *Trapping*, *Separation*, *Imposition* and *Screening*. Thus one typical *RIPing* Node is with JDF/@Type = “ProcessGroup” and JDF/@Category = “RIPing” as shown in the following example:

Example 6-8: RIPing

```
<JDF Type="ProcessGroup" Types="RIPing" Category="RIPing"
  ID="ID100" JobPartID="ID23" Status="Ready" Version="1.4" />
```

The *RIPing* Process consumes page descriptions and instructions for producing the graphical output. It parses the graphical contents in the page descriptions, renders the contents, and produces a rasterized image of the page. This raster MAY contain contone data and be represented upon output as a *ByteMap*. Alternatively, the *RIPing* Process MAY also perform halftone screening, in which case the output is in the form of a bitmap. It is also responsible for resolving all system Resource references that include font handling and Resource aliasing.

Instructions read by the RIP include information about the media, halftoning, color transformations, colorant controls and other items that affect that rasterized output. They do not, however, represent any specific controls for the physical output Device, nor do they deal with any instructions intended for the finishing Device.

In most cases, *RIPing* will be part of a Combined Process with a Process that specifies physical marking, (e.g., *DigitalPrinting* or *ImageSetting*). In this case, the interpreter SHOULD be able to act upon Device control instructions that affect the physical functioning of the printing Device such as media selection and page delivery. Media selection determines which type of medium is used for marking and where that medium can be obtained. Page delivery controls the location, orientation and quantity of physical output. The RIP is also responsible for resolving all system resource references. This includes handling font substitutions and dealing with Resource aliases. However, the RIP specifically does not get involved with any functions of the Device that could be considered finishing features such as stapling, duplexing and collating.

When a *RIPing* Process is comprised of only the *Interpreting* and *Rendering* Processes, various intermediary steps are needed before the output can be run through a *ConventionalPrinting* Process. In theory, however, a workflow could include no intermediary steps between a *RIPing* Process and a *DigitalPrinting* Process. The following workflow scenarios represent possible Process chains in each circumstance:

RIPing → *Screening* → *ImageSetting* → *ContactCopying* → *ConventionalPrinting*

RIPing → (*Screening*) → *DigitalPrinting*

Since *RIPing* is not a predefined JDF Process, see the Processes that contribute to the RIP for input and Output Resources.

6.4.33 Scanning

The *Scanning* Process creates bitmaps from analog images using a scanner.

Table 6-84: 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 6-85: Scanning – Output Resources

Name	Description
RunList	List of a ByteMap Resource or a LayoutElement Resource of <i>ElementType</i> = "Image".

6.4.34 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* Processes.

Table 6-86: 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 6-87: 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.

6.4.35 Separation

The *Separation* Process specifies the controls associated with the generation of color-separated data. It is designed to be flexible enough to allow a variety of possible methods for accomplishing this task. First of all, it sponsors host-

based PDF separating operations, in which a **RunList** of pre-separated PDF data is generated. It can also be combined with a RIP to allow control of In-RIP separations. In this scenario a **RunList** containing **ByteMap** Resources generated as the output. Yet another anticipated combination is with the Process to deal with incoming Device-dependent data. And finally, it MAY be part of a Combined Process with an **ImageReplacement** Process in order to do image substitution for omitted or proxy images.

Table 6-88: 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 6-89: Separation – Output Resources

Name	Description
RunList	List of separated pages or separated raster bytemaps.

6.4.36 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 1017 for details of this deprecated Process. In JDF 1.2 and beyond, soft proofing is a Combined Process.

6.4.37 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 6-90: Stripping – Input Resources (Section 1 of 2)

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 JobID and AssemblyIDs Attributes.
ColorantControl ? New in JDF 1.3	Contains information on the colors and separations. Useful when creating marks that need color information.
RunList (<i>Document</i>) ?	List of documents. When available, this list can be used to generate a Layout and populated RunList (no LayoutElement [@ElementType = " Reservation "]) which can be fed into a subsequent Imposition Process.

Table 6-90: Stripping – Input Resources (Section 2 of 2)

Name	Description
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 6-91: 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 <i>Imposition</i> Process.
RunList (Marks) ?	List of marks that are to be used as input of the following <i>Imposition</i> Process.

Example 6-9: Stripping: Simple Example

The first example specifies three Sheets based on folding catalog example F16-6. More examples can be found in Section N.6, Stripping.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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>
  </ResourcePool>
  <ResourceLinkPool>
    <StrippingParamsLink Usage="Input" rRef="FoldCatalogSample" />
  </ResourceLinkPool>
</JDF>
```

Example 6-10: Stripping: Complex Example

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* Work-Style.

WorkAndBack B2x4



WorkAndTurn B2x2



```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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>
  </ResourcePool>
</JDF>
```

```

</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>
</ResourcePool>
<ResourceLinkPool>
  <StrippingParamsLink Usage="Input" rRef="L1"/>
</ResourceLinkPool>
</JDF>

```

6.4.38 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 part of a Combined Process. For example, *Tiling* might be part of a Combined Process with *ImageSetting*. In that case, the input would be a **RunList** that contains **ByteMap** Resources for each surface.

Table 6-92: Tiling – Input Resources

Name	Description
RunList (<i>Surface</i>)	Structured list of imposed page contents or Byte Maps that are to be decomposed to produce the images for each tile. The <i>ElementType</i> value of the LayoutElement Resource MUST be <i>Surface</i> .
RunList (<i>Marks</i>) ?	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 6-93: Tiling – Output Resources

Name	Description
RunList	Structured list of portions of the decomposed surfaces. The value of the <i>ElementType</i> Attribute of the LayoutElement Resource MUST be <i>Tile</i> .

6.4.39 Trapping

Trapping is a prepress Process that modifies PDL files to compensate for a type of error that occurs on presses. Specifically, when more than one colorant is applied to a piece of media using more than one inking station, the media might not stay in perfect alignment when moving between inking stations. Any misalignment will result in an error called misregistration. The visual effect of this error is either that inks are erroneously layered on top of one another, or, more seriously, that gaps occur between inks that are intended to abut. In this second case, the color of the media is revealed in the gap and is frequently quite noticeable. *Trapping*, in short, is the Process of modifying PDL files so that abutting colorant edges intentionally overlap slightly, in order to reduce the risk of gaps.

The *Trapping* Process modifies a set of document pages to reduce or (ideally) eliminate visible misregistration errors in the final printed output. The Process MAY be part of a Combined Process with *RIPing* or specified as a stand-alone Process.

Table 6-94: 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 6-95: Trapping – Output Resources

Name	Description
RunList	Structured list of the modified page contents after <i>Trapping</i> 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 Sheet-Fed printing, Web Printing, Web/ribbon coating, converting and varnishing. Typically, each takes place after prepress and before postpress Processes. Direct imaging technology on press is modelled as a Combined Process of *ImageSetting* and *ConventionalPrinting*. Press machinery often includes postpress Processes (e.g., *WebInlineFinishing*, *Folding*, *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 in a Combined Process with the *WebInlineFinishing* Process. The following drawing gives an overview about Web Printing in general.

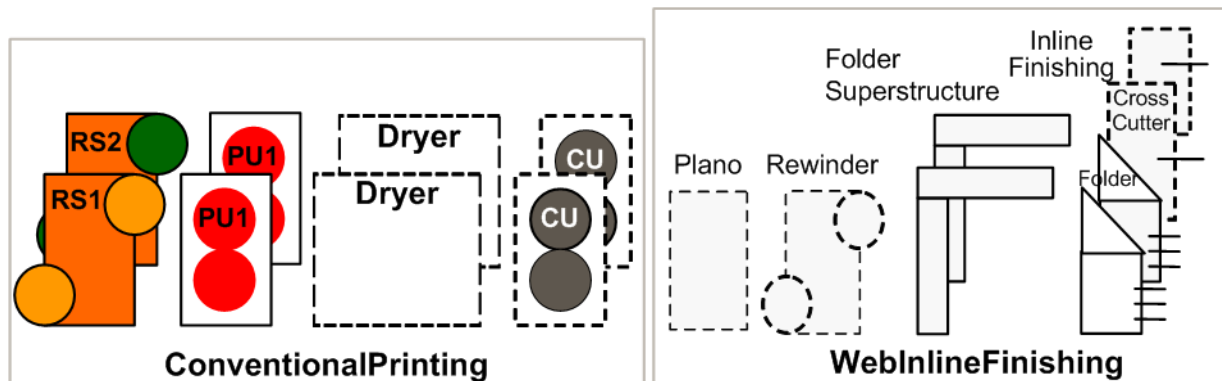


Figure 6-3: Overview of Web Printing

Table 6-96: 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 ? Modified in JDF 1.4	Various components in the form of preprints can be used in <i>ConventionalPrinting</i> in lieu of Media . Examples include waste or a set of preprinted Sheets. Modification note: starting with JDF 1.4, the input ComponentLink NEED NOT have <code>@ProcessUsage= "Input"</code> .
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 <i>ConventionalPrinting</i> . This Proof is produced by a prepress proofing Device. At most one of Component (Proof) or ExposedMedia (Proof) SHOULD be specified.

Table 6-96: ConventionalPrinting – Input Resources (Section 2 of 2)

Name	Description
ExposedMedia (<i>Sleeve</i>) ? New in JDF 1.4	Description of a sleeve.
Ink ? Modified in JDF 1.1	Information about the ink (e.g. brand, color) is useful to set up the press.
InkZoneProfile ?	The InkZoneProfile contains information about how much ink is needed along the printing cylinder of a specific printing press. It is only useful for Offset Lithography presses with ink key adjustment functions.
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 (<i>Input</i>) Resource is used.
Media (<i>MountingTape</i>) ? New in JDF 1.4	Description of a mounting tape for a sleeve.
PrintCondition ? New in JDF 1.2	Used to control the use of colorants when printing pages on a specific media. The Attributes and Elements of the PrintCondition Resource describe the aim values for a given printing Process.
Sheet ? Deprecated in JDF 1.1	Specific information about the Media (including type and color) and about the Sheet, (e.g., placement coordinates on the printing cylinder). Replaced by Layout in JDF 1.1.
TransferCurvePool ? New in JDF 1.1	Area coverage correction and coordinate transformations of the Device.

Table 6-97: 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 <i>ResourceLink</i> to this Resource indicates the number of copies of the entire Job which will be produced. Modification note: prior to JDF 1.2 this Component was marked with a <i>ProcessUsage</i> = "Good", which is OPTIONAL, but supported in JDF 1.2 and beyond.
Component (<i>Waste</i>) ? Deprecated in JDF 1.2	Produced waste of printed Sheets or ribbons. In JDF 1.2 and beyond, <i>ConventionalPrinting</i> 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-Fed and Web Presses can be used as machinery for *DigitalPrinting*.

DigitalPrinting is often used to image a small area on preprinted **Component** Resources to perform actions such as addressing or numbering another **Component**. This kind of Process can be executed by imaging with an inkjet printer during press, postpress or packaging operations. Therefore, *DigitalPrinting* is not only a press or pre-press 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 6-98: 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 * Modified in JDF 1.4	Various components can be used in <i>DigitalPrinting</i> instead of Media . Examples include preprinted covers, waste, precut Media , or a set of preprinted Sheets or webs. If multiple Component (<i>Input</i>) Resources are linked to one Process, the mapping of media to content is defined in the Partitions of DigitalPrintingParams . At least one of Component or Media MUST be specified as Input Modification note: starting with JDF 1.4, the input ComponentLink NEED NOT have <i>@ProcessUsage= "Input"</i> .
Component (<i>Proof</i>) ?	A Proof component is used if a proof was produced during an earlier print run, (see description in Section 6.5.1, <i>ConventionalPrinting</i>). 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 (<i>Proof</i>) 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 <i>DigitalPrinting</i> Process. In general, at most one of Component (<i>Proof</i>) or ExposedMedia SHOULD be specified
Ink ?	Ink or toner and information that is needed for <i>DigitalPrinting</i> .
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.

Table 6-98: DigitalPrinting – Input Resources (Section 2 of 2)

Name	Description
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 (<i>Input</i>) Resource is present. Unprinted Media used for covers are also defined as Media . At least one of Component or Media MUST be specified as Input Note: printing a Job on more than one Web or Sheet at the same time is parallel processing.
PrintCondition ?	Used to control the use of colorants when printing pages on a specific media. The Attributes and Elements of the PrintCondition Resource describe the aim values for a given printing Process.
RunList	Rendered data in Byte Maps that will be printed on the digital press are needed for <i>DigitalPrinting</i> . The RunList contains only ByteMap Elements.
Sheet ? Deprecated in JDF 1.1	Specific information about the Media (including type and color) and about the Sheet (placement coordinates on the printing cylinder). Replaced by Layout in JDF 1.1.
TransferCurvePool ? New in JDF 1.1	Area coverage correction and coordinate transformations of the Device.

Table 6-99: 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, <i>DigitalPrinting</i> 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 Varnishing

[New in JDF 1.4](#)

Varnishing is the Process of full color varnishing. Spot varnishing is described as *DigitalPrinting* or *ConventionalPrinting* with *Ink/@Family* = "Varnish".

Table 6-100: Varnishing – Input Resources

Name	Description
Component ?	The Component to be varnished. Exactly one of Component or Media MUST be specified.
ExposedMedia *	Various types of ExposedMedia MAY be specified for varnishing. See VarnishingParams/@VarnishMethod for details
Ink ?	Details of the colorant that is used for <i>Varnishing</i> . <i>Ink/@Family</i> SHOULD be "Varnish".
Media ?	The Media to be varnished. Exactly one of Component or Media MUST be specified.
VarnishingParams ?	Details of the setup of the varnishing device

Table 6-101: Varnishing – Output Resources

Name	Description
Component	The varnished Component .

6.5.4 IDPrinting

[Deprecated in JDF 1.1](#)

The IDPrinting Process was deprecated in JDF/1.1. Instead, implementations SHOULD use a Combined Process that includes the *DigitalPrinting* Process, thus improving interoperability by reducing one of the combinations of Processes. Also the *IDPrinting* Process defined a number of Resources and Subelements which are deprecated since they duplicate other Resources. See "IDPrinting" on page 1018 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 6-102: BlockPreparation – Input Resources

Name	Description
Component	The <i>BlockPreparation</i> Process consumes one Component and creates a book block.
BlockPreparationParams	Specific parameters to set up the machinery.

Table 6-103: BlockPreparation – Output Resources

Name	Description
Component	One Component is produced: the prepared book block. Its <i>ProductType</i> = " <i>BookBlock</i> "

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 6-104: BoxFolding – Input Resources

Name	Description
Component	The <i>BoxFolding</i> Process consumes one Component , the folding blank. Its <i>ProductType</i> = "BlankBox"
Component (<i>Application</i>) * Deprecated in JDF 1.4	This Process MAY consume additional Component Resources, such as windows, handles or inlets. These Component Resources MUST additionally be referenced from BoxFoldingParams/BoxApplication Elements. Deprecation note: starting with JDF 1.4, a Combined Process that includes the <i>BoxFolding</i> and <i>Inserting</i> Processes replaces BoxApplication .
BoxFoldingParams	Specific parameters to set up the folder gluer.

Table 6-105: 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 6-106: BoxPacking – Input Resources

Name	Description
BoxPackingParams	Specific parameters to set up the machinery.
Component +	The <i>BoxPacking</i> Process puts a set of Component Resources into the box Component . If more than one Component is specified, a Component/Bundle Resource MUST also be specified for each Component Modification note: starting with JDF 1.4, Component can occur more than once.
Component (<i>Box</i>) ?	Details of the box or carton.
Media (<i>Tie</i>) ? New in JDF 1.3	Protective Media can be placed between individual rows of Component Resources.
Media (<i>Underlay</i>) ? New in JDF 1.3	Protective Media can be placed between individual layers of Component Resources.

Table 6-107: 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 6-4: Bundle Creation



Figure 6-5: Bundle Transport

Table 6-108: 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 6-109: 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 6-110: CaseMaking – Input Resources

Name	Description
Component (<i>CoverMaterial</i>) ?	The cover material is either a preprinted or processed Sheet of paper. Exactly one of Media (<i>CoverMaterial</i>) or Component (<i>CoverMaterial</i>) MUST be specified.
CaseMakingParams	Specific parameters to set up the machinery.
Media (<i>CoverMaterial</i>) ?	The <i>CaseMaking</i> Process MAY also consume unprocessed Media as cover material. Exactly one of Media (<i>CoverMaterial</i>) or Component (<i>CoverMaterial</i>) MUST be specified.
Media (<i>CoverBoard</i>) Modified in JDF 1.1A	The cardboard Media used for the cover board.
Media (<i>SpineBoard</i>) ?	The cardboard Media used for the spine board. If not specified, the Media (<i>CoverBoard</i>) MUST be used for the spine board.

Table 6-111: CaseMaking – Output Resources

Name	Description
Component	One Component is produced: the produced book case. Its <i>ProductType</i> = "BookCase".

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 6-112: 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 6-113: 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 6-114: ChannelBinding – Input Resources

Name	Description
Component	The operation requires one component: the block of Sheets to be bound. If Component (<i>Cover</i>) is NOT provided and there is a cover, this Component MUST be Partitioned, and the first Partition of this Component MUST specify the Cover Modification note: starting with JDF 1.4, the input ComponentLink NEED NOT have @ <i>ProcessUsage</i> = "BookBlock".
Component (<i>Cover</i>) ?	The empty cover with the U-shaped clamp that might, for example, have been printed before it is used during the <i>ChannelBinding</i> Process.
ChannelBindingParams	Specific parameters to set up the machinery.

Table 6-115: 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 6-116: 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 6-117: 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 6-118: 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 6-119: 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 6-120: CoverApplication – Input Resources

Name	Description
CoverApplicationParams	Specific parameters to set up the machinery.
Component	The book block on which the cover is applied. If Component (<i>Cover</i>) is NOT provided, this Component MUST be Partitioned, and the first Partition of this Component MUST specify the Cover.
Component (<i>Cover</i>) ? Modified in JDF 1.4	The soft cover that is applied. Modification note: starting with JDF 1.4, this Component is optional because of the new rule about Partitioning the main Component specified above.

Table 6-121: 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 6-122: 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 <i>Creasing</i> Process.

Table 6-123: 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 6-124: 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.

Table 6-124: Cutting – Input Resources (Section 2 of 2)

Name	Description
CutBlock * Deprecated in JDF 1.1	One or more CutBlock Resources can be used to define the <i>Cutting</i> 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.
CuttingParams New in JDF 1.1	Details of the <i>Cutting</i> 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 6-125: 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 DieMaking

[New in JDF 1.4](#)

This Process describes the production of Tools for a die cutter e.g. in a die maker shop.

Table 6-126: DieMaking – Input Resources

Name	Description
DieLayout	A Resource describing the die cutter tool set

Table 6-127: DieMaking – Output Resources

Name	Description
Tool +	The set of tools for the die cutter.

6.6.15 Dividing

[Deprecated in JDF 1.1](#)

Dividing has been replaced by *Cutting*. See "Dividing" on page 1020 for details of this deprecated Process.

6.6.16 Embossing

[New in JDF 1.1](#)

The *Embossing* Process is performed after printing to stamp a raised or depressed image (artwork or typography) into the surface of paper using engraved metal embossing dies, extreme pressure and heat. Embossing styles include blind, deboss and foil-embossed.

Table 6-128: Embossing – Input Resources (Section 1 of 2)

Name	Description
Component	This Process consumes one Component which is embossed by the Process.
EmbossingParams	Parameters to setup the machinery.

Table 6-128: Embossing – Input Resources (Section 2 of 2)

Name	Description
Media * Modified in JDF 1.4	If foil stamping or foil embossing, the stamping foil materials are REQUIRED. Modification note: starting with JDF 1.4, Media can occur more than once.
Tool * Modified in JDF 1.4	The embossing stamps or calenders. Modification note: starting with JDF 1.4, Tool can occur more than once.

Table 6-129: Embossing – Output Resources

Name	Description
Component	One Component is created.

6.6.17 EndSheetGluing

EndSheetGluing finalizes the folded Sheet or book block in preparation for case binding. It requires three **Component** Resources – the back-end Sheet, the book block and the front-end Sheet – and information about how they are merged together. Back-end Sheets and front-end Sheets are in most cases Sheets folded once before *EndSheetGluing* takes place. The end Sheets serve as connections between the book block and the cover boards.

Table 6-130: EndSheetGluing – Input Resources

Name	Description
Component	A back-end Sheet and a front-end Sheet are glued onto the book block. Modification note: starting with JDF 1.4, the input ComponentLink NEED NOT have <i>@ProcessUsage= "BookBlock"</i> .
Component (<i>BackEndSheet</i>)	A back-end Sheet to be mounted on the book block.
Component (<i>FrontEndSheet</i>)	A front-end Sheet to be mounted on the book block.
EndSheetGluingParams	Specific parameters to set up the machinery.

Table 6-131: EndSheetGluing – Output Resources

Name	Description
Component	A book block is produced that includes the end Sheets.

6.6.18 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 part of a Combined Process with 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).

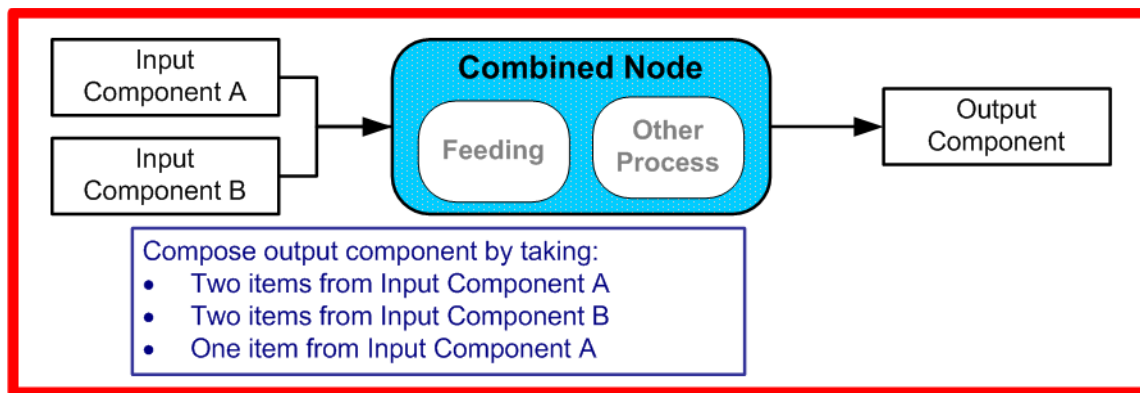


Figure 6-6: Combined Process with Feeding Process

In our example above, one input component (Component A) is a bundle component (*BundleType = Stack*) consisting of a collated set of three Sheets, the other one (Component B) is a collated set consisting of two Sheets per set. Both sets are oriented face-up (See Figure 6-7). Figure 6-8 shows the output for the case of *Gathering*.

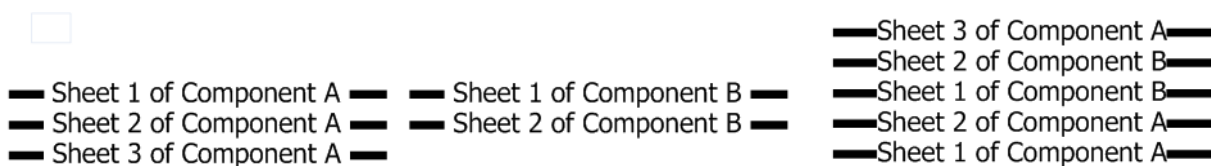


Figure 6-7: Input Components

Figure 6-8: Output Component

Note that, by default, none of the Sheets is flipped, so surfaces of Sheet 1 of **Component A** do not show in a different direction. To flip Sheets, *FeedingParams/CollatingItem/@Orientation* MAY be specified.

Table 6-132: 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 6-133: Feeding – Output Resources

Name	Description
Component *	Component (s) fed to the consuming Process.
Media *	Media fed to the consuming Process.

6.6.19 Folding

Buckle folders or knife folders are used for *Folding* Sheets. One or more Sheets can be folded at the same time. Web presses often provide in-line *Folding* equipment. Longitudinal *Folding* is often performed using a former, a plow folder or a belt. While jaw folding, chopper folding or drum folding equipment is used for folding the Sheets that have been divided.

The JDF *Folding* Process covers both operations done in stand-alone *Folding* machinery – typically found for processing printed materials from Sheet-Fed presses – and in-line equipment of Web Presses. Creasing and/or slot perforating are sometimes necessary parts of the *Folding* operation that guarantee exact Process execution. They

depend on the folder used, the **Media** and the folding layout. These operations are specified in the *Creasing* and *Perforating* Processes respectively.

Table 6-134: Folding – Input Resources

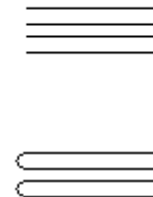
Name	Description
Component	Component Resources, including a printed Sheet or a pile of Sheets, are used in the <i>Folding</i> Process.
FoldingParams	Specific parameters to set up the machinery.

Table 6-135: Folding – Output Resources

Name	Description
Component Modified in JDF 1.1	The Process produces a Component , which in most cases is a folded Sheet.

6.6.20 Gathering

In the *Gathering* Process, ribbons, Sheets or other **Component** Resources are accumulated on a pile that will eventually be stitched or glued in some way to create an individual **Component**. The input **Component** Resources MAY be Output Resources of a Web-Printing Machine used in *Collecting* or of any Machine that executes a *ConventionalPrinting* or *DigitalPrinting* Process. In Sheet applications, a moving gathering channel is used to transport the pile. But no matter what the inception of the *Gathering* Process, the sequence of the input components dictates the produced component. The figure on the right shows typical gathered piles.

**Table 6-136: 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 <i>Gathering</i> 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 6-137: Gathering – Output Resources

Name	Description
Component	Components gathered together, (e.g., a pile of folded Sheets).

6.6.21 Gluing

[New in JDF 1.1](#)

Gluing describes arbitrary methods of applying glue to a **Component**.

Table 6-138: Gluing – Input Resources

Name	Description
Component	This Process consumes one Component : the printed Sheets.
GluingParams	Details of the <i>Gluing</i> Process.

Table 6-139: Gluing – Output Resources

Name	Description
Component	One Component is produced, the input Component with glue applied to it.

6.6.22 HeadBandApplication

[New in JDF 1.1](#)

Head bands are applied to the hard cover book block.

Table 6-140: HeadBandApplication – Input Resources

Name	Description
Component	The prepared book block.
HeadBandApplicationParams	Specific parameters to set up the machinery.

Table 6-141: HeadBandApplication – Output Resources

Name	Description
Component	One Component is produced: the hard cover block with head bands.

6.6.23 HoleMaking

A variety of Machines (e.g., those responsible for stamping and drilling) can perform the *HoleMaking* Process. This postpress Process is needed for different binding techniques, (e.g., spiral binding). One or several holes with different shapes can be made that are later on used for binding the book block together.

Table 6-142: HoleMaking – Input Resources

Name	Description
Component	One Component (e.g., a printed Sheet or a pile of Sheets) are modified in the <i>HoleMaking</i> Process.
HoleMakingParams	Specific parameters, including hole diameter and positions, used to set up the machinery.

Table 6-143: 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.

6.6.24 Inserting

This Process can be performed at several stages in postpress. The Process can be used to describe the labeling of products, labeling of packages or the gluing-in of a **Component**, (e.g., a card, Sheet or CD-ROM). Two **ComponentResources** 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 6-144: Inserting – Input Resources

Name	Description
Component Modified in JDF 1.4	Designates where to insert the child Component Modification note: starting with JDF 1.4, the input ComponentLink NEED NOT have @ProcessUsage= "Mother".
Component (Child)	The Component to be inserted in the mother Component .
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 6-145: Inserting – Output Resources

Name	Description
Component	A mother Component is produced containing the inserted child Component .

6.6.25 Jacketing

[New in JDF 1.1](#)

Jacketing is the Process where the book is wrapped by a jacket that needs to be folded twice. As long as the book is specified and the jacket dimensions are known, there are just a few important details. If the jacketing Device also creases the jacket, this can be described with a Combined Process of *Jacketing* and *Creasing*.

Table 6-146: 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 6-147: Jacketing – Output Resources

Name	Description
Component	The jacketed book.

6.6.26 Labeling

[New in JDF 1.1](#)

A label can be attached to a bundle. The label can contain information on the addressee, the product, the product quantities, etc., which can be different for each bundle.

Table 6-148: Labeling – Input Resources (Section 1 of 2)

Name	Description
Component	The <i>Labeling</i> Process labels one Component with a set of labels.
Component (Label) ?	The label to be attached to the Component .

Table 6-148: Labeling – Input Resources (Section 2 of 2)

Name	Description
LabelingParams	Specific parameters to set up the machinery.

Table 6-149: Labeling – Output Resources

Name	Description
Component	One Component is produced: the labeled Component .

6.6.27 Laminating

In the *Laminating* Process, a plastic film is bonded to one or both sides of a **Component** Resource's media, and adhered under pressure with either a thermal setting or pressure sensitive adhesive.

Table 6-150: Laminating – Input Resources

Name	Description
Component	A Component is REQUIRED for <i>Laminating</i> .
LaminatingParams	Specific parameters to set up the machinery.
Media ?	The laminating foil material.

Table 6-151: Laminating – Output Resources

Name	Description
Component	One Component is produced: the laminated component.

6.6.28 LongitudinalRibbonOperations

[Deprecated in JDF 1.1](#)

In version 1.1 of JDF and beyond, in-line finishing is described using the “standard” finishing Processes (e.g., *Creasing*, *Cutting*, *Folding*) or in a Combined Process Node with *ConventionalPrinting*. See "LongitudinalRibbonOperations" on page 1020 for details of this deprecated Process.

6.6.29 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 6-152: Numbering – Input Resources

Name	Description
Component	One Component (e.g., a printed Sheet or a pile of Sheets) are modified in the <i>Numbering</i> Process.
NumberingParams	Specific parameters to set up the machinery.

Table 6-153: Numbering – Output Resources

Name	Description
Component	One Component is produced: the numbered Sheet.

6.6.30 Palletizing

[New in JDF 1.1](#)

Bundles, stacks, piles or boxes can be loaded onto a pallet.

Table 6-154: Palletizing – Input Resources

Name	Description
Component + Modified in JDF 1.4	The <i>Palletizing</i> Process describes placing the bundle that is represented by the Component onto a pallet. If more than one Component is specified, a PalletizingParams/Bundle Resource MUST also be specified. Modification note: starting with JDF 1.4, Component can occur more than once.
PalletizingParams	Specific parameters to set up the machinery.
Pallet	The pallet.

Table 6-155: Palletizing – Output Resources

Name	Description
Component	One Component is produced. It represents the loaded pallet. If more than one input Component is supplied, a Component/Bundle Resource MUST also be supplied in the output Component

6.6.31 Perforating

[New in JDF 1.1](#)

Perforating describes any Process where a **Component** is perforated. *Perforating* includes production perforation applied as a preparation for *Folding*.

Table 6-156: Perforating – Input Resources

Name	Description
Component	This Process consumes one Component : the printed Sheets.
PerforatingParams	Details of the <i>Perforating</i> Process.

Table 6-157: Perforating – Output Resources

Name	Description
Component	One Component is produced.

6.6.32 PlasticCombBinding

In the *PlasticCombBinding* Process, a plastic insert wraps through prepunched holes in the substrate. Most often, these holes are rectangular and elongated. After the plastic comb is opened with a special tool, the prepunched block of Sheets – often together with a top and button cover – is inserted onto the “teeth” of the plastic comb. When released from the Machine, the teeth return to their original cylindrical positions with the points tucked into the back-side of the spine area. Special machinery can be used to reopen the plastic comb binding.

Table 6-158: 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 6-159: PlasticCombBinding – Output Resources

Name	Description
Component	One Component is produced: the plastic-comb-bound component forming an item such as a calendar.

6.6.33 PrintRolling

[.New in JDF 1.2](#)

The single products like Sheets, Signatures or partial products are rolled onto a roll stand. The Roll is the output component of the Process and is used to store the products. The single components of a Roll are used as input component of a consuming Process e.g., *Collecting*, *Gathering* or *Inserting*.

**Table 6-160: PrintRolling – Input Resources**

Name	Description
Component	Component to be rolled.
PrintRollingParams ?	Print rolling parameters.
RollStand ?	Roll stand to store the component(s) as rolls.

Table 6-161: PrintRolling – Output Resources

Name	Description
Component	The print Roll.

6.6.34 RingBinding

In this Process, prepunched Sheets are placed in a ring binder. Ring binders have different numbers of rings that are fixed to a metal backbone. In most cases, two, three or four metal rings hold the Sheets together as long as the binding is closed. Depending on the amount of Sheets to be bound together, ring binders of different thickness **MUST** be used.

Table 6-162: RingBinding – Input Resources

Name	Description
Component Modified in JDF 1.4	The operation requires one component: the pile of prepunched Sheets to be inserted into the ring binder. Modification note: starting with JDF 1.4, the input ComponentLink NEED NOT have <code>@ProcessUsage= "BookBlock"</code> .
Component (RingBinder) ?	The empty ring binder that might have been printed, for example, before it is used during the <i>RingBinding</i> Process.
RingBindingParams	Specific parameters to set up the Process/machinery.

Table 6-163: RingBinding – Output Resources

Name	Description
Component	One Component is produced: the ring-bound component forming an item such as a calendar.

6.6.35 SaddleStitching

[Deprecated in JDF 1.1](#)

SaddleStitching has been replaced by *Stitching* in JDF 1.1. See "SaddleStitching" on page 1020 for details of this deprecated Process.

6.6.36 ShapeCutting

[New in JDF 1.1](#)

The *ShapeCutting* Process can be performed using tools such as hollow form punching, perforating or die-cutting equipment.

Table 6-164: 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 <i>ShapeCutting</i> Process.
Tool * Modified in JDF 1.3	The set of tools (die, counter, blankers, strippers, etc.).

Table 6-165: ShapeCutting – Output Resources

Name	Description
Component + Modified in JDF 1.3	One or more Component Resources are produced by the <i>ShapeCutting</i> Process.

6.6.37 ShapeDefProduction

[New in JDF 1.4](#)

This process describes the structural design of a packaging or labels product e.g. a non rectangular label, a box, a display, a bag, a pouch, etc. Also, this process typically (but not exclusively) describes the process of designing the shape of a new box using a CAD application. The output of the *ShapeDefProduction* Process can be multiple **ShapeDef** Resources, e.g. when the design of the box results in multiple pieces, e.g. a box and an insert piece to fix to object to be packed in the box. Another example would be a multi-piece display. The *ShapeDefProduction* Process can be performed by a human operator using a CAD application. In some cases it can be an automated process. Note that *ShapeDefProduction* needs information stored in both **ShapeDefProductionParams** and **ShapeDef** to make a new structural design.

Table 6-166: ShapeDefProduction – Input Resources

Name	Description
LayoutElement ?	A rough drawing or outline (e.g. an EPS)of the ShapeDef that serves as the input for structural design.
ShapeDefProductionParams	Parameters for the structural design.

Table 6-167: ShapeDefProduction – Output Resources

Name	Description
ShapeDef +	A Resource describing the shape of the product to be produced

6.6.38 Shrinking

[New in JDF 1.1](#)

Shrink-wrap foil MUST be treated in order to shrink.

Table 6-168: Shrinking – Input Resources

Name	Description
Component	The <i>Shrinking</i> 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 6-169: Shrinking – Output Resources

Name	Description
Component	One Component is produced: the bundle including bundle including the shrunk shrink-wrap media.

6.6.39 SideSewing

[Deprecated in JDF 1.1](#)

Replaced by *ThreadSewing*. See "SideSewing" on page 1021 for details of this deprecated Process.

6.6.40 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 6-170: SpinePreparation – Input Resources

Name	Description
Component	The raw book block.
SpinePreparationParams	Specific parameters to set up the machinery.

Table 6-171: SpinePreparation – Output Resources

Name	Description
Component	The book block with a processed spine.

6.6.41 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 6-172: SpineTaping – Input Resources

Name	Description
Component	The book block that the spine is taped to.
SpineTapingParams	Specific parameters to set up the machinery.

Table 6-173: SpineTaping – Output Resources

Name	Description
Component	The book block with the spine.

6.6.42 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 6-174: 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 6-175: Stacking – Output Resources

Name	Description
Component	One Component is produced: the stack of input Components.

6.6.43 StaticBlocking

[New in JDF 1.4](#)

The *StaticBlocking* Process puts an electrical charge on a stack in order to hold it together for shipping.

Table 6-176: StaticBlocking – Input Resources

Name	Description
Component	The <i>StaticBlocking</i> Process puts an electrical charge on the specified Component .
StaticBlockingParams	Specific parameters for the electrical charging.

Table 6-177: StaticBlocking – Output Resources

Name	Description
Component	The resulting electrically charged Component .

6.6.44 Stitching

Gathered or collected Sheets or Signatures are stitched together with a cover.

Table 6-178: 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 6-179: Stitching – Output Resources

Name	Description
Component	One Component is produced: the gathered or collected Sheets including the cover stitched together.

Example 6-11: Stitching: Combined Process

Components containing staples of different characteristics like shape, width, etc. are defined by a Combined Process.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="CombinedStitch"
```

```

JobID="Stitching special" JobPartID="ID123" Type="Combined"
Types="Stitching Stitching" Status="Ready" Version="1.4">
<ResourcePool>
  <StitchingParams Class="Parameter" ID="Stitch1" NumberOfStitches="2"
    StapleShape="Butted" Status="Available" StitchPositions="100 700"
    StitchWidth="28.3" WireBrand="Steel" WireGauge="2.3"/>
  <StitchingParams Class="Parameter" ID="Stitch2" NumberOfStitches="2"
    StapleShape="Eyelet" Status="Available" StitchPositions="300 500"
    StitchWidth="42.5" WireBrand="Steel" WireGauge="2.3"/>
  <Component Class="Quantity" ID="Comp1" Status="Available"
    ComponentType="Sheet"/>
  <Component Class="Quantity" ID="Comp2" Status="Unavailable"
    ComponentType="Sheet"/>
</ResourcePool>
<ResourceLinkPool>
  <StitchingParamsLink CombinedProcessIndex="0" Usage="Input" rRef="Stitch1"/>
  <StitchingParamsLink CombinedProcessIndex="1" Usage="Input" rRef="Stitch2"/>
  <ComponentLink Usage="Input" rRef="Comp1"/>
  <ComponentLink Usage="Output" rRef="Comp2"/>
</ResourceLinkPool>
</JDF>

```

6.6.45 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 6-180: Strapping – Input Resources

Name	Description
Component	The <i>Strapping</i> 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 6-181: Strapping – Output Resources

Name	Description
Component	One Component is produced: the strapped Component.

6.6.46 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 6-182: 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 6-183: StripBinding – Output Resources

Name	Description
Component	One Component is produced: the strip-bound component forming an item such as a book.

6.6.47 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 6-184: ThreadSealing – Input Resources

Name	Description
Component	This Process consumes one Component : the printed Sheets.
ThreadSealingParams	Details of the <i>ThreadSealing</i> Process.

Table 6-185: ThreadSealing – Output Resources

Name	Description
Component	One Component is produced.

6.6.48 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 6-186: ThreadSewing – Input Resources

Name	Description
Component	The operation requires one component: the gathered Sheets.
ThreadSewingParams	Specific parameters to set up the machinery.

Table 6-187: ThreadSewing – Output Resources

Name	Description
Component	One Component is produced: the thread-sewn components forming an item such as a raw book block.

6.6.49 Trimming

The *Trimming* Process is performed to adjust a book block or Sheet to its final size. In most cases, it follows a block joining Process, and the Process is often executed as an in-line operation of a production chain. For example, the binding station might deliver the book blocks to the trimmer. A Combined Process in the trimming machinery would then execute a cut at the front, head and tail in a cycle of two operations. Closed edges of folded Signatures would then be opened while the book block is trimmed to its predetermined dimensions.

The separation of N-up Multiple Products is specified with a *Cutting* Process in front of a *Trimming* Process.

Table 6-188: Trimming – Input Resources (Section 1 of 2)

Name	Description
Component Modified in JDF 1.2	The bound book block or Sheet that will be trimmed.

Table 6-188: Trimming – Input Resources (Section 2 of 2)

Name	Description
TrimmingParams	Specific parameters (e.g., trim size) to set up the machinery.

Table 6-189: Trimming – Output Resources

Name	Description
Component	One Component is produced: the trimmed component.

6.6.50 WebInlineFinishing

[New in JDF 1.3](#)

The *WebInlineFinishing* Process combines all additional information about inline finishing functionality in connection with Web Printing. In order to describe the *WebInlineFinishing* functionality fully, it is necessary to combine additional Processes like *Stitching*, *Trimming*, *Gluing*, etc.

Table 6-190: 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 <i>WebInlineFinishing</i> 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 6-191: 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>

6.6.51 WireCombBinding

In *WireCombBinding* metal wire, wire with plastic or pure plastic is used to fasten prepunched Sheets of paper, cardboard or other such materials. The wire – often formed as a double wire – is inserted into the holes, then curled to create a circular enclosure.

Table 6-192: 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 6-193: WireCombBinding – Output Resources

Name	Description
Component	One Component is produced: the wire-comb bound component forming an item such as a calendar.

6.6.52 Wrapping

[New in JDF 1.1](#)

Single products, bundles or pallets can be wrapped by film or paper.

Table 6-194: Wrapping – Input Resources

Name	Description
Component	The <i>Wrapping</i> Process wraps a bundle that is represented by a Component .
WrappingParams	Specific parameters to set up the machinery.
Media ?	The wrapping material.

Table 6-195: 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 leaves 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:

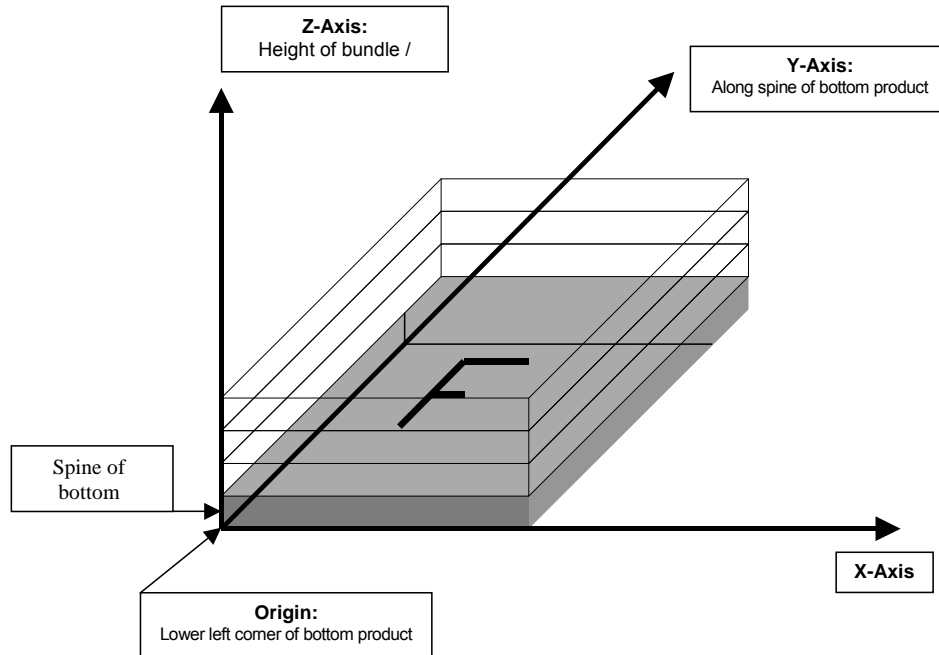


Figure 6-9: Packaging Process Coordinate System

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 Sheet-Fed printing presses. The following Processes describe these Sheet finishing operations. Sheet Processes include:

- *Creasing*
- *Cutting*
- *Embossing*
- *Feeding*

- *Folding*
- *Gathering*
- *Gluing*
- *Palletizing*
- *Perforating*
- *PrintRolling*
- *ShapeCutting*
- *ThreadSealing*

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 3.8, 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 Product 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 11 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 3.8.5, Resource Classes. A template of this list is shown below.

After the list describing the Resource properties, each section contains tables that outline the structure of each Resource and, when applicable, the abstract or Subelement information that pertains to the Resource structure. The first column contains the name of the Attribute or Element. A template of these tables is also provided below.

Note: for the Resource Properties Template below, the *italicized* text describes the actual text that would be in its place in an actual Resource definition.

Note 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 Partition Keys: For a complete list of Partition Keys, see the description of PartIDKeys in Table 3-27, “Partitionable Resource Element” on page 103. 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 Subelement 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 3.10.2, 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 Span Element to use is determined by the data type of the values to be recorded. Span Elements are defined to facilitate negotiation between buyer and provider as defined in Section 4.1.2, Defining Business Objects Using Intent Resources.

7.1.1.1 Abstract Span Element

Span Elements of Intent Resources have a common set of Attributes that define the priority, data type and requested identity of the Element. These common Attributes are described in Table 7-2, “Abstract Span Element”. In addition, Abstract Span Elements have at least four Attributes that define the data type dependent aspects of the span. The data type of these values depends on the data type of the span and is defined in the following sections:

Actual – The intended value agreed to by the producer of the product.

OfferRange – A proposed range of equivalent values in cost that are defined by the producer of the product.

Preferred – A preferred value defined by the recipient of the product.

Range – A proposed range of values defined by the recipient of the product.

Table 7-2: Abstract Span Element (Section 1 of 2)

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. Values are:										
		<table border="1"> <tbody> <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> </tbody> </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>NumberSpan</i>	<i>XYPairSpan</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>											

Table 7-2: Abstract Span Element (Section 2 of 2)

Name	Data Type	Description
<i>Priority?</i> Deprecated in JDF 1.2	enumeration	Indicates the importance of the specific intent. Values are: <i>None</i> <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> . <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. Deprecation note: starting with JDF 1.2, use <i>SettingsPolicy</i> .

7.1.1.2 Span Elements

The Data Type column of tables for *Intent Resources* (below) can contain the same data types as non-*Intent Resources* (namely data types defined in the "Data Structures" on page 13) as well as *Span Elements* that are listed in the Table 7-3, "List of Span Elements". In *Intent Resource* tables, *XXXSpan* Elements are treated as Attribute-like data types even though *Span Elements* are technically XML elements because the semantic usage of the *Span Elements* is equivalent to the usage of *Attributes* in *Process Resources*.

Each *Span Element* contains *Attributes* or *Subelements* listed in Table 7-2, "Abstract Span Element" and in the pertinent *Span Element* listed in Table 7-3, "List of Span Elements"

Table 7-3: List of Span Elements

Name	Page	Description
<i>DurationSpan</i> New in JDF 1.1	page 341	Describes a set of duration values.
<i>EnumerationSpan</i>	page 342	Describes a set of enumeration values.
<i>IntegerSpan</i>	page 343	Describes a numerical range of integer values.
<i>NameSpan</i>	page 343	Describes a set of NMTOKEN values.
<i>NumberSpan</i>	page 343	Describes a numerical range of values.
<i>OptionSpan</i>	page 344	Describes an intent in which the principal information is that a specific option is requested.
<i>ShapeSpan</i> New in JDF 1.1	page 344	Describes a set of shape values.
<i>StringSpan</i>	page 345	Describes a set of string values.
<i>TimeSpan</i>	page 345	Describes a set of dateTime values.
<i>XYPairSpan</i>	page 345	Describes a set of XYPair values.

7.1.1.2.1 DurationSpan

[New in JDF 1.1](#)

This *Span Subelement* is used to describe a selection of instances in time. It inherits from the *Abstract Span Element* described in Section 7.1.1.1, *Abstract Span Element*.

Table 7-4: DurationSpan Element (Section 1 of 2)

Name	Data Type	Description
<i>Actual?</i>	duration	The actual value selected for the quote.

Table 7-4: DurationSpan Element (Section 2 of 2)

Name	Data Type	Description
<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> .

7.1.1.2.2 EnumerationSpan

This Span Subelement is used to describe ranges of enumerative values. It inherits from the Abstract Span Element described in Section 7.1.1.1, 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. Default value is from: @Actual.
<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. Default value is from: @Preferred.

Example 7-1: EnumerationSpan

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="Product" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <BindingIntent Class="Intent" ID="BI1" Status="Available">
      <BindingType DataType="EnumerationSpan" Actual="Ring"/>
      <RingBinding>
        <HoleType DataType="EnumerationSpan" Range="R4m-DIN-A5 R6m-DIN-A5">
          <Comment Name="R4m-DIN-A5">
            4 equidistant holes on each side of a hexagonal piece of paper
          </Comment>
          <Comment Name="R6m-DIN-A5">
            6 equidistant holes on each side of a hexagonal piece of paper
          </Comment>
        </HoleType>
      </RingBinding>
    </BindingIntent>
    <Component Class="Quantity" ID="cI1" Status="Unavailable"
      ComponentType="Sheet"/>
  </ResourcePool>
  <ResourceLinkPool>
    <BindingIntentLink Usage="Input" rRef="BI1"/>
    <ComponentLink Usage="Output" rRef="cI1"/>
  </ResourceLinkPool>
</JDF>
```


7.1.1.2.3 IntegerSpan

This Span Subelement is used to describe ranges of integer values. It inherits from the Abstract Span Element described in Section 7.1.1.1, 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. Default value is from: @Actual.
<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. Default value is from: @Preferred.

7.1.1.2.4 NameSpan

This Span Subelement is used to describe name ranges. It inherits from the Abstract Span Element described in Section 7.1.1.1, 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. Default value is from: @Actual.
<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. Default value is from: @Preferred.

7.1.1.2.4.1 Specifying New Values in a NameSpan Subelement

NameSpan Elements generally define an open list of predefined values. If a custom value is specified, a Comment Element in the NameSpan defines the value with a *Name* Attribute in the Comment, as demonstrated in the following example:

7.1.1.2.5 NumberSpan

This Span Subelement is used to describe a numerical range of values. It inherits from the Abstract Span Element described in Section 7.1.1.1, Abstract Span Element.

Table 7-8: NumberSpan Element (Section 1 of 2)

Name	Data Type	Description
<i>Actual?</i>	double	The actual value selected for the quote.

Table 7-8: NumberSpan Element (Section 2 of 2)

Name	Data Type	Description
<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. Default value is from: @ <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. Default value is from: @ <i>Preferred</i> .

7.1.1.2.6 OptionSpan

This Span Subelement is used to describe a range of options or Boolean values. It inherits from the Abstract Span Element described in Section 7.1.1.1, 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 = " <i>true</i> ".
<i>OfferRange?</i> New in JDF 1.3	enumerations	Provides a set of the discreet Boolean values. Default value is from: @ <i>Actual</i> . Values are: <i>true</i> <i>false</i>
<i>Detail?</i> Deprecated in JDF 1.2	string	<i>Detail</i> provides information about the option. Deprecation note: starting with JDF 1.2, use <i>DescriptiveName</i> .
<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. Values are: <i>true</i> <i>false</i>

7.1.1.2.7 ShapeSpan

[New in JDF 1.1](#)

This Span Subelement is used to describe ranges of numerical value pairs. It inherits from the Abstract Span Element described in Section 7.1.1.1, Abstract Span Element.

Table 7-10: ShapeSpan Element (Section 1 of 2)

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. Default value is from: @ <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> .

Table 7-10: ShapeSpan Element (Section 2 of 2)

Name	Data Type	Description
<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. Default value is from: @Preferred.

7.1.1.2.8 StringSpan

This Span Subelement is used to describe string ranges. It inherits from the Abstract Span Element described in Section 7.1.1.1, 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. Default value is from: @Actual.
<i>Range</i> *	telem	Provides a set of discreet values that comprise all allowed values for the Span. Default value is from: @Preferred.

7.1.1.2.9 TimeSpan

This Span Subelement is used to describe a selection of instances in time. It inherits from the Abstract Span Element described in Section 7.1.1.1, 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. Default value is from: @Actual.
<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. Default value is from: @Preferred.

7.1.1.2.10 XYPairSpan

This Span Subelement is used to describe ranges of numerical value pairs. It inherits from the Abstract Span Element described in Section 7.1.1.1, Abstract Span Element.

Table 7-13: XYPairSpan Element (Section 1 of 2)

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. Default value is from: @Actual.

Table 7-13: XYPairSpan Element (Section 2 of 2)

Name	Data Type	Description
<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. Default value is from: @Preferred.

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:	Option
Input of Processes:	Any Product Intent Node
Output of Processes:	—

Table 7-14: ArtDeliveryIntent Resource (Section 1 of 4)

Name	Data Type	Description
ArtDeliveryDate ? New in JDF 1.1	TimeSpan	Specifies the latest time by which the transfer of the artwork will be made.
ArtDeliveryDuration ? New in JDF 1.1	DurationSpan	Specifies the latest time by which the transfer will be made relative to the date of the purchase order. Within an RFQ or a Quote, at most one of either ArtDeliveryDate or ArtDeliveryDuration MUST be specified. Within a purchase order, only ArtDeliveryDate is allowed.
ArtHandling ? New in JDF 1.1	EnumerationSpan	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 (@ContactTypes, "ArtReturn")]/ Address . 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.

Table 7-14: ArtDeliveryIntent Resource (Section 2 of 4)

Name	Data Type	Description
DeliveryCharge ? New in JDF 1.1 Modified in JDF 1.3	EnumerationSpan	Specifies who pays for a delivery being made by a third party. Values are from: <i>DeliveryIntent/DeliveryCharge</i> .
Method ? Modified in JDF 1.3	NameSpan	Specifies the delivery method, which can be a generic method. Values include: <i>EMail</i> <i>ExpressMail</i> <i>InterofficeMail</i> <i>OvernightService</i> <i>Courier</i> <i>CompanyTruck</i> <i>ISDNSoftware</i> <i>Local</i> – The files are already in place and a <i>DigitalDelivery</i> Process is not needed. New in JDF 1.3 <i>NetworkCopy</i> – This includes LAN and VPN. <i>WebServer</i> – Upload / download from HTTP / FTP server. <i>InstantMessaging</i> <i>UPS</i> – a delivery service brand. <i>DHL</i> – a delivery service brand. <i>FedEx</i> – a delivery service brand. <i>Vio</i> – a digital delivery service brand <i>WAMNET</i> – a digital delivery service brand
<i>PreflightStatus</i> = <i>"NotPerformed"</i> New in JDF 1.1 Modified in JDF 1.2	enumeration	Information about a <i>Preflight</i> Process probably applied to the artworks before being submitted. Values are: <i>NotPerformed</i> – No preflighting was applied. <i>WithErrors</i> – Preflighting resulted in error messages and possibly warning messages. <i>WithWarnings</i> – Preflighting resulted in warning messages and no errors. <i>WithoutErrors</i> – Preflighting was successful. No errors and no warnings occurred.

Table 7-14: ArtDeliveryIntent Resource (Section 3 of 4)

Name	Data Type	Description
<i>ReturnList</i> = "None" New in JDF 1.1	NMTOKENS	Type of printer created intermediate materials that are to be sent to the customer after usage. Values include: <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>Tool</i> – Tools needed for processing the Job, (e.g., a die for die cutting or embossing stamp). <i>None</i> – No intermediate materials are to be returned to the customer.
ReturnMethod ? New in JDF 1.1	NameSpan	Specifies a delivery method for returning the artwork if ArtHandling/@Actual = "Return" and for the printer created materials listed in ReturnList. Values include those from: Method
ServiceLevel ? New in JDF 1.2	StringSpan	The service level of the specific carrier. Values include: <i>Next Day</i> <i>2nd Day Air</i> <i>Ground.</i>
Transfer ? New in JDF 1.1	EnumerationSpan	Describes the responsibility of the transfer. Values are: <i>BuyerToPrinterDeliver</i> – The buyer delivers the artwork to the printer. The printer MAY specify in the quote a special Contact [contains (@ContactTypes, "Delivery")] to specify where the buyer is to send the artwork. <i>BuyerToPrinterPickup</i> – The printer picks up the artwork. The Contact [contains (@ContactTypes, "Pickup")] specifies where the printer has to pick up the artwork.
ArtDelivery + Modified in JDF 1.1	element	Individual delivery.
Company ? Deprecated in JDF 1.1	refelement	Address and further information of the art delivery. Company 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 .

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 (@ <i>ContactTypes</i> , "Delivery")]/ Address . At most one such Contact MUST <i>be specified</i> . The actual pickup address MUST be specified by Contact [contains (@ <i>ContactTypes</i> , "Pickup")]/ Address . At most one such Contact MUST be specified.

7.1.2.1 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
<i>Amount</i> ? Modified in JDF 1.2	integer	Number of physical objects to be delivered. Only valid if no detailed Resource description (e.g., ExposedMedia , RunList , ScanParams , DigitalMedia or Tool) is specified.
ArtDeliveryDate ? New in JDF 1.1	TimeSpan	Specifies the latest time by which the transfer of the artwork will be made.
ArtDeliveryDuration ? New in JDF 1.1	DurationSpan	Specifies the latest time by which the transfer will be made relative to the date of the purchase order. Within an RFQ or a Quote, at most one of either ArtDeliveryDate or ArtDeliveryDuration MUST be specified. Within a purchase order, only the ArtDeliveryDate is allowed.
<i>ArtDeliveryType</i> New in JDF 1.1 Modified in JDF 1.2	NMTOKEN	Type of artwork supplied. Values include: <i>DigitalFile</i> – Digital data irrespective of the delivery mechanism. The union of <i>DigitalMedia</i> and <i>DigitalNetwork</i> . 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	<p>Describes what is to happen to the artwork after usage.</p> <p>The address for the <i>Return</i> and <i>Pickup</i> values MUST be specified by Contact[contains (<i>@ContactTypes</i>, "ArtReturn")]/Address.</p> <p>Default value is from: ArtDeliveryIntent/ArtHandling.</p> <p>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>
DeliveryCharge ? New in JDF 1.1 Modified in JDF 1.3	Enumeration-Span	<p>Specifies who pays for a delivery being made by a third party.</p> <p>Default value is from: ArtDeliveryIntent/DeliveryCharge.</p> <p>Values are from: DeliveryIntent/DeliveryCharge.</p>
<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	<p>Specifies a delivery method. It MAY be a generic item from the list defined in <i>Method</i> in ArtDeliveryIntent.</p> <p>Values include those from: ArtDeliveryIntent/Method</p>
<i>PageList</i> ?	IntegerRange-List	<p>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 ArtDelivery (e.g., RunList) 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.</p>
PreflightOutput ? New in JDF 1.1	URL	<p>Pointer to the output information created by the preflight tool if <i>PreflightStatus</i> is either <i>WithoutErrors</i> or <i>WithErrors</i>.</p>
PreflightStatus ? New in JDF 1.1	enumeration	<p>Information about a <i>Preflight</i> Process.</p> <p>Default value is from: ArtDeliveryIntent/@PreflightStatus.</p> <p>Values are from: ArtDeliveryIntent/@PreflightStatus.</p>
ReturnMethod ? New in JDF 1.1	NameSpan	<p>Specifies a delivery method for returning the artwork if <i>ArtHandling/@Actual</i> = "Return".</p> <p>Default value is from: ArtDeliveryIntent/ReturnMethod.</p> <p>Values include those from: ArtDeliveryIntent/ReturnMethod</p>
ServiceLevel ? New in JDF 1.2	StringSpan	<p>The service level of the specific carrier.</p> <p>Default value is from: ArtDeliveryIntent/ServiceLevel.</p> <p>Values include those from: ArtDeliveryIntent/ServiceLevel</p>

Table 7-15: ArtDelivery Element (Section 3 of 3)

Name	Data Type	Description
Transfer ? New in JDF 1.1	Enumeration-Span	Describes the responsibility of the transfer. Default value is from: ArtDeliveryIntent/Transfer. Values are from: ArtDeliveryIntent/Transfer.
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. Default value is from: ArtDeliveryIntent/Contact...
DigitalMedia ? New in JDF 1.2	refelement	Description of any digital media, (e.g., CD or tape with artwork that will be delivered). If neither ExposedMedia , RunList , DigitalMedia , nor Tool are specified, no details of the ArtDelivery except the <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 a 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 Intent Node

Output of Processes:

—

Table 7-16: BindingIntent Resource (Section 1 of 3)

Name	Data Type	Description
BackCoverColor ? New in JDF 1.1	EnumerationSpan	Defines the color of the back cover material of the binding. Default value is from: @CoverColor Values are from: Table A-3, “NamedColor Enumeration Values” on page 870.
BackCoverColorDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If BackCoverColorDetails is supplied, BackCoverColor SHOULD also be supplied.
BindingColor ?	EnumerationSpan	Defines the color of the spine material of the binding. Values are from: Table A-3, “NamedColor Enumeration Values” on page 870.
BindingColorDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If BindingColorDetails is supplied, BindingColor SHOULD also be supplied.
BindingLength ?	EnumerationSpan	Indicates which side is to be bound when no content. Thus, no orientation is available, but a quote for binding is needed. Values are: <i>Long</i> <i>Short</i>
<i>BindingOrder = "Gathering"</i> New in JDF 1.1 Modified in JDF 1.4	enumeration	Specifies whether the child Component Resources are to be collected or gathered if multiple child Component Resources are combined. Values are: <i>None</i> – The child Component Resource are NOT bound together. Typically used for flatwork Jobs. New in JDF 1.4 <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.

Table 7-16: BindingIntent Resource (Section 2 of 3)

Name	Data Type	Description
BindingSide ?	EnumerationSpan	Indicates which side are to be bound. 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. Constraint: If both BindingSide and BindingLength are specified, BindingSide has precedence Default value is from: BindingLength, unless a non-empty BindList was specified. Values are: <i>Top</i> <i>Bottom</i> <i>Right</i> <i>Left</i> .
BindingType Modified in JDF 1.2	EnumerationSpan	Describes the desired binding for the Job. Values are from: Table 7-17, “BindingType Attribute Values”.
CoverColor ?	EnumerationSpan	Defines the color of the cover material of the binding. Values are from: Table A-3, “NamedColor Enumeration Values” on page 870.
CoverColorDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If CoverColorDetails is supplied, CoverColor SHOULD also be supplied.
AdhesiveBinding ? Deprecated in JDF 1.1	element	Details of AdhesiveBinding. Replaced with SoftCoverBinding in JDF 1.1.
BindList ? New in JDF 1.1	element	Details of binding of individual child Component Resources.
BookCase ? Deprecated in JDF 1.1	element	Details of the Book Case. Used in Combination with AdhesiveBinding, ThreadSewing or ThreadSealing. Replaced with HardcoverBinding in JDF 1.1.
ChannelBinding ?	element	Details of ChannelBinding.
CoilBinding ?	element	Details of CoilBinding.
EdgeGluing ? New in JDF 1.1	element	Details of EdgeGluing.
HardCoverBinding ? New in JDF 1.1	element	Details of HardcoverBinding.
PlasticCombBinding ?	element	Details of PlasticCombBinding.
RingBinding ?	element	Details of RingBinding.
SaddleStitching ?	element	Details of SaddleStitching.
SideSewing ?	element	Details of SideSewing.
SideStitching ?	element	Details of SideStitching.
SoftCoverBinding ? New in JDF 1.1	element	Details of SoftCoverBinding.
StripBinding ? New in JDF 1.1	element	Details of StripBinding.
Tabs ?	element	Details of Tabs.

Table 7-16: BindingIntent Resource (Section 3 of 3)

Name	Data Type	Description
Tape ? New in JDF 1.1	element	Details of Tape binding.
ThreadSealing ?	element	Details of ThreadSealing.
ThreadSewing ?	element	Details of ThreadSewing.
VeloBinding ? Deprecated in JDF 1.1	element	Details of VeloBinding. Renamed to StripBinding in JDF 1.1.
WireCombBinding ?	element	Details of WireCombBinding.

— Attribute: BindingType

Table 7-17: BindingType Attribute Values

Value	Description
Adhesive Deprecated in JDF 1.1	This type of binding can be handled with the <i>AdhesiveBinding</i> Process. It includes perfect binding. Deprecated in JDF 1.1 and replaced with <i>SoftCover</i> or <i>HardCover</i> .
ChannelBinding	This type of binding can be handled with the <i>ChannelBinding</i> Process.
CoilBinding	This type of binding can be handled with the <i>CoilBinding</i> Process.
CornerStitch New in JDF 1.2	Stitch in the corner that is at the clockwise end binding edge. For example, to stitch in the top left corner, set BindingSide/@Actual = "Left". This type of binding can be handled with the <i>Stitching</i> Process.
EdgeGluing	Gluing gathered Sheets at one edge of the pile. This Type of Binding can be handled with the <i>Gluing</i> Process. Products of this type are also referred to as padded.
HardCover	This type of binding defines a hard-cover bound book.
LooseBinding	This type of binding defines a stack of pages with no additional binding.
PlasticComb	This type of binding can be handled with the <i>PlasticCombBinding</i> Process.
Ring	This type of binding can be handled with the <i>RingBinding</i> Process.
SaddleStitch	This type of binding can be handled with the <i>Stitching</i> Process.
Sewn Deprecated in JDF 1.4	This type of binding can be handled with the <i>ThreadSewing</i> Process.
SideSewn Deprecated in JDF 1.4	This type of binding can be handled with the <i>ThreadSewing</i> Process.
SideStitch	This type of binding can be handled with the <i>Stitching</i> Process.
SoftCover	This type of binding defines a soft cover bound book. It includes perfect binding.
StripBind	This type of binding can be handled with the <i>StripBinding</i> Process.
Tape	This type of binding is an inexpensive version of the <i>SoftCover</i> .
ThreadSealing Deprecated in JDF 1.4	This type of binding can be handled with the <i>ThreadSealing</i> Process.
WireComb	This type of binding can be handled with the <i>WireCombBinding</i> Process.

7.1.3.1 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.

7.1.3.2 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
BindingType ?	EnumerationSpan	Describes the desired binding for the individual BindItem. Default value is from: BindingIntent/BindingType. Values are from: BindingIntent/BindingType.
ChildFolio ?	XYPair	Definition of the fold between two pages in the BindItem component that is bound to the cover. The two numbers (as integers) in the <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 <i>LayoutIntent/@FolioCount</i> of the child product. Defaults to the spine of the child.
ParentFolio	XYPair	Definition of the fold between two pages in the Cover Component that receive the BindItem. The two numbers (as integers) in the <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 <i>LayoutIntent/@FolioCount</i> of the cover product.
Transformation ?	matrix	Rotation and offset between the Component to be inserted and the parent Component . For details on transformations, see Section 2.5.2, Coordinates and Transformations
WrapPages ?	IntegerRangeList	List of pages of the Cover that wrap around a BindItem after all folds are 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.
ChannelBinding ?	element	Details of ChannelBinding.
CoilBinding ?	element	Details of CoilBinding.
EdgeGluing ?	element	Details of EdgeGluing.
HardCoverBinding ?	element	Details of HardcoverBinding.
PlasticCombBinding ?	element	Details of PlasticCombBinding.
RingBinding ?	element	Details of RingBinding.
SaddleStitching ?	element	Details of SaddleStitching.
SideSewing ?	element	Details of SideSewing.
SideStitching ?	element	Details of SideStitching.
SoftCoverBinding ?	element	Details of SoftCoverBinding.
StripBinding ?	element	Details of StripBinding.
Tabs ?	element	Details of Tabs.
Tape ?	element	Details of Tape binding.

Table 7-19: BindItem Element (Section 2 of 2)

Name	Data Type	Description
ThreadSealing ?	element	Details of ThreadSealing.
ThreadSewing ?	element	Details of ThreadSewing.
WireCombBinding ?	element	Details of WireCombBinding.

7.1.3.3 Element: AdhesiveBinding[Deprecated in JDF 1.1](#)

The table defining the deprecated AdhesiveBinding Subelement has been moved to "BindingIntent Deprecated Subelements" on page 1021.

7.1.3.4 Element: BookCase[Deprecated in JDF 1.1](#)

The table defining the deprecated BookCase Subelement has been moved to "BindingIntent Deprecated Subelements" on page 1021.

7.1.3.5 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 <i>ChannelBinding</i> includes a preassembled cover.
Thickness ?	NumberSpan	Specifies thickness of board which is wrapped as front and back covers of a case bound book, in points.

7.1.3.6 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 <i>CoilBinding</i> . 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.

7.1.3.7 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. Values are: <i>ColdGlue</i> <i>Hotmelt</i> <i>PUR</i> – Polyurethane rubber.

7.1.3.8 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. Values are from: Table A-3, "NamedColor Enumeration Values" on page 870.
HeadBandColorDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If HeadBandColorDetails is supplied, HeadBandColor SHOULD also be supplied.
Jacket ?	EnumerationSpan	Specifies whether a hard cover jacket is needed and how it is attached. Details of the jacket MAY be described in the Component with <i>ProcessUsage</i> = "Jacket". Values are: <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
JacketFoldingWidth ? New in JDF 1.3	NumberSpan	Dimension of the jacket folds. See JacketingParams for details.
JapanBind ?	OptionSpan	Bind the book block at the open edge, so that the folds are visible on the outside. If not specified, explicitly, this option is never selected.
SpineBrushing ?	OptionSpan	Brushing option for <i>SpinePreparation</i> .

Table 7-23: HardCoverBinding Element (Section 2 of 2)

Name	Data Type	Description
SpineFiberRoughing ?	OptionSpan	Fiber roughing option for <i>SpinePreparation</i> .
SpineGlue ?	EnumerationSpan	Glue type used to glue the book block to the cover. Values are: <i>ColdGlue</i> <i>Hotmelt</i> <i>PUR</i> – Polyurethane rubber.
SpineLevelling ?	OptionSpan	Leveling option for <i>SpinePreparation</i> .
SpineMilling ?	OptionSpan	Milling option for <i>SpinePreparation</i> .
SpineNotching ?	OptionSpan	Notching option for <i>SpinePreparation</i> .
SpineSanding ?	OptionSpan	Sanding option for <i>SpinePreparation</i> .
SpineShredding ?	OptionSpan	Shredding option for <i>SpinePreparation</i> .
StripMaterial ?	EnumerationSpan	Spine taping strip material. Values are: <i>Calico</i> <i>Cardboard</i> <i>CrepePaper</i> <i>Gauze</i> <i>Paper</i> <i>PaperlinedMules</i> <i>Tape</i>
Thickness ?	NumberSpan	Specifies thickness of board which is wrapped as front and back covers of a case bound book, in points.
TightBacking ?	EnumerationSpan	Definition of the geometry of the back of the book block. Values are: <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.

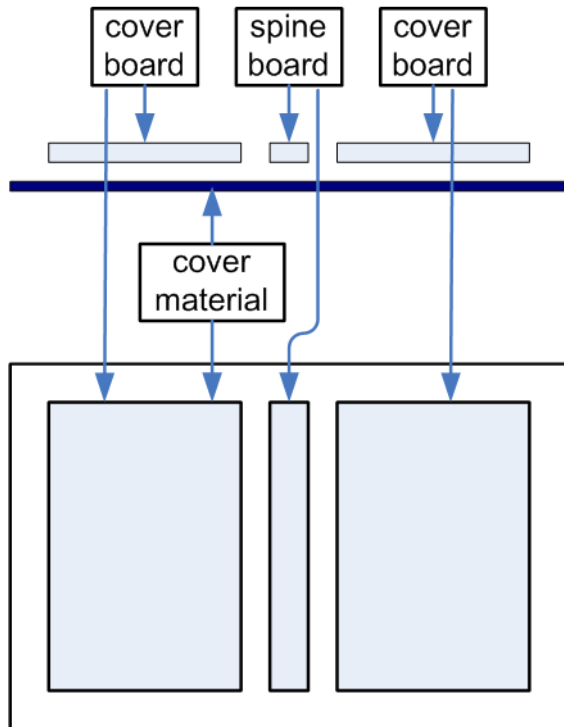


Figure 7-1: Structure of a normal hardcover book

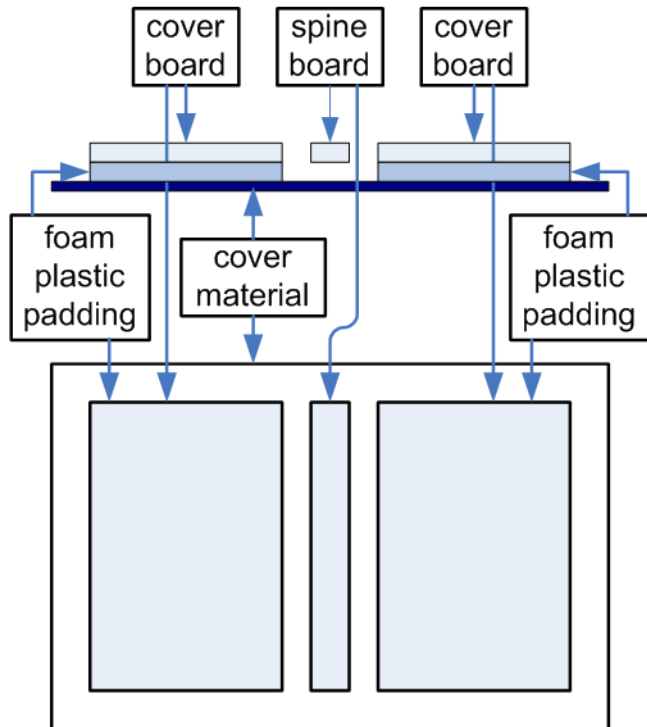


Figure 7-2: Structure of a padded hardcover book

7.1.3.9 Element: PlasticCombBinding

Table 7-24: PlasticCombBinding Element

Name	Data Type	Description
CombBrand ? New in JDF 1.3	StringSpan	Strings providing available brand names for the plastic comb.
PlasticCombType ? Modified in JDF 1.1	NameSpan	The distance between the “teeth” in <i>PlasticCombBinding</i> 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 929 exist: Values include: <i>P12m-rect-02</i> – Distance = 12 mm; Holes = 7 mm x 3 mm <i>P16_9i-rect-0t</i> – Distance = 14.28 mm; Holes = 8 mm x 3 mm <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
HoleList ? New in JDF 1.2	element	Details of the holes for the plastic comb. Note that <i>Shape</i> is always rectangular by design of the plastic combs.

7.1.3.10 Element: RingBinding

Table 7-25: RingBinding Element

Name	Data Type	Description
BinderBrand ? New in JDF 1.3	StringSpan	Strings providing available brand names for RingBinding.
BinderMaterial ?	NameSpan	The following describe <i>RingBinding</i> 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.
HoleType ? 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, see the values. Values are from: "JDF/CIP4 Hole Pattern Catalog" on page 929.
RingDiameter ?	NumberSpan	Size of the rings in points. The value used in production MUST be suitable for specified values of HoleType. Note that in ring shapes other than round, this size is specified by industry-standard method.
RingMechanic ?	OptionSpan	The ring binder used includes a lever for opening and closing.
RingShape ?	NameSpan	<i>RingBinding</i> shapes. Values include: <i>Round</i> <i>Oval</i> <i>D-shape</i> <i>SlantD</i>
RingSystem ? Deprecated in JDF 1.1	NameSpan	Values include: <i>2HoleEuro</i> <i>3HoleUS</i> <i>4HoleEuro</i> Deprecation note: starting with JDF 1.1, use HoleType.
RivetsExposed ?	OptionSpan	The following <i>RingBinding</i> 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.
ViewBinder ?	NameSpan	The values are <i>RingBinding</i> clear vinyl outer wrap types and are used on top of a colored base wrap. Values include: <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.

7.1.3.11 Element: SaddleStitching

Table 7-26: SaddleStitching Element

Name	Data Type	Description
StitchNumber ? New in JDF 1.1	IntegerSpan	Number of stitches used for saddle stitching.

7.1.3.12 Element: SideSewing

This is a placeholder that might be filled with private or future data.

Table 7-27: SideSewing Element

Name	Data Type	Description

7.1.3.13 Element: SideStitching

Table 7-28: SideStitching Element

Name	Data Type	Description
StitchNumber ? New in JDF 1.2	IntegerSpan	Number of stitches used for side stitching.

7.1.3.14 Element: SoftCoverBinding

[New in JDF 1.1](#)

Table 7-29: SoftCoverBinding Element (Section 1 of 2)

Name	Data Type	Description
BlockThreadSewing ?	OptionSpan	Specifies whether the block is also thread sewn.
EndSheets ? New in JDF 1.3	OptionSpan	Specified if end Sheets are applied. Additional details of the EndSheets MAY be specified by supplying an input Component with <i>ProcessUsage</i> = "EndSheet".
FoldingWidth ? New in JDF 1.3	NumberSpan	Definition of the dimension of the folding width of the front cover fold. See JacketingParams for details.
FoldingWidthBack ? New in JDF 1.3	NumberSpan	Definition of the dimension of the folding width of the back cover fold. If not specified, FoldingWidthBack defaults to FoldingWidth.
GlueProcedure ?	EnumerationSpan	Glue procedure used to glue the book block to the cover. 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 EndSheets are also specified. See Figure "Structure of a book with GlueProcedure = "SideOnly" (Layflat)" on page 362 <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
Scoring ?	EnumerationSpan	Scoring option for <i>SoftCoverBinding</i> . Values are based on viewing the cover in its flat, prebound state. Values are: <i>TwiceScored</i> <i>QuadScored</i> <i>None</i>
SpineBrushing ?	OptionSpan	Brushing option for <i>SpinePreparation</i> .
SpineFiberRoughing ?	OptionSpan	FiberRoughing option for <i>SpinePreparation</i> .
SpineGlue ?	EnumerationSpan	Glue type used to glue the book block to the cover. Values are: <i>ColdGlue</i> <i>Hotmelt</i> <i>PUR</i> – Polyurethane rubber.
SpineLevelling ?	OptionSpan	Leveling option for <i>SpinePreparation</i> .
SpineMilling ?	OptionSpan	Milling option for <i>SpinePreparation</i> .
SpineNotching ?	OptionSpan	Notching option for <i>SpinePreparation</i> .
SpineSanding ?	OptionSpan	Sanding option for <i>SpinePreparation</i> .
SpineShredding ?	OptionSpan	Shredding option for <i>SpinePreparation</i> .

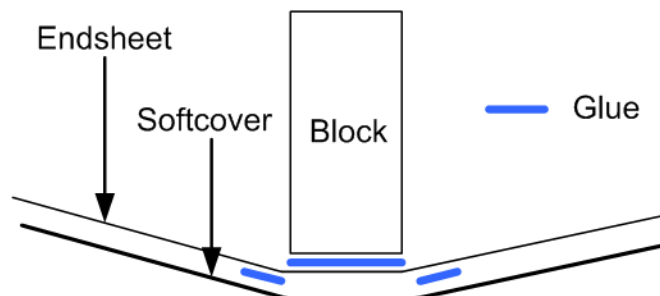


Figure 7-3: Structure of a book with GlueProcedure = "SideOnly" (Layflat)

7.1.3.15 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.

7.1.3.16 Element: Tabs

Specifies tabs in a bound document.

Table 7-31: Tabs Element

Name	Data Type	Description
<i>TabBanks</i> = "1" Deprecated in JDF 1.4	integer	Number of rows of tabs on the face of the book. Deprecation note: starting with JDF 1.4, <i>TabBanks</i> should be calculated from <i>TabCount</i> and <i>TabsPerBank</i> .
<i>TabBrand</i> ? New in JDF 1.3	StringSpan	Strings providing available brand names for the Tabs.
<i>TabCount</i> ? New in JDF 1.4	integer	Number of tabs across all banks. If <i>TabsPerSet</i> is not an even multiple of <i>TabsPerBank</i> , the last bank in each set is partially filled.
<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 " <i>true</i> ", the tab extension will be mylar reinforced.
<i>TabBindMylar</i> ?	OptionSpan	If " <i>true</i> ", the tab bind edge will be mylar reinforced.
<i>TabBodyCopy</i> ?	OptionSpan	If " <i>true</i> ", 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 " <i>true</i> ". Values are from: NamedColor (Section A.3.3.3, NamedColor).
<i>TabMylarColorDetails</i> ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If <i>TabMylarColorDetails</i> is supplied, <i>TabMylarColor</i> SHOULD also be supplied.

7.1.3.17 Element: Tape[New in JDF 1.1](#)

Table 7-32: Tape Element

Name	Data Type	Description
<i>TapeColor</i> ? Deprecated in JDF 1.4	EnumerationSpan	Defines the color of the tape material of the binding. Values are from: Table A-3, "NamedColor Enumeration Values" on page 870. Deprecation note: starting with JDF 1.4, use BindingIntent/@BindingColor .

7.1.3.18 Element: ThreadSealing

This is a placeholder that might be filled with private or future data.

Table 7-33: ThreadSealing Element

Name	Data Type	Description

7.1.3.19 Element: ThreadSewing

Table 7-34: ThreadSewing Element

Name	Data Type	Description
Sealing ?	OptionSpan	If "true", thermo-sealing is needed in <i>ThreadSewing</i> .

7.1.3.20 Element: WireCombBinding

Table 7-35: WireCombBinding Element

Name	Data Type	Description
WireCombBrand ? New in JDF 1.3	StringSpan	Strings providing available brand names for the <i>WireCombBinding</i> .
WireCombMaterial ?	EnumerationSpan	The material used for forming the <i>WireCombBinding</i> . Values are: <i>Steel-Silver</i> <i>ColorCoatedSteel</i>
WireCombShape ?	EnumerationSpan	The shape of the <i>WireCombBinding</i> . 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:	<i>Option, PageNumber, Side</i>
Input of Processes:	<i>Any Product Intent Node</i>
Output of Processes:	—

Table 7-36: ColorIntent Resource (Section 1 of 3)

Name	Data Type	Description
Coatings ? Modified in JDF 1.3	StringSpan	Material usually applied to a full surface on press as a protective or gloss-enhancing layer over ink. Values include: <i>Aqueous</i> <i>DullVarnish</i> <i>GlossVarnish</i> <i>Protective</i> <i>SatinVarnish</i> <i>Silicone</i> <i>UV</i> Note: The individual strings within Coatings are of type NMTO-KENS and MAY contain multiple entries from the list of values. Note: that spot coating is specified in ColorsUsed.
ColorStandard? Modified in JDF 1.2	NameSpan	The color process (i.e., printing condition) requested for the Job. ColorStandard does not imply values for ColorsUsed. For instance, if ColorStandard is <i>CMYK</i> , ColorsUsed MUST still contain the four process colors <i>Cyan</i> , <i>Magenta</i> , <i>Yellow</i> and <i>Black</i> . If both of ColorICCStandard and ColorStandard are specified, then ColorICCStandard defines the ICC specific details, whereas ColorStandard defines the generic color standard. Values include those from: Table 7-37, “ColorStandard Attribute Values” on page 367

Table 7-36: ColorIntent Resource (Section 2 of 3)

Name	Data Type	Description
ColorICCStandard ? New in JDF 1.2	StringSpan	<p>ColorICCStandard can be used to identify a specific standard printing condition, by reference to Characterization Data registered with the ICC (http://www.color.org/drsection1.html). This printing condition reference corresponds to the 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.</p> <p>Values include:</p> <p><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/m²) and paper type 2 (matte-coated, above 70 g/m²), screen frequency 60/cm. Appropriate for black-backing measurement.</p> <p><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/m²) and paper type 2 (matte-coated, above 70 g/m²), screen frequency 60/cm. Appropriate for self-backing measurement.</p> <p><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.</p> <p>Note: If both of ColorICCStandard or ColorsUsed are specified, the union of the two is specified. If both of ColorICCStandard and ColorStandard are specified, then ColorICCStandard defines the ICC specific details, whereas ColorStandard defines the generic color standard.</p>
Coverage ?	NumberSpan	<p>Cumulative colorant coverage percentage. For example, a full Sheet of 100% deep black in CMYK has Coverage/@Actual= "400". Typical coverages based on one color plane are:</p> <p><i>Light</i> = 1-9%</p> <p><i>Medium</i> = 10-35%</p> <p><i>Heavy</i> = 36+%</p>
InkManufacturer ? Deprecated in JDF 1.2	NameSpan	Name of the manufacturer of the ink requested, (e.g., "ACMEInk", "CIP4_Ink_Company" etc.).
ColorPool ? New in JDF 1.1	refelement	Additional details about the colors used. The ColorPool Resource MAY include some or all details about both ColorsUsed separation spot colors, spot colors contained in Job files that will be printed using process color equivalents and the ColorStandard process colors.

Table 7-36: ColorIntent Resource (Section 3 of 3)

Name	Data Type	Description
ColorsUsed ?	element	Array of colorant separation names that are requested. If not specified, the values are implied from ColorStandard. 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 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> Deprecated in JDF 1.2	[ISO12647-2:2004] offset standard.
<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> Deprecated in JDF 1.2	No marks. Used to define one-sided printing. Deprecation note: starting with JDF 1.2, use <i>LayoutIntent/@Sides</i> 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.

7.1.4.1 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 Coatings as SeparationSpec/@Name:</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. Spot 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:	<i>Any Product Intent Node</i>
Output of Processes:	—

Table 7-39: DeliveryIntent Resource (Section 1 of 4)

Name	Data Type	Description
<i>Accepted ?</i> Deprecated in JDF 1.3	boolean	<p>The quote that is specified by this DeliveryIntent has been accepted.</p> <p>Deprecation note: starting with JDF 1.3, contract negotiation information has been removed and will be handled by the business wrapper around JDF, e.g. PrintTalk.</p>
<i>AdditionalAmount = "1"</i> New in JDF 1.2 Deprecated in JDF 1.3	integer	<p>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 <i>DropIntent/DropItemIntent</i> and not to the total additional number of items.</p> <p>In JDF 1.3 and beyond, pricing information has been removed and will be handled by the business wrapper around JDF, e.g. PrintTalk.</p>

Table 7-39: DeliveryIntent Resource (Section 2 of 4)

Name	Data Type	Description
<i>BuyerAccount</i> ?	string	Account ID of the buyer with the delivery service.
<i>DeliveryCharge</i> ? New in JDF 1.1 Modified in JDF 1.2	EnumerationSpan	Specifies who pays for a delivery being made by a third party. 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 [@ <i>ContactTypes</i> = " <i>DeliveryCharge</i> "]. 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 Earliest or EarliestDuration 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 Earliest or EarliestDuration MUST be specified. Within a purchase order, EarliestDuration MUST NOT be specified.
Method ?	NameSpan	Specifies a delivery method, which can be a generic method. Values include: <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> <i>UPS</i> – a delivery service brand <i>DHL</i> – a delivery service brand <i>FedEx</i> – a delivery service brand
<i>Ownership</i> = "Origin"	enumeration	Point of transfer of ownership: Values are: <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.
<i>Pickup</i> ? Deprecated in JDF 1.1	boolean	Specifies whether the delivery brings or picks up the merchandise. If <i>Pickup</i> = " <i>false</i> ", the drop is delivered to the address specified in Company . If <i>Pickup</i> = " <i>true</i> ", 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.

Table 7-39: DeliveryIntent Resource (Section 3 of 4)

Name	Data Type	Description
Required ?	TimeSpan	Specifies the time by which the transfer is to be made. Within an RFQ or a Quote, exactly one of Required or RequiredDuration MUST be specified.
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 Required or RequiredDuration MUST be specified. Within a purchase order, RequiredDuration 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 SurplusHandling = "Return". Values are from: Method
ServiceLevel ? New in JDF 1.2	StringSpan	The service level of the specific carrier. Values include those from: ArtDeliveryIntent/ServiceLevel
SurplusHandling ? New in JDF 1.1	EnumerationSpan	Describes what is to happen with unused or redundant parts of the transfer specified with Transfer = "BuyerToPrinterDeliver" or "BuyerToPrinterPickup" after the Job. The return delivery or pickup address is specified by Contact[contains (@ContactTypes, "SurplusReturn")]. 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.

Table 7-39: DeliveryIntent Resource (Section 4 of 4)

Name	Data Type	Description
Transfer ? New in JDF 1.1	EnumerationSpan	Describes the direction and responsibility of the transfer. 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 (@ <i>ContactTypes</i> , "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 (@ <i>ContactTypes</i> , "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 (@ <i>ContactTypes</i> , "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 (@ <i>ContactTypes</i> , "Pickup")]. The Contact specifies where the buyer is to pick up the merchandise.
Underage ?	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> , "Delivery")]/ Address . The actual pickup address is specified as the Contact [contains (@ <i>ContactTypes</i> , "Pickup")]/ Address . At most one Contact [contains (@ <i>ContactTypes</i> , X)] MUST be specified for X equal to "Delivery", "Pickup" or "Billing",
DropIntent +	element	Includes all locations where the product will be delivered. Note that multiple DropIntent Elements specify multiple deliveries and not options for delivery.
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.

7.1.5.1 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 <i>DropIntent/DropItemIntent</i> and not to the total additional number of items. If not specified, defaults to the value of <i>DeliveryIntent/@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. <i>PrintTalk</i> .
<i>BuyerAccount</i> ? New in JDF 1.2	string	Account ID of the buyer with the delivery service. Default value is from: <i>DeliveryIntent/@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. Default value is from: <i>DeliveryIntentMethod</i>. Values include those from: <i>DeliveryIntentMethod</i>
<i>Pickup</i> ? Deprecated in JDF 1.1	boolean	If " <i>true</i> ", the merchandise is picked up. If <i>Pickup</i> = " <i>false</i> ", the <i>DropIntent</i> is delivered to the address specified in <i>Company</i> . If <i>Pickup</i> = " <i>true</i> ", the <i>DropIntent</i> describes an input to the Job, (e.g., a CD for inserting, a preprinted cover, etc.). In this case, <i>Company</i> describes the location where the merchandise is picked up.
<i>Required</i> ?	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.
<i>RequiredDuration</i> ?	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.
<i>ReturnMethod</i> ? 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> ". Default value is from: <i>DeliveryIntent/ReturnMethod</i>. Values include those from: <i>DeliveryIntent/ReturnMethod</i>
<i>ServiceLevel</i> ? New in JDF 1.2	StringSpan	The service level of the specific carrier. Default value is from: <i>DeliveryIntent/ServiceLevel</i>. Values include those from: <i>DeliveryIntent/ServiceLevel</i>
<i>SurplusHandling</i> ? New in JDF 1.1	Enumeration-Span	Describes what is to happen with unused or redundant parts of the transfer. Default value is from: <i>DeliveryIntentSurplusHandling</i>. Values are from: <i>DeliveryIntent/SurplusHandling</i>.

Table 7-40: DropIntent Element (Section 2 of 2)

Name	Data Type	Description
Transfer ? New in JDF 1.1	Enumeration-Span	Describes the direction and responsibility of the transfer. Default value is from: DeliveryIntent/Transfer. Values are: (see 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> , "Delivery")]/ Address . The actual pickup address is specified as the Contact [contains (@ <i>ContactTypes</i> , "Pickup")]/ Address . At most one Contact [contains (@ <i>ContactTypes</i> , X)] MUST be specified for X equal to "Delivery", "Pickup" or "Billing". Defaults to the DeliveryIntent/Contact
DropItemIntent +	element	A DropIntent MAY consist of multiple products, which are represented by their respective Physical 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.

7.1.5.2 Element: DropItemIntent

Table 7-41: DropItemIntent Element (Section 1 of 2)

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 <i>Pricing</i> . 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 PhysicalResource or 1 if this DropItemIntent 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 DropItemIntent refers to a proof that is specified in a ProofItem of the ProofingIntent of this Product Intent Node . Constraint: ProofingIntent/ProofItem/@ProofName MUST match <i>Proof</i> . Exactly one of PhysicalResource or <i>Proof</i> MUST be specified.
<i>Unit</i> ?	string	Unit of measurement for the <i>Amount</i> specified in the PhysicalResource . Default value is: value of <i>Unit</i> defined in the Resource described by the PhysicalResource

Table 7-41: DropltemIntent Element (Section 2 of 2)

Name	Data Type	Description
PhysicalResource ? Modified in JDF 1.1	refelement	Description of the individual item that is delivered. Constraint: exactly one of PhysicalResource or <i>Proof</i> MUST be specified. Note: PhysicalResource represents a Resource that MUST be an instance of a Physical Resource, (e.g., Component).
Pricing ? Deprecated in JDF 1.3	element	Pricing Element that defines the pricing of the DropltemIntent. Deprecation note: starting with JDF 1.3, pricing information has been removed and will be handled by the business wrapper around JDF, e.g. Print-Talk.

7.1.5.3 Element: Pricing

[Deprecated in JDF 1.3](#)

The table defining the deprecated Pricing Subelement has been moved to "DeliveryIntent Deprecated Subelements" on page 1022.

7.1.5.4 Element: Payment

[Deprecated in JDF 1.3](#)

The table defining the deprecated Payment Subelement has been moved to "DeliveryIntent Deprecated Subelements" on page 1022.

7.1.5.5 Element: CreditCard

[Deprecated in JDF 1.3](#)

The table defining the deprecated CreditCard Subelement has been moved to "DeliveryIntent Deprecated Subelements" on page 1022.

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:	<i>Any Product Intent Node</i>
Output of Processes:	—

Table 7-42: EmbossingIntent Resource

Name	Data Type	Description
EmbossingItem +	element	Each embossed image is described by one EmbossingItem.

7.1.6.1 Element: EmbossingItem

Table 7-43: EmbossingItem Element

Name	Data Type	Description
Direction Modified in JDF 1.3	EnumerationSpan	The direction of the image. 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.
EdgeAngle ?	NumberSpan	The angle of a beveled edge in degrees. Typical values are an angle of: 30, 40, 45, 50 or 60 degrees. If EdgeAngle is specified, EdgeShape = "Beveled" MUST be specified.
EdgeShape ?	EnumerationSpan	The transition between the embossed surface and the surrounding media can be rounded or beveled (angled). Values are: <i>Rounded</i> <i>Beveled</i>
EmbossingType Modified in JDF 1.4	StringSpan	The strings defined in EmbossingType are whitespace separated combinations of the following tokens. Values include: <i>Braille</i> – 6 dot Braille embossing. New in JDF 1.4 <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.
FoilColor ?	EnumerationSpan	Defines the color of the foil material which is used for embossing. Values are from: Table A-3, "NamedColor Enumeration Values" on page 870.
FoilColorDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If FoilColorDetails is supplied, FoilColor SHOULD also be supplied.
Height ?	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 Direction.
ImageSize ?	XYPairSpan	The size of the bounding box of one single image.
Level ?	EnumerationSpan	The level of embossing. Values are: <i>SingleLevel</i> <i>MultiLevel</i> <i>Sculpted</i>
Position ?	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 540.

Resource Properties

Resource Class:	Intent
Resource referenced by:	—
Process Resource Pairing:	CreasingParams, CuttingParams, Fold, FoldingParams, PerforatingParams
Example Partition:	Option
Input of Processes:	Any Product Intent Node
Output of Processes:	—

Table 7-44: FoldingIntent Resource

Name	Data Type	Description
FoldingCatalog	NameSpan	Describes the folding scheme according to the folding catalog in Figure 7-31, "Fold catalog part 1" on page 542 and Figure 7-32, "Fold catalog part 2" on page 543. Value format is: "Fn-i" where "n" is the number of finished pages and "i" is either an integer, which identifies a particular fold or the letter "X", which identifies a generic fold. E.g., "F6-2" describes a Z-fold of 6 finished pages, and "F6-X." describes a generic fold with 6 finished pages. Note: The folding scheme in this context refers to the folding of the finished product as seen after the cutting, not the folding, of the Sheet as seen in production. See LayoutIntent/@Foliocount for a discussion of pagination of folded end products.
Folds? Deprecated in JDF 1.1	XYPair	Number of folds in x and in y direction. This Attribute specifies the number of folds seen in the Sheet after folding, and not the number of fold operations needed to achieve that result. If not specified, it MUST be inferred from <i>FoldingCatalog</i> . If X and Y are the number of folds in the x and y directions, respectively, the product 2*(X+1)*(Y+1) 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:	Option
Input of Processes:	Any Product Intent Node
Output of Processes:	—

Table 7-45: HoleMakingIntent Resource

Name	Data Type	Description
<i>Extent</i> ? New in JDF 1.2	XYPair	Size (bounding box) of the hole in points when specifying a standard hole pattern in HoleType. If not specified the implied default defined in "JDF/CIP4 Hole Pattern Catalog" on page 929 is assumed. Ignored when HoleType/@Actual = "Explicit".
<i>HoleReferenceEdge</i> = "Left" New in JDF 1.1	enumeration	The edge of the media relative to where the holes are to be punched. Use with HoleType. 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 HoleType in "JDF/CIP4 Hole Pattern Catalog" on page 929 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). Values include: <i>Explicit</i> – Holes are defined in an array of Hole Elements. <i>2HoleEuro</i> – Replace by either R2m-DIN or R2m-ISO Deprecated in JDF 1.0 . <i>3HoleUS</i> – Replace by R3I-US Deprecated in JDF 1.0 <i>4HoleEuro</i> – Replace by R4m-DIN-A4 or R4m-DIN-A5 Deprecated in JDF 1.0 . Values are from: "JDF/CIP4 Hole Pattern Catalog" on page 929.
HoleList ?	element	Array of all Hole Elements. Used only when HoleType/@Actual = "Explicit", 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:	Option
Input of Processes:	Any Product Intent Node
Output of Processes:	—

Table 7-46: InsertingIntent Resource

Name	Data Type	Description
GlueType ?	EnumerationSpan	Glue used to fasten the insert. Values are: <i>Permanent</i> <i>Removable</i>
Method ?	EnumerationSpan	Values are: <i>BindIn</i> – Apply glue to fasten the insert <i>BlowIn</i> – Loose insert.
InsertList	element	List of individual inserts.

7.1.9.1 Element: InsertList

Table 7-47: InsertList Element

Name	Data Type	Description
Insert *	element	Individual insert description.

7.1.9.2 Element: Insert

Table 7-48: Insert Element (Section 1 of 2)

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/@Actual</i> = " <i>BlowIn</i> " and by the page that the glue is applied in case <i>Method/@Actual</i> = " <i>BindIn</i> ". In general, a list of Folios will only be supplied for <i>Method/@Actual</i> = " <i>BlowIn</i> ". The pages are counted in the order, which is described in <i>FolioCount</i> of the parent Component/Bundle .
GlueType ?	EnumerationSpan	Glue used to fasten the insert. Default value is from: InsertingIntent /GlueType. Values are: <i>Removable</i> <i>Permanent</i>
Method ?	EnumerationSpan	Inserting method. Default value is from: InsertingIntent /Method. 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 Component to be inserted and finished page identified by <i>Folio</i> 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.

Table 7-48: Insert Element (Section 2 of 2)

Name	Data Type	Description
WrapPages ? New in JDF 1.1	IntegerRangeList	List of finished pages of the cover that wrap around an <code>Insert</code> 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 <code>GlueLine</code> Elements used to glue in the insert. MUST NOT be specified in conjunction with <code>GlueType</code> .

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:	<code>LaminatingParams</code>
Example Partition:	<i>Option</i>
Input of Processes:	<i>Any Product Intent Node</i>
Output of Processes:	—

Table 7-49: LaminatingIntent Resource

Name	Data Type	Description
Laminated ? Deprecated in JDF 1.1	OptionSpan	If " <code>true</code> ", the product is laminated. If no <code>LaminatingIntent</code> is specified, the product MUST NOT be laminated.
Temperature ? Modified in JDF 1.3	EnumerationSpan	Temperature used in the <i>Laminating</i> Process. Values are: <i>Hot</i> <i>Cold</i>
Surface ?	EnumerationSpan	The surface to be laminated. Values are: <i>Front</i> <i>Back</i> <i>Both</i>
Texture ? New in JDF 1.3	NameSpan	The intended texture of the laminate. Values 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.
Thickness ?	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:	Option
Input of Processes:	Any Product Intent Node
Output of Processes:	—

Table 7-50: LayoutIntent Resource (Section 1 of 4)

Name	Data Type	Description
Dimensions ? 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, Dimensions for a Z-fold is the unfolded dimensions, while FinishedDimensions is the folded dimensions if known. Use Dimensions if FinishedDimensions is not known. The Dimensions Span Element is provided for the rare case that <i>FinishedDimensions</i> does not unambiguously define the finished product, due to complex folding schemes. If both values are specified, FinishedDimensions takes precedence
FinishedDimensions ? New in JDF 1.1	ShapeSpan	Specifies the width (X), height (Y) and depth (Z) in points, respectively, of the finished product Component after all finishing operations, including folding, trimming, etc. If the Z coordinate is 0, it is ignored. Only FinishedDimensions SHOULD be specified if both FinishedDimensions and Dimensions are known. Compatibility Warning. In JDF 1.1 height and width were erroneously switched in the description.
FinishedGrainDirection ? New in JDF 1.2	Enumeration-Span	Specifies the media grain direction of the finished page with respect to the binding edge. 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>l</u>ToBind</i>

Table 7-50: LayoutIntent Resource (Section 2 of 4)

Name	Data Type	Description
<p><i>FinishedPageOrientation?</i> Deprecated in JDF 1.1</p>	enumeration	<p>Indicates the desired orientation of the finished media.</p> <p>Values are:</p> <p><i>Portrait</i> – The short edges of the media are the top and bottom.</p> <p><i>Landscape</i> – The long edges of the media are the top and bottom.</p> <p>Note: In JDF 1.1, the finished page orientation is implied by the value of Dimensions and FinishedDimensions. If height (X) > width (Y), the product is portrait.</p>
<p><i>FolioCount</i> = "Booklet" New in JDF 1.1</p>	enumeration	<p>Defines the method used when counting finished pages.</p> <p>Values are:</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 7-31 and Figure 7-32) 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>

Table 7-50: LayoutIntent Resource (Section 3 of 4)

Name	Data Type	Description
<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>Pages multiplied with Dimensions then divided by two (2) identifies the amount of paper that is used in the product. Pages describes the paper usage regardless of document layout. This value MUST be an even number. For example, the value for Pages for a two-sided booklet with seven Reader Pages would be "8", whether the booklet were saddle stitched or glued.</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 Pages is used as the default. For example, if there are ten finished pages, in which three are identical, <i>PageVariance/@Actual="8"</i> since it would take eight master copies to produce the product.</p>
<p><i>RotatePolicy ?</i> New in JDF 1.2</p>	enumeration	<p>Specifies the policy to automatically rotate the image to optimize the fit of the image to the page container. For instance, individual landscape pages in a portrait document MAY automatically be rotated. The page container is one cell on the NUp grid of the Media defined in Dimensions or FinishedDimensions.</p> <p>Values are:</p> <p><i>NoRotate</i> – Do not rotate.</p> <p><i>RotateOrthogonal</i> – Rotate by 90° in either direction.</p> <p><i>RotateClockwise</i> – Rotate clockwise by 90°.</p> <p><i>RotateCounterClockwise</i> – Rotate counter-clockwise by 90°.</p>
<p><i>Sides ?</i> Modified in JDF 1.2</p>	enumeration	<p>Indicates whether contents are to be printed on one or both sides of the media.</p> <p>Values are:</p> <p><i>OneSided</i> – Page contents will only be imaged on the front side of the media.</p> <p><i>OneSidedBack</i> – Page contents will only be imaged on the back side of the media. New in JDF 1.2</p> <p><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.</p> <p><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.</p>

Table 7-50: LayoutIntent Resource (Section 4 of 4)

Name	Data Type	Description
SizePolicy ? New in JDF 1.2	Enumeration-Span	Allows printing even if the container size defined in Dimensions or FinishedDimensions does not match the requirements of the data. The page container is one cell on the NUp grid of the Media defined in Dimensions or FinishedDimensions. Values are: <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:	<i>Any Product Intent Node</i>
Output of Processes:	—

Table 7-51: MediaIntent Resource (Section 1 of 7)

Name	Data Type	Description
BackCoatings ?	EnumerationSpan	Identical to FrontCoatings, but applied to the back surface of the media. Default value is from: @FrontCoatings. Values are from: @FrontCoatings.
Brightness ?	NumberSpan	Reflectance percentage of diffuse blue reflectance as defined by [ISO2470:1999]. The reflectance is reported per [ISO2470:1999] as the diffuse blue reflectance factor of the paper or board in percent to the nearest 0.5% reflectance factor.

Table 7-51: MediaIntent Resource (Section 2 of 7)

Name	Data Type	Description
BuyerSupplied ?	OptionSpan	Indicates whether the customer will supply the media. Note that the Media Resource can be used to specify additional media requirements, particularly when the media is supplied by the customer.
Dimensions ? Deprecated in JDF 1.2	XYPairSpan	Specifies the size of the supplied media in points if BuyerSupplied evaluates to "true". Dimensions MUST be ignored if BuyerSupplied evaluates to "false". Note that the size of the finished product is always specified in LayoutIntent/FinishedDimensions . In JDF 1.2 and beyond the specifics of BuyerSupplied media SHOULD be specified using a Media Resource. The dimensions of the finished product are specified with LayoutIntent/Dimensions or LayoutIntent/FinishedDimensions .
Flute ? New in JDF 1.4	NameSpan	Single, capital letter that specifies the Flute type of corrugated media. Although the classification of flutes using a letter code "A", "B", etc., are used very frequently e.g., in the specification of the order for a box, there seems to be no agreement on the exact numerical specification of those categories. Slightly varying numbers for flute size and frequency can be found between regions (European versus US) and between vendors. Values include those from: Media/@Flute
FluteDirection ? New in JDF 1.4	EnumerationSpan	Direction of the fluting. Values are the same as Media/@FluteDirection with slightly different description. Values are: <i>LongEdge</i> – Along the longer axis as defined by LayoutIntent/Dimensions . <i>ShortEdge</i> – Along the shorter axis as defined by LayoutIntent/Dimensions . <i>XDirection</i> – Along the X-axis of the LayoutIntent coordinate system <i>YDirection</i> – Along the Y-axis of the LayoutIntent coordinate system
FrontCoatings ? Modified in JDF 1.2	EnumerationSpan	What pre-process coating has been applied to the front surface of the media. 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 3 of 7)

Name	Data Type	Description
Grade ?	IntegerSpan	<p>The intended grade of the media on a scale of 1 through 5. Grade is ignored if <i>MediaType/@Actual</i> is not "Paper". Grade of paper material is defined in accordance with the paper "types" set forth in [ISO12647-2:2004]. Offset printing paper types are defined with integer values:</p> <p>Note: [ISO12647-2:2004] paper type <i>AdditionalPrice</i> values do not align with U.S. GRACOL paper grade <i>AdditionalPrice</i> values, (i.e., [ISO12647-2:2004] type 1 does not equal U.S. GRACOL grade 1.)</p> <p>Values include:</p> <ul style="list-style-type: none"> 1 – Gloss-coated paper 2 – Matt-coated paper 3 – Gloss-coated, Web paper 4 – Uncoated, white paper 5 – Uncoated, yellowish paper
GrainDirection ? New in JDF 1.2 Modified in JDF 1.4	EnumerationSpan	<p>Direction of the grain in the coordinate system defined by LayoutIntent/Dimensions or LayoutIntent/FinishedDimensions.</p> <p>Values are:</p> <p><i>ShortEdge</i> – Parallel to the shorter axis of the finished page.</p> <p><i>LongEdge</i> – Parallel to the longer axis of the finished page.</p> <p><i>XDirection</i> – Along the X-axis of the LayoutIntent coordinate system New in JDF 1.4</p> <p><i>YDirection</i> – Along the Y-axis of the LayoutIntent coordinate system. New in JDF 1.4</p>
HoleCount ? Deprecated in JDF 1.1	IntegerSpan	<p>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.</p>
HoleType ? New in JDF 1.1	StringSpan	<p>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.)</p> <p>Values include:</p> <p><i>None</i> – no holes</p> <p>Values include those from: "JDF/CIP4 Hole Pattern Catalog" on page 929</p>
MediaColor ?	EnumerationSpan	<p>Color of the media. If more-specific, specialized or site-specific media color names are needed, use <i>MediaColorDetails</i>.</p> <p>Values are from: Table A-3, "NamedColor Enumeration Values" on page 870.</p>

Table 7-51: MediaIntent Resource (Section 4 of 7)

Name	Data Type	Description
MediaColorDetails ? New in JDF 1.2	StringSpan	A more specific, specialized or site-defined name for the media color. If MediaColorDetails is supplied, MediaColor SHOULD also be supplied. Note that there is a one-to-many relationship between entries in MediaColor and MediaColorDetails, (e.g., MediaColorDetails values of <i>Burgundy</i> and <i>Ruby</i> both correspond to a MediaColor of <i>DarkRed</i>).
MediaQuality ? New in JDF 1.4	StringSpan	Named quality description of the media. For folding carton quality, multiple named quality description systems are in use. E.g. GC1, SBB, etc. For an overview see http://www.procarton.com/files/fact_file_6.pdf .
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>UserMediaType</i> is "PreCutTabs", a <i>MediaSetCount</i> of 5 would indicate that each set includes 5 tab Sheets.
MediaType ? New in JDF 1.1 Modified in JDF 1.3	EnumerationSpan	Describes the medium being employed. 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 MediaTypeDetails 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 MediaType. 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. Values include those from: Media/@MediaTypeDetails Note: values from Media/@MediaTypeDetails are RECOMMENDED. However, some Process related values, such as <i>DryFilm</i> , SHOULD NOT be used for this Attribute.
MediaUnit ? Deprecated in JDF 1.2	EnumerationSpan	Describes the format of the media as it is delivered to the Device. Values are: <i>Roll</i> <i>Sheet</i> Deprecation note: deprecated because Intent Attributes and Span Elements pertain to finished product, not the raw media format. If BuyerSupplied = "true", then the Media Resource can be used to provide this Span Element.

Table 7-51: MediaIntent Resource (Section 5 of 7)

Name	Data Type	Description
Opacity ? Modified in JDF 1.2	EnumerationSpan	The opacity of the media. See OpacityLevel to specify the degree of opacity for any of these values. 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 RecycledPercentage.
RecycledPercentage ? New in JDF 1.2	NumberSpan	The percentage, between 0 and 100, of recycled material that the media is expected to contain.
StockBrand ?	StringSpan	Strings providing available brand names. The customer might know exactly what paper is to be used. Example is "Lustro" or "Warren Lustro" even though the manufacturer name is included.
StockType ?	NameSpan	Strings describing the available stock. Values include those from: Media/@StockType
Texture ?	NameSpan	The intended texture of the media. Values include those from: Media/@Texture.
Thickness ? New in JDF 1.1	NumberSpan	The thickness of the chosen medium. Measured in microns [µm].

Table 7-51: MediaIntent Resource (Section 6 of 7)

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.</p> <p>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>
USWeight ? Deprecated in JDF 1.2	NumberSpan	The intended weight of the media, measured in pounds per ream of basis size. At most one of Weight or USWeight MUST be specified. If known, Weight SHOULD be specified in grammage (g/m ² .) In JDF 1.2 and beyond, use Weight.
Weight ?	NumberSpan	The intended weight of the media, measured in grammage (g/m ²) of the media. See "North American and Japanese Media Weight Explained" on page 895 for an explanation of how to calculate the US weight from the grammage for different stock types.

Table 7-51: MediaIntent Resource (Section 7 of 7)

Name	Data Type	Description
MediaLayers ? New in JDF 1.4	element	Subelement describing the layer structure of media such as corrugated or self adhesive materials.

7.1.13 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:	<i>Any Product Intent Node</i>
Output of Processes:	—

Table 7-52: NumberingIntent Resource

Name	Data Type	Description
ColorName ?	EnumerationSpan	Defines the color of the numbering. Values are from: Table A-3, “NamedColor Enumeration Values” on page 870.
ColorNameDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If ColorNameDetails is supplied, ColorName SHOULD also be supplied.
ColorPool ?	refelement	Additional details about the colors used.
NumberItem +	element	Individual position of the numbers on the finished page.

7.1.13.1 Element: NumberItem

Table 7-53: NumberItem Element

Name	Data Type	Description
ColorName ?	EnumerationSpan	Defines the color of the numbering. Default value is from: NumberingIntent/@ColorName. Values are from: Table A-3, “NamedColor Enumeration Values” on page 870
ColorNameDetails ? New in JDF 1.4	StringSpan	A more specific, specialized or site-defined name for the color. If ColorNameDetails is supplied, ColorName SHOULD also be supplied.
Orientation?	NumberSpan	Rotation of the numbering machine in degrees. If Orientation/@Actual = 0, the top of the numbers is along the leading edge.
StartValue = "1"	string	First value of the numbering machine.
Step = "1"	integer	Number that specifies the difference between two subsequent numbers of the numbering machine.
XPosition ?	NumberSpan	Position of the number in the X direction of the product.
YPosition ?	NumberSpan	Position of the number in the Y direction of the product.
SeparationSpec ?	element	Specifies the name of the Color in the ColorPool that is used for Numbering.

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:	<i>Any Product Intent Node</i>
Output of Processes:	—

Table 7-54: PackingIntent Resource (Section 1 of 2)

Name	Data Type	Description
BoxedQuantity ?	IntegerSpan	How many units of <i>product</i> in a box.
BoxShape ?	ShapeSpan	Describes the length, width and height of the box, in points.
CartonQuantity ?	IntegerSpan	How many units of <i>product</i> in a carton.
CartonShape ?	ShapeSpan	Describes the length, width and height of the carton, in points. For example, 288 544 1012
CartonMaxWeight ?	NumberSpan	Maximum weight of an individual carton, in kilograms.
CartonStrength ?	NumberSpan	Strength of the carton, in kilograms.
FoldingCatalog ?	NameSpan	Describes the folding scheme for folding the product for packaging as specified in the folding catalog in Figure 7-31, “Fold catalog part 1” on page 542 and Figure 7-32, “Fold catalog part 2” on page 543. Value format is: “ <i>F_n-i</i> ” where “n” is the number of finished pages and “i” is either an integer, which identifies a particular fold or the letter “X”, which identifies a generic fold. E.g., “ <i>F6-2</i> ” describes a Z-fold of 6 finished pages, and “ <i>F6-X</i> .” describes a generic fold with 6 finished pages. Note: The folding scheme in this context refers to the folding of the finished product for packaging only. The folding has no effect on the page/Folio definition.
PalletCornerBoards ? New in JDF 1.3	NameSpan	Additional protective corner boards for packaging on a pallet: Values include: <i>Corners:</i> Corner boards on 8 corners of the pallet. <i>VerticalEdge:</i> Corner boards along the 4 vertical edges.
PalletQuantity ?	IntegerSpan	Number of <i>product</i> per pallet
PalletSize ?	XYPairSpan	Describes the length and width of the pallet, in points, (e.g., “3500 3500”).
PalletMaxHeight ?	NumberSpan	Maximum height of a loaded pallet, in points.
PalletMaxWeight ?	NumberSpan	Maximum weight of a loaded pallet, in kilograms.

Table 7-54: PackingIntent Resource (Section 2 of 2)

Name	Data Type	Description
PalletType ?	NameSpan	Type of pallet used. Values include: <i>2Way</i> – Two-way entry <i>4Way</i> – Four-way entry <i>Euro</i> – Standard 1*1 m Euro pallet
PalletWrapping ?	NameSpan	Wrapping of the completed pallet. Values include: <i>Banding</i> <i>None</i> – explicitly requests no wrapping. <i>StretchWrap</i>
WrappedQuantity ?	IntegerSpan	Number of units of product per wrapped package.
WrappingMaterial ?	NameSpan	Values 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:	<i>Any Product Intent Node</i>
Output of Processes:	—

Table 7-55: ProductionIntent Resource (Section 1 of 2)

Name	Data Type	Description
PrintPreference ?	EnumerationSpan	Intended result or goal. 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.

Table 7-55: ProductionIntent Resource (Section 2 of 2)

Name	Data Type	Description
PrintProcess ? Modified in JDF 1.3	NameSpan	Print Process requested. Values include: <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> Modification Note: starting with JDF 1.3, the data type of PrintProcess is expanded from EnumerationSpan to NameSpan.
Resource * New in JDF 1.3	refelement	Any production Resources that are provided by the customer. Some examples include buyer specified media or ink or specific parameter setups. Note that DeliveryIntent MUST be specified for any PhysicalResource Elements 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: *Option*

Input of Processes: *Any Product Intent 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.

7.1.16.1 Element: ProofItem

All parameters of **ProofingIntent** have been moved into ProofItem in JDF 1.1

Table 7-57: ProofItem Element (Section 1 of 3)

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".

Table 7-57: ProofItem Element (Section 2 of 3)

Name	Data Type	Description
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. 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. Values are: <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	Index list of pages that are to be proofed in the ArtDeliveryIntent/RunList/PageList . If no range is specified then all pages MUST 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> .
ProofType? Modified in JDF 1.1	EnumerationSpan	The kind of proof. Values are: <i>Page</i> – Page proof <i>Imposition</i> – Imposition proof <i>None</i> – No proof is needed.

Table 7-57: ProofItem Element (Section 3 of 3)

Name	Data Type	Description
Technology ? Modified in JDF 1.1	NameSpan	Technology used for making the proof. Values include: <i>BlueLine</i> <i>DyeSub</i> <i>InkJet</i> <i>Laser</i> <i>PressProof</i> <i>SoftProof</i>
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:	<i>Any Product Intent Node</i>
Output of Processes:	—

Table 7-58: PublishingIntent Resource

Name	Data Type	Description
<i>ContentDataRefs</i> ? New in JDF 1.4	IDREFS	IDs of ContentData Elements in the referenced ContentList . ContentData Elements provide metadata related to the product to be published. <i>ContentDataRefs</i> MUST NOT be specified if no ContentList is specified.
IssueDate	TimeSpan	Publication date of the issue.
IssueName	StringSpan	The name of a the publication.
IssueType Modified in JDF 1.4	NameSpan	Defines the product type of the issue. Values include: <i>Magazine</i> – The publication is a magazine <i>Newspaper</i> – The publication is a newspaper <i>Supplement</i> – The publication is a supplement to a magazine or newspaper. New in JDF 1.4 .
Circulation ?	IntegerSpan	Specifies the number of copies to be published.
ContentList ? New in JDF 1.4	refelement	ContentList with additional metadata.

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 Intent Node
Output of Processes:	—

Table 7-59: ScreeningIntent Resource

Name	Data Type	Description
DotSize ?	NumberSpan	Specifies the dot size of the screen in microns [μm] when FM screening is used, otherwise DotSize is ignored.
Frequency ?	NumberSpan	Specifies the line frequency of the screen in lines per inch (lpi) when AM screening is used, otherwise Frequency is ignored.
FrequencySelection ?	EnumerationSpan	Selects the AM or FM frequency range. 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
ScreeningType ?	EnumerationSpan	General type of screening. 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 Intent 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.

7.1.19.1 Element: ShapeCut

Table 7-61: ShapeCut Element

Name	Data Type	Description
<i>CutBox?</i>	rectangle	Specification of a rectangular window. (See Section A.2.32, rectangle 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.
<i>Material?</i>	StringSpan	Transparent material that fills a shape (e.g., an envelope window) that was cut out when <i>CutOut = "true"</i> .
<i>CutType?</i> Modified in JDF 1.1	EnumerationSpan	Type of cut or perforation used. 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. Values are: <i>Path</i> <i>Rectangular</i> <i>Round</i> <i>RoundedRectangle</i> – Rectangle with rounded corners. New in JDF 1.3
<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 3.8.5, *Resource Classes*, all Resources are derived from one of the following seven superclasses: *Intent*, *Parameter*, *Implementation*, *Consumable*, *Quantity*, *Handling* and *Placeholder*. All Resources inherit additional contents (i.e., zero or more Attributes or zero or more Elements) from their respective

superclasses, and those Attributes and Elements are not repeated in this section. Thus those Attributes associated with a Resource of Class *Parameter*, for example, can be found in Table 3-10, “Abstract Resource Element” on page 63. Note that this inheritance is only valid for atomic Resources, (i.e., Resources that reside directly in a ResourcePool).

Resource Elements are listed in separate sections if they can be referenced by more than one Resource. For an example, see the Resource Element **SeparationSpec**. If the Resource is not referenced by multiple Resources, it is described inside the Resource section of the Resource to which it belongs. For example, see the Structure of the BundleItem Element of the Bundle Resource. If an Element inside a Resource section of the Resource is needed to be referenced by multiple Resources in a revision of JDF, then that Element is promoted to its own section. For example, **ColorSpaceConversionOp** was a Subelement of **ColorSpaceConversionParams** in JDF/1.1. The Resource Class of an atomic Resource also defines the superclasses from which the Resource inherits additional contents. The Consumable Resource, Quantity Resource and Handling Resource inherit from the Physical Resource Element, which in turn inherits from the Resource Element. The Parameter Resource 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 7.2.72, *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 Partition Keys: For a complete list of Partition Keys, see the description of PartIDKeys in Table 3-27, “Partitionable Resource Element” on page 103. 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. Note: the “element” data type means that the specified Element MUST be an in-line Subelement within the Resource.</i>

Table 7-62: Template for Process Resource (Section 2 of 2)

Name	Data Type	Description
<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 3.10.2, 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</i>) <i>Cardinality</i>	refelement	<i>Information about the FileSpec Resource</i> Note: FileSpec/@ResourceUsage MUST match the “ <i>someValue</i> ” value specified in the parentheses. When a Resource potentially contains multiple FileSpec children, the value of FileSpec@ResourceUsage distinguishes the FileSpec Resources.
<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:	Location, Contact , Person
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. Values include those from: [ISO3166-1:1997] Note: 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 1024 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:	ProofItem, ConventionalPrintingParams, DigitalPrintingParams
Example Partition:	—
Input of Processes:	<i>Approval</i>
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.
MinApprovals = "1" New in JDF 1.2	integer	Minimum number of ApprovalPerson[@ ApprovalRole = "Group"] whose associated person MUST sign the ApprovalSuccess for the ApprovalSuccess to be Available.

7.2.4.1 Element: ApprovalPerson

Table 7-65: ApprovalPerson Element

Name	Data Type	Description
Obligated? Deprecated in JDF 1.2	boolean	If "true", the person has to sign this approval. In JDF 1.2 and beyond, use ApprovalRole.
ApprovalRole = "Obligated" New in JDF 1.2 Modified in JDF 1.3	enumeration	Role of the ApprovalPerson. Values are: <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 <i>Approval</i> Process, but the approval is still valid, even without his approval. <i>Obligated</i> – The approver MUST sign the approval.
ApprovalRoleDetails? New in JDF 1.3	string	Additional details on the ApprovalRole.
Contact	refelement	Contact (e.g., a customer, printer or manager) who MUST sign the approval. There MUST be a Contact[contains (@ContactTypes, "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

Name	Data Type	Description
ApprovalDetails * New in JDF 1.3	element	Container for details about the decision for each approver.
Contact * New in JDF 1.2 Deprecated in JDF 1.3	refelement	List of contacts that have signed off on this approval. Use ApprovalDetails/Contact in JDF 1.3 and above.
FileSpec ? Deprecated in JDF 1.3	refelement	The file that contains the approval signature. If FileSpec does not exist, ApprovalSuccess is a logical placeholder. Use ApprovalDetails/FileSpec in JDF 1.3 and above.

7.2.5.1 Element: ApprovalDetails

[New in JDF 1.3](#)

Table 7-67: ApprovalDetails Element

Name	Data Type	Description
ApprovalState	enumeration	Decision made by the approver specified in this ApprovalDetails/Contact. 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.
ApprovalStateDetails ?	string	Additional details on the decision made by the approver are specified in this ApprovalDetails/Contact. This value provides additional machine readable details of ApprovalState. Hand written comments and notes MAY be specified in ApprovalDetails/Comment or ApprovalDetails/@CommentURL.
Contact ?	refelement	Contact that signed off on this approval.
FileSpec ?	refelement	The file that contains the approval signature. If FileSpec does not exist, ApprovalSuccess is a logical placeholder.

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: Component, CutBlock, PageList
 Example Partition: —
 Input of Processes: *Collecting, Gathering, Stripping, WebInlineFinishing*
 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.
<i>AssemblyIDs?</i> New in JDF 1.3	NMTOKENS	Identification of the Assembly Elements if <i>Stripping</i> describes an imposition scheme for multiple Assembly Elements. <i>AssemblyIDs</i> MAY contain multiple NMTOKENS, when the Assembly Resource specifies an intermediate product that contains multiple final assemblies.
<i>BindingSide = "Left"</i>	enumeration	Indicates which side is to be bound. <i>BindingSide</i> is ignored when <i>Order = "None"</i> . Values are: <i>Left</i> <i>Right</i> <i>Top</i> <i>Bottom</i>
<i>JobID?</i>	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.
<i>JogSide = "Top"</i> New in JDF 1.3	enumeration	<i>JogSide</i> specifies the side on which the AssemblySection Elements will be aligned. Values are: <i>Left</i> <i>Right</i> <i>Top</i> <i>Bottom</i> <i>None</i>
<i>Order = "Gathering"</i>	enumeration	Ordering of the individual AssemblySection Elements. Order specifies the topology of the final Assembly . Values are: <i>Collecting</i> – The sections are placed within one another. The first section is on the outside. An example is a saddle-stitched brochure. See Section 6.6.10, "Collecting" on page 318. <i>Gathering</i> – The sections are placed on top of one another. The first section is on the top. An example is a perfect bound magazine. See Section 6.6.20, "Gathering" on page 323. <i>None</i> – The sections are not bound. Typically used for flatwork Jobs. <i>List</i> – More complex ordering of the sections. If multiple child AssemblySection Elements are provided, these are gathered on top of one another. The first AssemblySection is on the top. If nested AssemblySection Elements are provided, these are Collected into each other. The first AssemblySection is on the outside.

Table 7-68: Assembly Resource (Section 2 of 2)

Name	Data Type	Description
PhysicalSection ? New in JDF 1.3	IntegerList	Specifies the physical structure of a newspaper. The structure is based on a broadsheet production. For instance, <i>PhysicalSection</i> = "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 <i>Order</i> = "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 <i>Order</i> = "List".

7.2.6.1 Element: AssemblySection

Table 7-69: AssemblySection Element (Section 1 of 2)

Name	Data Type	Description
AssemblyID ? Deprecated in JDF 1.3	string	Identification of the AssemblySection if <i>Stripping</i> produces a multi-section Assembly. If not specified, it defaults to the value specified or implied in the parent Assembly or AssemblySection.
AssemblyIDs ? New in JDF 1.3	NMTOKENS	Identification of the AssemblySection Elements if <i>Stripping</i> describes an imposition scheme for a multi-section Assembly. If not specified, it defaults to the value specified or implied in the parent Assembly or AssemblySection. In general AssemblySection/@AssemblyIDs will contain only a single NMTOKEN value. AssemblyIDs MAY contain multiple NMTOKENS, when the AssemblySection specifies an intermediate product that contains multiple final products.
JobID ?	string	Identification of the original Job the AssemblySection belongs to. If not specified, it defaults to the value specified or implied in the parent Assembly or AssemblySection.
Order = "Gathering" Deprecated in JDF 1.4	enumeration	Ordering of the child AssemblySection Elements. Values are: <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. Deprecation note: starting with JDF 1.4, Sibling AssemblySection Elements are gathered whereas Child AssemblySection Elements are collected. Thus the Relationship of the AssemblySection Elements directly reflects the structure of the Assembly.
AssemblySection *	element	Additional child AssemblySection Elements which are collected to create this AssemblySection.

Table 7-69: AssemblySection Element (Section 2 of 2)

Name	Data Type	Description
PageAssignedList * 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.

7.2.6.2 Element: PageAssignedList[New in JDF 1.3](#)

PageAssignedList specifies the metadata related to assigned pages.

Table 7-70: PageAssignedList Element

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.
<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, PagePlacementName specifies an identifier for a single page on a Web-product level. Therefore, different PagePlacementName values might be assigned to one single PageList/PageData Element.

Example 7-2: Perfect Bound (Gathering)[New in JDF 1.4](#)

Cover wrapped around a perfect bound (gathering) body

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Assembly BindingSide="Left" Class="Parameter" ID="ASM000" Order="List"
      Status="Available">
      <AssemblySection AssemblyIDs="Ass_Cover" >
        <AssemblySection AssemblyIDs="Ass_Body1" />
        <AssemblySection AssemblyIDs="Ass_Body2" />
        <AssemblySection AssemblyIDs="Ass_Insert" />
        <AssemblySection AssemblyIDs="Ass_Body3" />
        <AssemblySection AssemblyIDs="Ass_Body4" />
      </AssemblySection>
    </Assembly>
  </ResourcePool>
  <ResourceLinkPool>
    <AssemblyLink Usage="Input" rRef="ASM000" />
  </ResourceLinkPool>
</JDF>
```

Example 7-3: Saddle-Stitched Brochure (Collecting)[New in JDF 1.4](#)

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Assembly BindingSide="Left" Class="Parameter" ID="ASM000" Order="List"
      Status="Available">
      <AssemblySection AssemblyIDs="Ass_Cover" >
        <AssemblySection AssemblyIDs="Ass_Body1" >
          <AssemblySection AssemblyIDs="Ass_Body2" >
            <AssemblySection AssemblyIDs="Ass_Body3" >
              <AssemblySection AssemblyIDs="Ass_Body4" >
                </AssemblySection>
              </AssemblySection>
            </AssemblySection>
          </AssemblySection>
        </AssemblySection>
      </AssemblySection>
    </Assembly>
  </ResourcePool>
  <ResourceLinkPool>
    <AssemblyLink Usage="Input" rRef="ASM000" />
  </ResourceLinkPool>
</JDF>

```

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 Type	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:	<i>ElementColorParams</i> , <i>RenderingParams</i> , <i>SeparationControlParams</i>
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-72: AutomatedOverPrintParams Resource

Name	Data Type	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 text is applied regardless of any values in the PDL. If "false", <i>TextSizeThreshold</i> and <i>TextBlackLevel</i> are ignored and PDL text 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:	<i>BarcodeReproParams</i> , <i>LayoutElementProductionParams</i> / <i>LayoutElementPart/BarcodeProductionParams</i>
Example Partition:	—
Input of Processes:	—

Output of Processes: —

Table 7-73: BarcodeCompParams Resource

Name	Data Type	Description
<i>CompensationProcess</i>	enumeration	Process that is bar width spread is compensated for. Values are: <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, DeviceMark**

Example Partition: —

Input of Processes: —

Output of Processes: —

Table 7-74: BarcodeReproParams Resource (Section 1 of 2)

Name	Data Type	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.

Table 7-74: BarcodeReproParams Resource (Section 2 of 2)

Name	Data Type	Description
<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:	<i>Bending</i>
Output of Processes:	—

Table 7-75: BendingParams Resource

Name	Data Type	Description
<i>Bend="true"</i>	boolean	If " <i>true</i> ", indicates that the Device MUST bend.
<i>Punch="true"</i>	boolean	If " <i>true</i> ", 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.

Each **BinderySignature** consumes a number of Pages from the **PageList**. If no **SignatureCell** Elements are specified, each **BinderySignature** consumes the number of pages as calculated from *NumberUp* ($X*Y*2$) or *FoldCatalog* (The integer value after the *F*, e.g. *F16-x* consumes 16 pages). If **SignatureCell** Elements are specified, the number of pages consumed is the sum of the number of pages for all unique **SignatureCell/@SectionIndex**. The number of pages for each **SignatureCell/@SectionIndex** is one more than the maximum value of any **SignatureCell/@FrontPages** or **SignatureCell/@BackPages** for that **SignatureCell/@SectionIndex** (it is one more because **SignatureCell/@FrontPages** and **SignatureCell/@BackPages** begin at zero)

Resource Properties

Resource Class:	Parameter
Resource referenced by:	StrippingParams
Example Partition:	<i>WebName</i>
Input of Processes:	—
Output of Processes:	—

Table 7-76: BinderySignature Resource (Section 1 of 4)

Name	Data Type	Description
<i>AlignmentReferenceWeb</i> ? New in JDF 1.4	NMTOKEN	The Partition <i>WebName</i> value of the reference web that <i>WebCellAlignment</i> refers to.
<i>BinderySignatureType</i> = "Fold" New in JDF 1.3	enumeration	The type of BinderySignature. 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. Values are: <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
<i>BindingOrientation</i> ? New in JDF 1.3	orientation	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> . For BinderySignature Elements defined by <i>FoldCatalog</i> or Fold Elements, the default value of <i>BindingOrientation</i> = "Rotate0" if the folded BinderySignature has a closed head, otherwise <i>BindingOrientation</i> = "Flip0". For BinderySignature Elements defined by SignatureCell Elements, the default value <i>BindingOrientation</i> = "Rotate0". For details, See “Matrices and Orientation values for describing the orientation of a Component” on page 30.
<i>FoldCatalog</i> ?	string	Describes the type of fold according to the folding catalog in Figure 7-31, “Fold catalog part 1” on page 542 and Figure 7-32, “Fold catalog part 2” on page 543 Value format is: "Fn-i" where “n” is the number of finished pages and “i” is either an integer, which identifies a particular fold or the letter "X", which identifies a generic fold. E.g., "F6-2" describes a Z-fold of 6 finished pages, and "F6-X." describes a generic fold with 6 finished pages. Constraint: At least one of SignatureCell, <i>FoldCatalog</i> or Fold MUST be specified. <i>FoldCatalog</i> MUST NOT be specified unless <i>BinderySignatureType</i> = "Fold".

Table 7-76: BinderySignature Resource (Section 2 of 4)

Name	Data Type	Description
<i>FoldLay?</i> New in JDF 1.4	Orientation	Specification of the orientation applied to the substrate of all stacked webs before applying folding (only specified at root BinderySignature node, and would default to Rotate0).
<i>JogEdge = "Top"</i> New in JDF 1.3	enumeration	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. Values are: <i>Left</i> <i>Right</i> <i>Top</i> <i>Bottom</i> <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> .
<i>NumberUp = "1 1"</i> Modified in JDF 1.3	XYPair	Specifies a regular, multi-up grid of <i>SignatureCell</i> Elements into which content pages are mapped. The first value specifies the number of columns of <i>SignatureCell</i> Elements, and the second value specifies the number of rows of <i>SignatureCell</i> Elements 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.
<i>OutsideGutter?</i> New in JDF 1.3	boolean	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.
<i>StaggerColumns?</i> New in JDF 1.3	DoubleList	A list of doubles describing the staggering for subsequent columns. The number of entries in the list describes the periodicity of the staggering. Each value gives a factor of the strip cell height ((y value of <i>TrimSize</i>) + <i>TrimHead</i> + <i>TrimFoot</i>) by which to shift the corresponding column (can be negative). E.g., <i>StaggerColumns = "0.0 0.333 0.666"</i> specifies to shift each "3*n column up by 0% "3*n+1 column up by 33.3% of the strip cell height "3*n column up by 66.6% of the strip cell height This Element MAY be present when <i>BinderySignatureType = "Grid"</i> . At most one of <i>StaggerColumns</i> or <i>StaggerRows</i> MUST be specified.

Table 7-76: BinderySignature Resource (Section 3 of 4)

Name	Data Type	Description
<i>StaggerContinuous</i>? New in JDF 1.3	boolean	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 m columns and n rows with <i>StaggerContinuous</i> = "true". If <i>StaggerColumns</i> is specified, the BinderySignature MUST be considered continuous with a height H equal to n multiplied by the strip cell height. If <i>StaggerColumns</i> has a value of y for a certain column, that column is shifted up (assuming $y > 0$) by an amount equal to y multiplied by the strip cell height (in the same way as described for <i>StaggerColumns</i>). All content (even partial cells) that falls above H (the top of BinderySignature) is shifted to the bottom such that the top of the shifted content is just below the original bottom cell in the column. For example, if y is 0.666, then the top 66.6% of the top cell is shifted to be just below the original bottom cell. Analogous for <i>StaggerRows</i> .
<i>StaggerRows</i>? New in JDF 1.3	DoubleList	A list of doubles describing the staggering for subsequent rows. The number of entries in the list describes the periodicity of the staggering. Each value gives a factor of the strip cell width ((x value of <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 $3*n$ row right by 0% $3*n+1$ row right by 33.3% of the strip cell width $3*n+2$ row right by 66.6% of the strip cell width This Element MAY be present when <i>BinderySignatureType</i> = "Grid". At most one of <i>StaggerColumns</i> or <i>StaggerRows</i> MUST be specified.
<i>WebCellAlignment</i>? New in JDF 1.4	XYPair	Zero based <i>SignatureCell</i> index (coordinate) that the bottom left <i>SignatureCell</i> in this web is aligned with in the full web (only specified at the <i>WebName</i> Partition, and would default to "0 0"). See Figure 7-4, "WebCellAlignment, Example 1" on page 411, Figure 7-5, "WebCellAlignment Example 2" on page 412 and Figure 7-6, "WebCellAlignment Example 3" on page 413. Also, the "stacking" of the webs is implied by the order of the webs within the BinderySignature . The back side of a <i>WebName</i> Partition of a BinderySignature will be touching the front side of the <i>WebName</i> partition of the BinderySignature that follows it in the JDF file.
<i>DieLayout</i>? New in JDF 1.3	refelement	The layout as defined by a pre-existing die. DieLayout MUST be present when <i>BinderySignatureType</i> = "Die".
Fold *	element	Describes the folding operations in the sequence in which they are to be carried out. When both Fold and <i>FoldCatalog</i> are specified, <i>FoldCatalog</i> 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 <i>SignatureCell</i> Elements are present. Fold MUST NOT be specified unless <i>BinderySignatureType</i> = "Fold".

Table 7-76: BinderySignature Resource (Section 4 of 4)

Name	Data Type	Description
SignatureCell *	element	Describes the SignatureCell Elements used in this BinderySignature. SignatureCell Elements are ordered in X-Y direction starting at the lower left-hand corner of the BinderySignature. When both SignatureCell and FoldCatalog are specified, FoldCatalog defines the topology of the folding scheme, and the specifics of each individual Signature cell are described by the SignatureCell Elements. The SignatureCell Elements MUST have precedence. SignatureCell MUST NOT be specified if Fold Elements are present.

```

<BinderySignature PartIDKeys="WebName" FoldCatalog="F16-6" AlignmentReferenceWeb="Web1">
  <BinderySignature NumberUp="4 2" WebCellAlignment="0 0" WebName="Web1"/>
  <BinderySignature NumberUp="4 2" WebCellAlignment="0 0" WebName="Web2"/>
</BinderySignature>
  
```

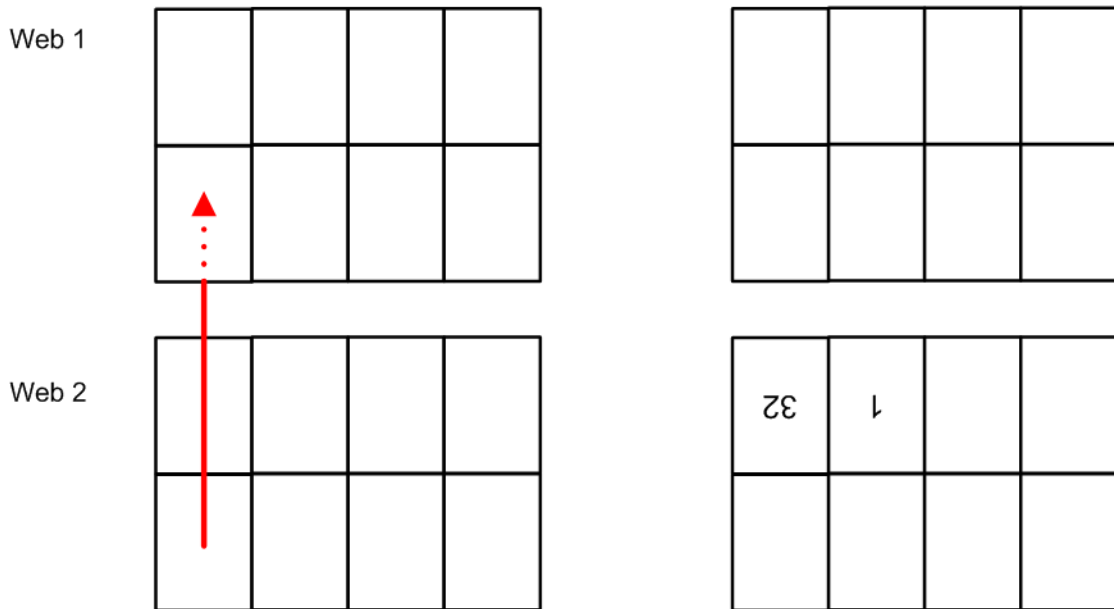


Figure 7-4: WebCellAlignment, Example 1

```

<BinderySignature PartIDKeys="WebName" FoldCatalog="F16-6" AlignmentReferenceWeb="Web1">
  <BinderySignature NumberUp="2 2" WebCellAlignment="1 0" WebName="Web1"/>
  <BinderySignature NumberUp="4 2" WebCellAlignment="0 0" WebName="Web2"/>
</BinderySignature>

```

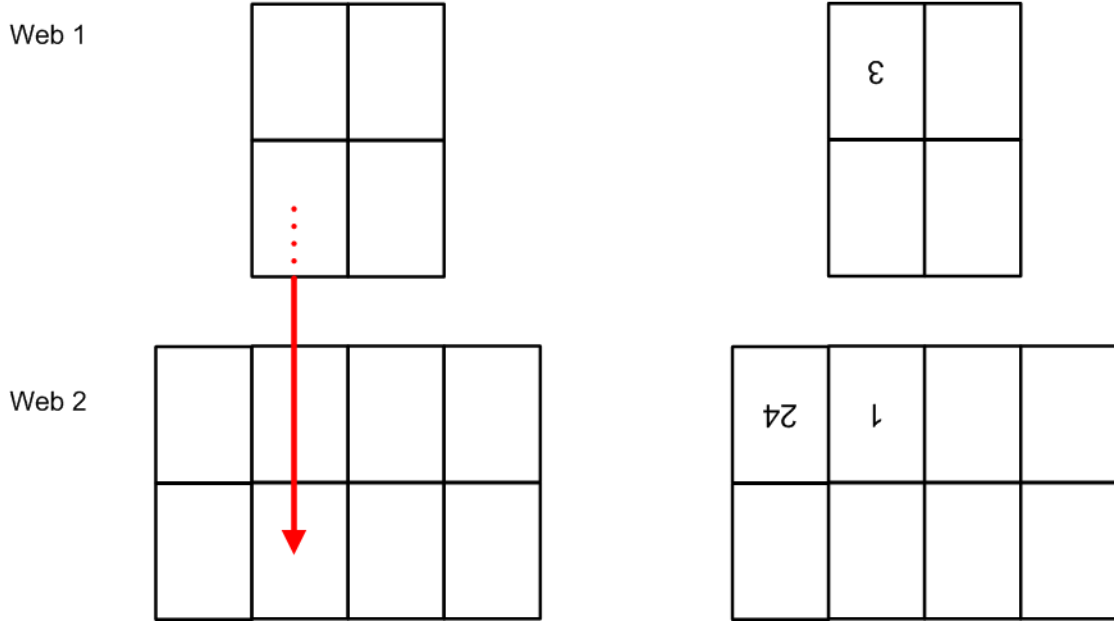


Figure 7-5: WebCellAlignment Example 2

```

<BinderySignature PartIDKeys="WebName" FoldCatalog="F16-6" AlignmentReferenceWeb="Web1">
  <BinderySignature NumberUp="2 2" WebCellAlignment="0 0" WebName="Web1"/>
  <BinderySignature NumberUp="4 2" WebCellAlignment="0 0" WebName="Web2"/>
</BinderySignature>

```

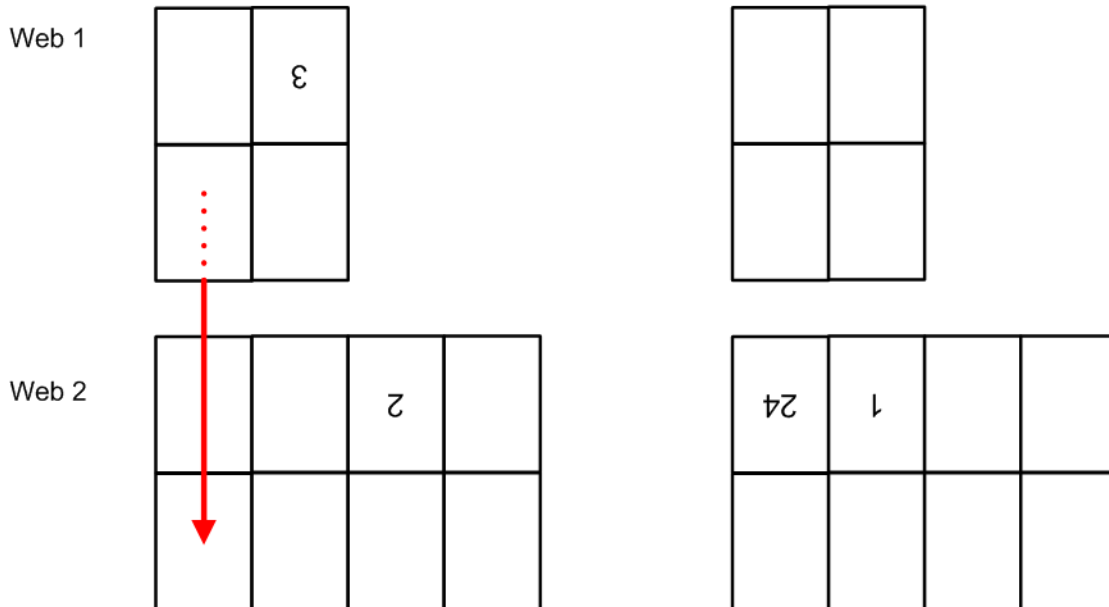


Figure 7-6: WebCellAlignment Example 3

Example 7-4: Pseudo Code to Generate Page Count from SignatureCell Elements

[New in JDF 1.4](#)

```

maxSectionIndexSeen = 0
maxSectionPages = [0]
for sc in BinderySignature/SignatureCell
  si = sc@SectionIndex
  if ( si > maxSectionIndexSeen)
    for index from maxSectionIndexSeen to si - 1:
      maxSectionPages.append(0)
    maxSectionIndexSeen = si
  for page in sc@FrontPages
    maxSectionPages[si] = max(maxSectionPages[si],page)
  for page in sc@BackPages
    maxSectionPages[si] = max(maxSectionPages[si], page)
totalPages = 0
for sectionIndex from 0 to maxSectionIndexSeen
  totalPages += 1 + maxSectionPages[sectionIndex]
return totalPages

```

7.2.12.1 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 from the PageList numbered from 0 to n, as opposed to Folio pages, which are the numbers that appear in print with the content of the document; the difference being that pages without Folio numbering are counted. As the BinderySignature is a reusable object, the page

numbers refer to finished pages numbered from 0 to n as if this **BinderySignature** were the only section of the **Assembly**. The consuming Device needs to calculate the final product page number using the **Assembly** and **StrippingParams/@SectionList**. The **BinderySignature** cells MUST NOT contain final page numbers unless **Assembly/@Order = "None"**.

Table 7-77: SignatureCell Element

Name	Data Type	Description
BackFacePages ? Deprecated in JDF 1.4	IntegerList	Page numbers for the back finished pages forming a foldout. Deprecation note: starting with JDF 1.4, use <i>FaceCells</i> to describe foldouts.
BackPages ?	IntegerList	Page numbers of the back finished pages of a <i>SignatureCell</i> . 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.
BottleAngle ?	double	Indicates the bottle angle, which is the slight rotation of the <i>SignatureCell</i> needed to compensate for the rotation fault introduced when making cross-folds.
BottleAxis ?	enumeration	Indicates the point around which the cell is bottled. Values are: <i>FaceFoot</i> <i>FaceHead</i> <i>SpineFoot</i> <i>SpineHead</i>
FaceCells ? New in JDF 1.4	IntegerList	List of indices of <i>SignatureCell</i> Elements that form a foldout together with this <i>SignatureCell</i> . The <i>SignatureCell</i> that contains <i>FaceCells</i> is the parent of the foldout, typically the Page that is attached to the spine. Details of each foldout Page are described by a <i>SignatureCell</i> Element.
FrontFacePages ? Deprecated in JDF 1.4	IntegerList	Page numbers for the front finished pages forming a foldout. Deprecation note: starting with JDF 1.4, use <i>FaceCells</i> to describe foldouts.
FrontPages ?	IntegerList	Page numbers of the front finished pages of a <i>SignatureCell</i> . Multiple page cells with the same properties except for the finished pages to which they are assigned MAY be summarized as one <i>SignatureCell</i> with multiple entries in <i>FrontPages</i> .
Orientation = "Up" Modified in JDF 1.3	enumeration	Indicates the orientation of the <i>SignatureCell</i> . Values are: <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.
SectionIndex = "0"	integer	Unique logical index of the page section that are to fill this <i>SignatureCell</i> . This is an indirect logical index. The actual section index is defined in StrippingParams/@SectionList .
StationName ? New in JDF 1.3	string	The name of the 1-up station in the die layout. Constraint: if BinderySignature/@BinderySignatureType = "Die" , this Element SHOULD be specified. Constraint: if BinderySignature/@BinderySignatureType = "Die" and BinderySignature/DieLayout contains more than 1 Station, this Attribute MUST be specified.

Example 7-5: StrippingParams: Foldout Using FaceCells

[New in JDF 1.4](#)

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <!--Stripping Foldout example corresponding to spec example n.6.5 - with new
      attribute FaceCells-->
    <StrippingParams Class="Parameter" ID="r000005"
      PartIDKeys="CellIndex" Status="Available">
      <BinderySignatureRef rRef="r000006"/>
      <StrippingParams CellIndex="0">
        <!--stripcell for the folded out foldout(front page=4)-->
        <StripCellParams TrimSize="200 400"/>
      </StrippingParams>
      <StrippingParams CellIndex="1">
        <!--stripcell for the inner page of the foldout foldout(front page=5)-->
        <StripCellParams TrimSize="300 400"/>
      </StrippingParams>
      <StrippingParams CellIndex="2">
        <!--stripcell for the inner page of the foldout foldout(front page=0)-->
        <StripCellParams TrimSize="320 400"/>
      </StrippingParams>
    </StrippingParams>
    <BinderySignature Class="Parameter" ID="r000006" Status="Available">
      <!--this is the foldout foldout cell-->
      <SignatureCell BackPages="3" FrontPages="4"/>
      <!--this cell is the inner page of the foldout, i.e. the page that is
        attached to the spine The new attribute FaceCells refers to the cell(s)
        that describe the foldout; in this case the cell to the left. The front
        and back pages of the foldout are listed in the respective cell(s)
      -->
      <SignatureCell BackPages="2" FaceCells="0" FrontPages="5"/>
      <!--this is the cell that has no foldout-->
      <SignatureCell BackPages="1" FrontPages="0"/>
    </BinderySignature>
  </ResourcePool>
  <ResourceLinkPool>
    <StrippingParamsLink Usage="Input" rRef="r000005"/>
  </ResourceLinkPool>
</JDF>
```

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.

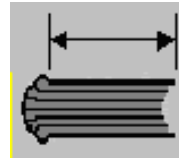
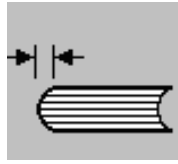
Kinds of Book Forms	Flat	Round	Flat and Backed	Rounded and Backed
<i>TightBacking=</i>	<i>Flat</i>	<i>Round</i>	<i>FlatBacked</i>	<i>RoundBacked</i>



For the rounding and for the backing there are two additional measurements:

Measurement	Rounding Way	Backing Way
-------------	--------------	-------------

Attribute

*Rounding**Backing***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. 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 7-8, “BoxFoldingType Attribute for values of Type00, Type01 and Type02” on page 420 through Figure 7-11, “BoxFoldingType Attribute for values of Type15 and Type20” on page 422. The first value of <i>BlankDimensionsX</i> is the position of the fold marked by X0 in a diagram, e.g., Figure 7-8. 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 7-8, “BoxFoldingType Attribute for values of Type00, Type01 and Type02” on page 420 through Figure 7-11, “BoxFoldingType Attribute for values of Type15 and Type20” on page 422. The first value of <i>BlankDimensionsY</i> is the position of the fold marked by Y0 in a diagram, e.g., Figure 7-8. 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.
<i>BoxFoldingType</i> ?	enumeration	<p>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.</p> <p>Each type is described with a sequence of <i>BoxFoldAction</i> Elements. The most common sequences (folding types) are predefined, All other are 'special' and MUST be described in detail.</p> <p>Values are:</p> <p>Type00 – Special type for boxes that are not pre-defined. See Figure 7-8</p> <p>Type01 – see Figure 7-8</p> <p>Type02 – see Figure 7-8</p> <p>Type03 – see Figure 7-9</p> <p>Type04 – see Figure 7-9</p> <p>Type10 – see Figure 7-9</p> <p>Type11 – see Figure 7-10</p> <p>Type12 – see Figure 7-10</p> <p>Type13 – see Figure 7-10</p> <p>Type15 – see Figure 7-11</p> <p>Type20 – see Figure 7-11</p>
<i>BoxApplication</i> * Deprecated in JDF 1.4	element	<p>Application work step in a Box folder-gluer. The sequence of <i>BoxFoldAction</i>, <i>BoxApplication</i> and <i>GlueLine</i> Elements defines the sequence of work steps. The first Element is applied first.</p> <p>Application SHOULD be described with a combined <i>Inserting</i> process.</p> <p>Deprecation note: starting with JDF 1.4, a Combined Process that includes the <i>BoxFolding</i> and <i>Inserting</i> Processes replaces <i>BoxApplication</i>.</p>

Table 7-79: BoxFoldingParams Resource (Section 2 of 2)

Name	Data Type	Description
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.

7.2.14.1 Element: BoxApplication

[Deprecated in JDF 1.4](#)

A BoxApplication describes the application of an external **Component** such as a window or handle to a folding box in the box folder-gluer. Note that a short description of the application SHOULD be specified in BoxApplication/@DescriptiveName. Application of an external **Component** SHOULD be described with a combined *Inserting* process.

Table 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 <i>BoxFolding</i> 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.

7.2.14.2 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
FoldIndex	XYPair	Identification of the upper right corner of the flap or fold that is affected by this BoxFoldAction. The first value of the XYPair refers to an indexed fold in <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.
Action	enumeration	Individual Action in the folder gluer. Values are from: Table 7-82, "Action Attribute Values" on page 419.
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 Values

Value	Description
<i>LongFoldLeftToRight</i>	For a drawing, see Table 7-7 on page 420
<i>LongFoldRightToLeft</i>	
<i>LongPreFoldLeftToRight</i>	
<i>LongPreFoldRightToLeft</i>	For a drawing, see Table 7-7 on page 420
<i>FrontFoldComplete</i>	For a drawing, see Table 7-7 on page 420
<i>FrontFoldDiagonal</i>	
<i>FrontFoldCompleteDiagonal</i>	For a drawing, see Table 7-7 on page 420
<i>BackFoldComplete</i>	For a drawing, see Table 7-7 on page 420
<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. For a drawing, see Table 7-7 on page 420
<i>Milling</i>	
<i>Rotate90</i>	90° counter-clockwise rotation
<i>Rotate180</i>	180° rotation
<i>Rotate270</i>	90° clockwise rotation

Example 7-6: BoxFoldingParams/BoxFoldAction

For instance, processing a Type01 blank (Figure 7-8, “BoxFoldingType Attribute for values of Type00, Type01 and Type02” on page 420) has the following actions:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <BoxFoldingParams Class="Parameter" ID="BFP000" Status="Available">
      <BoxFoldAction FoldIndex="0 -1" Action="LongPreFoldLeftToRight"/>
      <BoxFoldAction FoldIndex="2 -1" Action="LongPreFoldRightToLeft"/>
      <BoxFoldAction FoldIndex="1 -1" Action="LongFoldLeftToRight"/>
      <BoxFoldAction FoldIndex="3 -1" Action="LongFoldRightToLeft"/>
    </BoxFoldingParams>
  </ResourcePool>
  <ResourceLinkPool>
    <BoxFoldingParamsLink Usage="Input" rRef="BFP000"/>
  </ResourceLinkPool>
</JDF>
```

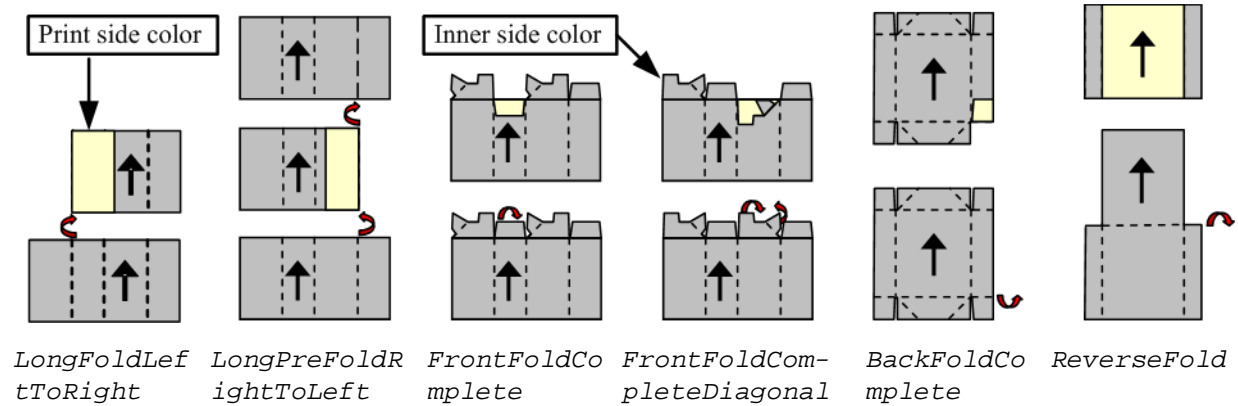


Figure 7-7: Folding examples for some values of BoxFoldAction/@Action

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

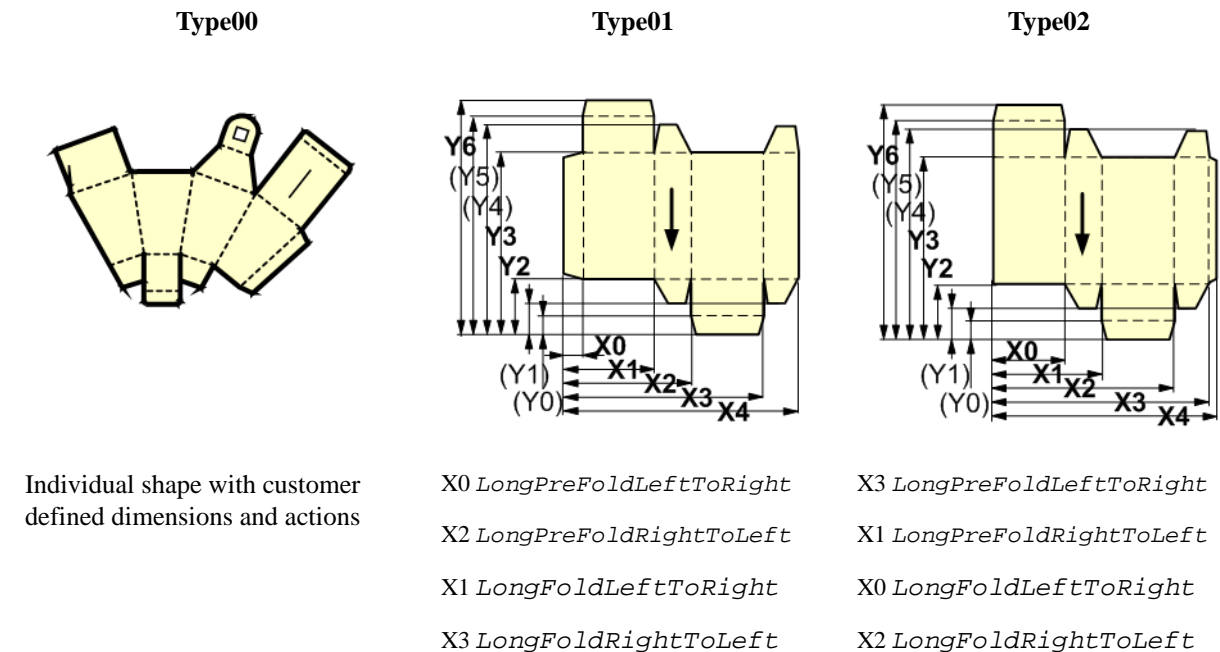
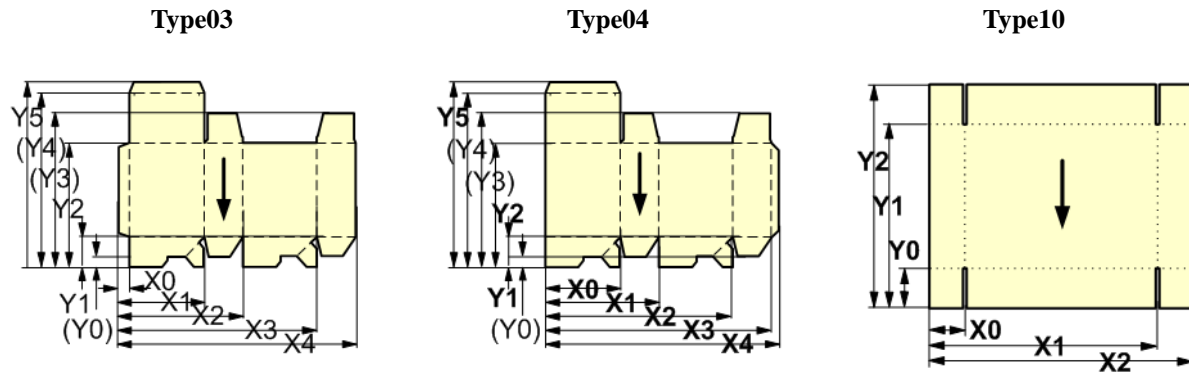


Figure 7-8: BoxFoldingType Attribute for values of Type00, Type01 and Type02

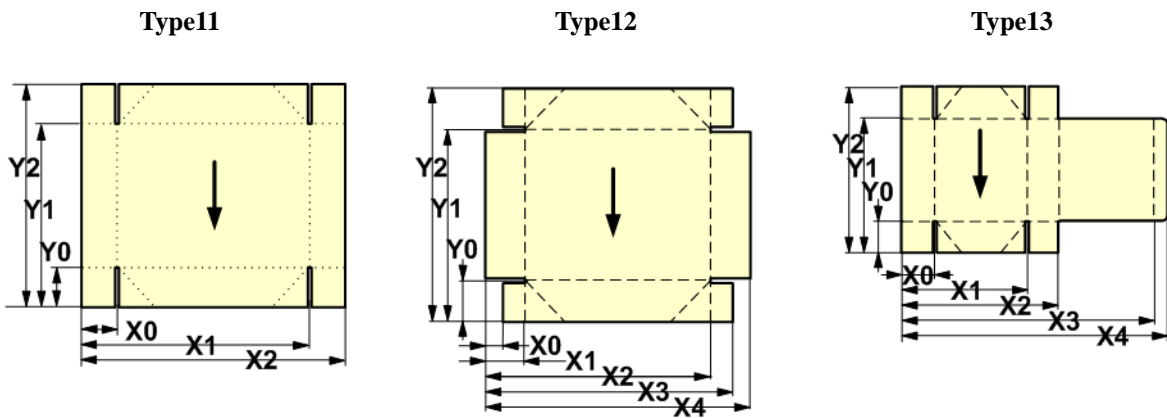


X0 LongPreFoldLeftToRight
 X2 LongPreFoldRightToLeft
 X2/Y1: FrontFoldComplete
 X4/Y1: FrontFoldComplete
 X1/Y1: FrontFoldCompleteDiagonal
 X3/Y1:
 FrontFoldCompleteDiagonal
 X1 LongFoldLeftToRight
 X3 LongFoldRightToLeft

X3 LongPreFoldLeftToRight
 X1 LongPreFoldRightToLeft
 X1/Y1: FrontFoldComplete
 X3/Y1: FrontFoldComplete
 X0/Y1:
 FrontFoldCompleteDiagonal
 X2/Y1:
 FrontFoldCompleteDiagonal
 X0 LongFoldLeftToRight
 X2 LongFoldRightToLeft

X0 LongPreFoldLeftToRight
 X1 LongPreFoldRightToLeft

Figure 7-9: BoxFoldingType Attribute for values of Type03, Type04 and Type10

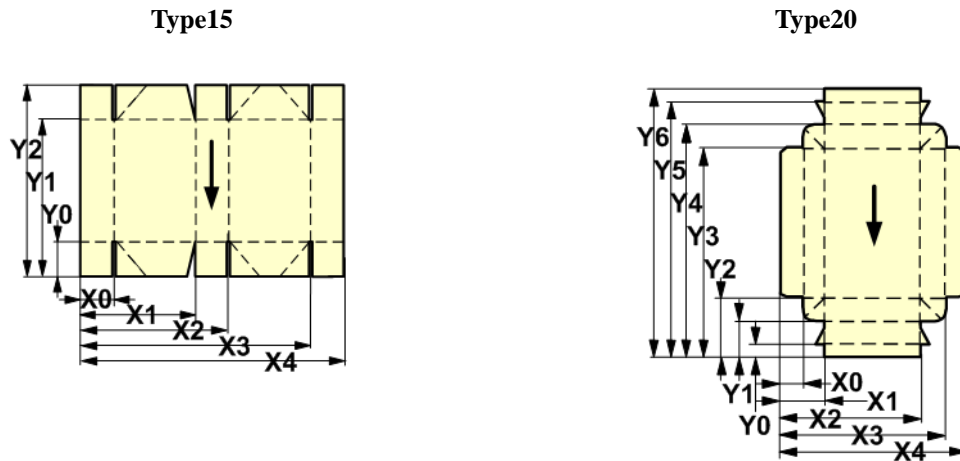


X0/Y0: FrontFoldComplete
 X2/Y0: FrontFoldComplete
 X0/Y2: BackFoldComplete
 X2/Y2: BackFoldComplete
 X1/Y0:
 FrontFoldCompleteDiagonal
 X1/Y2:
 BackFoldCompleteDiagonal
 X0 LongFoldLeftToRight
 X2 LongFoldRightToLeft

X1/Y0:
 FrontFoldCompleteDiagonal
 X1/Y2:
 BackFoldCompleteDiagonal
 X0 LongFoldLeftToRight
 X2 LongFoldRightToLeft

X0/Y0: FrontFoldComplete
 X2/Y0: FrontFoldComplete
 X0/Y2: BackFoldComplete
 X2/Y2: BackFoldComplete
 X1/Y0:
 FrontFoldCompleteDiagonal
 X1/Y2:
 BackFoldCompleteDiagonal
 X0 LongFoldLeftToRight
 X2 LongFoldRightToLeft

Figure 7-10: BoxFoldingType Attribute for values of Type 11, Type12 and Type13



X0/Y0: <i>FrontFoldComplete</i>	(continued from previous column)	X0 <i>LongFoldLeftToRight</i>
X2/Y0 <i>FrontFoldComplete</i>	X3/Y0 <i>FrontFoldCompleteDiagonal</i>	X3 <i>LongFoldRightToLeft</i>
X4/Y0 <i>FrontFoldComplete</i>	X1/Y2 <i>BackFoldCompleteDiagonal</i>	
X0/Y2 <i>BackFoldComplete</i>	X3/Y2 <i>BackFoldCompleteDiagonal</i>	
X2/Y2 <i>BackFoldComplete</i>	X0 <i>LongFoldLeftToRight</i>	
X4/Y2 <i>BackFoldComplete</i>	X3 <i>LongFoldRightToLeft</i>	
X1/Y0 <i>FrontFoldCompleteDiagonal</i>	X2 <i>LongFoldRightToLeft</i>	

Figure 7-11: 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 7-12. If the Components represent Bundles , the number of Bundles is specified.
<i>Columns</i> ? New in JDF 1.4	integer	Columns per shipping box. Columns are in the 3rd Dimension in Figure 7-12, and are thus not illustrated.
<i>ComponentOrientation</i> ? New in JDF 1.4	enumeration	Defines the coordinate pair that is facing the bottom of the box, defining the horizontal plane. Values are: <i>XY</i> – Axis X and Y <i>XZ</i> – Axis X and Z <i>YZ</i> – Axis Y and Z

Table 7-83: BoxPackingParams Resource (Section 2 of 2)

Name	Data Type	Description
<i>Copies</i> ? New in JDF 1.4	integer	Number of copies in the box. <i>Copies</i> MUST NOT be specified if <i>MaxWeight</i> is present.
<i>FillMaterial</i> ?	NMTOKEN	Material to fill boxes that are not completely filled, as illustrated by F in Figure 7-12. 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>
<i>Layers</i> ? New in JDF 1.3	integer	Layers per shipping box, as illustrated by L in Figure 7-12.
<i>MaxWeight</i> ? New in JDF 1.4	double	Maximum weight of a packed box in grams. <i>MaxWeight</i> MUST NOT be specified if <i>Copies</i> is present.
<i>Pattern</i> ?	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.
<i>Rows</i> ? New in JDF 1.3	integer	Rows per shipping box, as illustrated by R in Figure 7-12.
<i>Ties</i> ? 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.
<i>UnderLays</i> ? New in JDF 1.3	IntegerList	Number of underlay Sheets at each layer, as illustrated by U in Figure 7-12. 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.

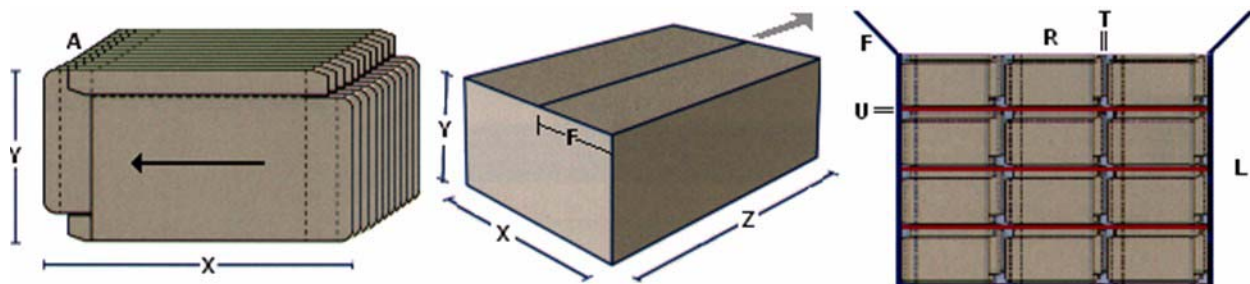


Figure 7-12: Box packing

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:	<i>Buffer</i>
Output of Processes:	—

Table 7-84: BufferParams Resource

Name	Data Type	Description
<i>MinimumWait ?</i>	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 **Components**. 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, PalletizingParams
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-85: Bundle Resource (Section 1 of 2)

Name	Data Type	Description
<i>BundleType = "Stack"</i>	enumeration	<p>Values are:</p> <p><i>BoundSet</i> – Stack of components that are bound together.</p> <p><i>Box</i></p> <p><i>Carton</i></p> <p><i>CollectedStack</i> – Components collected on a saddle, result of <i>Collecting</i> Process</p> <p><i>CompensatedStack</i> – Loose stack of compensated components</p> <p><i>Pallet</i></p> <p><i>Roll</i> – Rolled components on a print Roll.</p> <p><i>Sheet</i> – Multiple individual items printed onto one Sheet.</p> <p><i>Stack</i> – Loose stack of equally stacked components.</p> <p><i>StrappedStack</i> – Strapped stack of equally stacked components.</p> <p><i>StrappedCompensatedStack</i> – Strapped stack of compensated components.</p> <p><i>WrappedBundle</i></p>
<i>FolioCount ?</i>	integer	Total amount of individual finished pages that this bundle contains. If not specified, it MUST be calculated from the individual <i>BundleItem</i> Elements.

Table 7-85: Bundle Resource (Section 2 of 2)

Name	Data Type	Description
<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 .

7.2.17.1 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 30. 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 .

Example 7-7: Bundle: Boxing and Palletizing

The following example code shows a JDF that describes boxing and palletizing for 4200 books. The appropriate **Bundle** Elements have orange tags and magenta Attributes. The Resources have not yet been completely filled in.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Bundle" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID20" Version="1.4">
  <!-- The BoxPacking Process consumes the thing to pack and the boxes-->
  <!-- The BoxPacking Process creates packed boxes -->
  <JDF ID="n0235" Status="Waiting" Type="BoxPacking" JobPartID="ID21" >
    <ResourceLinkPool>
      <ComponentLink ProcessUsage="Box" Usage="Input" rRef="BoxID"/>
      <BoxPackingParamsLink Usage="Input" rRef="BoxParamsID"/>
      <ComponentLink Usage="Input" rRef="ComponentID"/>
      <ComponentLink Usage="Output" rRef="PackedBoxID"/>
    </ResourceLinkPool>
  <!-- The BoxPacking Process has the following local resources -->
  <ResourcePool>
    <BoxPackingParams Class="Parameter" ID="BoxParamsID"
      Status="Available"/>
    <Component Amount="100" Class="Quantity" ID="BoxID">
```

```

        Status="Available" ComponentType="Sheet"/>
    </ResourcePool>
</JDF>
<ResourcePool>
    <!-- This Component describes a Box with 42 Books -->
    <Component Amount="100" Class="Quantity" ID="PackedBoxID"
        Status="Unavailable" ComponentType="Sheet" >
        <Bundle BundleType="Box" TotalAmount="42">
            <BundleItem Amount="42">
                <ComponentRef rRef="ComponentID"/>
            </BundleItem>
        </Bundle>
    </Component>
    <Component Amount="4200" Class="Quantity" ID="ComponentID"
        Status="Available" ComponentType="Sheet" />
    <!-- This Component describes the contents of the pallet: 100
        Boxes w. 42 Books -->
    <Component Amount="10" Class="Quantity" ID="palletContentsID"
        Status="Unavailable" ComponentType="Sheet" >
        <Bundle BundleType="Pallet" TotalAmount="420">
            <BundleItem Amount="10">
                <ComponentRef rRef="PackedBoxID"/>
            </BundleItem>
        </Bundle>
    </Component>
</ResourcePool>
<JDF ID="n0239" Status="Waiting" Type="Palletizing" JobPartID="ID22">
    <ResourceLinkPool>
        <ComponentLink Usage="Input" rRef="PackedBoxID"/>
        <PalletLink Usage="Input" rRef="palletID"/>
        <PalletizingParamsLink Usage="Input" rRef="palletParamsID"/>
        <ComponentLink Usage="Output" rRef="palletContentsID"/>
    </ResourceLinkPool>
    <ResourcePool>
        <Pallet Amount="10" Class="Consumable" ID="palletID"
            Status="Available" PalletType="Euro800x600"/>
        <PalletizingParams Class="Parameter" ID="palletParamsID"
            Status="Available" />
    </ResourcePool>
</JDF>
</JDF>

```

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:	<i>Bundling</i>
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.

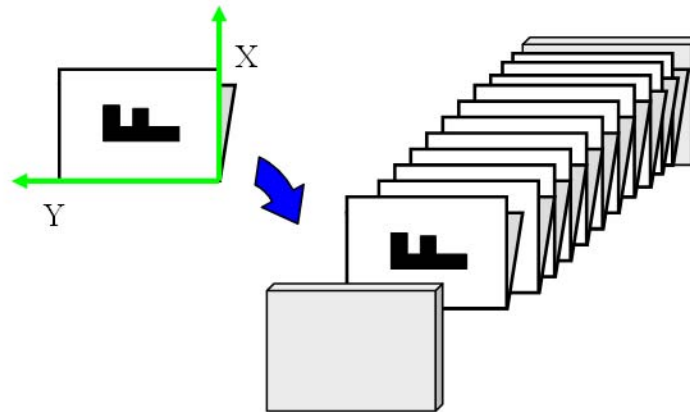


Figure 7-13: BundlingParams Coordinate System

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. <i>BandOrdering</i> is REQUIRED for non-interleaved data and MUST be ignored for interleaved data if specified. 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>ElementType</i> ? New in JDF 1.4	enumeration	Values are from: Table 7-254, “ElementType Attribute Values” on page 614. Note: values are the same as <code>LayoutElement@ElementType</code> .
<i>FrameHeight</i> ? Modified in JDF 1.4	integer	Height of the overall image that MAY be broken into multiple bands. Modification note: starting with JDF 1.4, <i>FrameHeight</i> is optional.
<i>FrameWidth</i> ? Modified in JDF 1.4	integer	Width of overall image that MAY be broken into multiple columns. Modification note: starting with JDF 1.4, <i>FrameWidth</i> is optional.
<i>Halftoned</i> ? Modified in JDF 1.4	boolean	Indicates whether or not the data has been halftoned. Modification note: starting with JDF 1.4, <i>Halftoned</i> is optional.
<i>Interleaved</i> ? Modified in JDF 1.4	boolean	If “ <i>true</i> ”, the data are interleaved or chunky. Otherwise the data are non-interleaved or planar. Modification note: starting with JDF 1.4, <i>Interleaved</i> is optional.
<i>PixelSkip</i> ?	integer	Number of bits to skip between pixels of interleaved data.
<i>Resolution</i> ? Modified in JDF 1.4	XYPair	Output resolution. Modification note: starting with JDF 1.4, <i>Resolution</i> is optional.
Band * Modified in JDF 1.4	element	Array of bands containing raster data. Modification note: starting with JDF 1.4, Band is optional.
<i>ColorPool</i> ? 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 * Modified in JDF 1.4	element	Ordered list containing information about which colorants are represented and how many bits per pixel are used. Modification note: starting with JDF 1.4, <i>PixelColorant</i> is optional.

7.2.19.1 Element: Band

Table 7-89: Band Element

Name	Data Type	Description
<i>Data?</i> Modified in JDF 1.4	URL	Actual bytes of data. Modification note: starting with JDF 1.4, <i>Data</i> is optional.
<i>Height?</i> Modified in JDF 1.4	integer	Height in pixels of the band. Modification note: starting with JDF 1.4, <i>Height</i> is optional.
<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> Modified in JDF 1.4	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. Modification note: starting with JDF 1.4, <i>WasMarked</i> is optional.
<i>Width?</i> Modified in JDF 1.4	integer	Width in pixels of the band Modification note: starting with JDF 1.4, <i>Width</i> is optional.

7.2.19.2 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:	<i>CaseMaking</i>
Output of Processes:	—

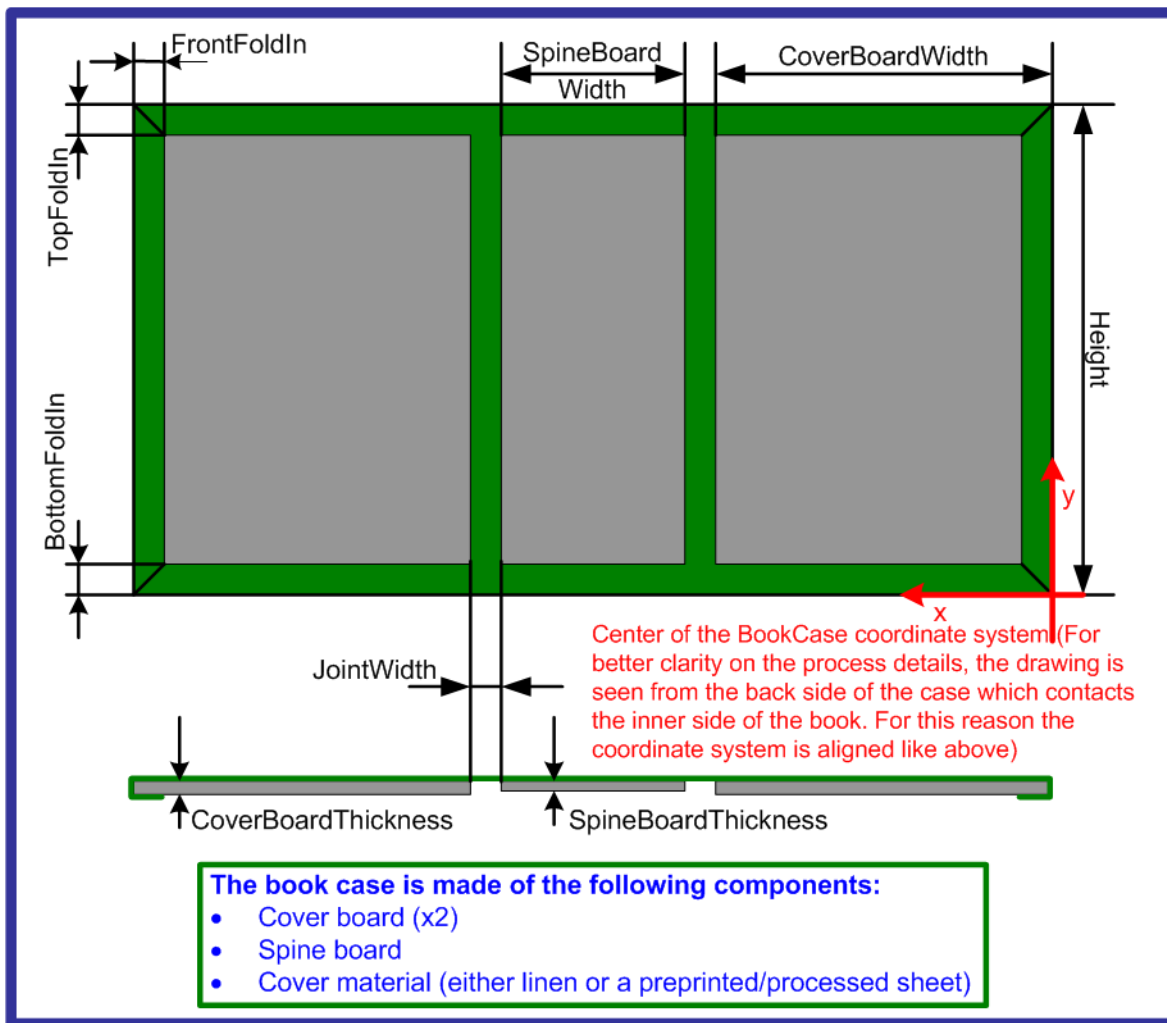


Figure 7-14: CaseMakingParams

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.
<i>CornerType</i> ?	NMTOKEN	Method of wrapping the corners of the cover material around the corners of the board. 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.

Table 7-91: CaseMakingParams Resource (Section 2 of 2)

Name	Data Type	Description
<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 7-15.

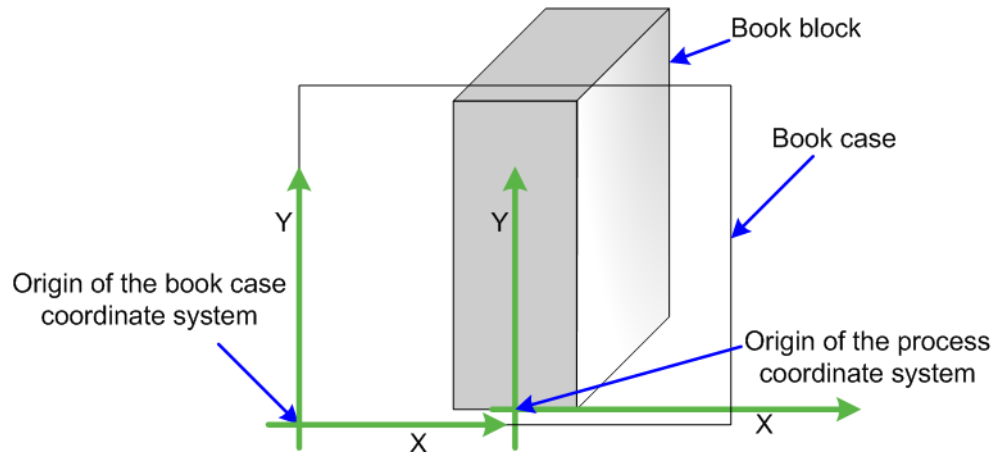


Figure 7-15: Parameters and coordinate system for CasingIn

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.
GlueApplication * New in JDF 1.4	refelement	Properties of the glue to attach the case.
GlueLine + Deprecated in JDF 1.4	refelement	Properties of the glue used. Deprecation note: starting with JDF 1.4, use GlueApplication .

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:	<i>ChannelBinding</i>

Output of Processes: —

Figure 7-16 depicts the *ChannelBinding* Process.

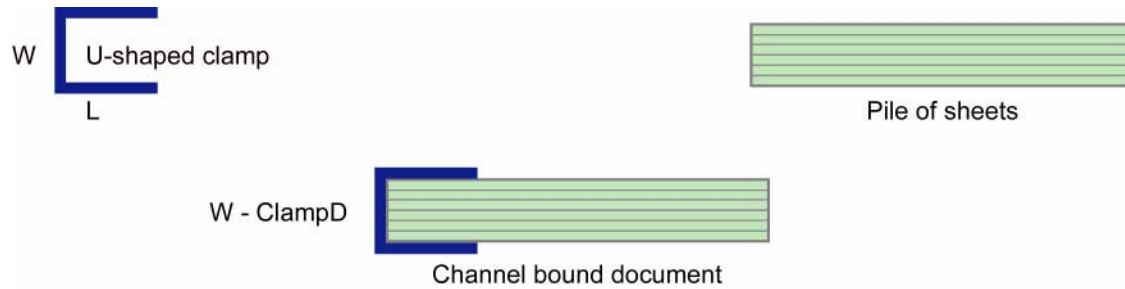


Figure 7-16: Parameters used for channel binding

The symbols W, L and ClampD of Figure 7-16 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> = "true", then the color of the cover is also meant.
<i>ClampColorDetails?</i> New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If <i>ClampColorDetails</i> is supplied, <i>ClampColor</i> SHOULD also be supplied.
<i>ClampD?</i>	double	The distance of the clamp that was "pressed away" (see Figure 7-16 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 7-16) 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 7-16). The third corresponds to the spine length (not visible in Figure 7-16). 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, Layout/MarkObject
Example Partition:	—
Input of Processes:	—
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 <code>MarkObject</code> that contains this mark. If the measuring field is defined within a <code>ColorControlStrip</code> , <i>Center</i> refers to the rectangle defined by <i>Center</i> and <i>Size</i> of the <code>ColorControlStrip</code> .
<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. 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> Deprecation note: starting with JDF 1.1, use <code>ColorMeasurementConditions/@DensityStandard</code> .
<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. 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 <code>ColorMeasurementConditions/@Observer</code>
<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. Deprecation note: starting with JDF 1.1, use details from <code>ColorMeasurementConditions</code>
<i>Tolerance?</i> Modified in JDF 1.1	double	Tolerance in ΔE .
<code>ColorMeasurementConditions?</code> 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:	<i>CoilBinding</i>
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>ColorDetails?</i> New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If <i>ColorDetails</i> is supplied, <i>Color</i> SHOULD also be supplied.
<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. Values are: <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 <i>HoleMakingParams/@HoleType</i> .
<i>Thickness?</i>	double	The thickness of the coil.
<i>Tucked="false"</i>	boolean	If " <i>true</i> ", the ends of the coils are "tucked in".
<i>HoleMakingParams?</i> New in JDF 1.2	refelement	Details of the holes in <i>CoilBinding</i> .

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:	<i>Collecting</i>
Output of Processes:	—

Table 7-96: CollectingParams Resource

Name	Data Type	Description

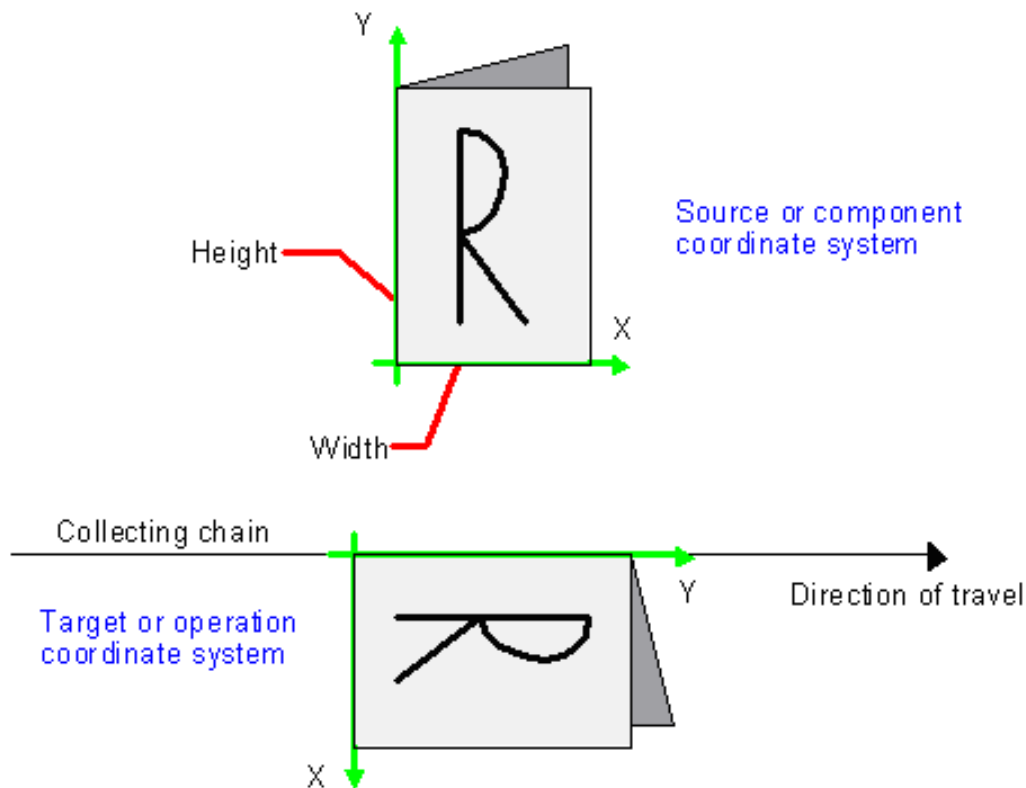


Figure 7-17: Coordinate systems used for collecting

7.2.26 Color

Color describes spot color inks, process color inks and any other coating, for instance varnish or gloss coating. Spot colors are named colors that can either be separated or converted to process colors. It is important to know the neutral density of the colorant for trapping and, in many cases, the *Lab* values for representing them on screen. If you know the *Lab* value, you can calculate the neutral density. When representing colors on screen, a conversion to process colors **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, LayoutPreparationParams/PageCell

Example Partition: —

Input of Processes: —

Output of Processes: —

Table 7-97: Color Resource (Section 1 of 4)

Name	Data Type	Description
<i>ActualColorName</i> ? New in JDF 1.3	string	Actual name of the color in the PDL. <i>ActualColorName</i> SHOULD be used to identify the color. If not specified, defaults to the value of <i>@Name</i> . Note: this Attribute was added to JDF 1.3 Errata.
<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 Resource 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 Values include: <i>CIP4 ColorBook Uncoated Grade 5</i> <i>PANTONE C</i> - an example <i>PANTONE C</i> - an example <i>Placeholder</i> - <i>Placeholder</i> is a special token that indicates that the Color/@Name is not a real color but a place holder like 'Spot1' that MUST be resolved when the content arrives. Added in JDF 1.3 Modification note: starting with JDF 1.2, the data type changes from NMTOKEN to string.
<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.

Table 7-97: Color Resource (Section 2 of 4)

Name	Data Type	Description
<i>ColorName</i> ? New in JDF 1.1	NamedColor	Mapping to a color name. Allowed values are defined in Section A.3.3.3, NamedColor.
<i>ColorDetails</i> ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If <i>ColorDetails</i> is supplied, <i>Color</i> SHOULD also be supplied.
<i>ColorType</i> ? Modified in JDF 1.2	enumeration	A name that characterizes the colorant. Values are: <i>DieLine</i> – Marks made with colorants of this type are ignored for trapping. <i>Trapping</i> 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. <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. <i>Transparent</i> – Marks made with colorants of this type are to be ignored for trapping. <i>Trapping</i> Processes are not to generate a color plane for this colorant. This value SHOULD be used for varnish. <i>Opaque</i> – Marks covered by colorants of this type are ignored for trapping. <i>Opaque</i> can be used for metallic inks. <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.
<i>Density</i> ? New in JDF 1.2	double	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.
<i>Gray</i> ? New in JDF 1.4	double	Gray 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>MappingSelection</i> = " <i>UseProcessColorValues</i> " and ColorantControl /@ <i>ProcessColorModel</i> = " <i>DeviceGray</i> ". Uses a subtractive color model: 0.0 means 100% coverage with colorant, while 1.0 means no coverage.
<i>Lab</i> ?	LabColor	L, a, b value of the 100% tint value of the colorant.

Table 7-97: Color Resource (Section 3 of 4)

Name	Data Type	Description
<i>MappingSelection</i> = "UsePDLValues" New in JDF 1.2	enumeration	This value specifies the mapping method to be used for this Color. <i>MappingSelection</i> 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. 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.
<i>MediaType</i> ? Modified in JDF 1.2	string	Specifies the media type. 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.
<i>RawName</i> ? 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.

Table 7-97: Color Resource (Section 4 of 4)

Name	Data Type	Description
<i>UsePDALternateCS</i> ? 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 <i>DeviceNColor</i> of this <i>Color</i> MUST be used depending on the value of <i>ColorantControl/@ProcessColorModel</i> . In JDF 1.2 and beyond, use <i>MappingSelection</i> .
<i>ColorMeasurementConditions</i> ? New in JDF 1.1	refelement	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>ColorantControl/@ProcessColorModel</i> = " <i>DeviceN</i> ".
<i>FileSpec (ColorProfile)</i> ?	refelement	A <i>FileSpec</i> 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.
<i>FileSpec (TargetProfile)</i> ?	refelement	A <i>FileSpec</i> Resource pointing to an ICC profile that defines the target output Device in case the object that uses the <i>Color</i> has been color space converted to a Device color space. <i>FileSpec (TargetProfile)</i> applies to the alternate color defined by the value of <i>MappingSelection</i> .
<i>PrintConditionColor</i> * 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.).
<i>TransferCurve</i> * 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"

7.2.26.1 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 <i>DeviceN</i> 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 <i>ColorantControl</i> 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 <i>ColorantControl/DeviceNSpace/@Name</i> .

7.2.26.2 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
<code>CMYK?</code>	CMYKColor	<code>CMYK</code> of the <code>PrintConditionColor</code> . Default value is from: parent <code>Color/@CMYK</code>
<code>ColorBook?</code>	string	<code>ColorBook</code> of the <code>PrintConditionColor</code> . Default value is from: parent <code>Color/@ColorBook</code>
<code>ColorBookEntry?</code>	string	<code>ColorBookEntry</code> of the <code>PrintConditionColor</code> . Default value is from: parent <code>Color/@ColorBookEntry</code>
<code>ColorBookPrefix?</code>	string	<code>ColorBookPrefix</code> of the <code>PrintConditionColor</code> . Default value is from: parent <code>Color/@ColorBookPrefix</code>
<code>ColorBookSuffix?</code>	string	<code>ColorBookSuffix</code> of the <code>PrintConditionColor</code> . Default value is from: parent <code>Color/@ColorBookSuffix</code>
<code>Density?</code>	double	<code>Density</code> of the <code>PrintConditionColor</code> . Default value is from: parent <code>Color/@Density</code>
<code>Lab?</code>	LabColor	<code>Lab</code> of the <code>PrintConditionColor</code> . Default value is from: parent <code>Color/@Lab</code>
<code>MappingSelection?</code> New in JDF 1.2	enumeration	This value specified the mapping method to be used for this <code>Color</code> . Default value is from: parent <code>Color/@MappingSelection</code> . Values are: <code>UsePDLValues</code> – Use color values specified in the PDL for this color. See [ColorPS]. <code>UseLocalPrinterValues</code> – Use the Printer's best local mapping for this <code>Color</code> . <code>UseProcessColorValues</code> – Use the values defined in this <code>Color</code> .
<code>MediaSide = "Both"</code>	enumeration	Media front and back surfaces can be different, affecting color results. If the <code>Media/@FrontCoatings</code> , <code>Media/@BackCoatings</code> or <code>Media/@Gloss</code> Attributes indicate differences in surface then <code>MediaSide</code> can be used to specify the side of the media to which the <code>PrintConditionColor</code> Attributes pertain. Values are: <code>Front</code> <code>Back</code> <code>Both</code>
<code>NeutralDensity?</code>	double	<code>NeutralDensity</code> of the <code>PrintConditionColor</code> . Default value is from: parent <code>Color/@NeutralDensity</code>

Table 7-99: PrintConditionColor Element (Section 2 of 2)

Name	Data Type	Description
<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/@Name . 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/@sRGB .
DeviceNColor *	element	DeviceNColor of the PrintConditionColor. If not specified, defaults to the parent Color/@DeviceNColor .
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/@MediaType , 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 .

Example 7-8: Color

This is an example of the structure for **Color**. The transfer curves in this example are defined for process CMYK and sRGB, independently.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool> >
    <Color Class="Parameter" ID="C000" Status="Available" CMYK="0.2 0.3 0.4 0.5"
      Density="3.14" Lab="20. 30. 40." MediaType="Coated"
      Name="PANTONE Deep Blue" sRGB="0.6 0.7 0.9">
      <TransferCurve Curve="0 0 .5 .4 1 1" Separation="Cyan"/>
      <TransferCurve Curve="0 0 .5 .6 1 1" Separation="Magenta"/>
      <TransferCurve Curve="0 0 1 1" Separation="Yellow"/>
      <TransferCurve Curve="0 0 1 1" Separation="Black"/>
      <TransferCurve Curve="0 0 1 1" Separation="sRed"/>
      <TransferCurve Curve="0 0 1 1" Separation="sGreen"/>
      <TransferCurve Curve="0 0 1 1" Separation="sBlue"/>
    </Color>
  </ResourcePool>
  <ResourceLinkPool>
    <ColorLink Usage="Input" rRef="C000"/>
  </ResourceLinkPool>
</JDF>
```

Example 7-9: ColorantControl: Content-Ignorant MIS

[New in JDF 1.4](#)

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool> >
    <!--Simple colorantcontrol from MIS: CMYK + 1 spot+ 1 black text
      version; knows no more-->
    <ColorantControl Class="Parameter" ID="r000004" ProcessColorModel="DeviceCMYK"
      Status="Available">
```

```

    <!--Note that all Strings in ColorantParams etc. use Color/@Name,
    NOT Color/@ActualColorName-->
    <ColorantParams>
      <SeparationSpec Name="Spot1" />
      <SeparationSpec Name="BlackText" />
    </ColorantParams>
  </ColorantControl>
</ResourcePool>
<ResourceLinkPool>
  <ColorantControlLink Usage="Input" rRef="r000004"/>
</ResourceLinkPool>
</JDF>

```

Example 7-10: ColorantControl: Synchronized with Input

[New in JDF 1.4](#)

Example of initial (previous) **ColorantControl** after synchronizing with input. This example specifies the replacement color name with a new *ActualColorName* Attribute in the **Color** Element. This approach has the disadvantage of needing a new Attribute. However, it has the following advantages

- no ambiguity in case of multiple names (**ColorantAlias** is used only as a pure aliasing mechanism)
- The name is localized in the **ColorPool**, which should be more central and not differ, e.g. between proofing and final imaging.
- it is “easier” to implement

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool> >
    <!--ColorantControl after prepress has correctly set ActualColorName based
    on pdl content-->
    <ColorantControl Class="Parameter" ID="r000004" ProcessColorModel="DeviceCMYK"
      Status="Available">
      <!--Note that all Strings in ColorantParams etc. use Color/@Name,
      NOT Color/@ActualColorName-->
      <ColorantParams>
        <SeparationSpec Name="Spot1" />
        <SeparationSpec Name="BlackText" />
      </ColorantParams>
      <ColorPoolRef rRef="r000005" />
    </ColorantControl>
    <ColorPool Class="Parameter" ID="r000005" Status="Available">
      <!--Color that maps the predefined separation Black
      ActualColorName is the new attribute that replaces
      ExposedMedia/@DescriptiveName as the "Main" PDL color
      -->
      <Color ActualColorName="Schwarz" CMYK="0 0 0 1" Class="Parameter"
        Name="Black" />
      <Color ActualColorName="Gelb" CMYK="0 0 1 0" Class="Parameter"
        Name="Yellow" />
      <!--ActualColorName defaults to Name if not specified-->
      <Color CMYK="1 0 0 0" Class="Parameter" Name="Cyan" />
      <Color Class="Parameter" Name="Magenta" />
      <Color ActualColorName="Acme Aqua" CMYK="0.7 0.2 0.03 0.1"
        Class="Parameter" Name="Spot1" />
      <Color ActualColorName="VersionsText" CMYK="0 0 0 1" Class="Parameter"
        Name="BlackText" />
    </ColorPool>
  </ResourcePool>
  <ResourceLinkPool>
    <ColorantControlLink Usage="Input" rRef="r000004"/>
  </ResourceLinkPool>
</JDF>

```

Example 7-11: ColorantControl: Synchronized with Input with Alias[New in JDF 1.4](#)Example of initial **ColorantControl** after synchronizing with input that contains an alias

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool> >
    <ColorantControl Class="Parameter" ID="r000004" ProcessColorModel="DeviceCMYK"
      Status="Available">
      <!--ColorantControl after prepress has correctly set ActualColorName based
        on pdl content-->
      <!--Note that all Strings in ColorantParams etc. use Color/@Name,
        NOT Color/@ActualColorName-->
      <ColorantParams>
        <SeparationSpec Name="Spot1"/>
        <SeparationSpec Name="BlackText"/>
      </ColorantParams>
      <ColorPoolRef rRef="r000005"/>
      <!--ColorantAlias that maps the additional representations
        (noir, schwarz) to the predefined separation Black-->
      <ColorantAlias Class="Parameter" RawNames="6E6F6972 73636877E4727A"
        ReplacementColorantName="Black">
        <SeparationSpec Name="noir"/>
        <SeparationSpec Name="schwarz"/>
      </ColorantAlias>
    </ColorantControl>
    <ColorPool Class="Parameter" ID="r000005" Status="Available">
      <!-- ColorPool is same as previous example -->
    </ColorPool>
    <!-- ... -->
  </ResourcePool>
  <ResourceLinkPool>
    <ColorantControlLink Usage="Input" rRef="r000004"/>
  </ResourceLinkPool>
</JDF>

```

Example 7-12: ColorantControl: with ColorantAlias/ReplacementColorantName[New in JDF 1.4](#)Example of many-to one substitution with **ColorantAlias/@ReplacementColorantName**

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool> >
    <ColorantControl Class="Parameter" ID="r000004" ProcessColorModel="DeviceCMYK"
      Status="Available">
      <!--ColorantAlias that maps the predefined separation Black-->
      <ColorantAlias ReplacementColorantName="Black">
        <SeparationSpec Name="Schwarz"/>
        <SeparationSpec Name="schwarz"/>
      </ColorantAlias>
    </ColorantControl>
  </ResourcePool>
  <ResourceLinkPool>
    <ColorantControlLink Usage="Input" rRef="r000004"/>
  </ResourceLinkPool>
</JDF>

```

Example 7-13: ColorantControl: with Invalid ColorantAlias/ReplacementColorantName[New in JDF 1.4](#)Invalid example of many-to one substitution with **ColorantAlias/@ReplacementColorantName**

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool> >
    <ColorantControl Class="Parameter" ID="r000004" ProcessColorModel="DeviceCMYK"
      Status="Available">
      <!--ColorantAlias that maps the predefined separation Black-->
      <ColorantAlias ReplacementColorantName="Black">
        <SeparationSpec Name="Schwarz"/>
      </ColorantAlias>
      <ColorantAlias ReplacementColorantName="Black">
        <SeparationSpec Name="schwarz"/>
      </ColorantAlias>
    </ColorantControl>
  </ResourcePool>
  <ResourceLinkPool>
    <ColorantControlLink Usage="Input" rRef="r000004"/>
  </ResourceLinkPool>
</JDF>
```

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 Subelement of **ColorantControl** to a top level Resource in JDF 1.2.

Resource Properties

Resource Class: Parameter
Resource referenced by: ColorantControl, ElementColorParams
Example Partition: —
Input of Processes: —
Output of Processes: —

Table 7-100: ColorantAlias Resource

Name	Data Type	Description
<i>RawNames</i> ? New in JDF 1.4	hexBinaryList	Whitespace-separated list of hexBinary values. Each token represents the original 8-bit byte stream of the color specified in SeparationSpec . Used to transport the original byte representation of a color name when moving JDF tickets between computers with different locales. Exactly one token MUST be specified for each SeparationSpec in this ColorantAlias . The order of tokens MUST be identical to the order of the related SeparationSpec .
<i>ReplacementColorantName</i>	string	The value of the colorant name string to be substituted for the colorant name strings in the SeparationSpec Resource list.
SeparationSpec + Modified in JDF 1.2	element	The names of the colorants to be replaced in PDL files.

Example 7-14: ColorantAlias/@RawNames[New in JDF 1.4](#)

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
```

```

    Type="ProcessGroup" JobPartID="ID300" Version="1.4">
<ResourcePool> >
  <ColorantAlias Class="Parameter" ID="r000004" RawNames="4772FC6E 6772FC6E"
    ReplacementColorantName="Green" Status="Available">
    <!-- ColorantAlias that maps the additional representation (grün, Grün)
      to the predefined separation Green -->
    <SeparationSpec Name="Grün" />
    <SeparationSpec Name="grün" />
  </ColorantAlias>
</ResourcePool>
<ResourceLinkPool>
  <ColorantAliasLink Usage="Input" rRef="r000004" />
</ResourceLinkPool>
</JDF>

```

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: *ColorCorrection, ColorSpaceConversion, ConventionalPrinting, DigitalPrinting, ImageSetting, Interpreting, PreviewGeneration, Separation, Stripping, Trapping*

Output of Processes: *ColorSpaceConversion*

Table 7-101: ColorantControl Resource (Section 1 of 3)

Name	Data Type	Description
<i>ForceSeparations = "false"</i>	boolean	If " <i>true</i> ", 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 " <i>false</i> " specifies to respect the parameters specified in ColorantControl and elsewhere in the JDF.
<i>ProcessColorModel?</i> Modified in JDF 1.4	NMTOKEN	Specifies the model to be used for rendering the colorants defined in color spaces into process colorants. Values include: <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> <i>None</i> – No Colorants other than those specified in ColorantParams MUST be output. New in JDF 1.4
ColorantAlias *	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. Multiple ColorantAlias Elements with identical values of ColorantAlias/@ReplacementColorantName MUST NOT be specified in the same ColorantControl resource context.
<i>ColorantConvertProcess?</i> New in JDF 1.4	element	List of colors that MUST be converted to process colors. Defaults to all colors that are neither listed in ColorantParams nor implied by <i>ProcessColorModel</i> . Application can issue a warning for all PDL Colors that are not in (ColorantParams + ColorantConvertProcess + implied by <i>ProcessColorModel</i>) lists.

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 Product 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 <i>ColorPool/Color/@MappingSelection = "UseProcessColorValues"</i>. 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, such as press modules. Note that this MUST be synchronized with the Device output ICC profile. All of the named colorants MUST occur in <i>ColorantOrder</i> if it is present. If <i>ColorantOrder</i> is not present, then all of the named colorants MUST occur in the <i>ColorantParams</i> list, or be implied by the <i>ProcessColorModel</i> . If the <i>DeviceColorantOrder</i> Element is not specified, the order for laying down colorants defaults to <i>ColorantOrder</i> .
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> .

7.2.28.1 Element: ColorantConvertProcess[New in JDF 1.4.](#)**Table 7-102: ColorantConvertProcess Element**

Name	Data Type	Description
SeparationSpec *	element	The names of the colorants that define the respective lists.

7.2.28.2 Element: ColorantOrder**Table 7-103: ColorantOrder Element**

Name	Data Type	Description
SeparationSpec *	element	The names of the colorants that define the respective lists.

7.2.28.3 Element: ColorantParams**Table 7-104: ColorantParams Element**

Name	Data Type	Description
SeparationSpec *	element	The names of the colorants that define the respective lists.

7.2.28.4 Element: DeviceColorantOrder**Table 7-105: DeviceColorantOrder Element**

Name	Data Type	Description
SeparationSpec *	element	The names of the colorants that define the respective lists.

7.2.28.5 Element: ColorSpaceSubstitute**Table 7-106: ColorSpaceSubstitute Element (Section 1 of 2)**

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).

Table 7-106: ColorSpaceSubstitute Element (Section 2 of 2)

Name	Data Type	Description
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 7-107: 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	—	Yellow Black	Cyan Magenta (If <i>DeviceColorantOrder</i> is not present then lay down order will be Cyan first, Magenta last.)
DeviceCMYK	Spot1 Spot2	Cyan Magenta Yellow Black Spot2	—	Spot1	Cyan Magenta Yellow Black Spot2
DeviceCMYK	Spot1 Spot2	Cyan Magenta Yellow Black Spot2	Cyan Magenta	Spot1 Spot2 Yellow Black	Cyan Magenta
DeviceGray	Spot1 Spot2	Black Spot2	—	Spot1	Black Spot2
DeviceN (with example N = 2 colorants as identified in <i>DeviceNSpace</i>)	Spot1 Spot2	Spot2 DeviceN 1 DeviceN 2	—	Spot1	DeviceN 1 DeviceN 2 Spot2 The reordering is accomplished using <i>DeviceColorantOrder</i> .

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:	Layout/MarkObject
Example Partition:	—
Input of Processes :	—
Output of Processes:	—

Table 7-108: ColorControlStrip Resource

Name	Data Type	Description
<i>Center</i> ? Modified in JDF 1.4	XYPair	Position of the center of the color control strip in the coordinates of the <i>MarkObject</i> that contains this mark. Modification note: starting with JDF 1.4, <i>Center</i> is optional.
<i>Rotation</i> ?	double	Rotation in degrees. Positive graduation figures indicate counter-clockwise rotation; negative figures indicate clockwise rotation.
<i>Size</i> ? Modified in JDF 1.4	XYPair	Size, in points, of the color control strip. Modification note: starting with JDF 1.4, <i>Size</i> is optional.
<i>StripType</i> ?	NMTOKEN	Type of color control strip. This Attribute MAY be used for specifying a predefined, company-specific color control strip.
<i>CIELABMeasuringField</i> * New in JDF 1.1	refelement	Details of a CIELAB measuring field that is part of this <i>ColorControlStrip</i> .
<i>DensityMeasuringField</i> * New in JDF 1.1	refelement	Details of a density measuring field that is part of this <i>ColorControlStrip</i> .
<i>SeparationSpec</i> * New in JDF 1.4	element	Ordered list of separations that comprise the <i>ColorControlStrip</i> . If neither <i>CIELABMeasuringField</i> nor <i>DensityMeasuringField</i> are specified, the geometry is implied by the value of <i>StripType</i> .

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 893 Attribute description.

Note: These color adjustments are not available in any Intent Resource, (e.g., **ColorIntent**). In order to request such adjustment in a Product Intent Job ticket supplied to a print provider, attach to a Product Intent Node an incomplete **ColorCorrection** Process with a **ColorCorrectionParams** Resource specifying the requested “AdjustXXX” Attributes.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	<i>DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName</i>
Input of Processes:	<i>ColorCorrection</i>
Output of Processes:	—

Table 7-109: 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. Values include those from: <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. ColorCorrectionOp SHOULD contain the complete set of parameters for a given color correction operation. Otherwise the results are implementation dependent.

7.2.30.1 Element: ColorCorrectionOp

Table 7-110: 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. 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>
<i>AdjustCyanRed?</i> 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.)
<i>AdjustMagentaGreen?</i> 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.)
<i>AdjustYellowBlue?</i> 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.)
<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.)

Table 7-110: ColorCorrectionOp Element (Section 2 of 2)

Name	Data Type	Description
<i>ObjectTags</i> ? New in JDF 1.4	NMTOKENS	Tags associated with individual objects that this <i>ColorCorrectionOp</i> MUST be applied to. Each tag specified in <i>ObjectTags</i> is logically anded with the object type(s) specified by <i>SourceObjects</i> , enabling first qualification by object type (such as image), and then tags associated with those objects. The values of <i>ObjectTags</i> depends on the PDL that the color correction is applied to.
<i>FileSpec</i> (<i>AbstractProfile</i>)? New in JDF 1.2	refelement	A <i>FileSpec</i> 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.)
<i>FileSpec</i> (<i>DeviceLinkProfile</i>)? New in JDF 1.2	refelement	A <i>FileSpec</i> 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 7-111: ColorMeasurementConditions Resource (Section 1 of 2)

Name	Data Type	Description
<i>DensityStandard</i> = "ANSIT"	enumeration	Density filter standard used during density measurements. 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>

Table 7-111: ColorMeasurementConditions Resource (Section 2 of 2)

Name	Data Type	Description
<i>Illumination</i> = "D50"	enumeration	Illumination used during spectral measurements. Values are: <i>D50</i> <i>D65</i> <i>Unknown</i>
<i>InkState</i> ?	enumeration	State of the ink during color measurements. Values are: <i>Dry</i> <i>Wet</i> <i>NA</i> 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. 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. Values are: <i>Black</i> <i>White</i> <i>NA</i> Deprecated in JDF 1.2
<i>WhiteBase</i> ?	enumeration	Reference for white calibration used for density measurements. 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:	ColorIntent, NumberingIntent, ByteMap, ColorantControl, FormatConversionParams, LayoutElement, PageList
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-112: 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/@ICColorStandard or ColorIntent/@ColorStandard .

7.2.33 ColorSpaceConversionOp

Note: **ColorSpaceConversionOp** was elevated from a Subelement 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, ElementColorParams
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-113: ColorSpaceConversionOp Resource (Section 1 of 5)

Name	Data Type	Description
<i>IgnoreEmbeddedICC = "false"</i>	boolean	If " <i>true</i> ", specifies that embedded source ICC profiles MUST be ignored as part of the selection criteria for this ColorSpaceConversionOp . If " <i>true</i> ", FileSpec (PDLSourceProfile) is ignored.
<i>ObjectTags ?</i> New in JDF 1.4	NMTOKENS	Tags associated with individual objects that this ColorSpaceConversionOp MUST be applied to. Each tag specified in <i>ObjectTags</i> is logically anded with the object type(s) specified by <i>SourceObjects</i> , enabling first qualification by object type (such as image), and then tags associated with those objects. The values of <i>ObjectTags</i> depends on the PDL that the color space conversion is applied to.

Table 7-113: ColorSpaceConversionOp Resource (Section 2 of 5)

Name	Data Type	Description
<p><i>Operation?</i> Modified in JDF 1.2</p>	enumeration	<p>Controls which of five functions the color space conversion utility performs.</p> <p>Values are:</p> <p><i>Convert</i> – Transforms graphical elements to final target color space.</p> <p><i>Tag</i> – Associates appropriate working space profile with uncharacterized graphical element.</p> <p><i>Untag</i> – Removes all profiles and color characterizations from graphical elements.</p> <p><i>Retag</i> – Equivalent to a sequence of <i>Untag</i> → <i>Tag</i>, where the <i>Untag</i> → <i>Tag</i> sequence is only applied to those objects selected by this ColorSpaceConversionOp.</p> <p><i>ConvertIgnore</i> – Equivalent to a sequence of <i>Untag</i> → <i>Convert</i>.</p> <p>Constraint: <i>Operation</i> MUST be specified in the context of ColorSpaceConversionParams/ColorSpaceConversionOp and MUST NOT be specified in the context of ElementColorParams/ColorSpaceConversionOp.</p>
<p><i>PreserveBlack = "false"</i> New in JDF 1.1</p>	boolean	<p>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.</p>
<p><i>RenderingIntent = "ColorSpaceDependent"</i> Modified in JDF 1.3</p>	enumeration	<p>Identifies the rendering intents associated with <i>SourceObjects</i> Elements.</p> <p>Values are (ICC-defined [ICC.1] rendering intent values):</p> <p><i>Saturation</i></p> <p><i>Perceptual</i></p> <p><i>RelativeColorimetric</i></p> <p><i>AbsoluteColorimetric</i></p> <p><i>ColorSpaceDependent</i> – The rendering intent is dependent on the color space. The dependencies are implementation specific. Modified in JDF 1.3</p>

Table 7-113: ColorSpaceConversionOp Resource (Section 3 of 5)

Name	Data Type	Description
<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 <i>ColorSpaceConversion</i> Process in determining how to allocate RGB values to the black (K) channel. After the <i>ColorSpaceConversion</i> Process is completed, the <i>Rendering</i> Process uses <i>AutomatedOverPrintParams</i> 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. Values are from: Table 7-114, "SourceCS Attribute Values"; Note: see Table 7-115, "Mapping of SourceCS enumeration values to color spaces in the most common input file formats".
<i>SourceObjects</i> = "All"	enumerations	List of object Classes that identifies which incoming graphical objects will be operated on. 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>

Table 7-113: ColorSpaceConversionOp Resource (Section 4 of 5)

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. Default value is from: <i>@RenderingIntent</i> . Values are (ICC-defined [ICC.1] rendering intent values): <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 Note: The <i>SourceRenderingIntent</i> will pertain to the source profile used in a particular <i>ColorSpaceConversion</i> Process, (e.g., sources can be the native original color space, an intermediate working color space or an reference output simulation color space).
<i>DeviceNSpace</i> ? New in JDF 1.2	refelement	DeviceNSpace Resource that describe the DeviceN color space on which to operate when <i>SourceCS</i> = " <i>DeviceN</i> ". 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 de-scribes the characterization of an Abstract Profile for specifying a preference adjustment.
FileSpec (<i>DeviceLinkProfile</i>)* New in JDF 1.3 Modified in JDF 1.4	refelement	A FileSpec Resource pointing to a Device Link ICC profile [ICC.1]. The Source colorspace of the referenced Device Link Profile SHOULD match that of the profile identified by FileSpec (<i>PDLSourceProfile</i>) and the destination color space SHOULD match that of the destination profile identified by ColorSpaceConversionParams . Multiple Device Link ICC profiles should be provided where each profile specifies a different rendering intent. This is important in the case where multiple PDL content objects of the colorspace specify different rendering intents. FileSpec (<i>DeviceLinkProfile</i>) MUST be ignored when the <i>Operation</i> is " <i>Tag</i> " or " <i>Retag</i> ". Note: an ICC Device Link profile contains only one transform with one color rendering intent. Note: this Element was added to JDF 1.3 Errata. Modification note: starting with JDF 1.4, multiple FileSpec (<i>DeviceLinkProfile</i>) Elements are allowed.

Table 7-113: ColorSpaceConversionOp Resource (Section 5 of 5)

Name	Data Type	Description
FileSpec (<i>PDLSourceProfile</i>)? New in JDF 1.4	refelement	A FileSpec Resource describing an ICC profile that describes a profiled source space that this ColorSpaceConversionOp should operate on. When present, only objects that specify the <i>SourceCS</i> along with the specified profile are selected. Note: the FileSpec/@Checksum or FileSpec/@FileVersion Attributes SHOULD be used to positively identify the ICC profile when available (the specified attribute's value should match the ICC ProfileID field).
FileSpec (<i>SourceProfile</i>)?	refelement	A FileSpec Resource pointing to an ICC profile [ICC.1] that describes the assumed source color space. If FileSpec (<i>SourceProfile</i>) is specified, it MUST be used as the profile for the source object's color space during a <i>Convert</i> , <i>Tag</i> , or <i>Retag</i> operation, as specified by <i>@Operation</i> . FileSpec (<i>SourceProfile</i>) MUST be present for <i>Tag</i> or <i>Retag</i> operations, as specified by <i>@Operation</i> .
SeparationSpec * New in JDF 1.2	element	SeparationSpec Resource(s) defining on which separation(s) to operate when <i>SourceCS</i> = " <i>Separation</i> ".

— Attribute: SourceCS

Table 7-114: SourceCS Attribute Values (Section 1 of 2)

Value	Description
<i>All</i> New in JDF 1.4	Operates on all source colorspaces. This is useful when specifying a <i>Convert</i> operation using all PDL source-supplied characterizations with a JDF-supplied final target device profile.
<i>CalGray</i> New in JDF 1.3	defines a calibrated Device independent representation of Gray.
<i>CalRGB</i> New in JDF 1.3	defines a calibrated based Device independent representation of RGB. 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> New in JDF 1.2	Operates on <i>CalGray</i> and <i>CalRGB</i> color spaces.
<i>CIEBased</i> New in JDF 1.2	Operates on CIE-based color spaces (<i>CIEBasedA</i> , <i>CIEBasedABC</i> , <i>CIEBasedDEF</i> and <i>CIEBasedDEFG</i>).
<i>CMYK</i>	Operates on DeviceCMYK color spaces. If <i>IgnoreEmbeddedICC</i> is " <i>true</i> ", then PDL DeviceCMYK color spaces that have an ICC-based characterization (/DefaultCMYK in PDF having an ICC-based characterization) are treated as uncharacterized DeviceCMYK color spaces, and operated on by <i>SourceCS</i> = " <i>CMYK</i> ".
<i>DeviceN</i> New in JDF 1.2	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 refelements MUST also be present.
<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>).

Table 7-114: SourceCS Attribute Values (Section 2 of 2)

Value	Description
<i>Gray</i>	Operates on <i>DeviceGray</i> . If <i>IgnoreEmbeddedICC</i> is "true", then PDL DeviceGray color spaces that have an ICC-based characterization (/DefaultGray in PDF having an ICC-based characterization) are treated as uncharacterized DeviceGray color spaces, and operated on by <i>SourceCS</i> = "Gray".
<i>ICCBased</i> New in JDF 1.2	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 defined in the PDL, and a ColorSpaceConversionOp where <i>SourceCS</i> = "DevIndep" is not to be applied unless that color space is also Device independent.
<i>ICCMYK</i> New in JDF 1.3	Operates on DeviceCMYK PDL data having an associated ICC-based characterization (/DefaultCMYK in PDF having an ICC-based characterization).
<i>ICCGray</i> New in JDF 1.3	Operates on DeviceGray PDL data having an associated ICC-based characterization (/DefaultGray in PDF having an ICC-based characterization).
<i>ICCLAB</i> New in JDF 1.3	defines an ICC based Device independent representation of LAB
<i>ICCRGB</i> New in JDF 1.3	Operates on DeviceRGB PDL data having an associated ICC-based characterization (/DefaultRGB in PDF having an ICC-based characterization).
<i>Lab</i> New in JDF 1.2	Operates on <i>Lab</i> .
<i>RGB</i> Modified in JDF 1.2	Operates on <i>DeviceRGB</i> . If <i>IgnoreEmbeddedICC</i> is "true", then PDL DeviceRGB color spaces that have an ICC-based characterization (/DefaultRGB in PDF having an ICC-based characterization) are treated as uncharacterized DeviceRGB color spaces, and operated on by <i>SourceCS</i> = "RGB".
<i>Separation</i> New in JDF 1.2	Operates on <i>Separation</i> 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> New in JDF 1.2	Operates on <i>YUV</i> (Also known as YCbCr). See [CCIR601-2]

Notes:

DevIndep has been retained for backwards compatibility with JDF 1.1 and because there will probably be cases where the same processing is to be applied to all Device independent spaces. An equivalent "DevDep" has not been added because it's less likely that all Device-dependent spaces are to be treated in the same way. The following table summarizes how the *SourceCS* Attribute is mapped to/from different file formats.

Table 7-115: 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

Table 7-115: 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>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
<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>ICCCMYK</i> <i>ICCGray</i> <i>ICCLAB</i> <i>ICCRGB</i>	PDF (2)	ICCBased (For <i>ICCCMYK</i> , <i>ICCGray</i> , and <i>ICCRGB</i> , this refers to objects where the object's resource dictionary's ColorSpace dictionary supplies an ICC-based interpretation for DefaultCMYK, DefaultGray or DefaultRGB, respectively)
	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) When *IgnoreEmbeddedICC* = "false", *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.

7.2.34 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:	<i>DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName</i>
Input of Processes:	<i>ColorSpaceConversion</i>
Output of Processes:	—

Table 7-116: ColorSpaceConversionParams Resource (Section 1 of 3)

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. Values include those from: <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.
<i>ConvertDevIndepColors?</i> Deprecated in JDF 1.1	boolean	When "true", 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> = "DevIndep" in JDF 1.1.

Table 7-116: ColorSpaceConversionParams Resource (Section 2 of 3)

Name	Data Type	Description
<i>ICCProfileUsage</i> = "UsePDL" New in JDF 1.2	enumeration	<p>This Attribute specifies where to obtain the destination profile for the current iteration of the <i>ColorSpaceConversion</i> Process, (i.e., either from the PDL (PDF/X output intent dictionary) or supplied in the JDF <i>ColorSpaceConversionParams</i> Resource). <i>ICCProfileUsage</i> provides an order precedence.</p> <p>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(ReferenceOutputProfile). 3 Use the profile specified in the LayoutElement/ElementColorParams/FileSpec(ActualOutputProfile). 4 Use the Device Link profile specified in a ColorSpaceConversionOp/FileSpec(DeviceLinkProfile). 5 Use the profile specified in ColorSpaceConversionParams/FileSpec(FinalTargetDevice). 6 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(ReferenceOutputProfile). 2 Use the profile specified in the LayoutElement/ElementColorParams/FileSpec(ActualOutputProfile). 3 Use the Device Link profile specified in a ColorSpaceConversionOp/FileSpec(DeviceLinkProfile). 4 Use the profile specified in ColorSpaceConversionParams/FileSpec(FinalTargetDevice). 5 Use the system specified profile.
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.

Table 7-116: ColorSpaceConversionParams Resource (Section 3 of 3)

Name	Data Type	Description
ColorSpaceConversionOp *	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. The XML order of ColorSpaceConversionOp Elements is significant, and when multiple elements apply to the same object, they are applied in that XML order.</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> <p>ColorSpaceConversionOp SHOULD contain the complete set of parameters for a given color space conversion operation. Otherwise the results are implementation dependent.</p>

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, CustomerInfo/CustomerMessage, Person
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-117: ComChannel Resource (Section 1 of 2)

Name	Data Type	Description
ChannelType Modified in JDF 1.2	enumeration	<p>Type of the communication channel.</p> <p>Values are:</p> <p><i>Phone</i> - Telephone number.</p> <p><i>Email</i> - Email address.</p> <p><i>Fax</i> - Fax machine.</p> <p><i>WWW</i> - WWW home page or form.</p> <p><i>JMF</i> - JMF messaging channel.</p> <p><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.</p> <p><i>InstantMessaging</i> - IM service address. The <i>ChannelTypeDetails</i> Attribute has the name of the IM service vendor</p>

Table 7-117: ComChannel Resource (Section 2 of 2)

Name	Data Type	Description
<i>ChannelTypeDetails</i> ? New in JDF 1.2	NMTOKEN	Description of the value of the <i>ChannelType</i> Attribute. Consumer treats this value as the service vendor name if <i>ChannelType</i> is <i>PrivateDirectory</i> or <i>InstantMessaging</i> . Values include those from: Table 7-118, “ChannelTypeDetails Attribute – predefined values for certain ChannelType values” on page 465
<i>ChannelUsage</i> ? New in JDF 1.2	NMTOKENS	Communication channel usage. 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.
<i>Locator</i> 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: Values include: “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 7-118: 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.

Example 7-15: ComChannel for Telephone

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool> >
    <ComChannel Class="Parameter" ID="cc000004" ChannelType="Phone"
      ChannelTypeDetails="Mobile" ChannelUsage="Business"
      Locator="tel:+44-07808-907-919" Status="Available"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ComChannelLink Usage="Input" rRef="cc000004" />
  </ResourceLinkPool>
</JDF>
```

Example 7-16: ComChannel for Instant Messaging

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool> >
    <ComChannel Class="Parameter" ID="cc000004" ChannelType="InstantMessaging"
      ChannelTypeDetails="MyIMService" ChannelUsage="Private"
      Locator="123456789" Status="Available"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ComChannelLink Usage="Input" rRef="cc000004"/>
  </ResourceLinkPool>
</JDF>

```

7.2.36 Company

Specifies contacts to a company including detailed information about contact persons and addresses. Use *@ProductID* when a unique identifier for the **Company** is required. This structure can be used in many situations where addresses or contact persons are needed. Examples of contacts are customer, supplier, company and address-ees. 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 7-119: 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.

Glossary – Component

The descriptions of **Component**-specific Attributes use some terms whose meaning depends on the culture in which they are used. For example, different cultures mean different things when they refer to the “front” side of a magazine. Other terms (e.g., binding) are defined by the production process and, therefore, do not depend on the culture.

Whenever possible, this specification endeavors to use culturally independent terms. In cases where this is not possible, Western style (left-to-right writing) is assumed. Please note that these terms might have a different meaning in other cultures, (i.e., those writing from right to left).

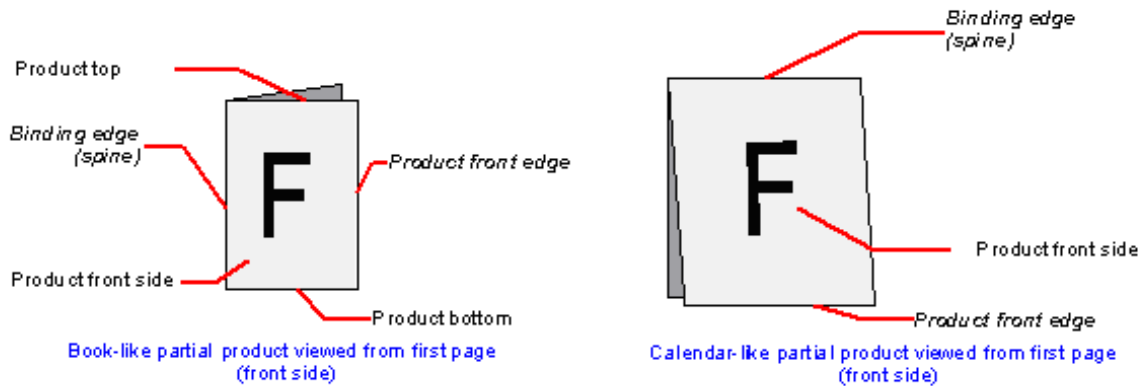


Figure 7-18: Component – terms and definitions

The table below describes the terms used to define the components.

Table 7-120: Glossary – Component

Term	Definition
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:** Bundle/BundleItem, DigitalPrintingParams, FeedingParams/Feeder, FeedingParams/CollatingItem
- Example Partition:** Condition, RibbonName, SheetName, SignatureName, WebName
- Input of Processes:** Any Product Intent Node, ConventionalPrinting, DigitalPrinting, Varnishing, BlockPreparation, BoxFolding, BoxPacking, Bundling, CaseMaking, CasingIn, ChannelBinding, CoilBinding, Collecting, CoverApplication, Creasing, Cutting, Embossing, EndSheetGluing, Feeding, Folding, Gathering, Gluing, HeadBandApplication, HoleMaking, Inserting, Jacketing, Labeling, Laminating, Numbering, Palletizing, Perforating, PlasticCombBinding, PrintRolling, RingBinding, ShapeCutting, Shrinking, SpinePreparation, SpineTaping, Stacking, StaticBlocking, Stitching, Strapping, StripBinding, ThreadSealing, ThreadSewing, Trimming, WebInlineFinishing, WireCombBinding, Wrapping
- Output of Processes:** , Any Product Intent Node, ConventionalPrinting, DigitalPrinting, Varnishing, BlockPreparation, BoxFolding, BoxPacking, Bundling, CaseMaking, CasingIn, ChannelBinding, CoilBinding, Collecting, CoverApplication, Creasing, Cutting, Embossing, EndSheetGluing, Feeding, Folding, Gathering, Gluing, HeadBandApplication,

HoleMaking, Inserting, Jacketing, Labeling, Laminating, Numbering, Palletizing, Perforating, PlasticCombBinding, PrintRolling, RingBinding, ShapeCutting, Shrinking, SpinePreparation, SpineTaping, Stacking, StaticBlocking, Stitching, Strapping, StripBinding, ThreadSealing, ThreadSewing, Trimming, WebInlineFinishing, WireCombBinding, Wrapping

Table 7-121: Component Resource (Section 1 of 3)

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 423) of the two top flaps of a carton for shipping. MUST NOT be specified unless <i>ProductType</i> = "Carton".
<i>ComponentType</i> Modified in JDF 1.3	enumerations	Specifies the category of the component. 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. New in JDF 1.3 <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. <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. Modified in JDF 1.3 Constraint: 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 first five enumerations specified as values.
<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> Deprecated in JDF 1.4	boolean	If "true", the Component is waste from a previous Process that can be used to set up a Machine. Deprecation note: starting with JDF 1.4, use Partitioning with <i>Condition</i> instead of <i>IsWaste</i> .
<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).

Table 7-121: Component Resource (Section 2 of 3)

Name	Data Type	Description
Overfold ? New in JDF 1.1	double	Expansion of the overfold of a Component . This Attribute is needed for the <i>Inserting</i> or other postpress Processes.
OverfoldSide ? New in JDF 1.1	enumeration	Specifies the longer side of a folded component. Values are: <i>Front</i> <i>Back</i>
PageListIndex ? New in JDF 1.3	IntegerRange-List	List of the indices of the PageData Elements of the PageList specified in this Component .
ProductType ?	NMTOKEN	Type of product that this component specifies. Values include those from: Table 7-122, "ProductType Attribute Values" on page 470).
ProductTypeDetails ? New in JDF 1.3	string	Additional details of the product: If <i>ProductType</i> = "BlankBox" or <i>ProductType</i> = "FlatBox", <i>ProductTypeDetails</i> specifies a box type: e.g. [ECMA], [FEFCO] or company internal box type standard. Values include: <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> , "Block") 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> , "Ribbon"). <i>RibbonName</i> of the ribbon used in this Component . Deprecation note: starting with JDF 1.3, use a direct reference to the Layout Partition that represents the ribbon.
SourceSheet ? Deprecated in JDF 1.3	string	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 . Deprecation note: starting with JDF 1.3, use a direct reference to the Layout Partition that represents the Sheet.
SourceWeb ? Deprecated in JDF 1.3	string	MUST NOT be specified unless contains (@ <i>ComponentType</i> , "Ribbon"). <i>WebName</i> of the ribbon used in this Component . Deprecation note: starting with JDF 1.3, use a direct reference to the Layout Partition that represents the Web.
SpineThickness ? New in JDF 1.4	double	Thickness
SurfaceCount ? New in JDF 1.1	integer	Total amount of individual surfaces that this Component contains. Note: a sheet always has 2 Surfaces regardless of the number of images or reader pages. In case of homogeneous Component Elements, <i>SurfaceCount</i> refers to surfaces with a size of Component/@Dimensions

Table 7-121: Component Resource (Section 3 of 3)

Name	Data Type	Description
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 <code>ResourceLink/@Transformation</code> or <code>ResourceLink/@Orientation</code> .
Assembly ? New in JDF 1.3	refelement	Specifies the assembly of the Component . In case of a newspaper-Web Press, the output Component MAY already be built up of several “booklets”. <i>AssemblyIDs</i> 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 (<code>@ComponentType, "Sheet"</code>) or contains (<code>@ComponentType, "Block"</code>). Replaced with Layout in JDF 1.2 and beyond. The Sheet in the referenced Layout is accessed by matching <i>SourceSheet</i> with <code>Layout/Signature/Sheet/@Name</code>
Layout ? New in JDF 1.2	refelement	Specifies the original Layout of the source Sheet of the Component if it contains (<code>@ComponentType, "Sheet"</code>) or contains (<code>@ComponentType, "Block"</code>). The original Sheet is the Layout Partition Element where <i>SourceSheet</i> matches the <code>Layout/@SheetName</code> used in this Component
Media ? New in JDF 1.4	refelement	Media for the component. The coordinate system of Media coincides with the coordinate system of the component.
PageList ? New in JDF 1.3	refelement	Specification of page metadata for pages described by this Component .

— Attribute: ProductType

Table 7-122: ProductType Attribute Values (Section 1 of 2)

Value	Description
<i>BackCover</i>	
<i>BlankBox</i> New in JDF 1.3	Cut, Unfolded box, input for folder-gluer
<i>BlankSheet</i> New in JDF 1.4	A sheet with connected blanks after a die cutting
<i>BlankWeb</i> New in JDF 1.4	A web with connected blanks after a die cutting.

Table 7-122: ProductType Attribute Values (Section 2 of 2)

Value	Description
<i>Body</i> New in JDF 1.2	Generic content inside of a cover.
<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> New in JDF 1.3	A folded and glued blank (not opened). Output from a box folder-gluer.
<i>FrontCover</i>	
<i>Insert</i> New in JDF 1.2	
<i>Jacket</i>	Hard cover case jacket.
<i>Label</i>	
<i>Pallet</i> New in JDF 1.3	Loaded pallet of Boxes, Cartons or Component Resources
<i>Poster</i>	
<i>Stack</i> New in JDF 1.4	Stacked Component .
<i>Newspaper</i> New in JDF 1.3	A newspaper-product
<i>Unknown</i> Deprecated in JDF 1.2	

Example 7-17: Component: Partitioned by the Condition Partition Key[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 Partitioned Resources. See Section 3.10.5, Description of Partitioned Resources.)

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="IS123" Status="Waiting"
  Type="Product" JobPartID="ID20" Version="1.4">
  <ResourcePool>
    <Component Class="Quantity" ID="Res6" PartIDKeys="Condition" Status="Available"
      ComponentType="Sheet">
      <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>
      </AmountPool>
    </ComponentLink>
  </ResourceLinkPool>
</JDF>
```

```
<PartAmount Amount="87">
  <Part Condition="Waste" />
</PartAmount>
</AmountPool>
</ComponentLink>
</ResourceLinkPool>
</JDF>
```

7.2.38 Contact

Element describing a contact to a person or address.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	Abstract Physical Resource, ArtDeliveryIntent , ArtDeliveryIntent/ArtDelivery , DeliveryIntent , DeliveryIntent/DropIntent , ApprovalParams/ApprovalPerson , ApprovalSuccess/ApprovalDetails , ContentList/ContentData/ContentMetadata , CustomerInfo , DeliveryParams , DeliveryParams/Drop , DigitalDeliveryParams , OrderingParams
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-123: Contact Resource

Name	Data Type	Description
<i>ContactTypes</i> Modified in JDF 1.4	NMTOKENS	Classification of the contact. 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>Agency</i> – The contact is an employee of an Agency. New in JDF 1.4 <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>Author</i> – New in JDF 1.4 <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>Designer</i> – New in JDF 1.4 <i>Editor</i> – New in JDF 1.4 <i>Illustrator</i> – New in JDF 1.4 <i>Owner</i> – The owner of a Resource. <i>Photographer</i> – New in JDF 1.4 <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. <i>TelephoneSanitizer</i> – New in JDF 1.4
<i>ContactTypeDetails</i> ? New in JDF 1.2	string	Details of the Contact's role or roles. For instance, if contains (@ <i>ContactTypes</i> , " <i>Delivery</i> ") this could be a description for which delivery location this Contact is responsible.
<i>Address</i> ?	refelement	Element describing the address.
<i>ComChannel</i> * 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.
<i>Company</i> ? New in JDF 1.1	refelement	Company that this Contact is associated with.
<i>Person</i> ?	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 7-124: ContactCopyParams Resource

Name	Data Type	Description
<i>ContactScreen = "false"</i>	boolean	<i>ContactScreen = "true"</i> 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. Values are: <i>On</i> <i>Off</i>
<i>PolarityChange = "true"</i>	boolean	<i>PolarityChange = "true"</i> if the copy is to change polarity with respect to the original image.
<i>RepeatStep = "1 1"</i>	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.
<i>ScreeningParams ?</i>	refelement	Properties of the halftone screen on film. Ignored if <i>ContactScreen = "false"</i> .

7.2.40 ContentList

[New in JDF 1.3](#)

ContentList provides a list of **ContentData** Elements.

Resource Properties

Resource Class: Parameter
 Resource referenced by: **PublishingIntent, LayoutElement, PageList**
 Example Partition: —
 Input of Processes: —
 Output of Processes: —

Table 7-125: ContentList Resource

Name	Data Type	Description
ContentData + Modified in JDF 1.4	element	Details of the individual content element. A ContentData Element is referred to by its ID, i.e. the value of ContentData/@ID . Modification note: before JDF 1.4, a ContentData Element was referred to by its index in ContentList with the warning that ContentData elements not be removed or inserted in a position other than the end of the list.

7.2.40.1 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 7-126: 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>ContentRefs</i> ? New in JDF 1.4	IDREFS	List of <i>ContentData/ID</i> values that specify the <i>ContentData</i> Elements children of this <i>ContentData</i> Element. For instance, a book may refer to individual chapters. The reference <i>ContentData</i> object MUST reside in the same ContentList as this <i>ContentData</i> .
<i>ContentType</i> ?	NMTOKEN	Type of content. Values include those from: Table 7-127, “ <i>ContentType</i> Attribute Values” on page 475.
<i>HasBleeds</i> ?	boolean	If " <i>true</i> ", the file has bleeds.
<i>ID</i> ? New in JDF 1.4	ID	For reference by <i>ContentRefs</i>
<i>IsBlank</i> ?	boolean	If " <i>true</i> ", the <i>ContentData</i> has no content marks and is blank.
<i>IsTrapped</i> ?	boolean	If " <i>true</i> ", the file has been trapped.
<i>JobID</i> ?	string	ID of the Job that this <i>ContentData</i> belongs to.
<i>ProductID</i> ?	string	An ID of the <i>ContentData</i> as defined in the MIS system.
<i>ContentMetadata</i> ? New in JDF 1.4	element	Container for document related metadata such as ISBN, Author etc.
<i>ElementColorParams</i> ?	refelement	Color details of the <i>ContentData</i> Element.
<i>ImageCompressionParams</i> ?	refelement	Specification of the image compression properties.
<i>ScreeningParams</i> ?	refelement	Specification of the screening properties.
<i>SeparationSpec</i> *	element	List of separation names defined in the Element.

— Attribute: **ContentType**

Table 7-127: ContentType Attribute 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.).
<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.

Table 7-127: ContentType Attribute Values (Section 2 of 2)

Value	Description
<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.

7.2.40.2 Element: ContentMetadata

[New in JDF 1.4](#)

ContentMetadata is a container for metadata pertaining to this ContentData Element. Additional metadata fields may be created using GeneralID.

Table 7-128: ContentMetadata Element

Name	Data Type	Description
<i>ISBN10?</i>	string	The 10 digit ISBN (see ref ISBN)
<i>ISBN13?</i>	string	The 13 digit ISBN (see ref ISBN)
<i>Title?</i>	string	The title of the content
Comment?	element	If required, an abstract should be specified in Comment[@Name = "Abstract"].
Contact *	refelement	The person who is responsible for this content.
Employee *	refelement	If required, the author should be specified in an Employee[contains(@Roles, "Author")].
Part?	element	If present, conserves the values of the specified Partition Keys related to the content being processed. It is illegal to set Partition Key values where that key is used to explicitly Partition the referencing Resource, or is implied by that Resource. Note: this allows Partition Keys and values to be conserved in a RunList (Surfaces) or Component.

Example 7-18: ContentList

[New in JDF 1.4](#)

```
<?xml version="1.0" encoding="UTF-8"?>
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1"
  ID="n071030_02242300_000002" JobPartID="n071030_02242300_000002"
  Status="Waiting" Type="Approval" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="Approval">
  <!--Generated by the CIP4 Java open source JDF Library version : CIP4 JDF
    Writer Java 1.3 BLD 47-->
  <AuditPool>
    <Created AgentName="CIP4 JDF Writer Java" AgentVersion="1.3 BLD 47"
      Author="CIP4 JDF Writer Java 1.3 BLD 47"
      ID="a071030_02242331_000003" TimeStamp="2007-10-30T14:24:23+01:00"/>
  </AuditPool>
</ResourcePool>
```

```

<ApprovalParams Class="Parameter" ID="ID345" Status="Available"/>
<ApprovalSuccess Class="Parameter" ID="ID346" Status="Unavailable"/>
<RunList Class="Parameter" ID="r071030_02242378_000004"
  Status="Available">
  <PageListRef rRef="PageList"/>
</RunList>
<PageList Class="Parameter" ID="PageList" Status="Available">
  <ContentListRef rRef="ContentList"/>
</PageList>
<ContentList Class="Parameter" ID="ContentList" Status="Available">
  <ContentData ContentListIndex="1 2 3">
    <ContentMetadata ISBN="0123456789" Title="book thing">
      <Comment ID="c071030_022423109_000005" Name="Abstract">
        Abstract of the book in english
      </Comment>
      <Contact Class="Implementation" ContactTypes="Editor">
        <Person Class="Parameter" DescriptiveName="authorName"
          FamilyName="authorName"/>
      </Contact>
    </ContentMetadata>
  </ContentData>
  <ContentData>
    <ContentMetadata Title="Chapter 1">
      <Contact Class="Implementation" ContactTypes="Customer">
        <Person Class="Parameter" DescriptiveName="authorName1"
          FamilyName="authorName1"/>
      </Contact>
    </ContentMetadata>
  </ContentData>
  <ContentData>
    <ContentMetadata Title="Chapter 2">
      <Contact Class="Implementation" ContactTypes="Customer">
        <Person Class="Parameter" DescriptiveName="authorName2"
          FamilyName="authorName2"/>
      </Contact>
    </ContentMetadata>
  </ContentData>
  <ContentData>
    <ContentMetadata Title="Chapter 3">
      <Contact Class="Implementation" ContactTypes="Customer" >
        <Person Class="Parameter" DescriptiveName="authorName3"
          FamilyName="authorName3"/>
      </Contact>
    </ContentMetadata>
  </ContentData>
</ContentList>
</ResourcePool>
<ResourceLinkPool>
  <RunListLink Usage="Input" rRef="r071030_02242378_000004"/>
  <ApprovalParamsLink Usage="Input" rRef="ID345"/>
  <ApprovalSuccessLink Usage="Output" rRef="ID346"/>
</ResourceLinkPool>
</JDF>

```

Example 7-19: ContentList: Extended with ISBN, Author, etc.

[New in JDF 1.4](#)

Example of ContentList with ISBN, Author, etc.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool >
    <!-- Information about the input (file, author) -->
    <RunList ID="NodeIDRunList" Status="Available" Class="Parameter" >

```

```

<LayoutElementRef rRef="NodeIDLE" />
<PageList>
  <ContentList>
    <ContentData>
      <!-- String for title -->
      <new:DocumentInfo Title="This is the title of the book"
        ISBN="0123456789" xmlns:new="new schema URI">
        <!-- Multi-lines string for Abstract -->
        <new:DocumentAbstract>
          This is the abstract of the book
          It has several lines...
        </new:DocumentAbstract>
        <!-- List of authors. Using a PersonRef allows reusing the same
          Person element -->
        <new:Author Subject="Preface">
          <PersonRef rRef="AuthorID1" />
        </new:Author>
        <new:Author Subject="Content">
          <new:PersonRef rRef="AuthorID2" />
          <new:PersonRef rRef="AuthorID3" />
        </new:Author>
      </new:DocumentInfo>
    </ContentData>
  </ContentList>
</PageList>
</RunList>
<LayoutElement ID="NodeIDLE" Status="Available" Class="Parameter" >
  <FileSpec URL="file:///hotfolder/files/Document2747.pdf"
    MimeType="application/pdf" UserFileName="JDF1.3.pdf" />
</LayoutElement>
<!-- Information about the authors -->
<Person ID="AuthorID1" Class="Parameter" Status="Available" FirstName="James"
  FamilyName="Smith" JobTitle="Author" />
<Person ID="AuthorID2" Class="Parameter" Status="Available" FirstName="John"
  FamilyName="Smith" JobTitle="Author" />
<Person ID="AuthorID3" Class="Parameter" Status="Available" FirstName="William"
  FamilyName="Smith" JobTitle="Author" />
<!-- Media: A3 white paper coated on both sides, 70 gr/m2 -->
<Media ID="MediaID" Class="Consumable" Status="Available" Weight="70"
  Dimension="1190 842" MediaType="Paper" MediaColorName="White"
  FrontCoatings="Coated" BackCoatings="Coated" />
<!-- Media: A4 yellow paper for Banner Page -->
<Media ID="MediaID2" Class="Consumable" Status="Available" Weight="70"
  Dimension="595 842" MediaType="Paper" MediaColorName="Yellow" />
<!-- Booklet layout + banner page with ISBN and Authors printed on it -->
<LayoutPreparationParams ID="NodeIDLPP" Class="Parameter" Status="Available"
  Sides="TwoSidedFlipY" NumberUp="2 1" BindingEdge="Left"
  PresentationDirection="FoldCatalog" FoldCatalog="F4-1"
  FinishingOrder="GatherFold" PageDistributionScheme="Saddle">
  <InsertSheet SheetType="JobSheet" SheetUsage="Header" IsWaste="true"
    SheetFormat="Standard" >
    <Layout>
      <MediaRef rRef="MediaID2" />
      <MarkObject CTM="1 0 0 1 0 0" >
        <JobField ShowList="new:ISBN new:Authors" />
      </MarkObject>
    </Layout>
  </InsertSheet>
</LayoutPreparationParams>
</ResourcePool>
<ResourceLinkPool>
  <LayoutPreparationParamsLink Usage="Input" rRef="NodeIDLPP"/>
  <RunListLink Usage="Input" rRef="NodeIDRunList"/>
  <MediaLink Usage="Input" rRef="MediaID"/>

```



```
</ResourceLinkPool>
</JDF>
```

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 3.10.5, 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 7-129: ConventionalPrintingParams Resource (Section 1 of 3)

Name	Data Type	Description
<i>DirectProof = "false"</i>	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 <i>ConventionalPrinting</i> 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 = "WaitForApproval"</i> .
<i>Drying?</i>	enumeration	The way in which ink is dried after a print run. 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>
<i>FirstSurface?</i> Modified in JDF 1.2	enumeration	Printing order of the surfaces on the Sheet. Values are: <i>Either- Deprecated in JDF 1.2 Omit FirstSurface to specify either.</i> <i>Front</i> <i>Back</i>
<i>FountainSolution?</i>	enumeration	State of the fountain solution module in the printing units. Values are: <i>On</i> <i>Off</i>

Table 7-129: ConventionalPrintingParams Resource (Section 2 of 3)

Name	Data Type	Description
<i>MediaLocation</i> ?	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 3.10.6.4, Locations of Physical Resources) there SHOULD be a match between one <i>Location</i> Partition Key and this <i>MediaLocation</i> value. Values include those from: Table C-21, “Input Tray and Output Bin Names” on page 889. Note: the specified values are for printer locations.
<i>ModuleAvailableIndex</i> ? New in JDF 1.1 Deprecated in JDF 1.4	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> . Deprecation note: starting with JDF 1.4, the skipping of press modules is now handled by specifying ColorantControl/DeviceColorantOrder/SeparationSpec with no <i>@Name</i>
<i>ModuleDrying</i> ?	enumeration	The way in which ink is dried in individual modules. 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>
<i>ModuleIndex</i> ? Deprecated in JDF 1.4	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> . Deprecation note: starting with JDF 1.4, the skipping of press modules is now handled by specifying ColorantControl/DeviceColorantOrder/SeparationSpec with no <i>@Name</i>
<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 ConventionalPrinting Process. The Media origin is unaffected by <i>NonPrintableMarginBottom</i> .
<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>PerfectingModule</i> ? New in JDF 1.1	integer	Index of the perfecting module if <i>WorkStyle</i> = <i>Perfecting</i> and multiple perfecting modules are installed.
<i>Powder</i> ?	double	Quantity of powder in%.

Table 7-129: ConventionalPrintingParams Resource (Section 3 of 3)

Name	Data Type	Description
PrintingTechnology? New in JDF 1.4	enumeration	Printing technology of the press or press module. Values are: <i>Flexo</i> <i>Gravure</i> <i>Offset</i> <i>Screen</i>
PrintingType Modified in JDF 1.3	enumeration	Type of printing Machine. 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
SheetLay?	enumeration	Lay of input media. Reference edge of where paper is placed in a feeder. Values are: <i>Left</i> <i>Right</i> <i>Center</i>
Speed? 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.
WorkStyle?	WorkStyle	The direction in which to turn the press Sheet.
ApprovalParams? New in JDF 1.2	refelement	Details of the direct Approval Process, when <i>DirectProof</i> = "true".
Ink * Modified in JDF 1.2 Deprecated in JDF 1.4	refelement	Details of varnishing. Defines the varnish to be used for coatings on printed sides. Coatings are applied after printing all the colors. Other coating sequences 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 <i>Ink@Family</i> = "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. Deprecation note: starting with JDF 1.4, use Varnishing .

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:	Notification, ResourceInfo, JobPhase, Device , Employee
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-130: 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:	<i>CoverApplication</i>
Output of Processes:	—

Table 7-131: 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.

7.2.43.1 Element: Score

Table 7-132: Score Element

Name	Data Type	Description
<i>Offset</i>	double	Position of scoring given in the operation coordinate system.
<i>Side="FromInside"</i>	enumeration	Specifies the side from which the scoring tool works. Values are: <i>FromInside</i> <i>FromOutside</i>

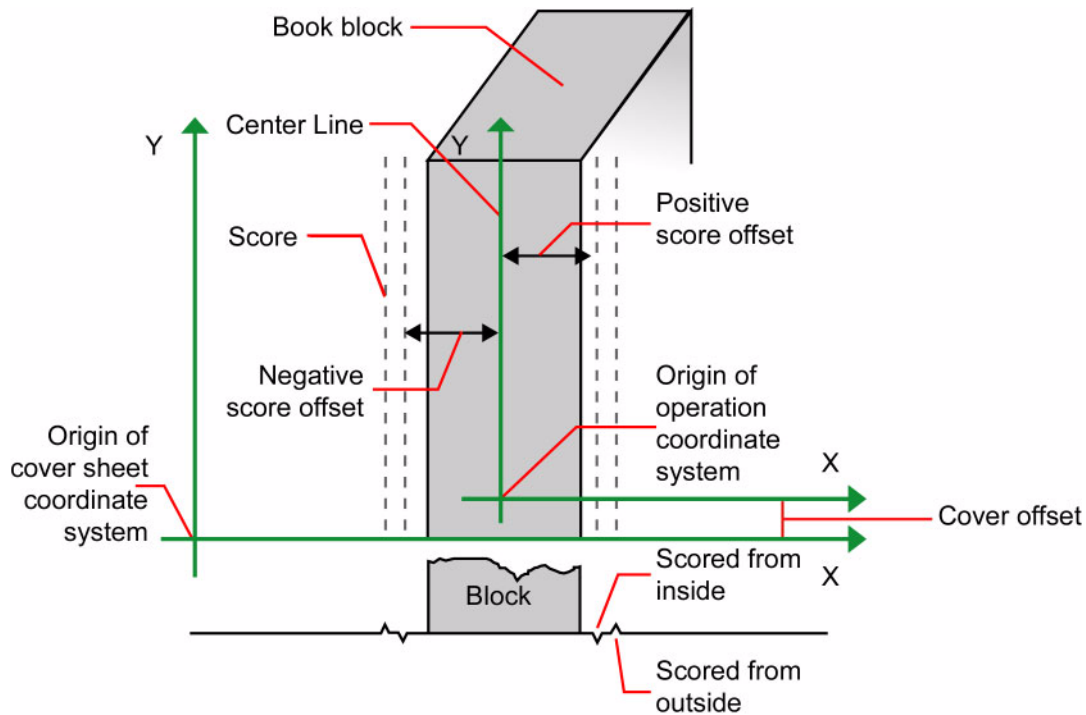


Figure 7-19: Parameters and coordinate system for cover application

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:	<i>BlockName, RibbonName, SheetName, SignatureName, WebName</i>
Input of Processes:	<i>Creasing</i>
Output of Processes:	—

Table 7-133: CreasingParams Resource

Name	Data Type	Description
Crease *	element	Defines one or more Crease lines.

7.2.44.1 Element: Crease

Crease defines an individual crease line on a **Component**.

Table 7-134: Crease Element (Section 1 of 2)

Name	Data Type	Description
<i>Depth?</i> New in JDF 1.2	double	Depth of the crease, measured in microns [μm].

Table 7-134: Crease Element (Section 2 of 2)

Name	Data Type	Description
<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. 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.

Before JDF 1.3, **CustomerInfo** was a Sub-element of a JDF Node, and “inherited” down to child Nodes. Starting with JDF 1.3, **CustomerInfo** became a Resource that must be linked like any other Resource; there is no “inheritance”. Any Node **MAY** link to the same **CustomerInfo** Resource as its parent. A normative **CustomerInfo** is specified by a



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?

linked Resource. An informative **CustomerInfo** MAY be retrieved by searching for **CustomerInfo** of parent Nodes or Ancestor Elements

Resource Properties

Resource Class:	Parameter
Resource referenced by:	Ancestor
Example Partition:	—
Input of Processes:	<i>Any Process</i>
Output of Processes:	—

Table 7-135: 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 Contact Elements is specified in Contact .
Contact * New in JDF 1.1	refelement	Resource Element describing contacts associated with the customer. There SHOULD be one Contact [contains (@ <i>ContactTypes</i> , "Customer")]. 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.

7.2.45.1 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 7-136: CustomerMessage Element (Section 1 of 2)

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 .

Table 7-136: CustomerMessage Element (Section 2 of 2)

Name	Data Type	Description
<i>MessageEvents</i>	NMTOKENS	Defines the set of events that trigger a message that is defined or specified by the system. Values include those from: Table C-20, “MessageEvents and MilestoneType Values” on page 886.
<i>ShowList?</i> Modified in JDF 1.4	NMTOKENS	List of parameters to display in the CustomerMessage. Values include those from: Table I-1, “Predefined variables used in @XXXTemplate and @ShowList” on page 909 New in JDF 1.4 Modification note: starting with JDF 1.4, the values come from a common list rather than a list that is custom to this Element.

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 7-137: CutBlock Resource (Section 1 of 2)

Name	Data Type	Description
<i>AssemblyIDs?</i> New in JDF 1.3	NMTOKENS	The <i>AssemblyIDs</i> of the Assembly , AssemblySection or StrippingParams[@BinderySignatureName] 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. 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.

Table 7-137: CutBlock Resource (Section 2 of 2)

Name	Data Type	Description
<i>BlockType</i>	enumeration	Block type. 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.
<i>CutWidth</i> ? New in JDF 1.4	double	Width in points of u-shaped knife, saw blade, etc.
<i>Assembly</i> ? New in JDF 1.3	refelement	Assembly that is referred to by <i>AssemblyIDs</i> or contains the <i>AssemblySection</i> 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/MarkObject
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-138: CutMark Resource

Name	Data Type	Description
<i>Blocks</i> ? Modified in JDF 1.1	NMTOKENS	Values of the <i>BlockName</i> Partition Attributes of the blocks defined by the CutMark Resource.
<i>MarkType</i>	enumeration	Cut mark type. 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 <i>MarkObject</i> 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 7-139: 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:	<i>Cutting</i>
Output of Processes:	—

Table 7-140: CuttingParams Resource (Section 1 of 2)

Name	Data Type	Description
<i>NUpSeparation</i> ? New in JDF 1.4	XYPair	Defines the number of CutBlock Elements in x and y direction. For example, a 2-up book sawed apart would have <i>NUpSeparation</i> = "2 1".
CutBlock *	refelement	One or several CutBlock Elements can be used to find the <i>Cutting</i> sequence. The CutBlock Elements MUST NOT be written if Cut Elements are specified.

Table 7-140: CuttingParams Resource (Section 2 of 2)

Name	Data Type	Description
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. Cut Elements MUST NOT be specified if CutBlock Elements are specified.

7.2.48.1 Element: Cut

Cut describes one straight cut with an arbitrary tool.

Table 7-141: Cut Element

Name	Data Type	Description
<i>CutWidth</i> ? New in JDF 1.4	double	Width in points of u-shaped knife, saw blade, etc.
<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.
<i>WorkingDirection</i>	enumeration	Direction from which the tool is working. 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: *PlateLayout, ProductionRun, Separation*
Input of Processes: —
Output of Processes: *CylinderLayoutPreparation*

Table 7-142: CylinderLayout Resource

Name	Data Type	Description
<i>DeviceID</i> ?	NMTOKEN	Specifies the Device that this CylinderLayout belongs to.
<i>CylinderPosition</i> + <i>Layout</i> ?	element refelement	Specifies the position of a plate on a cylinder of a newspaper-Web Press. References the Layout that describes the plates to be mounted.

7.2.49.1 Element: CylinderPosition

Table 7-143: CylinderPosition Element

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	<p>Specifies whether the plate contains content data or represents a dummy plate. Additionally, it indicates where in the workflow it will be produced.</p> <p>Values are:</p> <p><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.</p> <p><i>Exposed</i> – Indicates that the plate contains content data and MUST be exposed by an ImageSetting Process.</p>
<i>PlateUsage</i> = "Original"	enumeration	<p>Specifies, whether a plate has to be produced for a specific Web run or not.</p> <p>Values are:</p> <p><i>Original</i> – indicates that the plate is to be produced specifically for this run.</p> <p><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.</p>
<i>DeviceModuleIndex</i>	integer	Defines a Module with <i>ModuleType</i> = "PrintModule" within the Device specified by CylinderLayout/@DeviceID . In a newspaper-Web Press, <i>PrintModule</i> corresponds to a single cylinder.

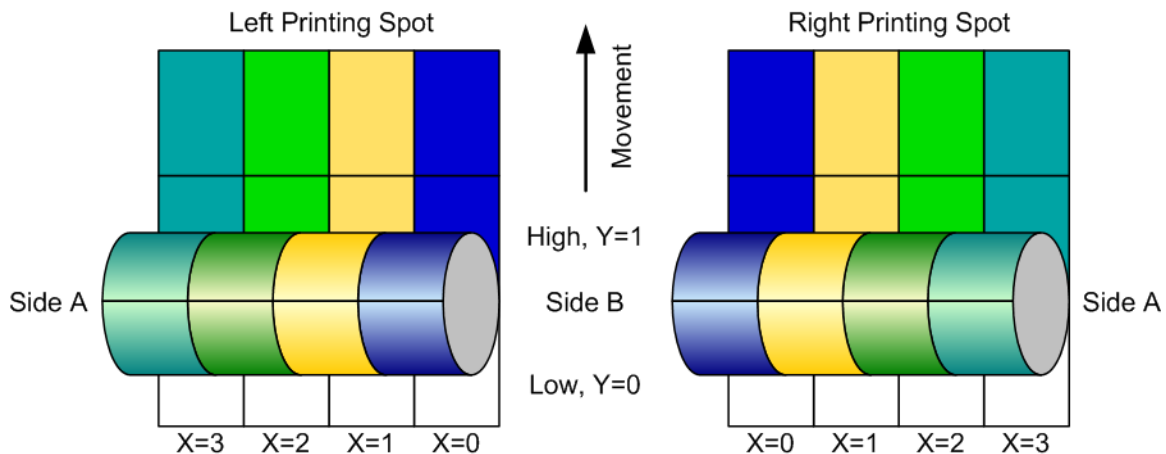


Figure 7-20: Definition of the PlatePosition Attribute on a newspaper-Web Press

In Figure 7-20, 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 7-21 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.

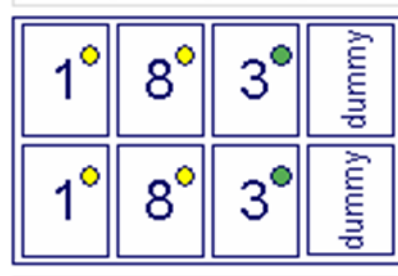


Figure 7-21: Example of a single physical section of eight pages

Example 7-20: CylinderLayout

The following **CylinderLayout** is an example representation of the cylinder layout as shown in the sketch.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Bundle" Status="Waiting"
  Type="CylinderLayoutPreparation" JobPartID="ID20" Version="1.4">
  <ResourcePool>
    <CylinderLayoutPreparationParams ID="CL002" Class="Parameter"
      Status="Available" >
      <ProductionPath/>
    </CylinderLayoutPreparationParams>
    <RunList ID="R-002" Class="Parameter" Status="Available" />
    <Device ID="DEV-001" Manufacturer="MAN" ModelName="GEOMAN" Status="Available"
      Class="Implementation" DeviceID="DEV-001">
      <Module ModuleIndex="0" ModuleType="Folder" ModelName="Folder 1">
        <Module ModuleIndex="1" ModuleType="PrintUnit"
          DescriptiveName="PU-1">
          <Module ModuleIndex="2" SubModuleIndex="0"
            ModuleType="PrintModule" DescriptiveName="PM-1"/>
          <Module ModuleIndex="3" SubModuleIndex="1"
            ModuleType="PrintModule" DescriptiveName="PM-2"/>
        </Module>
      </Module>
    </Device>
  </ResourcePool>
</JDF>
```

```

    <Module ModuleIndex="4" SubModuleIndex="2"
      ModuleType="PrintModule" DescriptiveName="PM-3" />
    <Module ModuleIndex="5" SubModuleIndex="3"
      ModuleType="PrintModule" DescriptiveName="PM-4" />
    <Module ModuleIndex="6" SubModuleIndex="4"
      ModuleType="PrintModule" DescriptiveName="PM-5" />
    <Module ModuleIndex="7" SubModuleIndex="5"
      ModuleType="PrintModule" DescriptiveName="PM-6" />
    <Module ModuleIndex="8" SubModuleIndex="6"
      ModuleType="PrintModule" DescriptiveName="PM-7" />
    <Module ModuleIndex="9" SubModuleIndex="7"
      ModuleType="PrintModule" DescriptiveName="PM-8" />
  </Module>
</Module>
</Device>
<Layout ID="L-001" Class="Parameter" Status="Available" />
<CylinderLayout ID="CL-001" Class="Parameter" Status="Available"
  PartIDKeys="WebSetup PlateLayout Separation"
  DeviceID="DEV-001">
  <LayoutRef rRef="L-001" />
  <CylinderLayout WebSetup="Run-1">
    <CylinderLayout PlateLayout="PL-001">
      <CylinderLayout Separation="Yellow">
        <CylinderPosition DeviceModuleIndex="2" PlatePosition="0 0"
          PlateType="Exposed" PlateUsage="Original" />
        <!-- page 1 -->
        <CylinderPosition DeviceModuleIndex="2" PlatePosition="0 1"
          PlateType="Exposed" PlateUsage="Original" />
        <!-- page 1 -->
      </CylinderLayout>
    </CylinderLayout>
    <CylinderLayout PlateLayout="PL-002">
      <CylinderLayout Separation="Yellow">
        <CylinderPosition DeviceModuleIndex="2" PlatePosition="1 0"
          PlateType="Exposed" PlateUsage="Original" />
        <!-- page 8 -->
        <CylinderPosition DeviceModuleIndex="2" PlatePosition="1 1"
          PlateType="Exposed" PlateUsage="Original" />
        <!-- page 8 -->
      </CylinderLayout>
    </CylinderLayout>
    <CylinderLayout PlateLayout="PL-003">
      <CylinderLayout Separation="HKS57">
        <CylinderPosition DeviceModuleIndex="2" PlatePosition="2 0"
          PlateType="Exposed" PlateUsage="Reuse" />
        <!-- page 3 -->
        <CylinderPosition DeviceModuleIndex="2" PlatePosition="2 1"
          PlateType="Exposed" PlateUsage="Reuse" />
        <!-- page 3 -->
      </CylinderLayout>
    </CylinderLayout>
    <CylinderPosition DeviceModuleIndex="2" PlatePosition="3 0"
      PlateType="Dummy" PlateUsage="Reuse" />
    <CylinderPosition DeviceModuleIndex="2" PlatePosition="3 1"
      PlateType="Dummy" PlateUsage="Reuse" />
  </CylinderLayout>
</CylinderLayout>
</ResourcePool>
<ResourceLinkPool>
  <DeviceLink Usage="Input" rRef="DEV-001" />
  <LayoutLink Usage="Input" rRef="L-001" />
  <RunListLink Usage="Input" rRef="R-002" />
  <CylinderLayoutPreparationParamsLink Usage="Input" rRef="CL002" />
  <CylinderLayoutLink Usage="Output" rRef="CL-001" />

```

```
</ResourceLinkPool>
</JDF>
```

Example 7-21: CylinderLayout: Double-Spread-Page Plate

In case of a double-spread-page plate (or double-truck-page plate) the *CylinderPosition* MAY be set as:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Bundle" Status="Waiting"
  Type="CylinderLayoutPreparation" JobPartID="ID20" Version="1.4">
  <ResourcePool>
    <CylinderLayout ID="CL-001" Class="Parameter" Status="Available"
      PartIDKeys="WebSetup PlateLayout Separation"
      DeviceID="DEV-001">
      <!-- ... -->
      <!-- PlatePosition (XYPairRangeList)-->
      <CylinderPosition DeviceModuleIndex="2" PlatePosition="0 0 ~ 1 0"
        PlateType="Exposed" PlateUsage="Original"/>
      <!-- ... -->
    </CylinderLayout>
    <Layout ID="L-001" Class="Parameter" Status="Available"/>
    <Runlist ID="R-002" Class="Parameter" Status="Available"/>
  </ResourcePool>
  <ResourceLinkPool>
    <CylinderLayoutLink Usage="Output" rRef="CL-001"/>
    <LayoutLink Usage="Input" rRef="L-001"/>
    <RunlistLink Usage="Input" rRef="R-002"/>
  </ResourceLinkPool>
</JDF>
```

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 7-144: 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 7-145: 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: *DBDocTemplateLayout, Collecting, Gathering, Inserting*

Output of Processes: —

Table 7-146: DBRules Resource

Name	Data Type	Description
Comment +	element	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: *DBDocTemplateLayout, Verification*

Output of Processes: —

Table 7-147: DBSchema Resource

Name	Data Type	Description
<i>DBSchemaType</i>	enumeration	Database type. Values are: <i>CommaDelimited</i> <i>SQL</i> <i>XML</i>
Comment +	element	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, Collecting, Gathering, Inserting, Verification</i>
Output of Processes:	<i>Verification</i>

Table 7-148: DBSelection Resource

Name	Data Type	Description
<i>DataBase</i>	URL	URL of the database
<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:	<i>Delivery</i>
Output of Processes:	—

Table 7-149: 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. Values include those from: <i>DeliveryIntent/@Method</i>
<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.

Table 7-149: DeliveryParams Resource (Section 2 of 2)

Name	Data Type	Description
<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. Values include those from: DeliveryIntent/ServiceLevel
<i>Transfer?</i> New in JDF 1.2	enumeration	Describes the direction and responsibility of the transfer. 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, "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, "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, "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, "Pickup"</i>)]. This Contact specifies where the buyer is to pick up the merchandise.
<i>Company?</i> Deprecated in JDF 1.1	refelement	Address and further information of the addressee. In JDF 1.1 and beyond, use Contact/Company
<i>Contact *</i> 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.

7.2.55.1 Element: Drop

Table 7-150: Drop Element (Section 1 of 2)

Name	Data Type	Description
<i>Earliest?</i>	dateTime	Specified the earliest time after which the delivery is to be made. Default value is from: DeliveryIntent/@Earliest.
<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 without blank spaces in this Attribute. Default value is from: DeliveryParams/@Method Values include those from: 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. Default = DeliveryParams/@Pickup . Replaced with <i>Transfer</i> in JDF 1.2.

Table 7-150: Drop Element (Section 2 of 2)

Name	Data Type	Description
<i>Required?</i>	dateTime	Specifies the time by which the delivery is intended to be made. Default value is from: DeliveryParams/@Required
<i>ServiceLevel?</i> New in JDF 1.2	string	The service level of the specific carrier. Default value is from: DeliveryParams/@ServiceLevel Values include those from: DeliveryParams/@ServiceLevel
<i>TrackingID?</i> 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.
<i>Transfer?</i> New in JDF 1.2	enumeration	Describes the direction and responsibility of the transfer. Default value is from: DeliveryParams/@Transfer. Values are from: DeliveryParams/@Transfer.
Company? Deprecated in JDF 1.1	refelement	Address and further information of the addressee. Defaults to the value of Company specified in the root DeliveryParams Resource.
Contact* New in JDF 1.1	refelement	Address and further information of the Contact responsible for this delivery. Default = DeliveryParams/Contact .
DropItem +	element	A Drop MAY consist of multiple products, which are represented by their respective PhysicalResource Elements. Each DropItem describes an individual Resource that is part of this Drop.

7.2.55.2 Element: DropItem

Table 7-151: DropItem Element (Section 1 of 2)

Name	Data Type	Description
<i>ActualAmount?</i> 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.
<i>ActualTotalAmount?</i> 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.
<i>Amount?</i>	integer	Specifies the number of Physical Resources ordered. If <i>Amount</i> is not specified, defaults to the total amount of the Resource that is referenced by PhysicalResource .
<i>TotalAmount?</i> 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, pallets etc.). Whereas <i>TotalAmount</i> specifies the number of final products, (e.g., books, magazines etc.).
<i>TotalDimensions?</i> New in JDF 1.3	Shape	Total dimensions in points of all individual items including packaging delivered in this drop.
<i>TotalVolume?</i> New in JDF 1.3	double	Total volume in liters of all individual items including packaging delivered in this drop.

Table 7-151: Droptem Element (Section 2 of 2)

Name	Data Type	Description
<i>TotalWeight</i> ? New in JDF 1.3	double	Total weight in gram of all individual items including packaging delivered in this drop.
<i>TrackingID</i> ? 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 <i>Drop/@TrackingID</i> .
<i>Unit</i> ?	string	Unit of measurement for the <i>Amount</i> of the Resource that is referenced by <i>PhysicalResource</i> . Default value is from: <i>PhysicalResource/@Unit</i>
<i>PhysicalResource</i> ? 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:	<i>ColorControlStrip</i> , <i>Layout/MarkObject</i>
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-152: 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 <i>MarkObject</i> that contains this mark. If the measuring field is defined within a <i>ColorControlStrip</i> , <i>Center</i> refers to the rectangle defined by <i>Center</i> and <i>Size</i> of the <i>ColorControlStrip</i> .
<i>Density</i> Modified in JDF 1.1A	DoubleList	Density value for each process color measured with filter. The data type was modified to <i>NumberList</i> 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 <i>CMYKColor</i> .
<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 <i>DensityMeasuringField</i> to. When <i>DensityMeasuringField</i> is used as an Element, it is a standard Attribute, otherwise when <i>DensityMeasuringField</i> is used as a Resource, <i>Separation</i> MUST be defined as a <i>Separation</i> Partition Key.

Table 7-152: DensityMeasuringField Resource (Section 2 of 2)

Name	Data Type	Description
<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).
<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, ImageSetting</i>
Output of Processes:	—

Table 7-153: 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 3.8.5.3, Implementation Resource and Section 4.8, Capability and Constraint Definitions.

Resource Properties

Resource Class:	Implementation
Resource referenced by:	PhaseTime, DeviceFilter, IDInfo, Occupation, DeviceInfo, Queue, QueueFilter, DieLayout , DieLayoutProductionParams/ ConvertingConfig, InkZoneCalculationParams , PrintCondition , RollStand , StrippingParams
Example Partition:	—
Input of Processes:	<i>Any Process</i>
Output of Processes:	—

Table 7-154: Device Resource (Section 1 of 2)

Name	Data Type	Description
<i>DeviceFamily</i>? 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.
<i>DeviceID</i> ?	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 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.
<i>DeviceType</i> ?	string	Manufacturer type ID, including a revision stamp. Type of the Device. Used for grouping and filtering of Devices
<i>Directory</i>? 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 913 and [FileURL].
<i>FriendlyName</i>? New in JDF 1.1 Deprecated in JDF 1.4	string	Short user-friendly title. Deprecation note: starting with JDF 1.4, use <i>DescriptiveName</i> .
<i>ICSVersions</i>? New in JDF 1.3	NMTOKENS	CIP4 Interoperability Conformance Specification (ICS) Versions that this Device complies with. Values include those from: <i>JDF/@ICSVersions</i> (Table 3-5, "JDF Node" on page 47)
<i>JDFErrorURL</i>? New in JDF 1.2	URL	URL where, by default, the Device will post JDF output Job tickets that are aborted or in error and when <i>NodeInfo/@TargetRoute</i> is not specified. If <i>JDFErrorURL</i> is specified in the "file:" scheme, it MUST specify a directory. If not specified, <i>JDFErrorURL</i> defaults to the value of <i>JDFOutputURL</i> .
<i>JDFInputURL</i>? New in JDF 1.2	URL	URL where, by default, the Device can accept JDF input Job tickets. If <i>JDFInputURL</i> is specified in the "file:" scheme, it MUST specify a directory. The persistence of JDF tickets in this location is implementation dependent. If not specified, the Device does not accept JDF without a JMF SubmitQueueEntry.
<i>JDFOutputURL</i>? New in JDF 1.2	URL	URL where, by default, the Device will post JDF output Job tickets that are successfully completed and when <i>NodeInfo/@TargetRoute</i> is not specified. If <i>JDFOutputURL</i> is specified in the "file:" scheme, it MUST specify a directory.
<i>JDFVersions</i>? 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).

Table 7-154: Device Resource (Section 2 of 2)

Name	Data Type	Description
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.
ModelDescription? New in JDF 1.1	string	Long description for end user.
ModelName? New in JDF 1.1	string	Model name.
ModelNumber? New in JDF 1.1	string	Model number.
ModelURL? New in JDF 1.1	string	Web site for model.
SerialNumber? New in JDF 1.1	string	Serial number of the Device.
PresentationURL? New in JDF 1.1	string	<i>PresentationURL</i> specifies a URL to a Device-provided user interface for configuration, status, etc. For instance, if the Device has an embedded Web server, this is a URL to the configuration page hosted on that Web server.
SecureJMFURL? New in JDF 1.3	URL	URL of the Device port that will accept JMF Messages via the “https” protocol.
UPC? New in JDF 1.1	string	Universal Product Code for the Device. A 12-digit, all-numeric code that identifies the consumer package. Managed by the Uniform Code.
CostCenter?	element	MIS cost center ID.
DeviceCap* New in JDF 1.1	element	Description of the capabilities of the Device. The <i>DeviceCap</i> Elements are combined with a logical OR, (i.e., if a JDF resides within any parameter space defined by a <i>DeviceCap</i> , the Device can process the Job). For details see Section 7.3, Device Capability Definitions.
IconList? New in JDF 1.1	element	List of locations of icons that can be used to represent the Device.
Location? New in JDF 1.4	element	Description of the Device location.
Module* New in JDF 1.3	element	Individual Modules that are represented by this Device .

7.2.58.1 Element: IconList[New in JDF 1.1](#)The *IconList* is a list of individual icon descriptions.

Table 7-155: IconList Element

Name	Data Type	Description
Icon +	element	Individual icon description.

7.2.58.2 Element: Icon

[New in JDF 1.1](#)

An Icon represents a Device in the user interface.

Table 7-156: Icon Element

Name	Data Type	Description
<i>Size</i>	XYPair	Height and width of the icon.
<i>BitDepth</i>	integer	Bit depth of one color.
<i>IconUsage?</i>	enumerations	Definition of the <i>Status</i> of the Device that this Icon represents. Any combination of values are allowed. Default value is: a list of all values (i.e. no limit on Icon use). Values are: <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> Note: The meaning of the individual enumerations is described in the DeviceInfo Message Element. See Section 5.8.3, KnownDevices.
<i>FileSpec</i>	element	Details of the file containing the icon data.

7.2.58.3 Element: Module

[New in JDF 1.3](#)

A Module represents a physical Machine or part of a Device.

Table 7-157: Module Element (Section 1 of 2)

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.

Table 7-157: Module Element (Section 2 of 2)

Name	Data Type	Description
<i>ModuleID</i> ?	string	Name of the Module. This is a unique identifier within the workflow. <i>ModuleID</i> MUST be the same over time for a specific Device instance, (i.e., MUST survive reboots). At least one of <i>ModuleID</i> or <i>ModuleIndex</i> MUST be specified. If multiple logical Devices share a physical Module, <i>ModuleID</i> MUST be identical. <i>ModuleID</i> 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 <i>ModuleID</i> or <i>ModuleIndex</i> MUST be specified. <i>ModuleIndex</i> SHOULD be used to specify identical modules, e.g., print modules in a complex Device.
<i>ModuleType</i> ?	NMTOKEN	Type of Module. Values include those from: Section C.2, "ModuleType Supported Strings" on page 879.. Note: the allowed values depend on the type of Device. Each type of Device has a separate table of values.
<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/MarkObject, LayoutPreparationParams, LayoutPreparationParams/PageCell
Example Partition:	<i>Side</i>
Input of Processes:	—
Output of Processes:	—

Table 7-158: DeviceMark Resource (Section 1 of 2)

Name	Data Type	Description
<i>Anchor</i> ?	Anchor	Anchor point within this DeviceMark that MarkObject/@CTM refers to.
<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> ?	double	The size of the font that is to be used for the DeviceMark , in points ≥ 0 . Modification note: starting with JDF 1.4, the data type is no longer integer.

Table 7-158: DeviceMark Resource (Section 2 of 2)

Name	Data Type	Description
HorizontalFitPolicy ? New in JDF 1.4	enumeration	Values are from: StripMark/@HorizontalFitPolicy
MarkJustification ? Deprecated in JDF 1.4	enumeration	Description of the preferred DeviceMark justification. Interpreted in context of the <i>MarkOrientation</i> . Values are: <i>Center</i> <i>Left</i> <i>Right</i> Deprecation note: starting with JDF 1.4, use <i>Anchor</i> .
MarkOffset ? Deprecated in JDF 1.4	XYPair	Description of the preferred DeviceMark offset. Interpreted in context of the Device dependent default position in the coordinate system defined by <i>MarkOrientation</i> . Deprecation note: starting with JDF 1.4, use <i>Anchor</i> .
MarkOrientation ? Deprecated in JDF 1.4	enumeration	Description of the preferred DeviceMark orientation. Values are: <i>Horizontal</i> <i>Vertical</i> Deprecation note: starting with JDF 1.4, use <i>Anchor</i> .
MarkPosition ? Deprecated in JDF 1.4	enumeration	Description of the preferred DeviceMark position. Values are: <i>Top</i> <i>Bottom</i> <i>Left</i> <i>Right</i> Deprecation note: starting with JDF 1.4, use <i>Anchor</i> .
VerticalFitPolicy ? New in JDF 1.4	enumeration	Values are from: StripMark/@VerticalFitPolicy
BarcodeReproParams ? New in JDF 1.4	refelement	Reproduction parameters for Barcodes specified in the parent MarkObject/IdentificationField .

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/DeviceNSpace/@ColorList** Attribute gives the set of **DeviceNSpace** colorant percent values necessary to construct the,
- **ColorantControl/@ColorantParams** colorant (also named **ColorantControl/ColorPool/Color/@Name**) in using **DeviceNSpace** colorants.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	ColorantControl, ColorSpaceConversionOp
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-159: 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:	<i>DieDesign, DieMaking</i>
Output of Processes:	<i>DieDesign, DieLayoutProduction</i>

Table 7-160: DieLayout Resource (Section 1 of 2)

Name	Data Type	Description
<i>DieSide</i> ? New in JDF 1.4	enumeration	Determines the die side for which the DieLayout is made. Values are: <i>Up</i> – the DieLayout is made with the knives pointing upwards. <i>Down</i> – the DieLayout is made with the knives pointing downwards.
<i>MediaSide</i> ? New in JDF 1.4	enumeration	Determines the printing side for which the DieLayout is made. Values are: <i>Front</i> – for a box this corresponds to the outside of a box. <i>Back</i> – for a box this corresponds to the inside of a box.
<i>Rotated</i> ? New in JDF 1.4	boolean	Indicates if some of the structural designs are oriented cross grain/flute in the layout.

Table 7-160: DieLayout Resource (Section 2 of 2)

Name	Data Type	Description
<i>Waste</i> ? New in JDF 1.4	double	The percent of the material that is wasted. Inner waste i.e. cut out windows are not included in the waste.
<i>Device</i> * New in JDF 1.4	refelement	The Devices for which this DieLayout was made (printing press and die cutter). Typically only the type of Device would be used e.g. the model of the die cutter.
<i>FileSpec</i> ?	refelement	Reference to an external URL that represents the die.
<i>Media</i> ? New in JDF 1.4	refelement	Media for which this DieLayout was intended. The Media description defines important design parameters as the type of Media , dimensions, grain direction or flute direction.
<i>RuleLength</i> * New in JDF 1.4	element	Elements describing the length of die rules for the different types of rules. Each <i>RuleLength</i> Element describes the accumulated length of all rules of a certain type.
<i>Station</i> *	element	Description of the stations in a DieLayout . One <i>Station</i> produces one shape.

7.2.61.1 Element: RuleLength

[New in JDF 1.4](#)

Table 7-161: RuleLength Element

Name	Data Type	Description
<i>DDESCutType</i>	integer	Type of rule. Values include: a number between "0" and "999" corresponding to a line type as defined in DDES.
<i>Length</i>	double	Accumulated length of the rules of this type in the DieLayout (pt).

7.2.61.2 Element: Station

Table 7-162: Station Element

Name	Data Type	Description
<i>AssemblyIDs</i> ? New in JDF 1.3	NMTOKENS	The list of <i>AssemblyIDs</i> of the graphic elements that are processed by this <i>Station</i> . Note: <i>AssemblyIDs</i> was added to JDF 1.3 Errata.
<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 .
<i>ShapeDef</i> ? New in JDF 1.4	refelement	The ShapeDef corresponding to this station in the DieLayout .

7.2.62 DieLayoutProductionParams

[New in JDF 1.4](#)

Parameters for the die layout.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—

Input of Processes: *DieLayoutProduction*

Output of Processes: —

Table 7-163: DieLayoutProductionParams Resource

Name	Data Type	Description
<i>Estimate ?</i>	boolean	Determines if the Process runs in estimate mode or not. When in estimate mode multiple solutions are generated.
<i>Position ?</i>	Anchor	The position of layout on the sheet.
ConvertingConfig +	element	A ConvertingConfig Element describes a range of Sheet sizes that can be taken into account to create a new DieLayout . Typically a ConvertingConfig will correspond to 1 (printing press, die cutter) combination.
RepeatDesc +	element	Step and repeat parameters for a ShapeDef . There is either a single RepeatDesc giving the parameters for all ShapeDef Resources at the input or there is exactly 1 RepeatDesc per ShapeDef in the input in which case the sequence of both determines which RepeatDesc should be used for a ShapeDef .

7.2.62.1 Element: ConvertingConfig

[New in JDF 1.4](#)

The ConvertingConfig Element describes a range of Sheet sizes.

Table 7-164: ConvertingConfig Element

Name	Data Type	Description
<i>MarginBottom ?</i>	double	The bottom margin for positioning the layout on the Sheet.
<i>MarginLeft ?</i>	double	The left margin for positioning the layout on the Sheet.
<i>MarginRight ?</i>	double	The right margin for positioning the layout on the Sheet.
<i>MarginTop ?</i>	double	The top margin for positioning the layout on the Sheet.
<i>SheetHeight</i>	DoubleRange	The minimum to maximum Sheet height (pt).
<i>SheetWidth</i>	DoubleRange	The minimum to maximum Sheet width (pt).
Device *	refelement	The target devices (printing press and die cutter) corresponding to this configuration. Typically only the type of Device would be used e.g. the model of the die cutter.

7.2.62.2 Element: RepeatDesc

[New in JDF 1.4](#)

The RepeatDesc Element describes the layout specs for a **ShapeDef**.

Table 7-165: RepeatDesc Element (Section 1 of 2)

Name	Data Type	Description
<i>AllowedRotate ?</i>	enumeration	Allowed methods to rotate structural designs in with respect to grain/flute. Values are: <i>None</i> – No Rotation at all. <i>Grain</i> – 0° or 180° Rotation. <i>MinorGrain</i> – device dependent small rotations that retain the general grain direction, e.g. +/- 10° <i>CrossGrain</i> – Cross grain rotations, e.g. 90° are acceptable.
<i>GutterX ?</i>	double	Gutter between columns (see also <i>GutterX2</i>)

Table 7-165: RepeatDesc Element (Section 2 of 2)

Name	Data Type	Description
<i>GutterX2?</i>	double	Secondary gutter between columns. When the <i>LayoutStyle</i> = "Reverse2ndColumn", the gutter between columns (2n+1) and (2n+2) is <i>GutterX</i> and between columns (2n+2) and (2n+3) is <i>GutterX2</i> . When <i>GutterX2</i> is not specified <i>GutterX2</i> = <i>GutterX</i> . See Figure 7-28, "RepeatDesc/@GutterX2 and @GutterY2: Secondary Gutters" on page 512
<i>GutterY?</i>	double	Gutter between rows (see also <i>GutterY2</i>).
<i>GutterY2?</i>	double	Secondary gutter between rows. When the <i>LayoutStyle</i> = "Reverse2ndRow" the gutter between rows (2n+1) and (2n+2) is <i>GutterY</i> and between rows (2n+2) and (2n+3) <i>GutterY2</i> . When <i>GutterY2</i> is not specified <i>GutterY2</i> = <i>GutterY</i> . See Figure 7-28, "RepeatDesc/@GutterX2 and @GutterY2: Secondary Gutters" on page 512
<i>LayoutStyle?</i>	NMTOKENS	The allowed styles for the Layout Values include: <i>StraightNest</i> <i>Reverse2ndRow</i> <i>Reverse2ndRowAligned</i> <i>Reverse2ndColumn</i> <i>Reverse2ndColumnAligned</i> Note: for diagrams of the above values, see Figure 7-22, "Basic Shape for RepeatDesc/@LayoutStyle Examples" on page 509 and the following five figures
<i>OrderQuantity?</i>	integer	The order quantity for the 1-up for which this layout will be optimized. This information needs to be present when a Layout is being made for more than 1 ShapeDef .
<i>UseBleed?</i>	boolean	If true, the print bleed defined in the structural design is used to calculate the layout. If false, the outer cut is used.

The following Figure shows the basic shape for subsequent Figures. that relate to RepeatDesc.

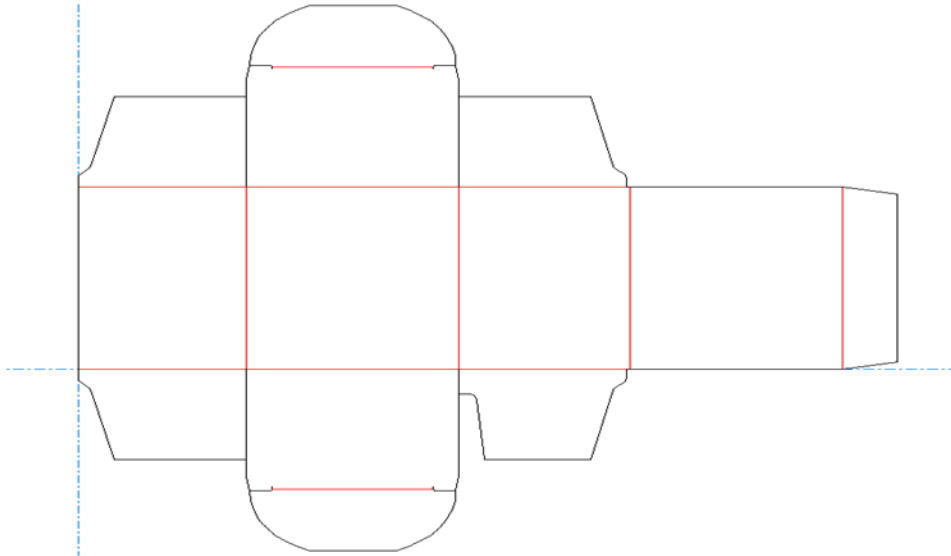


Figure 7-22: Basic Shape for RepeatDesc/@LayoutStyle Examples

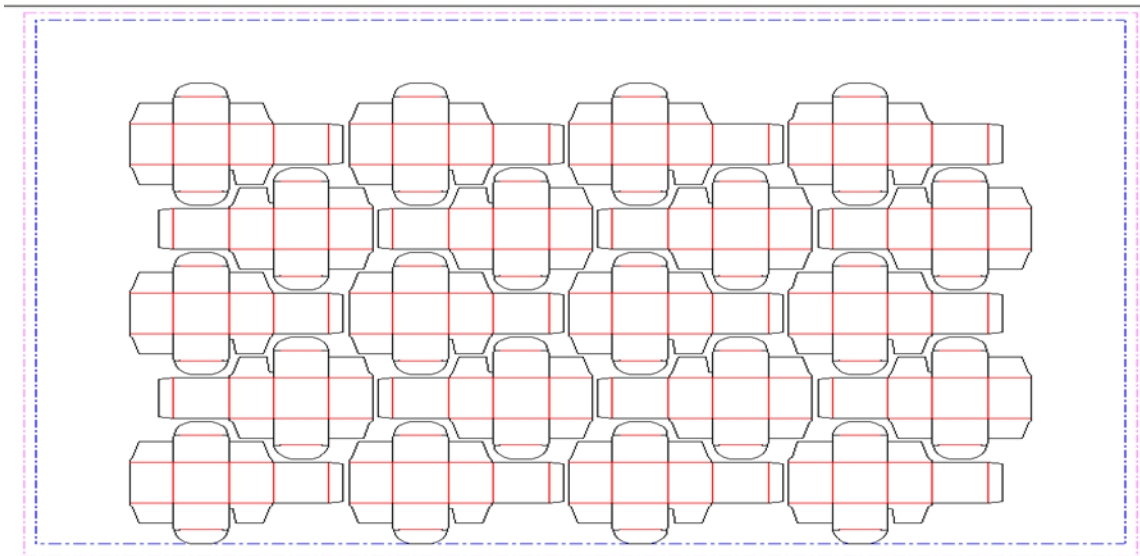


Figure 7-23: RepeatDesc/@LayoutStyle = "StraightNest"

In the following Figure, 1-ups on even rows are rotated 180 degrees. Even rows are shifted horizontally and vertically to obtain optimal nesting

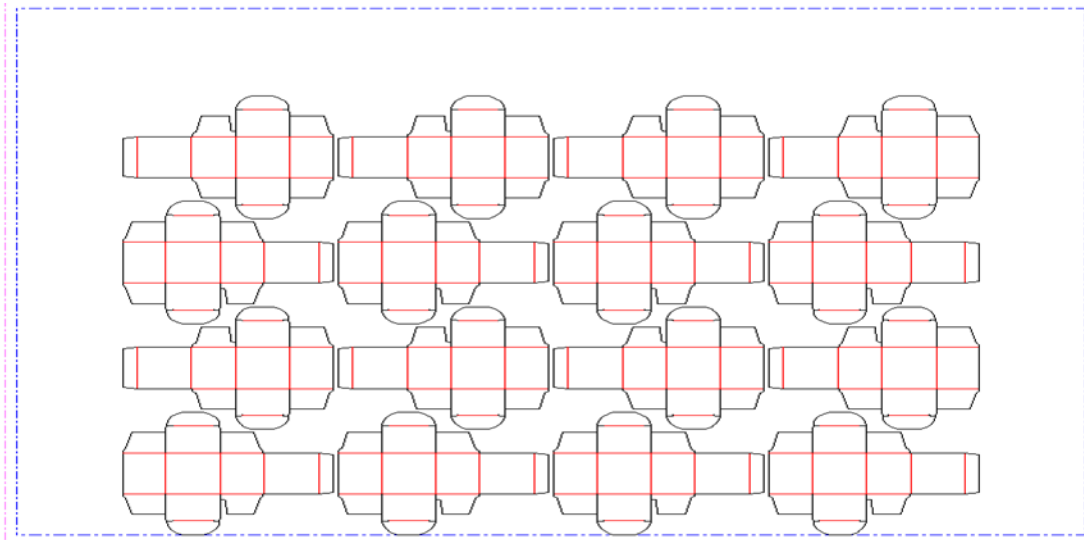


Figure 7-24: RepeatDesc/@LayoutStyle = "Reverse2ndRow"

In the following Figure, 1-ups on even rows are rotated 180 degrees. Even rows are shifted vertically to obtain optimal nesting. The even rows are not shifted horizontally. (Left and right edges are aligned between rows)

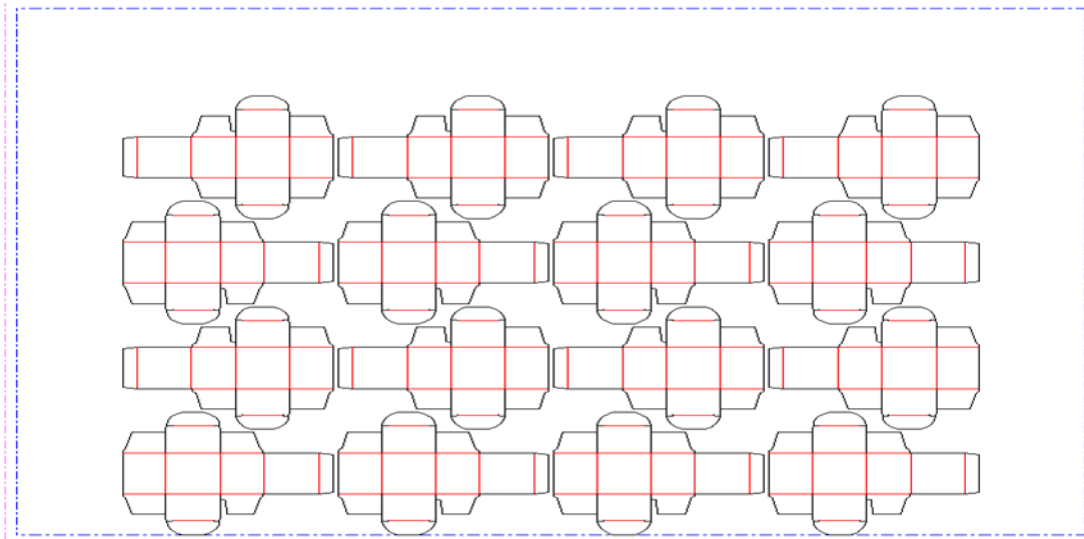


Figure 7-25: RepeatDesc/@LayoutStyle = "Reverse2ndRowAligned"

In the following Figure, 1-ups on even columns are rotated 180 degrees. Even columns are shifted vertically and horizontally to obtain optimal nesting.

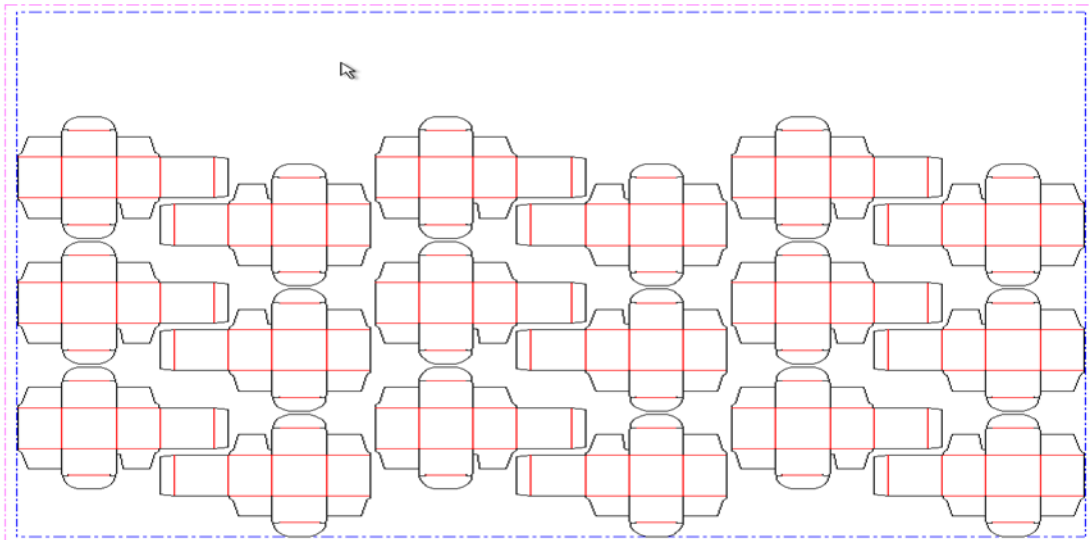


Figure 7-26: RepeatDesc/@LayoutStyle = "Reverse2ndColumn"

In the following Figure, 1-ups on even columns are rotated 180 degrees. Even columns are shifted horizontally to obtain optimal nesting. No vertical shifting of even columns is done (top and bottom edges are aligned between columns).

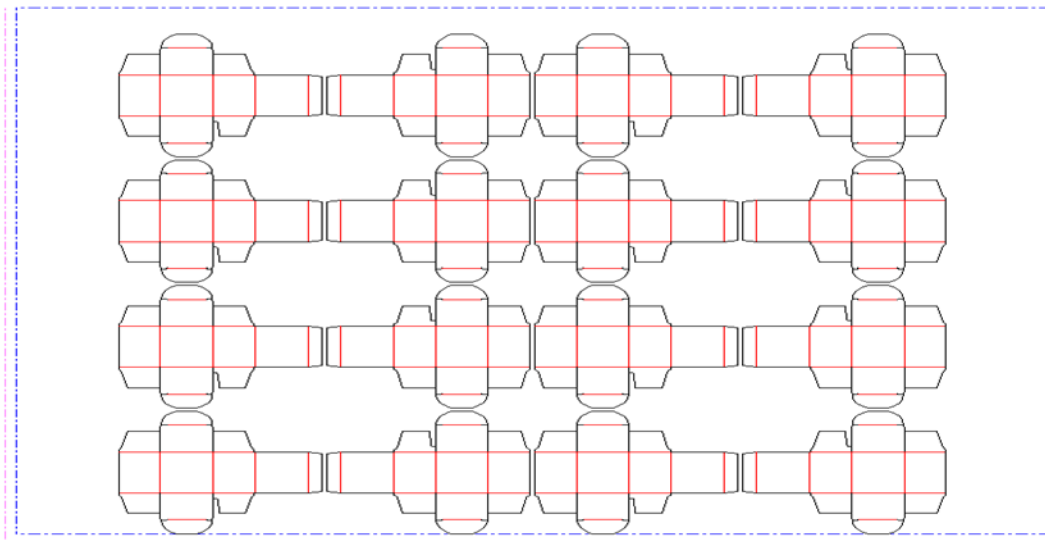


Figure 7-27: RepeatDesc/@LayoutStyle = "Reverse2ndColumnAligned"

In the following Figure, *LayoutStyle* = "Reverse2ndRow", *GutterY* = "15", *GutterY2* = "0"

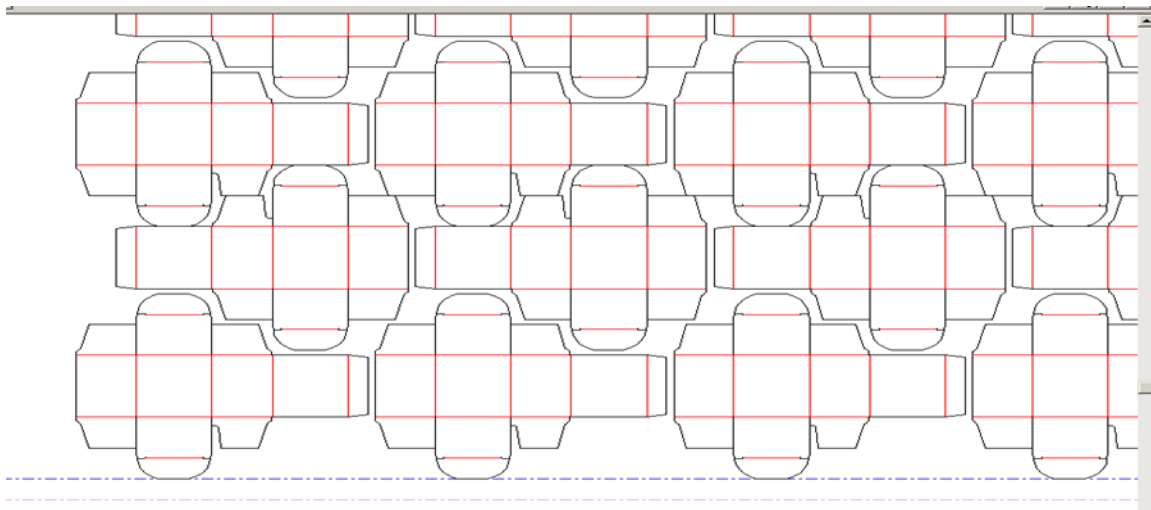


Figure 7-28: RepeatDesc/@GutterX2 and @GutterY2: Secondary Gutters

7.2.63 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:	<i>DigitalDelivery</i>
Output of Processes:	—

Table 7-166: DigitalDeliveryParams Resource (Section 1 of 2)

Name	Data Type	Description
<i>DigitalDeliveryDirection</i> ?	enumeration	Describes which side activates the delivery. Values are: <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>

Table 7-166: DigitalDeliveryParams Resource (Section 2 of 2)

Name	Data Type	Description
<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> <i>Vio</i> – a digital delivery service brand <i>WAMNET</i> – a digital delivery service brand
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 <i>DigitalDelivery</i> 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 974. for a set of examples.

7.2.64 DigitalMedia[New in JDF 1.2](#)

This Resource represents a processed removable digital media-based Handling Resource such as tape or removable disk.

Resource Properties

Resource Class:	Handling
Resource referenced by:	ArtDeliveryIntent/ArtDelivery
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-167: DigitalMedia Resource (Section 1 of 2)

Name	Data Type	Description
<i>Capacity?</i>	integer	Size of the digital media in megabytes.
<i>MediaLabel?</i>	string	Electronic label of the media.

Table 7-167: DigitalMedia Resource (Section 2 of 2)

Name	Data Type	Description
<i>Media Type</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.

7.2.65 DigitalPrintingParams

This Resource contains Attributes and Elements used in executing the *DigitalPrinting* Process. The *PrintingType* Attribute in this Resource defines two types of printing: *SheetFed* and *WebFed*. The principal difference between them is the shape of the paper each is equipped to accept. Presses that execute *WebFed* Processes use substrates that are continuous and cut after printing is accomplished. Most newspapers are printed on Web Presses. *SheetFed* printing, on the other hand, accepts precut substrates.

7.2.65.1 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 7-168: DigitalPrintingParams Resource (Section 1 of 4)

Name	Data Type	Description
<p><i>Collate</i> ? New in JDF 1.1</p>	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>Values are:</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>Example: two documents, A and B, each have two Sheets, A1, A2 and B1, B2. The number of document copies requested is one for both documents and the number of Job copies requested is three (Component/@Amount = 3). The Job contains no Document Set boundaries.</p> <p>If <i>Collate</i> = "<i>None</i>", the Sheet order will be: A1A1A1 A2A2A2 B1B1B1 B2B2B2</p> <p>If <i>Collate</i> = "<i>Sheet</i>", the Sheet order will be: A1A2 A1A2 A1A2 B1B2 B1B2 B1B2</p> <p>If <i>Collate</i> = "<i>SheetAndSet</i>" or "<i>SheetSetAndJob</i>", the Sheet order will be: A1A2 B1B2 A1A2 B1B2 A1A2 B1B2</p>
<p><i>DirectProofAmount</i> = "0" New in JDF 1.2</p>	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>
<p><i>ManualFeed</i> = "<i>false</i>" New in JDF 1.1</p>	boolean	<p>Indicates whether the media will be fed manually.</p>

Table 7-168: DigitalPrintingParams Resource (Section 2 of 4)

Name	Data Type	Description
<i>NonPrintableMarginBottom</i> ? 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 <i>DigitalPrinting</i> Process. The <i>DigitalPrinting</i> Process MUST put marks up to, but not in, the non-printable margin area. The <i>Media</i> 's origin is unaffected by <i>NonPrintableMarginBottom</i> . These margins are independent of the PDL content.
<i>NonPrintableMarginLeft</i> ? New in JDF 1.2	double	Same as <i>NonPrintableMarginBottom</i> except for the left margin.
<i>NonPrintableMarginRight</i> ? New in JDF 1.2	double	Same as <i>NonPrintableMarginBottom</i> except for the right margin.
<i>NonPrintableMarginTop</i> ? New in JDF 1.2	double	Same as <i>NonPrintableMarginBottom</i> except for the top margin.
<i>OutputBin</i> ? New in JDF 1.1 Modified in JDF 1.2	NMTOKEN	Specifies the bin to which the finished document is to be output. Values include those from: Table C-21, "Input Tray and Output Bin Names" on page 889
<i>PageDelivery</i> ? New in JDF 1.1	enumeration	Indicates how pages are to be delivered to the output bin or finisher. Note: these values refer to the orientation of the entire stack being output from the press, not individual sheets. For example, <i>SameOrderFaceDown</i> means that the stack can be picked up and turned over to find the output sheets in the same order as the input <i>RunList</i> with the first page on top facing up Values are: <i>FanFold</i> – The output is alternating face-up, face down. <i>SameOrderFaceUp</i> – Order as defined by the <i>RunList</i> , with the "Front" sides of the media up and the first Sheet on top. <i>SameOrderFaceDown</i> – Order as defined by the <i>RunList</i> , with the "Front" 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 "Front" sides of the media up and the last Sheet on top. <i>ReverseOrderFaceDown</i> – Sheet order reversed compared to <i>SameOrderFaceDown</i> , with the "Front" sides of the media down and the last Sheet on the bottom.
<i>PrintingType</i> ? Modified in JDF 1.2	enumeration	Type of printing Machine. Values are: <i>ContinuousFed</i> – connected Sheets including fan fold. New in JDF 1.2 <i>SheetFed</i> <i>WebFed</i>

Table 7-168: DigitalPrintingParams Resource (Section 3 of 4)

Name	Data Type	Description
PrintQuality? Deprecated in JDF 1.1	enumeration	Indicates how pages are to be delivered to the output bin or finisher. 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 Deprecation note: starting with JDF 1.1, use InterpretingParams/@PrintQuality
SheetLay?	enumeration	Lay of input media. Reference edge of where paper is placed in feeder. Values are: <i>Left</i> <i>Right</i> <i>Center</i>
Sides? New in JDF 1.3	enumeration	Indicates whether the ByteMap MUST be imaged on one or both sides of the media. If the RunList(Surfaces) input to DigitalPrinting is Partitioned by <i>Side</i> (either explicitly or implicitly using the RunList/@SheetSides Attribute), then the input RunList provides a binding of front and back surfaces to sheets. If @Sides = " <i>OneSidedFront</i> " or " <i>OneSidedBack</i> ", then that binding is ignored and one surface is imaged per sheet. If the RunList (Surfaces) does not provide the binding of surfaces to sides, then the Sides Attribute specifies the binding to be applied. When a different value for this Attribute is encountered, it 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. Values are from: LayoutPreparationParams/@Sides
ApprovalParams? New in JDF 1.2	refelement	Details of the direct approval Process, when <i>DirectProofAmount</i> > 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 Deprecated in JDF 1.4	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. Deprecation note: starting with JDF 1.4, use the Varnishing Process .

Table 7-168: DigitalPrintingParams Resource (Section 4 of 4)

Name	Data Type	Description
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.

7.2.66 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, StackingParams**

Example Partition: —

Input of Processes: —

Output of Processes: —

Table 7-169: 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 <i>OffsetDirection</i> simultaneously.
OffsetDirection ?	enumeration	Offset-shift action for the first component. A component can be offset to one of two positions—left or right. 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 <i>Inserting</i> 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.

Table 7-169: Disjuncting Resource (Section 2 of 2)

Name	Data Type	Description
InsertSheet ?	refelement	Some kind of physical marker (e.g., a paper strip or a yellow paper Sheet) that separates the components.

7.2.67 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: ResourcePullParams, QueueSubmissionParams, SubmitQueueEntry/QueueSubmissionParams, FileSpec, RunList

Example Partition: —

Input of Processes: —

Output of Processes: —

Table 7-170: Disposition Resource (Section 1 of 2)

Name	Data Type	Description
<i>DispositionAction</i> = "Delete"	enumeration	Values are: <i>Delete</i> – The asset is deleted when disposition time occurs. <i>Archive</i> – The asset is archived when disposition time occurs.
<i>DispositionUsage</i> ?	enumeration	Specifies the usage of the asset by the Process. Default behavior: Disposition applies to all Processes that link to the Disposition Resource (if <i>DispositionUsage</i> not specified). Values are: <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.

Table 7-170: Disposition Resource (Section 2 of 2)

Name	Data Type	Description
<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.

7.2.68 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 1026 for details of this deprecated Resource.

7.2.69 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/ContentData, LayoutElement, PageList, PageList/ PageData
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-171: 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. Values include those from: the ICC Manufacturer's Signature Registry at http://www.color.org . Example values: "ACME" for the Acme Corp. CMM.
<i>ICCOutputProfileUsage?</i>	enumeration	This Attribute specifies the usage of the output intent profile or specified printing condition from the PDL. 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.
<i>AutomatedOverPrintParams?</i>	refelement	A Resource that provides controls for the automated selection of overprinting of black text or graphics.

Table 7-171: ElementColorParams Resource (Section 2 of 2)

Name	Data Type	Description
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 Resource. Multiple ColorantAlias elements with identical values of ColorantAlias/@ReplacementColorantName MUST NOT be specified in the same ElementColorParams resource context.
ColorSpaceConversionOp ?	element	List of ColorSpaceConversionOp Subelements, each of which identifies a type of object, defines the source color space for that type of object, and specifies the behavior of the conversion operation for that type of object. If not present, the default conversion behavior is derived from ColorStandard . ColorSpaceConversionOp/@Operation is ignored in the context of ElementColorParams .
FileSpec (<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 .

7.2.70 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 7-172: EmbossingParams Resource

Name	Data Type	Description
Emboss *	element	One Emboss Element is specified for each impression.

7.2.70.1 Element: Emboss

Table 7-173: Emboss Element (Section 1 of 2)

Name	Data Type	Description
<i>Direction</i> Modified in JDF 1.3	enumeration	The direction of the image. 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). Values are: <i>Rounded</i> <i>Beveled</i>
<i>EmbossingType</i> Modified in JDF 1.3	enumeration	Values are: <i>BlindEmbossing</i> – Embossed forms that are not inked or foiled. The color of the image is the same as the paper. <i>Braille</i> – 6 dot <i>Braille</i> embossing. Note: <i>Braille</i> was added to JDF 1.3 Errata. New in JDF 1.3 <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. 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 .
<i>IdentificationField</i> ? New in JDF 1.4	refelement	If <i>EmbossingType</i> = " <i>Braille</i> ", <i>IdentificationField</i> describes the content of the Braille Element.

Table 7-173: Emboss Element (Section 2 of 2)

Name	Data Type	Description
Media ? New in JDF 1.4	refelement	If the <i>EmbossingType</i> = "FoilEmbossing" or "FailStamping", Media describes the foil.
Tool ? New in JDF 1.4	refelement	The tool used to make the embossing described by this Element.

7.2.71 Employee

Information about a specific Device or Machine operator (see Section 3.8.5.3, Implementation Resource). **Employee** is also used to describe the contact person who is responsible for executing a Node, as defined in **NodeInfo**.

Resource Properties

Resource Class: Implementation

Resource referenced by: Abstract Audit, Notification, PhaseTime, ModulePhase, JMF, Message, Occupation, DeviceInfo, ModuleStatus, **ContentList/ContentData/ContentMetadata**, **NodeInfo**

Example Partition: —

Input of Processes: *Any Process*

Output of Processes: —

Table 7-174: Employee Resource

Name	Data Type	Description
<i>PersonID?</i>	string	ID of the relevant MIS employee.
Roles ? New in JDF 1.2 Modified in JDF 1.4	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>StandBy</i> – Employee who is allocated to a specific task on demand. New in JDF 1.4
<i>Shift?</i>	string	Defines the shift to which the employee belongs.
CostCenter ?	element	MIS cost center ID.
Person ?	refelement	Describes the employee. If no Person Resource is specified, the Employee Resource represents any employee who fulfills the selection criteria.

7.2.72 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:	<i>EndSheetGluing</i>
Output of Processes:	—

Table 7-175: EndSheetGluingParams Resource

Name	Data Type	Description
EndSheet (<i>Front</i>)	element	Information about the front-end Sheet. The <i>Side</i> Attribute of this Element MUST be " <i>Front</i> ".
EndSheet (<i>Back</i>)	element	Information about the back-end Sheet. The <i>Side</i> Attribute of this Element MUST be " <i>Back</i> ".

7.2.72.1 Element: EndSheet**Table 7-176: EndSheet Element**

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 Element's ComponentLink.
<i>Side</i>	enumeration	Location of the end Sheet. Values are: <i>Front</i> <i>Back</i>
GlueLine	element	Description of the glue line.

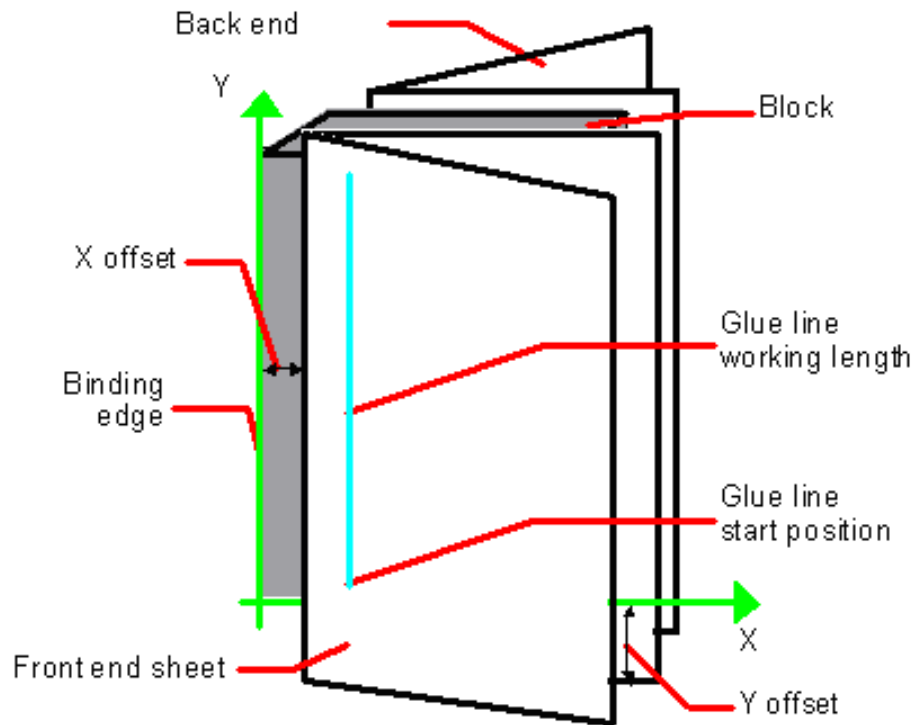


Figure 7-29: Parameters and coordinate system used for end-Sheet gluing

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).

7.2.73 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:	ArtDeliveryIntent/ArtDelivery
Example Partition:	<i>DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, Separation, SheetName, Side, SignatureName, TileID, WebName</i>
Input of Processes:	<i>Bending, ContactCopying, ConventionalPrinting, DigitalPrinting, ImageSetting, PreviewGeneration, Scanning, Varnishing</i>
Output of Processes:	<i>Bending, ContactCopying, ImageSetting</i>

Table 7-177: ExposedMedia Resource

Name	Data Type	Description
<i>ColorType</i> ?	enumeration	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 <i>PageData</i> Elements of the <i>PageList</i> specified in this <i>ExposedMedia</i> .
<i>PlateType</i> ? New in JDF 1.3	enumeration	Specifies whether a plate is exposed or a dummy plate. Values are: <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 <i>ExposedMedia</i> specifies a proof, <i>ProofName</i> is the name of the <i>ProofingIntent</i> / <i>ProofItem</i> that specified this proof in the Product Intent section.
<i>ProofQuality</i> ? Modified in JDF 1.2	enumeration	This Attribute is present if the <i>ExposedMedia</i> Resource describes a proof. 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	Values are: <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. Values include: <i>Bacher</i> <i>Stoesser</i> If not specified, no holes are punched.
<i>Resolution</i> ?	XYPair	Resolution of the output.
<i>FileSpec</i> (<i>OutputProfile</i>) ?	refelement	A <i>FileSpec</i> Resource pointing to an ICC profile that describes the output Process for which this media was exposed.
<i>Media</i>	refelement	Describes media specifics such as size and type.
<i>PageList</i> ? New in JDF 1.3	refelement	Specification of page metadata for pages described by this <i>ExposedMedia</i> .
<i>ScreeningParams</i> ?	refelement	Used to describe the screening in case of rasterized media.

7.2.74 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 7-178: 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.

7.2.75 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:	<i>Feeding</i>
Output of Processes:	—

Table 7-179: 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.

7.2.75.1 Element: Feeder

Table 7-180: Feeder Element (Section 1 of 3)

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.

Table 7-180: Feeder Element (Section 2 of 3)

Name	Data Type	Description
<i>FeederSynchronization</i> = "Primary"	enumeration	Specifies the synchronization of multiple Feeder Elements with identical Component Elements: Values are: <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> ? Modified in JDF 1.4	NMTOKEN	Specifies the feeder type. Values include: <i>AddOn</i> – Add on feeder, (e.g., CDs). <i>BookBlock</i> – A feeder for book blocks. New in JDF 1.4 <i>Folding</i> – A folding feeder that folds the input Component or Media . <i>Gluing</i> – A gluing feeder <i>Sheet</i> – Single Sheet feeder. <i>Signature</i> – Single Signature feeder.
<i>Loading</i> ?	NMTOKEN	Specifies the feeder loading. Values include: <i>Bundle</i> – Stream feeder, using the output of the <i>Bundling</i> Process. <i>FanFold</i> – Automatic loading of <i>FanFold Media</i> . <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 <i>PrintRolling</i> Process.
<i>Opening</i> = "None"	enumeration	Specifies the opening of Signatures: Values are: <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.
<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.).
Component ?	refelement	Specifies the Component that is to be loaded into this Feeder. This Component MUST be an input of the <i>Feeding</i> Process. Exactly one of Component or Media MUST be specified.
<i>FeederQualityParams</i> ?	element	Definition of the setup and policy for feeding quality.

Table 7-180: Feeder Element (Section 3 of 3)

Name	Data Type	Description
<i>Media</i> ?	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.

7.2.75.2 Element: FeederQualityParams

The **FeederQualityParams** Element defines the setup and policy for feeding quality control. It can be specified individually for each **Feeder**.

Table 7-181: FeederQualityParams Element

Name	Data Type	Description
<i>IncorrectComponentQuality</i> ?	enumeration	Defines the operation of the incorrect components quality control: 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. Values are from: @ <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. Values are from: @ <i>IncorrectComponentQuality</i>
<i>BadFeeds</i> ?	integer	Number of consecutive bad feeds until the Device stops.

7.2.75.3 Element: CollatingItem**Table 7-182: CollatingItem Element (Section 1 of 2)**

Name	Data Type	Description
<i>Amount</i> = "1"	integer	Determines, how many consecutive items shall be consumed.

Table 7-182: CollatingItem Element (Section 2 of 2)

Name	Data Type	Description
<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. A <i>BundleDepth</i> value of "0" means the Component itself. A value of "1" addresses the BundleItem Elements referenced from the Component , i.e. the Component/Bundle/BundleItem/Component(Ref) , and so on
<i>Orientation</i> ?	Orientation	Named <i>Orientation</i> of the CollatingItem relative to the input coordinate system. For details see Table 2-4 on page 30. 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 ResourceLink .
<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 ResourceLink . At most one of <i>Orientation</i> and <i>Transformation</i> MUST be specified. If neither is specified, no transformation is applied.
<i>TransformationContext</i> = "StackItem"	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. Values are: <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 CollatingItem individually. <i>CollateItem</i> – apply to a CollatingItem as a whole. Note: If <i>Amount</i> = "1", Component and CollatingItem are referring to the same object and, therefore, result in the same output.
Component ?	refelement	References one of the input components to the Process to be (partially) consumed by the CollatingItem Element. This Component MUST be an input of the Feeding Process. Exactly one of Component or Media MUST be specified.
Media ?	refelement	References one of the input media to the Process to be consumed by the CollatingItem 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 **PrintRolling** Process.

7.2.76 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:	ApprovalSuccess/ApprovalDetails, AssetListCreationParams, ByteMap, Color, Color/PrintConditionColor, ColorCorrectionParams, ColorCorrectionParams/ColorCorrectionOp, ColorSpaceConversionOp, ColorSpaceConversionParams, DBMergeParams, Device/IconList/Icon, DieLayout, ElementColorParams, ExposedMedia, ExternalImpositionTemplate, FileSpec/Container, FileSpec/FileAlias, FormatConversionParams/TIFFFormatParams/TIFFEmbeddedFile, ImageReplacementParams, LayoutElement, PDLResourceAlias, PrintCondition, QualityControlResult, ScanParams, ShapeDef, ShapeDefProductionParams/ObjectModel, ShapeDefProductionParams/ShapeTemplate
Example Partition:	<i>Separation</i>
Input of Processes:	—
Output of Processes:	—

Table 7-183: 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.
<i>AppOS?</i> Modified in JDF 1.2	string	Operating system of the application that created the file. 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> Note: Additional values can be used from the IANA Operating System Names [iana-os] which allows up to 40 uppercase US ASCII alphabetical values as well as “-”, “_” and “/” — but only for values not covered by the above values. For example, “OS/2”. See “AppOS and OSVersion Attributes” on page 739 for combinations of <i>AppOS</i> and <i>OSVersion</i> values.

Table 7-183: FileSpec Resource (Section 2 of 6)

Name	Data Type	Description
<i>AppVersion?</i>	string	Version of the value of the <i>Application</i> Attribute. Values include: "8.1" "8.1 (4331)" "9.0.3 SR3437" Note: the values are only examples.
<i>Checksum?</i> 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.
<i>Compression = "None"</i> Modified in JDF 1.2	NMTOKEN	Indicates the compression or encoding for the entire file. This is not compression used internally within the file. Values include: <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].
<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. 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. Deprecation note: starting with JDF 1.2, retention of assets is specified in the Disposition Resource.
<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.

Table 7-183: FileSpec Resource (Section 3 of 6)

Name	Data Type	Description
<i>Encoding ?</i> New in JDF 1.4	string	Encoding or code page of the file contents. Values include those from: http://www.iana.org/assignments/character-sets .
<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. Values are from: "Generating strings with Format and Template" on page 909 Constraint: 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.
<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. Default behavior: the file is Device independent Value format is from: IEEE 1284-2000 Device ID string. Note: 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. Example: it shows 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)
<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. Constraint: 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. Values are from: "Generating strings with Format and Template" on page 909.
<i>FileVersion ?</i> New in JDF 1.1	string	Version of the file referenced by this FileSpec .

Table 7-183: FileSpec Resource (Section 4 of 6)

Name	Data Type	Description
<i>MimeType</i> ? Modified in JDF 1.2	string	MIME type or file type of the file (or files of identical type when specifying a sequence of file names using the <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. If the file format has a MIME Media Type [iana- <i>mt</i>] registered with IANA, that value MUST be used. The [RFC2046] defines that MIME Media Types are case-insensitive. If the file format does not have a MIME Media Type registered with IANA, then the JDF spec defines string values, called file types, which MUST be used. Values include those from: "MimeType and MimeTypeVersion Attributes" on page 903.
<i>MimeTypeVersion</i> ? New in JDF 1.2	string	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: "PDF/1.3", "PDF/1.4" and "PDF/X-1a:2001" for <i>MimeType</i> = "application/pdf" "TIFF-IT/FP:1998", "TIFF-IT/CT:1998" and "TIFF-IT/LW/P1:1998" for <i>MimeType</i> = "TIFF/IT" Values include those from: "MimeType and MimeTypeVersion Attributes" on page 903.
<i>OSVersion</i> ? Modified in JDF 1.2	string	Version of the operating system specified by <i>AppOS</i> . The IANA Registry provides a list. Values include those from: Table S-1, "AppOS and OSVersion Examples" on page 739
<i>OverwritePolicy</i> ? New in JDF 1.2	enumeration	Policy that specifies the policy to follow when a file already exists and the FileSpec is used as an Output Resource. Values are: <i>Overwrite</i> – Overwrite the old file. <i>RenameNew</i> – Rename the new file. <i>RenameOld</i> – Rename the old file. <i>NewVersion</i> – Create a new file version. Only valid when the FileSpec references a file on a version aware file system. <i>OperatorIntervention</i> – Present a dialog to an operator. <i>Abort</i> – Abort the Process without modifying the old file.
<i>PageOrder</i> ?	enumeration	Indicates the order of pages in the file containing pages. Values are: <i>Ascending</i> – The first page in the file is the lowest numbered page. <i>Descending</i> – The first page in the file is the highest numbered page.
<i>Password</i> ? New in JDF 1.3	string	Password or decryption key that is needed to read the file contents. Note: since this password string is not encrypted, it SHOULD only be passed around within a protected environment.

Table 7-183: 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	<p>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.</p> <p>Values include those from: Table 7-184, “ResourceUsage Attribute Values” on page 536</p>
<i>SearchDepth</i> ? New in JDF 1.2	integer	<p>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.</p>
<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>Values include:</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>Constraint: 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 7-183: 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 913 and "FileSpec Attributes and Container Subelement" on page 939 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. May be used by an agent to identify a file on a Device without knowing its internal location.
<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 913 for details.
<i>Disposition</i> ? 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).

— Attribute: ResourceUsage

Table 7-184: ResourceUsage Attribute Values (Section 1 of 2)

Value	Description
<i>AbstractProfile</i>	Used for ColorCorrectionOp/FileSpec and ColorSpaceConversionOp/FileSpec
<i>ActualOutputProfile</i>	Used for ElementColorParams/FileSpec
<i>ColorProfile</i>	Used in Color/FileSpec
<i>CorrectionProfile</i>	Used for ScanParams/FileSpec
<i>DeviceLinkProfile</i>	Used for ColorCorrectionOp/FileSpec
<i>FinalTargetDevice</i>	Used for ColorCorrectionParams/FileSpec and ColorSpaceConversionParams/FileSpec
<i>InputFormat</i>	Used for FormatConversionParams/FileSpec

Table 7-184: ResourceUsage Attribute Values (Section 2 of 2)

Value	Description
<i>OutputFormat</i>	Used for FormatConversionParams/FileSpec
<i>OutputProfile</i>	Used for ExposedMedia/FileSpec
<i>RasterFileLocation</i>	Used for ByteMap/FileSpec
<i>ReferenceOutputProfile</i>	Used for ElementColorParams/FileSpec
<i>ScanProfile</i>	Used for ScanParams/FileSpec
<i>SearchPath</i>	Used for AssetListCreationParams/FileSpec and ImageReplacementParams/FileSpec
<i>SourceProfile</i>	Used for ColorSpaceConversionOp/FileSpec
<i>TargetProfile</i>	Used for PrintCondition/FileSpec , ScanParams/FileSpec , Color/FileSpec and PrintConditionColor/FileSpec
<i>WorkingColorSpace</i>	Used for ColorCorrectionParams/FileSpec and ColorSpaceConversionParams/FileSpec

7.2.76.1 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 7-185: Container Element

Name	Data Type	Description
FileSpec	reference	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 913 for details.

7.2.76.2 Element: FileAlias**Table 7-186: FileAlias Element (Section 1 of 2)**

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. 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. Deprecation note: starting with JDF 1.2, use FileSpec/Disposition .
<i>MimeType?</i> Deprecated in JDF 1.2	string	MIME type of the file. Deprecation note: starting with JDF/1.2, 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.

Table 7-186: FileAlias Element (Section 2 of 2)

Name	Data Type	Description
URL ? Deprecated in JDF 1.2	URL	The URL which identifies the file the alias refers to. In JDF/1.2 and beyond, use FileSpec/@URL .
FileSpec ? New in JDF 1.2	refeement	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.).

7.2.77 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**, **Layout/PlacedObject**, **LayoutPreparationParams**, **LayoutPreparationParams/PageCell**, **RasterReadingParams**

Example Partition: —

Input of Processes: —

Output of Processes: —

Table 7-187: FitPolicy Resource (Section 1 of 2)

Name	Data Type	Description
ClipOffset ?	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 , <i>ClipOffset</i> is ignored when PageCell/@ImageShift is specified.
GutterPolicy = "Fixed"	enumeration	Allows printing of NUp grids even if the media size does not match the requirements of the data. <i>GutterPolicy</i> MUST NOT be specified when FitPolicy is referenced from a Layout Resource. Values are: <i>Distribute</i> – The gutters can grow or shrink to the value specified in <i>MinGutter</i> . <i>Fixed</i> – The gutters are fixed.
MinGutter ?	XYPair	Minimum width in points of the horizontal and vertical gutters formed between rows and columns of pages of a multi-up Sheet layout. The first value specifies the minimum width of all horizontal gutters and the second value specifies the minimum width of all vertical gutters. <i>MinGutter</i> MUST NOT be specified when FitPolicy is referenced from a Layout resource.

Table 7-187: FitPolicy Resource (Section 2 of 2)

Name	Data Type	Description
<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. Values are: <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. Values are: <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.

7.2.78 Fold

[New in JDF 1.1](#)

Fold describes an individual folding operation of the **Component**.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	FoldingIntent, BinderySignature, FoldingParams
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-188: Fold Resource (Section 1 of 2)

Name	Data Type	Description
<i>From</i>	enumeration	Edge from which the page is folded. Values are: <i>Front</i> <i>Left</i>

Table 7-188: Fold Resource (Section 2 of 2)

Name	Data Type	Description
<i>To</i>	enumeration	Direction in which it is folded. 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.

7.2.79 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 31.

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 7-30, below.

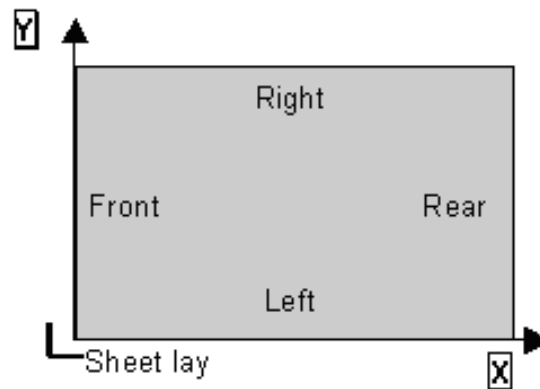


Figure 7-30: Names of the reference edges of a Sheet in the FoldingParams Resource

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	<i>BlockName, RibbonName, SheetName, SignatureName, WebName</i>
Input of Processes:	<i>Folding</i>
Output of Processes:	—

Table 7-189: FoldingParams Resource

Name	Data Type	Description
DescriptionType ? Deprecated in JDF 1.2	enumeration	How the folding operations are described. Values are: <i>FoldProc</i> – Detailed description of each individual fold. <i>FoldCatalog</i> – Selection of fold procedure from <i>FoldCatalog</i> . Deprecation note: starting with JDF 1.2, 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 ? Modified in JDF 1.4	string	Describes the type of fold according to the folding catalog in Figure 7-31, “Fold catalog part 1” on page 542 and Figure 7-32, “Fold catalog part 2” on page 543. In case of any ambiguity, the folding notation takes precedence over the graphic illustration in the aforementioned Figures. Value format is: “ <i>F_n-i</i> ” where “n” is the number of finished pages and “i” is either an integer, which identifies a particular fold or the letter “X”, which identifies a generic fold. E.g., “ <i>F6-2</i> ” describes a Z-fold of 6 finished pages, and “ <i>F6-X</i> .” describes a generic fold with 6 finished pages. Modification note: starting with JDF 1.4, the letter “X” is added for a generic fold
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.
<i>SheetLay</i> = “ <i>Left</i> ”	enumeration	Lay of input media. Values are: <i>Left</i> <i>Right</i> Note: <i>SheetLay</i> does not modify the coordinate references of the <i>Folding</i> 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 <i>Folding</i> procedure, rather than specifying a Combined Process that combines multiple <i>Folding</i> 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.

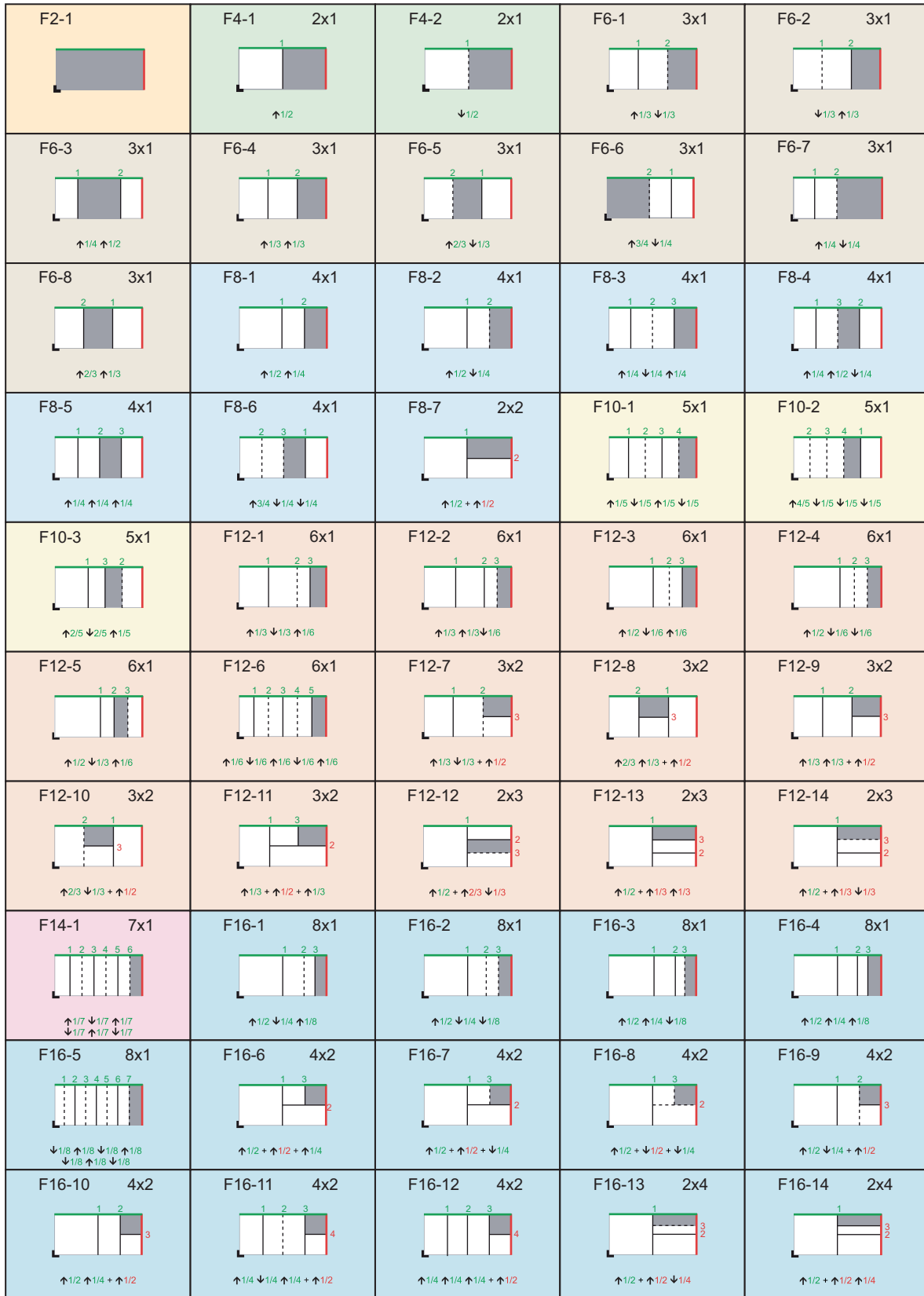


Figure 7-31: Fold catalog part 1

<p>F18-1 9x1</p> <p>↑1/8 ↓1/8 ↑1/8 ↓1/8 ↑1/8 ↓1/8 ↑1/8 ↓1/8</p>	<p>F18-2 9x1</p> <p>↑2/3 ↓1/3 ↑1/9 ↓1/9</p>	<p>F18-3 9x1</p> <p>↑1/3 ↓1/3 ↑2/9 ↓1/9</p>	<p>F18-4 9x1</p> <p>↑1/3 ↓1/3 ↑1/9 ↓1/9</p>	<p>F18-5 3x3</p> <p>↑1/3 ↓1/3 + ↑1/3 ↓1/3</p>
<p>F18-6 3x3</p> <p>↑1/3 ↓1/3 + ↑2/3 ↓1/3</p>	<p>F18-7 3x3</p> <p>↑1/3 ↑1/3 + ↑1/3 ↓1/3</p>	<p>F18-8 3x3</p> <p>↑1/3 ↑1/3 + ↑2/3 ↓1/3</p>	<p>F18-9 3x3</p> <p>↑2/3 ↓1/3 + ↑2/3 ↓1/3</p>	<p>F20-1 5x2</p> <p>↑2/5 ↓2/5 ↑1/5 + ↑1/2</p>
<p>F20-2 5x2</p> <p>↑1/5 ↓1/5 ↑1/5 ↓1/5 + ↑1/2</p>	<p>F24-1 6x2</p> <p>↑1/3 ↓1/3 + ↑1/2 + ↑1/6</p>	<p>F24-2 6x2</p> <p>↑1/3 ↑1/3 + ↑1/2 + ↑1/6</p>	<p>F24-3 6x2</p> <p>↑1/3 ↓1/3 ↑1/6 + ↑1/2</p>	<p>F24-4 6x2</p> <p>↑1/3 ↓1/3 ↓1/6 + ↑1/2</p>
<p>F24-5 6x2</p> <p>↑1/3 ↑1/3 ↓1/6 + ↑1/2</p>	<p>F24-6 6x2</p> <p>↓1/6 ↓1/6 ↑1/6 ↓1/6 ↑1/6 + ↑1/2</p>	<p>F24-7 6x2</p> <p>↑1/3 + ↑1/2 + ↑1/3 ↓1/6</p>	<p>F24-8 3x4</p> <p>↑1/3 ↓1/3 + ↑1/2 ↓1/4</p>	<p>F24-9 3x4</p> <p>↑2/3 ↑1/3 + ↑1/2 ↓1/4</p>
<p>F24-10 3x4</p> <p>↑1/3 ↑1/3 + ↑1/2 ↓1/4</p>	<p>F24-11 4x3</p> <p>↑1/2 + ↑2/3 ↓1/3 + ↑1/4</p>	<p>F28-1 7x2</p> <p>↑1/7 ↓1/7 ↑1/7 ↓1/7 ↑1/7 ↑1/7 + ↑1/2</p>	<p>F32-1 16x1</p> <p>↑1/2 ↓1/4 ↑1/8 ↓1/16</p>	<p>F32-2 8x2</p> <p>↑1/2 ↓1/4 + ↑1/2 + ↑1/8</p>
<p>F32-3 8x2</p> <p>↑1/2 ↓1/4 + ↑1/2 ↓1/8</p>	<p>F32-4 4x4</p> <p>↑1/2 + ↑1/2 + ↑1/4 + ↑1/4</p>	<p>F32-5 4x4</p> <p>↑1/2 + ↑1/2 + ↓1/4 + ↓1/4</p>	<p>F32-6 4x4</p> <p>↑1/2 + ↑1/2 + ↑1/4 + ↓1/4</p>	<p>F32-7 4x4</p> <p>↑1/4 ↓1/4 ↑1/4 + ↑1/2 ↓1/4</p>
<p>F32-8 4x4</p> <p>↑1/2 ↓1/4 + ↑1/2 ↓1/4</p>	<p>F32-9 4x4</p> <p>↑1/2 + ↑1/2 ↓1/4 + ↑1/4</p>	<p>F36-1 9x2</p> <p>↑1/3 ↓1/3 ↑1/9 ↓1/9 + ↑1/2</p>	<p>F36-2 6x3</p> <p>↑1/3 ↓1/3 + ↑1/3 ↓1/3 + ↑1/6</p>	<p>F40-1 5x4</p> <p>↑1/5 ↓1/5 ↑1/5 ↓1/5 + ↑1/2 ↓1/4</p>
<p>F48-1 6x4</p> <p>↑1/3 ↓1/3 + ↑1/4 ↓1/4 ↑1/4 + ↑1/6</p>	<p>F48-2 4x6</p> <p>↑1/4 ↓1/4 ↑1/4 + ↑1/3 ↓1/3 ↑1/6</p>	<p>F64-1 8x4</p> <p>↑1/2 + ↑1/4 ↓1/4 ↑1/4 + ↑1/4 ↓1/8</p>	<p>F64-2 8x4</p> <p>↑1/4 ↓1/4 ↑1/4 + ↑1/4 ↓1/4 ↑1/4 + ↑1/8</p>	

LEGEND

- Fold up
- - - - - Fold down
- █ Finished format folded sheet
- 1, 2, 3 Folds in numeric order
- L Lay
- Green: open sheet length
- Red: open sheet width

Example: F32-3, 8x2
 F32-3: Signature with 32 pages
 8x2: Split: 8 sheet parts lengthwise 2 sheet parts cross
 ↑1/2: Fold up with 1/2 of the open sheet format length
 ↓1/4: Fold down with 1/4 of the open sheet format length
 + : Fold direction change: 90...
 ↑1/2: Fold up with 1/2 of the open sheet format
 + : Fold direction change: 90...
 ↓1/8: Fold down with 1/8 of the open sheet format length

Figure 7-32: Fold catalog part 2

7.2.80 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:	<i>PSToPDFConversion</i>
Output of Processes:	—

Table 7-190: 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. 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 " <i>true</i> ", 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 " <i>true</i> ", font subsetting is enabled. If " <i>false</i> ", 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> .

7.2.81 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>Interpreting, Trapping</i>
Output of Processes:	—

Table 7-191: 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.

7.2.82 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 7-192: 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 <i>FormatConversion</i> 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 <i>FormatConversion</i> Process.
FileSpec (<i>OutputFormat</i>) ? Deprecated in JDF 1.2	refelement	The format of the converted file is specified in a FileSpec with <i>ResourceUsage = OutputFormat</i> . A URL SHOULD NOT be specified because the list of files is given by the output RunList of the <i>FormatConversion</i> 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 .

Table 7-192: FormatConversionParams Resource (Section 2 of 2)

Name	Data Type	Description
TIFFFormatParams ? New in JDF 1.2	element	Parameters specific to conversion of rasters to TIFF files. (See below.) <i>FormatConversion</i> SHOULD NOT be used to convert non-raster files to TIFF. The appropriate <i>Interpreting</i> and <i>Rendering</i> 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**.

7.2.82.1 Element: TIFFFormatParams

[New in JDF 1.2](#)

Table 7-193: TIFFFormatParams Element (Section 1 of 2)

Name	Data Type	Description
<i>ByteOrder</i> ?	enumeration	Byte order of the TIFF file. Values are: <i>II</i> – Low byte first. <i>MM</i> – high byte first. Note: the identifier values 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> : Values are: 1 – “Chunky” format, which is pixel interleaved. 2 – “Planar” format, which is strip interleaved.
<i>WhitelsZero</i> = "true"	boolean	When writing monochrome or grayscale files, this flag indicates whether the data is to be written as “WhitelsZero” or “BlackIsZero.”
<i>Segmentation</i> ?	enumeration	How the image data are segmented. 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>Striped</i> – Data are segmented into strips. <i>Tiled</i> – Data are segmented into tiles.

Table 7-193: TIFFFormatParams Element (Section 2 of 2)

Name	Data Type	Description
<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 Subelement, 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 <i>FormatConversion</i> , or because the <i>FormatConversion</i> Process forms a part of a compound process with a <i>Separation</i> 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.

7.2.82.2 Element: TIFFtag

[New in JDF 1.2](#)

Table 7-194: 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
<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.

Table 7-194: TIFFtag Element (Section 2 of 2)

Name	Data Type	Description
<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.

7.2.82.3 Element: TIFFEmbeddedFile

[New in JDF 1.2](#)

Table 7-195: 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).
<i>FileSpec</i>	reference	Reference to the file to be embedded.

7.2.83 GatheringParams

This Resource contains the Attributes of the *Gathering* Process.

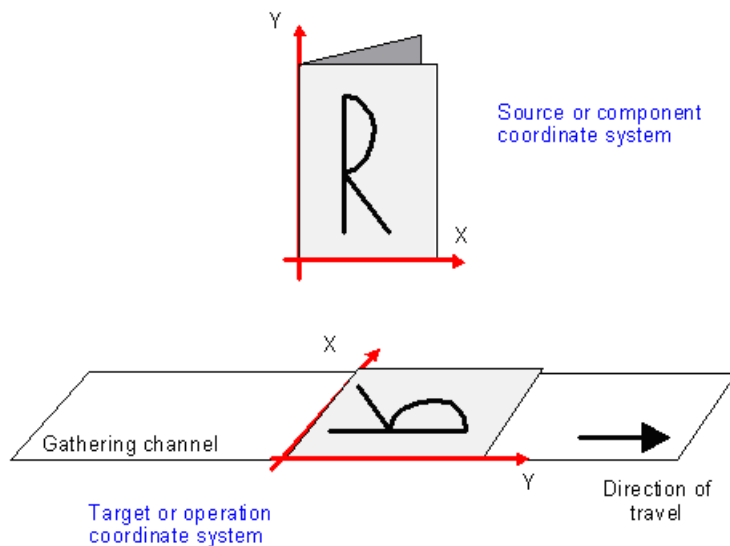


Figure 7-33: Coordinate system used for Gathering

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>Gathering</i>
Output of Processes:	—

Table 7-196: 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.

7.2.84 GeneralID

[New in JDF 1.3](#)

[Modified in JDF 1.4](#)

Modification note: starting with JDF 1.4, GeneralID becomes an Element, and is no longer a Resource. See “GeneralID” on page 43. GeneralID becomes a child of any Element. See Table 3-1, “Any Element (generic content)” on page 40. The remainder of this section has the text that was present in JDF 1.3, except that the table has been moved to “GeneralID” on page 43.

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:	Any Element (generic content)
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

7.2.85 GlueApplication

[New in JDF 1.1](#)

This Resource specifies glue application in hard and soft cover book production.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	CasingInParams, CoverApplicationParams, GluingParams/Glue, SpineTapingParams
Input of Processes:	—
Output of Processes:	—

Table 7-197: GlueApplication Resource

Name	Data Type	Description
<i>GluingTechnique</i>	enumeration	Type or technique of gluing application. Values are: <i>SpineGluing</i> <i>SideGluingFront</i> <i>SideGluingBack</i>
GlueLine	refelement	Structure of the glue line.

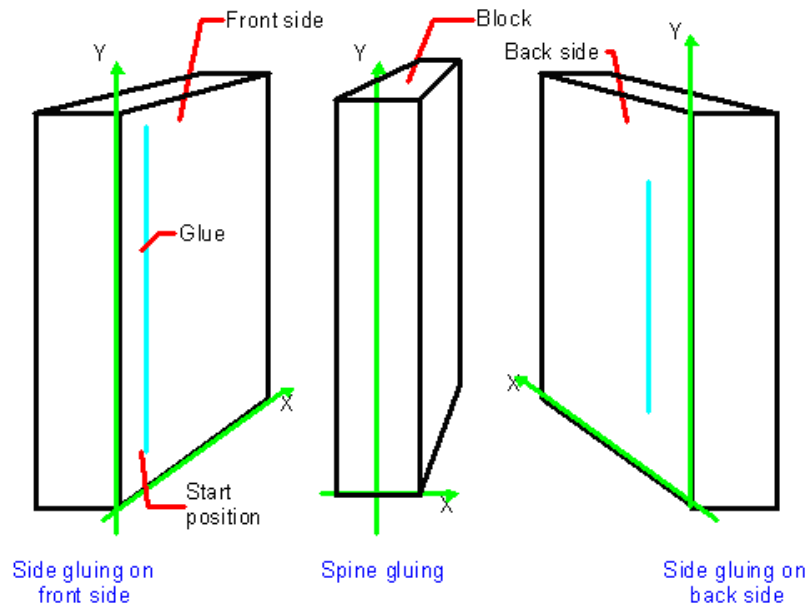


Figure 7-34: Parameters and coordinate system for glue application

7.2.86 GlueLine

This Resource provides the information to determine where and how to apply glue.

Resource Properties

Resource Class: Parameter

Resource referenced by: Insert, **BoxFoldingParams**, **BoxFoldAction**, **CaseMakingParams**, **CoverApplicationParams**, **EndSheetGluingParams/EndSheet**, **FoldingParams**, **GluingParams/Glue**, **HeadBandApplicationParams**, **InsertingParams**, **Media/MediaLayers**, **SpineTapingParams**, **ThreadSealingParams**, **ThreadSewingParams**

Example Partition: —

Input of Processes: —

Output of Processes: —

Table 7-198: GlueLine Resource (Section 1 of 2)

Name	Data Type	Description
<i>AreaGlue</i> = "false" 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.
<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. 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

Table 7-198: GlueLine Resource (Section 2 of 2)

Name	Data Type	Description
GluingPattern ? Modified in JDF 1.3	NumberList	Glue line pattern defined by the length of a glue line segment (1st Element, 3rd and all odd elements of the NumberList) and glue line gap (2nd Element, 4th and all even elements of the NumberList). A solid line is expressed by the pattern (1 0). <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.
MeltingTemperature ?	integer	Temperature needed for melting the glue, in degrees centigrade. Used only when <i>GlueType</i> = "Hotmelt" or <i>GlueType</i> = "PUR" .
RelativeStartPosition ? 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 .
RelativeWorkingPath ? 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 .
StartPosition ? Modified in JDF 1.2	XYPair	Start position of glue line. The start position is given in the coordinate system of the mother Sheet. If both <i>StartPosition</i> and <i>RelativeStartPosition</i> are specified, <i>RelativeStartPosition</i> is ignored.
WorkingPath ? 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.

7.2.87 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:	<i>Gluing</i>
Output of Processes:	—

Table 7-199: GluingParams Resource

Name	Data Type	Description
GluingProductionID ? New in JDF 1.3	string	Defines a gluing scheme for production.
Glue *	element	Definition of one or more Glue line applications.

7.2.87.1 Element: Glue

The Glue Element describes how to apply a line of glue.

Table 7-200: Glue Element

Name	Data Type	Description
<i>WorkingDirection</i>	enumeration	Direction from which the tool is working. 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.

7.2.88 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:	<i>HeadBandApplication</i>
Output of Processes:	—

Table 7-201: 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>BottomColorDetails</i> ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If <i>BottomColorDetails</i> is supplied, <i>BottomColor</i> SHOULD also be supplied.
<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>TopColorDetails</i> ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If <i>TopColorDetails</i> is supplied, <i>TopColor</i> SHOULD also be supplied.
<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 7-201: HeadBandApplicationParams Resource (Section 2 of 2)

Name	Data Type	Description
<i>StripMaterial?</i>	enumeration	Strip material. 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 *	reference	The carrier can be applied to the book block with glue. The coordinate system for the GlueLine is defined in the Section 7.2.72, EndSheetGluingParams.

7.2.89 Hole

The **Hole** Resource describes an individual hole.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	HoleLine, HoleList, HoleMakingParams
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-202: 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. Values are: <i>Elliptic</i> <i>Round</i> <i>Rectangular</i>

7.2.90 HoleLine

[New in JDF 1.1](#)

Line hole punching generates a series of holes with identical distance (pitch) running parallel to the edge of a Web, which is mainly used to transport paper through continuous-feed printers and finishing Devices (form processing). The final product typically is a Web with two lines of holes, one at each edge of the Web. The parameters for one line of holes are specified in the **HoleLine** Resource. The distance between holes within each line of holes is identical (constant pitch).

Resource Properties

Resource Class:	Parameter
Resource referenced by:	HoleList, HoleMakingParams
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-203: 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.

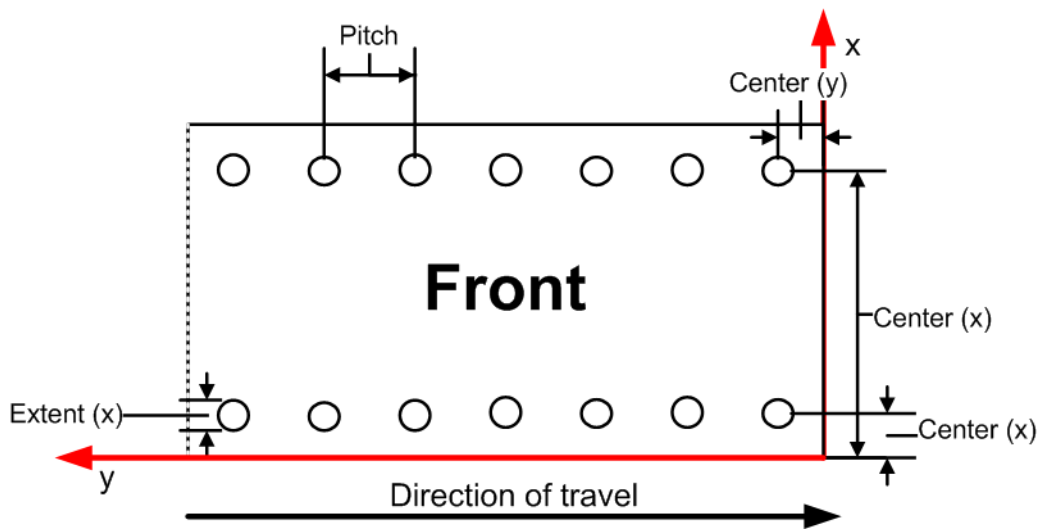


Figure 7-35: Hole line parameters

However, sometimes Line Hole Punching is performed for multiple webs before dividing the Web after the *HoleMaking* Process as illustrated in Figure 7-36 below:

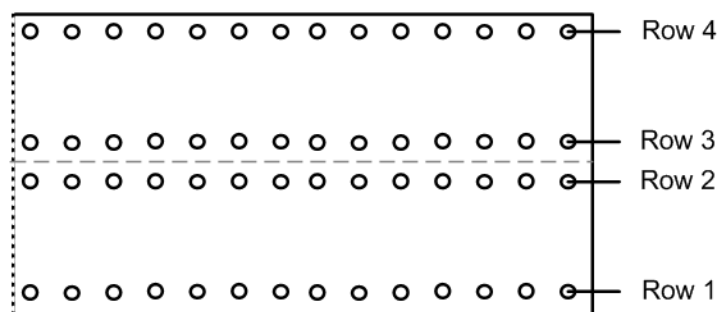


Figure 7-36: Line hole punching for multiple webs

7.2.91 HoleList

This Resource is used to describe holes or rows of holes in Intent Resources or Media. Note that it was an Intent Resource Subelement prior to JDF 1.2.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	BindingIntent/CoilBinding, BindingIntent/PlasticCombBinding, BindingIntent/StripBinding, BindingIntent/WireCombBinding, BindingIntent/BindList/BindItem/CoilBinding, BindingIntent/BindList/BindItem/PlasticCombBinding, BindingIntent/BindList/BindItem/StripBinding, BindingIntent/BindList/BindItem/WireCombBinding, HoleMakingIntent, Media
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-204: HoleList Resource

Name	Data Type	Description
Hole * Modified in JDF 1.1	refelement	Description of individual holes. See Section 7.2.89, Hole.
HoleLine * New in JDF 1.1	refelement	Array of all HoleLine Elements. See Section 7.2.90, HoleLine.

7.2.92 HoleMakingParams

This Resource specifies where to make a hole of what shape in components. This information is used by the *HoleMaking* Process.

Default behavior for *HoleCount*: For dealing with the Default case of *HoleCount* (i.e. when not supplied), intelligent systems will take into consideration Job parameters like the length of the binding edge or distance of holes to the paper edges to calculate the appropriate number of holes. For production of the holes and selection/production of the matching binding Element, the “system specified” values need to match 100% between the *HoleMaking* and the binding Process for obvious reasons. In practice, if no details are specified for *HoleMaking*, they SHOULD also be absent for binding. In this case, either the operator provides the missing value when setting up the binding Device for the Job, or the Device itself needs to have some kind of automatic hole detection mechanism.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	CoilBindingParams, PlasticCombBindingParams, RingBindingParams, StripBindingParams, WireCombBindingParams
Example Partition:	<i>SheetName, SignatureName</i>
Input of Processes:	<i>HoleMaking</i>
Output of Processes:	—

Table 7-205: HoleMakingParams Resource (Section 1 of 3)

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 = "TrailingEdge"</i> New in JDF 1.1	enumeration	Defines the reference coordinate system for <i>Center</i> . Values are: <i>TrailingEdge</i> – Physical coordinate system of the component. <i>RegistrationMark</i> – The center is relative to a registration mark.

Table 7-205: HoleMakingParams Resource (Section 2 of 3)

Name	Data Type	Description
<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. 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 s h h". "0 3 3 3 3" = "s s h h h s s s h h h". Default behavior: see "Default behavior for <i>HoleCount</i> "
<i>HoleReferenceEdge?</i> New in JDF 1.1 Deprecated in JDF 1.2	enumeration	The edge of the media relative to where the holes are to be punched. Use with <i>HoleType</i> . Default value: if <i>HoleType</i> is <i>Explicit</i> , <i>Pattern</i> ; otherwise <i>Left</i> . 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 929 is used. Deprecation note: starting with JDF 1.1, use 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.
<i>HoleType</i> New in JDF 1.1	enumerations	Predefined hole pattern. Multiple hole patterns are specified as one NMTOKENS string, (e.g., 3-hole ring binding and 4-hole ring binding holes on one piece of media). Values are: <i>Explicit</i> – Holes are defined in an array of Hole Elements. <i>2HoleEuro</i> – Replaced by either <i>R2m-DIN</i> or <i>R2m-ISO</i> . Deprecated in JDF 1.0 <i>3HoleUS</i> – Replaced by <i>R3I-US</i> . Deprecated in JDF 1.0 <i>4HoleEuro</i> – Replaced by either <i>R4m-DIN-A4</i> or <i>R4m-DIN-A5</i> . Deprecated in JDF 1.0 Values are from: "JDF/CIP4 Hole Pattern Catalog" on page 929

Table 7-205: HoleMakingParams Resource (Section 3 of 3)

Name	Data Type	Description
<i>Shape</i> ? Modified in JDF 1.1	enumeration	Shape of the holes if <i>HoleType</i> is not <i>Explicit</i> . Default value is: value implied by <i>HoleType</i> . Values are: <i>Elliptic</i> <i>Round</i> <i>Rectangular</i>
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> .

7.2.93 IdentificationField

This Resource contains information about a mark on a document (e.g., a bar code) used for OCR-based verification purposes or document separation.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	Abstract Physical Resource, Disjointing , EmbossingParams/Emboss , Layout/MarkObject , LayoutElementProductionParams/ LayoutElementPart/BarcodeProductionParams
Example Partition:	—
Input of Processes:	<i>Verification</i>
Output of Processes:	—

Table 7-206: IdentificationField Resource (Section 1 of 3)

Name	Data Type	Description
<i>BoundingBox</i> ?	rectangle	<p>Box that provides the boundaries of the mark that indicates where the IdentificationField is placed. If the IdentificationField is specified in a Layout, the coordinate system is defined by the MarkObject containing the IdentificationField. If no Layout context is available, the origin of the coordinate system is defined as the lower left corner of the Resource surface that <i>Position</i> specifies when the specified surface is viewed in its natural orientation.</p> <p>Each item in the list below specifies a value of <i>Position</i> and the corner that is the origin for the specified value when the viewer is positioned in front of the front surface. For example, when <i>Position</i> = "Left", the origin is the bottom-back corner of left surface when viewed from the front surface of the Resource and lower-left corner when viewed from the left surface.</p> <ul style="list-style-type: none"> • <i>Front</i> – Bottom-left corner • <i>Left</i> – Bottom-back corner • <i>Back</i> – Bottom-right corner • <i>Right</i> – Bottom-front corner • <i>Top</i> – Front-left corner • <i>Bottom</i> – Back left corner <p>If no <i>BoundingBox</i> is defined and the IdentificationField is specified</p> <ul style="list-style-type: none"> • outside the context of a Layout, the complete visible surface MUST be scanned for an appropriate bar code. • within the context of a Layout, the implied <i>BoundingBox</i> is specified by MarkObject/@ClipBox. <p>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.</p> <p>Modification note: starting with JDF 1.4, all text is new.</p>
<i>Encoding</i> Modified in JDF 1.4	enumeration	<p>Encoding of the information.</p> <p>Values are:</p> <p><i>ASCII</i> – Plain-text font.</p> <p><i>Barcode</i> – Any bar code. New in JDF 1.3</p> <p><i>BarCode1D</i> – One-dimensional bar code. Deprecated in JDF 1.3</p> <p><i>BarCode2D</i> – Two-dimensional bar code. Deprecated in JDF 1.3</p> <p><i>Braille</i> – Braille text. New in JDF 1.4</p> <p><i>RFID</i> – Radio Frequency Identification tag. New in JDF 1.3</p>
<i>EncodingDetails</i>	NMTOKEN	<p>Details about the encoding type. An example is the bar code scheme.</p> <p>Values include those from: Table 7-207, "EncodingDetails Attribute Values"</p>

Table 7-206: IdentificationField Resource (Section 2 of 3)

Name	Data Type	Description
<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 I dentificationField. The coordinate system is defined in the system of the Sheet or component where the I dentificationField resides. The <i>Orientation</i> is used only as metadata when searching or scanning I dentificationField Elements and not used when generating I dentificationField Elements in a <i>LayoutElementProduction</i> Process.
<i>Position?</i>	enumeration	Position with respect to the Instance Document or Physical Resource to which the Resource refers. 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 I dentificationField 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. 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 I dentificationField, (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.

Table 7-206: IdentificationField Resource (Section 3 of 3)

Name	Data Type	Description
<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. Values are from: "Generating strings with Format and Template" on page 909. Constraint: 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. Values are from: "Generating strings with Format and Template" on page 909. Constraint: 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 I dentificationField.

— Attribute: EncodingDetails

Table 7-207: EncodingDetails Attribute Values (Section 1 of 2)

Value	Description	Value	Description
<i>BOBST</i>		<i>ITF_14</i>	
<i>BrailleASCII</i> New in JDF 1.4	A binary representation for 6 dot Braille messages. See http://en.wikipedia.org/wiki/Braille_ASCII	<i>ITF_6</i>	
<i>BrailleUnicode</i> New in JDF 1.4	A binary representation for Braille messages. See http://www.unicode.org/charts/PDF/U2800.pdf#search=%22braille%20unicode%22	<i>ITF_16</i>	
<i>CODABAR</i>		<i>KURANDT</i>	
<i>CODABAR_Tradional</i>		<i>LAETUS_PHARMA</i>	
<i>CODABLOCK</i>		<i>MSI</i>	
<i>CODABLOCK_F</i>		<i>NDC_HRI</i>	
<i>Code128</i>		<i>PARAF</i>	
<i>Code25</i>		<i>Plessey</i>	
<i>Code39</i>		<i>PDF417</i>	
<i>Code39_Extended</i>		<i>PZN</i>	
<i>DATAMATRIX</i> Deprecated in JDF 1.3	Deprecation note: starting with JDF 1.3, use " <i>HIBC_DATAMATRIX</i> "	<i>QR</i>	
<i>EAN</i>	includes Bookland_EAN and ISSN.	<i>RSS_14</i>	
<i>EAN_13</i>		<i>RSS_14_Stacked</i>	

Table 7-207: EncodingDetails Attribute Values (Section 2 of 2)

Value	Description	Value	Description
<i>EAN_8</i>		<i>RSS_14_Stacked_Omnidir</i>	
<i>EAN_Coupon</i>		<i>RSS_14_Truncated</i>	
<i>EAN_128</i>		<i>RSS_Limited</i>	
<i>HIBC_Code39</i>		<i>RSS_Expanded</i>	
<i>HIBC_Code128</i>		<i>RSS_Expanded_Stacked</i>	
<i>HIBC_Code39_2</i>		<i>UPC_A</i>	
<i>HIBC_CODABLOCK_F</i>		<i>UPC_Coupon</i>	
<i>HIBC_QR</i>		<i>UPC_E</i>	
<i>HIBC_DATAMATRIX</i>		<i>UPC_SCS</i>	
<i>Interleave25</i>			

7.2.93.1 Element: BarcodeDetails

Table 7-208: BarcodeDetails Element (Section 1 of 2)

Name	Data Type	Description
<i>BarcodeVersion?</i>	NMTOKEN	The version of a barcode. Values include those from: Table 7-211, “BarcodeVersion Values – for HIBC_DATAMATRIX” on page 563 Values include those from: Table 7-212, “BarcodeVersion Values – for QR barcodes” on page 563
<i>ErrorCorrectionLevel?</i>	NMTOKEN	Error correction level for barcodes having a separately definable error correction level. Each value can be used only for certain values of IdentificationField/@EncodingDetails . Values include: <i>PDF417_EC_0</i> – for <i>EncodingDetails</i> = <i>PDF417</i> <i>PDF417_EC_1</i> – for <i>EncodingDetails</i> = <i>PDF417</i> <i>PDF417_EC_2</i> – for <i>EncodingDetails</i> = <i>PDF417</i> <i>PDF417_EC_3</i> – for <i>EncodingDetails</i> = <i>PDF417</i> <i>PDF417_EC_4</i> – for <i>EncodingDetails</i> = <i>PDF417</i> <i>PDF417_EC_5</i> – for <i>EncodingDetails</i> = <i>PDF417</i> <i>PDF417_EC_6</i> – for <i>EncodingDetails</i> = <i>PDF417</i> <i>PDF417_EC_7</i> – for <i>EncodingDetails</i> = <i>PDF417</i> <i>PDF417_EC_8</i> – for <i>EncodingDetails</i> = <i>PDF417</i> <i>QR_EC_L</i> – for <i>EncodingDetails</i> = <i>QR</i> <i>QR_EC_M</i> – for <i>EncodingDetails</i> = <i>QR</i> <i>QR_EC_Q</i> – for <i>EncodingDetails</i> = <i>QR</i> <i>QR_EC_H</i> – for <i>EncodingDetails</i> = <i>QR</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.

Table 7-208: BarcodeDetails Element (Section 2 of 2)

Name	Data Type	Description
<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.

7.2.93.2 Element: ExtraValues

Table 7-209: ExtraValues Element

Name	Data Type	Description
<i>Usage</i>	NMTOKEN	The usage of the value. 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 7-210: Usage of barcode Attributes for certain barcode types (Section 1 of 2)

EncodingDetails values (barcode types)	Height	Magnification	Ratio
<i>Code25</i> <i>Code39</i> <i>Code39_Extended</i> <i>Interleave25</i> <i>MSI</i> <i>Plessey</i>	Used	Used	Used
<i>CODABAR</i> <i>Code128</i> <i>EAN_13</i> <i>EAN_8</i> <i>EAN_128</i> <i>HIBC_Code39</i> <i>HIBC_Code128</i> <i>ITF_14</i> <i>ITF_16</i> <i>NDC_HRI</i> <i>PARAF</i> <i>UPC_A</i> <i>UPC_E</i> <i>UPC_SCS</i> <i>UPC_SCS</i>	Used	Used	Not used

Table 7-210: Usage of barcode Attributes for certain barcode types (Section 2 of 2)

EncodingDetails values (barcode types)	Height	Magnification	Ratio
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 HIBC_DATAMATRIX**

The following table specifies valid values of BarcodeDetails/@BarcodeVersion for a DATAMATRIX barcode:

Modification note: starting with JDF 1.3, these values are for "HIBC_DATAMATRIX" rather than "DATAMATRIX"

Table 7-211: BarcodeVersion Values – for HIBC_DATAMATRIX

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

— **Attribute: BarcodeVersion – for QR barcodes**

The following table specifies valid values of BarcodeDetails/@BarcodeVersion for a QR barcode.

Table 7-212: BarcodeVersion Values – for QR barcodes

Values							
QR_1	QR_6	QR_11	QR_16	QR_21	QR_26	QR_31	QR_36
QR_2	QR_7	QR_12	QR_17	QR_22	QR_27	QR_32	QR_37
QR_3	QR_8	QR_13	QR_18	QR_23	QR_28	QR_33	QR_38
QR_4	QR_9	QR_14	QR_19	QR_24	QR_29	QR_34	QR_39
QR_5	QR_10	QR_15	QR_20	QR_25	QR_30	QR_35	QR_40

Example 7-22: Barcode

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" JobPartID="ID34" Type="LayoutElementProduction"
  Version="1.4">
```

```

<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>

```

7.2.94 IDPrintingParams

[Deprecated in JDF 1.1](#)

See "IDPrintingParams" on page 1026 for details of this deprecated Resource.

7.2.95 ImageCompressionParams

Prior to JDF 1.2 the filtering in ImageCompressionParams was based on the terminology in PostScript and PDF. Many image compression and decompression filters require additional information in the form of a filter parameter dictionary, and additional filter parameters have been added to meet this need.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	ContentList/ContentData, FormatConversionParams, LayoutElement, PageList, PageList/PageData
Example Partition:	DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName
Input of Processes:	ImageReplacement, PDLCreation, PSToPDFConversion,
Output of Processes:	—

Table 7-213: ImageCompressionParams Resource

Name	Data Type	Description
ImageCompression *	element	Specifies how images are to be compressed.

7.2.95.1 Element: ImageCompression

Table 7-214: 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. 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 7-214: 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. 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". 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. Modification note: starting with JDF 1.1, the data type changes from enumeration 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 7-214: ImageCompression Element (Section 3 of 3)

Name	Data Type	Description
<i>ImageType</i>	enumeration	Specifies the kind of images that are to be manipulated. Values are: <i>Color</i> <i>Grayscale</i> <i>Monochrome</i>
<i>JPXQuality?</i> 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.
<i>CCITTFaxParams?</i> 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.
<i>DCTParams?</i> 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.
<i>FlateParams?</i> 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.
<i>JBIG2Params?</i> New in JDF 1.3	element	Provides the JBIG2 compression parameters.
<i>JPEG2000Params?</i> New in JDF 1.3	element	Provides the JPEG2000 compression parameters.
<i>LZWParams?</i> 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

7.2.95.2 Element: CCITTFaxParams[New in JDF 1.2](#)**Table 7-215: CCITTFaxParams Element**

Name	Data Type	Description
<i>Uncompressed = "false"</i>	boolean	A flag to indicate whether the file generated can use uncompressed encoding when advantageous.
<i>K = "0"</i>	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

7.2.95.3 Element: DCTParams

[New in JDF 1.2](#)

Table 7-216: DCTParams Element

Name	Data Type	Description
<i>SourceCSs</i>	enumerations	Identifies which of the incoming color spaces will be operated on. Values are from: Table 7-217, “SourceCSs Attribute Values” on page 568 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 = "1.0"</i>	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 = "Automatic"</i>	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: 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 7-217: SourceCSs Attribute 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 7-217: SourceCSs Attribute 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].

7.2.95.4 Element: FlateParams[New in JDF 1.2](#)**Table 7-218: 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.

7.2.95.5 Element: JBIG2Params[New in JDF 1.3](#)**Table 7-219: 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.

7.2.95.6 Element: JPEG2000Params[New in JDF 1.3](#)**Table 7-220: 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 = "1"</i>	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. Values are: <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.

7.2.95.7 Element: LZWParams

[New in JDF 1.2](#)

Table 7-221: LZWParams Element

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.
<i>Predictor</i> = "1"	integer	A code that selects the predictor function: 1 – No predictor (normal encoding or decoding). 2 – TIFF Predictor 2. 10 – PNG predictor, None function. 11 – PNG predictor, Sub function. 12 – PNG predictor, Up function. 13 – PNG predictor, Average function. 14 – PNG predictor, Path function. 15 – PNG predictor in which the encoding filter automatically chooses the optimum function separately for each row. Note: On 1X PNG predictors, these values select the specific PNG predictor function(s) to be used, as indicated above. When decoding, the predictor function is explicitly encoded in the incoming data.

7.2.96 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 7-222: ImageReplacementParams Resource (Section 1 of 2)

Name	Data Type	Description
<i>ImagePreScanStrategy?</i> New in JDF 1.2	NMTOKEN	Specifies the image pre-scanning strategy to be used on the input document data before starting the <i>RIPing</i> Process. Values include: <i>NoPreScan</i> – Do not pre-scan the document looking for references to images. <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. <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.
<i>ImageReplacementStrategy</i>	enumeration	Identifies how externally referenced images will be handled within the associated Process. 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.
<i>MaxResolution?</i> Deprecated in JDF 1.1	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.
<i>ResolutionReductionStrategy?</i> Deprecated in JDF 1.1	enumeration	Identifies the mechanism used for reducing the image resolution. Values are: <i>Downsample</i> <i>Subsample</i> <i>Bicubic</i> Deprecation note: starting with JDF 1.1, use a link to ImageCompressionParams in the Process.

Table 7-222: ImageReplacementParams Resource (Section 2 of 2)

Name	Data Type	Description
<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. The values are examples: Values include: .lay .e .samp
<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>) + New in JDF 1.1	refelement	Specification of the paths to search when trying to locate the referenced data. The FileSpec replaces the <i>SearchPath</i> text element.
<i>SearchPath *</i> Deprecated in JDF 1.1	telem	String that identifies the paths to search when trying to locate the referenced data.

7.2.97 ImageSetterParams

This Resource specifies the settings for the imagesetter. A number of settings are OEM-specific, while others are so widely used they MAY be supported between vendors. Both filmsetter settings and platesetter settings are described with this Resource.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	PreviewGenerationParams
Example Partition:	—
Input of Processes:	<i>ImageSetting</i>
Output of Processes:	—

Table 7-223: ImageSetterParams Resource (Section 1 of 3)

Name	Data Type	Description
<i>AdvanceDistance ?</i>	double	Additional media advancement beyond the media dimensions on a Web-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. 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 (Web-Fed).

Table 7-223: ImageSetterParams Resource (Section 2 of 3)

Name	Data Type	Description
<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. 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 <i>ImageSetting</i> 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). Values are: <i>Positive</i> <i>Negative</i>
<i>Punch</i> ? Deprecated in JDF 1.3	boolean	If "true", indicates that the Device MUST create registration punch holes. Use a Combined Process with a <i>Bending</i> 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 Process a <i>Bending</i> 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</i> ="OneSidedFront" 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. Values are from: Table 7-224, "Sides Attribute Values" on page 575.
<i>SourceWorkStyle</i> ? New in JDF 1.2	WorkStyle	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

Table 7-223: ImageSetterParams Resource (Section 3 of 3)

Name	Data Type	Description
<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 7-224: Sides Attribute 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> ”.

7.2.98 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:	—
Example Partition:	<i>FountainNumber, Separation, SheetName, Side, SignatureName, WebName</i>
Input of Processes:	<i>ConventionalPrinting, DigitalPrinting, Varnishing</i>
Output of Processes:	—

Table 7-225: Ink Resource

Name	Data Type	Description
<i>ColorName</i> ? Deprecated in JDF 1.4	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 870. Deprecation note: starting with JDF 1.4, use <i>Separation</i> Partition Key.
<i>Family</i> ?	NMTOKEN	Ink family. Values include: <i>HKS</i> – ink <i>PANTONE</i> – ink <i>Toyo</i> – ink <i>ISO</i> – ink [ISO2846-1:1997] (used by SWOP) <i>InkJet</i> – ink <i>Varnish</i> – liquid that is similar to ink <i>Silicon</i> – liquid that is similar to ink <i>Toner</i> – liquid that is similar to ink
<i>InkName</i> ?	string	The fully qualified ink name including the ink <i>Family</i> name. For instance, " <i>PANTONE 138 C</i> " is a member of the PANTONE family.
<i>SpecialInk</i> ? Modified in JDF 1.4	NMTOKENS	Specific ink Attributes. 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> Modification note: starting with JDF 1.4, the data type was expanded from NMTOKEN to NMTOKENS.
<i>SpecificYield</i> ?	double	Weight per area at total coverage in g/m ² .

7.2.99 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:	<i>InkZoneCalculation</i>

Output of Processes: —

Table 7-226: 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.
<i>Device</i> ? 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.

7.2.100 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: *FountainNumber, Separation, SheetName, Side, SignatureName, WebName*

Input of Processes: *ConventionalPrinting*Output of Processes: *InkZoneCalculation*

Table 7-227: 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.
<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.

Table 7-227: InkZoneProfile Resource (Section 2 of 2)

Name	Data Type	Description
<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.

7.2.101 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.

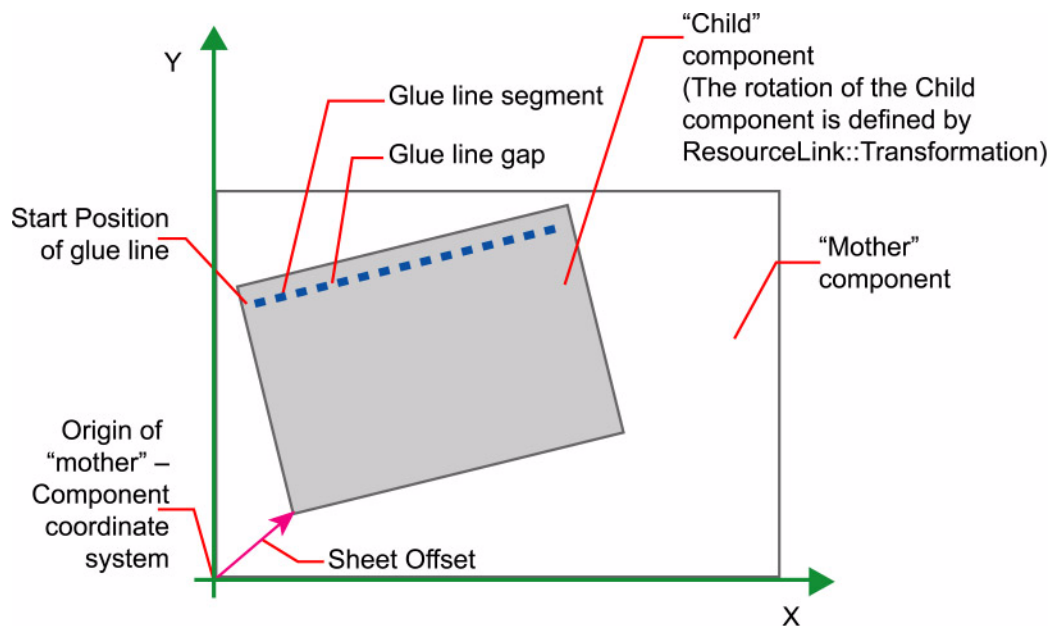


Figure 7-37: Parameters and coordinate system used for Inserting

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:	<i>Inserting</i>
Output of Processes:	—

Table 7-228: InsertingParams Resource (Section 1 of 2)

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 .

Table 7-228: InsertingParams Resource (Section 2 of 2)




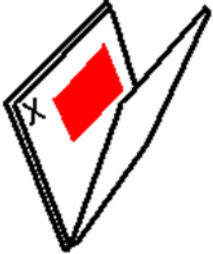
Name	Data Type	Description
<u>InsertLocation</u> <u>Modified in JDF 1.2</u>	enumeration	Where to place the “child” Sheet. Values are: <i>Back</i> <i>FinishedPage</i> – Place the child exactly onto the page specified in <i>FinishedPage</i> . <u>New in JDF 1.2</u> <i>Front</i> <i>Overfold</i> – Place onto the overfold. Replaces <i>OverfoldLeft</i> and <i>OverfoldRight</i> . <u>New in JDF 1.2</u> <i>OverfoldLeft</i> – <u>Deprecated in JDF 1.2</u> <i>OverfoldRight</i> – <u>Deprecated in JDF 1.2</u> Modification note: starting with JDF 1.2, this Attribute is renamed from <i>Location</i> due to a name clash with the <i>Location</i> Partition Key.
<i>Method</i> = "BlowIn"	enumeration	Inserting method. Values are: <i>BindIn</i> – Apply glue to fasten the insert. <i>BlowIn</i> – Loose insert.
<u>SheetOffset ?</u> <u>Deprecated in JDF 1.1</u>	XYPair	Offset between the Sheet to be inserted and the “mother” Sheet. <i>SheetOffset</i> is implied by the Transformation matrix in <i>ResourceLink/@Transformation</i> of the child’s <i>ComponentLink</i> .
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 7-229: Location of Inserts

Front	Back	Overfold	Finished Page
			
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.

7.2.102 InsertSheet

InsertSheet Resources define Device generated images and Sheets which can be produced along with the Job. **InsertSheet** Elements include separators Sheets, error Sheets, accounting Sheets and Job Sheets. The information provided on the Sheet depends on the type of Sheet. In some cases, an *Imposition* Process can encounter **RunList** Elements that do not provide enough finished pages to complete a **Layout** Resource or its children. **InsertSheet** Resources are used to provide a standard way of completing such **Layout** Resources. **InsertSheet** Resources MAY also be used to start new Sheet Resources, (e.g., to ensure that a new chapter starts on a right-hand page). In addition, **InsertSheet** MAY specify whether new media are to be inserted after the current Sheet, Signature, Instance Document or Job is completed.

InsertSheet Elements MAY be used at the beginning or end of **RunList** with a *SheetUsage* Attribute of *Header* or *Trailer*. When an **InsertSheet** appears both in a **RunList** and in a **Layout**, the following precedence applies:

- 1 The **InsertSheet** with *Usage FillSurface* from the **RunList** is applied first.
- 2 The **InsertSheet** with *Usage FillSheet* from the **RunList** is applied.
- 3 The **InsertSheet** with *Usage FillSignature* from the **RunList** is applied.
- 4 After completely processing the **RunList** **InsertSheet** Elements once, apply the **Layout**' Partition's **InsertSheet** Elements.

If the **RunList** of the **InsertSheet** does not supply enough content to fill a Sheet, Signature or surface, the **RunList** will be reapplied until no *PlacedObject* slots remain to be filled. When an **InsertSheet** is used in a **RunList** of a Process that does not use a **Layout** or **LayoutPreparationParams** Resource (i.e., that Process is not a part of a Combined Process with *Imposition* or *LayoutPreparation*), only *Usage Header* or *Trailer* are valid.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	Disjointing, LayoutPreparationParams, RunList
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-230: 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.</p> <p>Values are:</p> <p><i>After</i> – This InsertSheet is to be included in the BundleItem that occurs after this InsertSheet. "After" is equivalent to "None" 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. "Before" is equivalent to "None" 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> = "Before" occurs immediately after this InsertSheet.</p>
<i>IsWaste ?</i>	boolean	<p>Specifies whether the InsertSheet is waste that is to be removed from the document before further processing. If "true", 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.</p> <p>Values include:</p> <p><i>CIELABMeasuringField</i></p> <p><i>ColorControlStrip</i></p> <p><i>ColorRegisterMark</i></p> <p><i>CutMark</i></p> <p><i>DensityMeasuringField</i></p> <p><i>IdentificationField</i></p> <p><i>JobField</i></p> <p><i>PaperPathRegisterMark</i></p> <p><i>RegisterMark</i></p> <p><i>ScavengerArea</i></p>

Table 7-230: InsertSheet Resource (Section 2 of 2)

Name	Data Type	Description
SheetFormat ? New in JDF 1.1 Modified in JDF 1.2	NMTOKEN	<p>Identifies that Device-dependent information is to be included on the InsertSheet.</p> <p>Values include:</p> <p><i>Blank</i></p> <p><i>Brief</i></p> <p><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 Sheet (duplex), then the InsertSheet has both sides duplicated. New in JDF 1.2</p> <p><i>Full</i></p> <p><i>Standard</i></p>
SheetType New in JDF 1.1	enumeration	<p>Identifies the type of Sheet.</p> <p>Values are:</p> <p><i>AccountingSheet</i> – A Sheet that reports accounting information for the Job.</p> <p><i>ErrorSheet</i> – A Sheet that reports errors for the Job.</p> <p><i>FillSheet</i> – A Sheet that fills ContentObject Elements with no matching entry in the content RunList.</p> <p><i>InsertSheet</i> – A Sheet that is inserted to the Job, (e.g., a pre-printed cover).</p> <p><i>JobSheet</i> – A Sheet that delimits the Job.</p> <p><i>SeparatorSheet</i> – A Sheet that delimits pages, sections, copies or Instance Documents of the Job.</p>
SheetUsage New in JDF 1.1 Modified in JDF 1.2	enumeration	<p>Indicates where this InsertSheet is to be produced and inserted into the set of output pages.</p> <p>Values are from: Table 7-231, "SheetUsage Attribute Values".</p>
Usage ? Deprecated in JDF 1.1	enumeration	<p>Values are from: <i>@SheetUsage</i>.</p> <p>Deprecation note: starting with JDF 1.1, use <i>SheetUsage</i>.</p>
Layout ? New in JDF 1.3	refelement	<p>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.</p>
RunList ?	refelement	<p>A RunList that provides the content for the InsertSheet. Any InsertSheet Resources referenced by this RunList are ignored.</p>
Sheet ? Deprecated in JDF 1.3	refelement	<p>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.</p> <p>Deprecation note: starting with JDF 1.3, use Layout.</p>

— Attribute: SheetUsage

Table 7-231: SheetUsage Attribute 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 next Finished front Page of the current Sheet before forcing the next page of the content RunList to the next Finished back Page if not already on a Finished back Page. Modification note: starting with JDF 1.4, this value applies to Finished pages rather than sheet surfaces.
<i>FillForceFront</i>	Valid for <i>SheetType</i> = "FillSheet". Contents of the RunList of the InsertSheet are used to fill the next Finished back Page of the current Sheet before forcing the next Page of the content RunList to the next Finished front Page if not already on a Finished front Page. A typical use is to start a chapter on the front side of the Finished Page. Modification note: starting with JDF 1.4, this value applies to Finished pages rather than sheet surfaces.
<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> New in JDF 1.2	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".
<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".

Table 7-231: SheetUsage Attribute Values (Section 2 of 2)

Value	Description
<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".
<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> . Note: use <i>SheetType</i> = "ErrorSheet" and <i>SheetUsage</i> = "Trailer" to always produce a Sheet that contains error or success information even if no errors or warnings occurred.

7.2.103 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:	—

7.2.104 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:	<i>Interpreting</i>
Output of Processes:	—

Table 7-232: 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 <i>FitPolicy</i> / <i>@SizePolicy</i> = "ClipToMaxPage" and clipping is specified.

Table 7-232: InterpretingParams Resource (Section 2 of 3)

Name	Data Type	Description
<i>FitToPage?</i> 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 .
<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. 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 specified polarity. Note that this is a polarity change in the RIP and not a polarity change in the hardware of the output Device. 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. 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.

Table 7-232: InterpretingParams Resource (Section 3 of 3)

Name	Data Type	Description
<i>ScalingOrigin</i> ?	XYPair	A pair of real values that identifies the point in the unscaled PDL page that remains at the same position after scaling. This point is defined in the coordinate system of the PDL page. For example, The <i>ScalingOrigin</i> of a PDL page with dimensions "300 400" scaled from the PDL page center would be "150 200", regardless of the value of <i>Scaling</i> . Modification note: starting with JDF 1.4, 1) the default value MAY be set to an implementation defined value; the default value is no longer specified as "0 0" in this document; 2) the phrase "PDL page" replaces "Page"; 3) this attribute specifies the point which is not shifted when scaling is applied and doesn't specify a new Origin, i.e. lower left of the page.
<i>FitPolicy</i> ? 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 <i>FitPolicy</i> Resource MUST be ignored in a Combined Process with <i>LayoutPreparation</i> .
<i>Media</i> * 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 <i>Interpreting</i> . 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 <i>Media</i> are specified, <i>Media/@MediaType</i> defines the type of <i>Media</i> . If multiple <i>Media</i> with <i>Media/@MediaType</i> = "Paper" are specified in a proofing environment, the first <i>Media</i> is the proofer paper and the second <i>Media</i> is the final Device paper.
<i>ObjectResolution</i> *	refelement	Indicates the resolution at which the PDL contents will be interpreted in DPI. These Elements MAY be different from the <i>ObjectResolution</i> 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.

7.2.104.1 Element: PDFInterpretingParams

[New in JDF 1.1](#)

Table 7-233: PDFInterpretingParams Element (Section 1 of 3)

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

Table 7-233: PDFInterpretingParams Element (Section 2 of 3)

Name	Data Type	Description
<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 <i>OCGControl</i> Subelements, are to be included in the InterpretedPDLData produced by the <i>Interpreting</i> Process. Values are: <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.
<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: <i>Design</i> – as described in [PDF1.6]. <i>View</i> – 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 <i>Interpreting</i> 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 InterpretedPDLData : Values are: <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 InterpretedPDLData . A <i>OCGZoom</i> value of 1.0 is assumed to be a magnification of 100%.
<i>PrintPDFAnnotations</i> = "false" Modified in JDF 1.3	boolean	Indicates whether the contents of annotations on PDF pages MUST be included in the output. This only refers to annotations that are set to print in the PDF file excluding trap annotations. Trap annotations are controlled with <i>PrintTrapAnnotations</i> .
<i>PrintTrapAnnotations</i> ? New in JDF 1.3	boolean	Indicates whether the contents of trap annotations on PDF pages MUST be included in the output.

Table 7-233: PDFInterpretingParams Element (Section 3 of 3)

Name	Data Type	Description
<i>TransparencyRenderingQuality</i> ?	double	Values are 0 to 1. A value of 0 represents the lowest allowable quality; 1 represents the highest desired quality.
OCGControl * New in JDF 1.3	element	Provides a list of the OCGs (layers) that are to be explicitly included or excluded in the InterpretedPDLData . Any OCGs not listed in an OCGControl Element will follow the rules set by <i>OCGDefault</i> .

7.2.104.2 Element: OCGControl

[New in JDF 1.3](#)

Table 7-234: 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 InterpretedPDLData . 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> .

7.2.104.3 More about PDFInterpretingParams

7.2.104.3.1 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.

7.2.105 JacketingParams

[New in JDF 1.1](#)

Description of the setup of the jacketing machinery. Jacket height and width (1 and 4 in the figure to the right) are specified within the **Component** that describes the jacket.

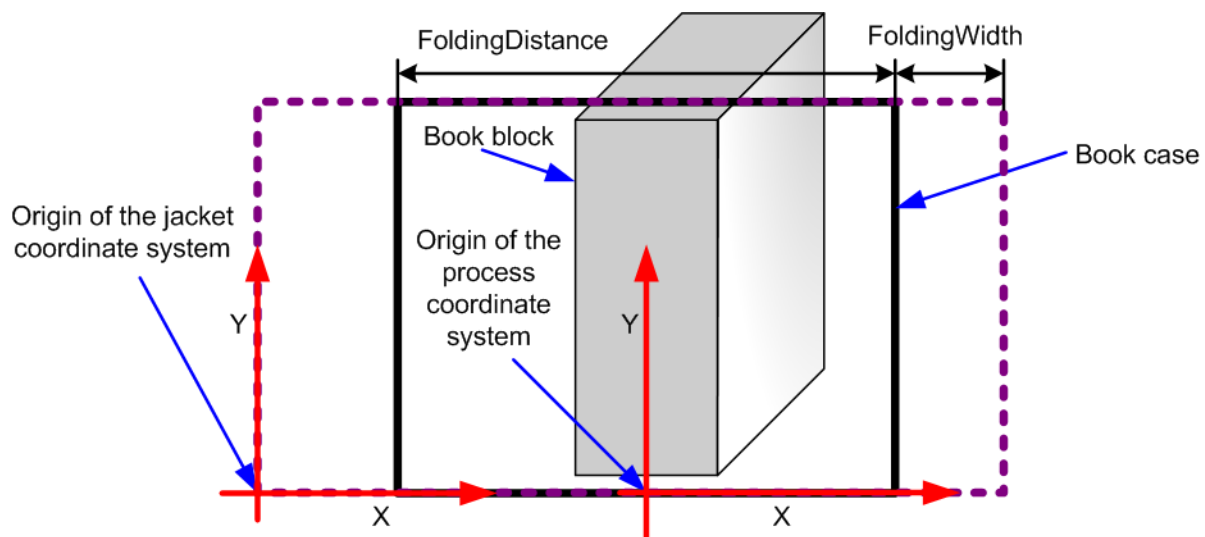
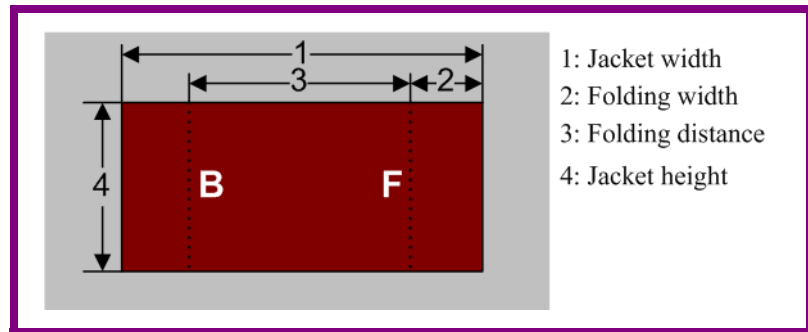


Figure 7-38: Parameters and coordinate system for jacketing

Resource Properties

Resource Class: Parameter
 Resource referenced by: —
 Example Partition: —
 Input of Processes: *Jacketing*
 Output of Processes: —

Table 7-235: JacketingParams Resource

Name	Data Type	Description
<i>FoldingDistance?</i> New in JDF 1.4	double	Distance from the fold at <i>FoldingWidth</i> to the other fold. If not specified, it defaults to width of the Jacket minus two times <i>FoldingWidth</i> (symmetrical folds).
<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.

7.2.106 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/MarkObject, LayoutPreparationParams, StrippingParams/StripMark**

Example Partition: —

Input of Processes: —

Output of Processes: —

Table 7-236: JobField Resource

Name	Data Type	Description
JobFormat ? New in JDF 1.4	string	A formatting string used with <i>JobTemplate</i> to generate a string. Values are from: "Generating strings with Format and Template" on page 909.
JobTemplate ? New in JDF 1.4	string	A list of values used with <i>JobFormat</i> to generate a string. Values are from: "Generating strings with Format and Template" on page 909.
OperatorText ?	string	Text from the operator. Note that this was erroneously described as text to the operator in JDF 1.1 and below. Constraint: starting with JDF 1.4, if <i>JobFormat</i> and <i>JobTemplate</i> are specified, <i>ShowList</i> , <i>OperatorText</i> and <i>UserText</i> MUST NOT be specified.
ShowList ? Modified in JDF 1.4	NMTOKENS	List of elements to display in the JobField . Constraint: starting with JDF 1.4, if <i>JobFormat</i> and <i>JobTemplate</i> are specified, <i>ShowList</i> <i>OperatorText</i> and <i>UserText</i> MUST NOT be specified. Values include those from: Table I-1, "Predefined variables used in @XXXTemplate and @ShowList" on page 909 New in JDF 1.4 Modification note: starting with JDF 1.4, the values come from a common list rather than a list that is custom to this Resource. In addition, <i>ShowList</i> becomes optional.
UserText ?	string	User-defined text to output with JobField . Constraint: starting with JDF 1.4, if <i>JobFormat</i> and <i>JobTemplate</i> are specified, <i>ShowList</i> , <i>OperatorText</i> and <i>UserText</i> MUST NOT be specified.
DeviceMark ? Modified in JDF 1.3 Deprecated in JDF 1.4	refelement	DeviceMark defines the formatting parameters for the mark. If not specified, the settings defined in LayoutPreparationParams/DeviceMark are assumed. Deprecation note: starting with JDF 1.4, DeviceMark MUST be specified in the parent MarkObject Element.

7.2.107 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:	<i>Labeling</i>
Output of Processes:	—

Table 7-237: 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. Values are: <i>Back</i> <i>Bottom</i> <i>Front</i> <i>Left</i> <i>Right</i> <i>Top</i>

7.2.108 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:	<i>Laminating</i>
Output of Processes:	—

Table 7-238: LaminatingParams Resource (Section 1 of 2)

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.

Table 7-238: LaminatingParams Resource (Section 2 of 2)

Name	Data Type	Description
<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. Values are: <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 <i>Laminating</i> Process.
<i>Temperature</i> ?	double	Temperature used in the <i>Laminating</i> Process, in ° Centigrade.

7.2.109 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: **LayoutIntent, Component, CylinderLayout, InsertSheet**

Example Partition: *SignatureName, WebName, RibbonName, SheetName, Side, PartVersion*

Input of Processes: *ConventionalPrinting, CylinderLayoutPreparation, DigitalPrinting, Imposition, InkZoneCalculation*

Output of Processes: *LayoutPreparation, Stripping*

Table 7-239: Layout Resource (Section 1 of 4)

Name	Data Type	Description
<i>Automated</i> ?	boolean	If " <i>true</i> ", the <i>Imposition</i> Process is expected to perform automated imposition. Layout/@Automated MUST only be specified in the root Partition of Layout . Default value is: " <i>false</i> " in the root Partition of Layout
<i>BaseOrdReset</i> = <i>"PagePool"</i> New in JDF 1.4	enumeration	Policy about how the <i>Ord</i> Attribute of an entry MUST be calculated when extracting a page from a RunList and positioning it in the Layout . Values are: <i>PagePool</i> – The Base Ord is reset to point to the first page entry of the Page Pool at the beginning of each Page Pool processed by the Imposition Template. <i>PagePoolList</i> – At the beginning of processing of the Imposition Template, the Base Ord is reset to point to the first page entry of the first Page Pool to be processed by the Imposition Template. This results in all Page Pools that will be processed by the Imposition Template to be treated as a Page Pool List.

Table 7-239: Layout Resource (Section 2 of 4)

Name	Data Type	Description
<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 <i>Sheet</i> Resource.
<i>MaxCollect</i> ? New in JDF 1.4	integer	Maximum number of Sheets that will be collected into a signature. <i>MaxCollect</i> modifies the pagination when automated imposition is selected. Specifying <i>MaxCollect</i> can effectively cause a Page Pool or Page Pool List to be broken into "sub" Page Pools. Each of these "sub" Page Pools provides the set of pages mapped onto a single Collect, and are processed sequentially out of the "parent" Page Pool (or Page Pool List). Thus each sub-Page Pool effectively restarts the ord counting within the Imposition Template (i.e. treat a sub-Page Pool as if a new Page Pool were being started with the Imposition Template). If not specified, all sheets MUST be collected.
<i>MaxDocOrd</i> = "1" New in JDF 1.1 Deprecated in JDF 1.4	integer	Zero-based maximum number of Instance Documents that are consumed from a <i>RunList</i> each time the <i>Layout</i> is executed, assuming the <i>Imposition</i> Process is automated. Deprecation note: see <i>MaxOrd</i> .
<i>MaxOrd</i> ? Deprecated in JDF 1.4	integer	Zero-based maximum number of placed objects that are consumed from a <i>RunList</i> each time the <i>Layout</i> is executed, assuming the <i>Imposition</i> Process is automated. If not specified, it MUST be calculated from the <i>Ord</i> values of the <i>ContentObject</i> Elements in the <i>Layout</i> . Deprecation note: <i>MaxOrd</i> has no meaning if negative <i>Ord</i> values exist in an automated <i>Layout</i> . The consumer MUST calculate the implied 2 values for increasing and decreasing the explicit <i>Ord</i> values in an automated <i>Layout</i> by evaluating the actual values of <i>ContentObject/@Ord</i> . Increment from Front = $1 + \max(\text{ContentObject}/@Ord_+)$ where " <i>Ord_+</i> " specifies positive values of <i>Ord</i> ; Decrement from Back = $\max(\text{abs}(\text{ContentObject}/@Ord_-))$ where " <i>Ord_-</i> " specifies negative values of <i>Ord</i> . See ref automated <i>Layout</i> for details.
<i>MaxSetOrd</i> = "1" New in JDF 1.1 Deprecated in JDF 1.4	integer	Zero-based maximum number of Document Sets that are consumed from a <i>RunList</i> each time the <i>Layout</i> is executed, assuming the <i>Imposition</i> Process is automated. Deprecation note: see <i>MaxOrd</i> .
<i>MinCollect</i> ? New in JDF 1.4	integer	Minimum number of Sheets that will be collected into a signature. <i>MinCollect</i> modifies the pagination when automated imposition is selected.
<i>Name</i> ? New in JDF 1.1 Deprecated in JDF 1.4	string	Unique name of the <i>Layout</i> . The <i>Name</i> is used for external reference to a <i>Layout</i> . Deprecation note: starting with JDF 1.4, use <i>DescriptiveName</i> .

Table 7-239: Layout Resource (Section 3 of 4)

Name	Data Type	Description
<i>OrdsConsumed?</i> New in JDF 1.4	Integer-RangeList	Range of <i>Ord</i> values of the RunList (<i>Document</i>) that are consumed by this Layout section. MUST NOT be specified unless <i>@Automated = "true"</i> .
<i>SheetCountReset?</i> New in JDF 1.4	enumeration	<p>Policy as to when the automated imposition variables <i>SheetCount</i> and <i>TotalSheetCount</i> are reset. See Section 6.4.17.2, "Variables for Automated Imposition" on page 289.</p> <p>Values are:</p> <p><i>Continue</i> – <i>SheetCount</i> continues to increment for each sheet generated by the current Imposition Template.</p> <p><i>PagePool</i> – <i>SheetCount</i> is reset to zero upon start of processing of a new Page Pool and <i>TotalSheetCount</i> is determined for that new Page Pool..</p> <p><i>PagePoolList</i> – <i>SheetCount</i> is reset to zero upon start of processing of an Imposition Template, and <i>TotalSheetCount</i> is recalculated.</p> <p>Note that the value of <i>TotalSheetCount</i> may depend on the sheets generated from successive Imposition Templates (for example, if the current Imposition Template has <i>SheetCountReset = "PagePoolList"</i>, and the subsequent Imposition Template has <i>SheetCountReset = "Continue"</i>, <i>TotalSheetCount</i> will include the sheets generated by both Imposition Templates.</p>
<i>SheetNameFormat?</i> New in JDF 1.4	string	<p>A formatting string used with <i>SheetNameTemplate</i> to algorithmically construct <i>SheetName</i>. <i>SheetNameFormat</i> and <i>SheetNameTemplate</i> are used to identify individual parts of the Layout in an automated environment.</p> <p>Values are from: "Generating strings with Format and Template" on page 909.</p>
<i>SheetNameTemplate?</i> New in JDF 1.4	string	<p>A list of values used with <i>SheetNameFormat</i> to algorithmically construct <i>SheetName</i>. <i>SheetNameFormat</i> and <i>SheetNameTemplate</i> are used to identify individual parts of the Layout in an automated environment.</p> <p>Values are from: "Generating strings with Format and Template" on page 909.</p>
<i>SourceWorkStyle?</i> New in JDF 1.3	WorkStyle	Indicates which <i>WorkStyle</i> was used to create the Layout . This is only informative and can be useful when creating double sided proofs.
<i>SurfaceContentsBox?</i> New in JDF 1.3	rectangle	<p>This box, specified in Layout-coordinate space, defines the area into which <i>MarkObject</i> or <i>ContentObject</i> 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".</p> <p><i>SurfaceContentsBox</i> MAY imply clipping.</p> <p>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.</p>

Table 7-239: Layout Resource (Section 4 of 4)

Name	Data Type	Description
InsertSheet * Deprecated in JDF 1.4	refelement	Additional Sheets that are to be inserted before and/or after a document. Depending on which Partition level the InsertSheet is defined, it specifies how to complete the Sheet or surface in an automated printing environment. Deprecation note: starting with JDF 1.4, use Layout/PageCondition for <i>FillSheet</i> , <i>FillSurface</i> , and <i>FillSignature</i> operations; use Layout/SheetCondition for an insert sheet.
LayerList ? New in JDF 1.1	element	List of <i>LayerDetails</i> Elements.
LogicalStackParams ? New in JDF 1.4	element	When specified, configures the imposition engine to place content onto one or more Logical Stacks distributed on a common set of sheets. Layout/LogicalStackParams MUST only be specified in the root Layout element AND only when Layout/@Automated = "true". All Logical Stacks defined by LogicalStackParams MUST be used in all Imposition Templates." See Section 6.4.17.4.1, "Using Logical Stacks" on page 292.
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.
PageCondition * New in JDF 1.4	element	The PageCondition Elements are used only with automated imposition. They define restrictions on which page content may be placed in a Layout/ContentObject and. If any PageCondition restricts placing a page into a ContentObject , the Page MUST NOT be filled into that ContentObject .
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 608. Note: PlacedObject is not a container but an Abstract type.
SheetCondition ? New in JDF 1.4	element	Specifies the conditions under which the optional sheet defined by this Layout is produced. MUST only be present when Layout/@Automated = "true", and MUST be contained within a Layout branch partitioned by <i>SheetName</i> .
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.

7.2.109.1 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 7-240: LayerList Element

Name	Data Type	Description
<i>LayerDetails</i> *	element	Details of the individual layers.

7.2.109.2 Element: LayerDetails[New in JDF 1.1](#)

This Element provides information about individual layers.

Table 7-241: LayerDetails Element

Name	Data Type	Description
<i>Name</i> ?	string	Unique name of the layer.

7.2.109.3 Element: LogicalStackParams[New in JDF 1.4](#)**Table 7-242: LogicalStackParams Element**

Name	Data Type	Description
<i>MaxStackDepth</i> ?	integer	Maximum number of imposed sheets to generate as an Imposed Sheet Set (the size of the Logical Stack). Implementations MUST generate the minimum stack size to accommodate the available number of Logical Sheets if the total number of required sheets for the last stack is smaller than <i>MaxStackDepth</i> .
<i>Restrictions</i> = "None"	enumeration	Describes any restrictions set on the placement of a Recipient Set's Logical Sheets within or across Imposed Sheet Sets. Values are: <i>None</i> – a recipient set's Logical Sheets may be placed across both Logical Stacks and Imposed Sheet Sets. <i>WithinImposedSheetSet</i> – a Recipient Set's Logical Sheets MUST be placed within a single Imposed Sheet Set <i>WithinLogicalStack</i> – a Recipient Set's Logical Sheets MUST be placed within a single Logical Stack.
<i>Stack</i> +	element	Describes parameters to control the sequencing of Logical Sheets onto individual Logical Stacks.

7.2.109.4 Element: Stack[New in JDF 1.4](#)**Table 7-243: Stack Element (Section 1 of 2)**

Name	Data Type	Description
<i>LogicalStackOrd</i>	integer	0-based Logical Stack identifier that specifies which Logical Stack is controlled by this Stack Element. The value of <i>Stack/@LogicalStackOrd</i> MUST correspond to a <i>PlacedObject/@LogicalStackOrd</i> value.

Table 7-243: Stack Element (Section 2 of 2)

Name	Data Type	Description
<i>LogicalStackSequence</i> = " <i>SheetIndex</i> "	enumeration	Specifies how Logical Sheets MUST be placed onto the Logical Stack. Values are: <i>SheetIndex</i> – Logical Sheets are placed in the order of ascending <i>SheetIndex</i> . <i>DescendingSheetIndex</i> – Logical Sheets are placed in the order of descending <i>SheetIndex</i> .

7.2.109.5 Element: PageCondition

[New in JDF 1.4](#)

The PageCondition Element defines restrictions on when page content MUST NOT be placed in a ContentObject of a **Layout**. Before placing page content from a **RunList** into a ContentObject the PageCondition/@*RestrictedContentObjects* Attribute MUST be checked for the *Ord* of the ContentObject. If the *Ord* of the ContentObject is in the *RestrictedContentObjects* Attribute Value, the alternate content, if any, MUST be placed in the ContentObject. After skipping a restricted ContentObject, the **Imposition** Process MUST then place the current page content into the location defined by the next ContentObject (after that specified by the *RestrictedContentObject*). This corresponds to incrementing the effective *Ord* value of the page in the **RunList** by 1, effectively incrementing the total number of pages of the **RunList**. If the next ContentObject is also restricted then the process is repeated. PageCondition Elements are processed in their XML order.

Table 7-244: PageCondition Element (Section 1 of 2)

Name	Data Type	Description
<i>RestrictedContentObjects</i>	IntegerList	List of @ <i>Ord</i> values of those ContentObject Elements into which page content that matches the conditions as specified in Part or PageCondition/@ <i>Condition</i> MUST NOT be placed.
<i>Condition</i> ?	enumeration	Specifies the conditions when the PageCondition applies. Condition MUST NOT be specified if Part Elements are present. Values are: <i>PagePoolStart</i> – the condition is true when the <i>Ord</i> refers to the first page of a Page Pool in the RunList . <i>PagePoolEnd</i> – after processing of the Page Pool is completed, the condition is true for all unused <i>Ord</i> positions in the current Collect. <i>PagePoolListStart</i> – the condition is true when the <i>Ord</i> refers to the first page of an aggregated set of Page Pools in the RunList . <i>PagePoolListEnd</i> – after processing of the Page Pool list is completed, the condition is true for all unused <i>Ord</i> positions in the current Collect.
<i>RunList</i> ?	refelement	Alternate page content that MUST be placed into the ContentObject Elements that are specified in <i>RestrictedContentObjects</i> when the PageCondition evaluates to " <i>true</i> ". The first page of the referenced RunList MUST be used.

Table 7-244: PageCondition Element (Section 2 of 2)

Name	Data Type	Description
Part *	element	Specifies the conditions when the PageCondition applies. Multiple Part Elements specify alternate page conditions (ORing of them). Part Elements MUST NOT be specified if <i>Condition</i> is present.

Example 7-23: PageCondition[New in JDF 1.4](#)

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Layout Class="Parameter" ID="L000004" Status="Available"
      PartIDKeys="SheetName Side">
      <PageCondition RestrictedContentObjects="1 -1">
        <!-- Force document with Chapter starts onto a right hand side (facing)
          page -->
        <Part DocTags="Chapter" DocRunIndex="0"/>
      </PageCondition>
      <Layout SheetName="Mysheet">
        <Layout Side="Front">
          <ContentObject CTM="1 0 0 1 0 0" Ord="-1"/> <!-- Outside left -->
          <ContentObject CTM="1 0 0 1 595 0" Ord="0" /> <!-- outside right -->
        </Layout>
        <Layout Side="Back">
          <ContentObject CTM="1 0 0 1 0 0" Ord="1"/> <!-- inside left-->
          <ContentObject CTM="1 0 0 1 595 0" Ord="-2"/> <!-- inside right-->
        </Layout>
      </Layout>
    </Layout>
  </ResourcePool>
  <ResourceLinkPool>
    <LayoutLink Usage="Input" rRef="L000004"/>
  </ResourceLinkPool>
</JDF>

```


7.2.109.6 Element: SheetCondition

[New in JDF 1.4](#)

Table 7-245: SheetCondition Element

Name	Data Type	Description
<i>Condition?</i>	enumerations	<p>When present, defines an optional sheet by specifying each condition (equivalent to a logical or) under which the optional sheet is produced.</p> <p>Values include:</p> <p><i>Begin</i> – At beginning of imposition processing (all sets in case of multiple Recipient Sets)</p> <p><i>End</i> – At the end of imposition processing.</p> <p><i>BeginSet</i> – At beginning of processing of an individual Recipient Set.</p> <p><i>EndSet</i> – At end of processing of an individual Recipient Set.</p> <p><i>PagePoolBegin</i> – At beginning of processing of a Page Pool.</p> <p><i>PagePoolEnd</i> – At the end of processing of a Page Pool.</p> <p><i>PagePoolListBegin</i> – At beginning of processing of a Page Pool List</p> <p><i>PagePoolListEnd</i> – At end of processing of a Page Pool List</p> <p><i>LogicalStackBegin</i> – adds a Logical Sheet to the beginning of each Logical Stack generated as part of an Imposed Sheet Set.</p> <p><i>LogicalStackEnd</i> – adds a Logical Sheet to the end of each Logical Stack generated as part of an Imposed Sheet Set.</p> <p><i>LogicalStackSetBegin</i> – At beginning of generation of a set of Logical Stacks. Note that <i>LogicalStackOrd</i> MUST be used to indicate the Logical Stack on which the conditional sheet is placed.</p> <p><i>LogicalStackSetEnd</i> – At end of generation of a set of Logical Stacks. Note that <i>LogicalStackOrd</i> MUST be used to indicate the Logical Stack on which the conditional sheet is placed.</p> <p><i>ImposedSheetSetBegin</i> – At beginning of generation of an Imposed Sheet Set. Note that this generates a separator sheet not counted as part of the Imposed Sheet Set. Any MarkObject Elements specifying <i>LogicalStackOrd</i> are ignored.</p> <p><i>ImposedSheetSetEnd</i> – At end of generation of an Imposed Sheet Set. Note that this generates a separator sheet not counted as part of the Imposed Sheet Set. Any MarkObject Elements specifying <i>LogicalStackOrd</i> are ignored.</p>
<i>RunList?</i>	refelement	Supplies content for any ContentObject Elements specified within the optional sheet definition. All ContentObject Elements in the optional sheet definition MUST reference content supplied by this RunList .

7.2.109.7 Element: PlacedObject

The marks that are to be placed on the designated surface of a **Layout** come in two varieties: **ContentObject** or **MarkObject** Elements. All inherit characteristics from the **Abstract PlacedObject** which is described below.

7.2.109.7.1 Element: Abstract PlacedObject

Table 7-246: Abstract PlacedObject Element (Section 1 of 3)

Name	Data Type	Description
<i>Anchor</i> ? New in JDF 1.4	Anchor	Specifies the anchor point of the PlacedObject that remains in place on the surface when the value of <i>TrimSize</i> changes. <i>Anchor</i> is specified in the coordinate system of the PlacedObject prior to application of the <i>CTM</i> .
<i>ClipBox</i> ?	rectangle	Clipping rectangle in the coordinates of the <i>SurfaceContentsBox</i> . <i>ClipBox</i> MUST NOT be present if PlacedObject/@ClipBoxFormat is supplied.
<i>ClipBoxFormat</i> ? New in JDF 1.4	string	A formatting string used with <i>ClipBoxTemplate</i> to algorithmically construct <i>ClipBox</i> . <i>ClipBoxFormat</i> MUST ONLY be present if PlacedObject/@ClipBox is not supplied and Layout/@Automated = "true" . Values are from: "Generating strings with Format and Template" on page 909.
<i>ClipBoxTemplate</i> ? New in JDF 1.4	string	A list of values used with <i>ClipBoxFormat</i> to algorithmically construct <i>ClipBox</i> . <i>ClipBoxTemplate</i> MUST ONLY be present if PlacedObject/@ClipBox is not supplied and Layout/@Automated = "true" . Values are from: "Generating strings with Format and Template" on page 909.
<i>ClipPath</i> ? 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).
<i>CompensationCTMFormat</i> ? New in JDF 1.4	string	A formatting string used with <i>CompensationCTMTemplate</i> to algorithmically construct a compensation CTM that MUST be concatenated to CTM. <i>CompensationCTMFormat</i> MAY be present if Layout/@Automated = "true" . Values are from: "Generating strings with Format and Template" on page 909.

Table 7-246: Abstract PlacedObject Element (Section 2 of 3)

Name	Data Type	Description
<i>CompensationCTMTemplate</i> ? New in JDF 1.4	string	A list of values used with <i>CompensationCTMFormat</i> to algorithmically construct a compensation CTM that MUST be concatenated to CTM. <i>CompensationCTMTemplate</i> MAY be present if <i>Layout/@Automated</i> = "true". Values are from: "Generating strings with Format and Template" on page 909.
<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 31. 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 28. 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> specifies the index of the Layout/LayerList/LayerDetails Element in Layout/LayerList that a ContentObject belongs to, (e.g., the language layer version). The details of the layers are specified in the Layout/LayerList/LayerDetails Element.
<i>LogicalStackOrd</i> ? New in JDF 1.4	integer	0-based Logical Stack identifier that this PlacedObject belongs to. <i>LogicalStackOrd</i> MUST match the <i>LogicalStackOrd</i> of an entry in Layout/LogicalStackParams/Stack .
<i>OrdID</i> ? New in JDF 1.1	integer	If a Layout supports layering (e.g., for versioning), elements that belong to the same final page SHOULD have a matching <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>TrimClipPath</i> ? New in JDF 1.4	PDFPath	The die cutting 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>). That path can be used for proofing purpose.

Table 7-246: Abstract PlacedObject Element (Section 3 of 3)

Name	Data Type	Description
<i>TrimCTM</i> ? New in JDF 1.1	matrix	The transformation matrix of the trim box to be applied to the object's referenced content in the coordinate system of <i>SurfaceContentsBox</i> . Note that imposition programs that execute the Layout MUST recalculate the <i>CTM</i> in case the referenced content is replaced with new referenced content having different dimensions, otherwise the position of the content inside the trim box will shift. This recalculation is based on <i>Anchor</i> , <i>TrimCTM</i> , <i>TrimSize</i> and trim box.
<i>TrimSize</i> ? New in JDF 1.2	XYPair	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. 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). Usage of both <i>PlacedObject</i> Elements <i>TrimCTM</i> and <i>TrimSize</i> Attributes will allow page replacements on any type of imposition Layout .
<i>Type</i> ? Deprecated in JDF 1.1	enumeration	Describes the kind of <i>PlacedObject</i> . Values are: <i>Content</i> <i>Mark</i>
<i>FitPolicy</i> ? New in JDF 1.4	refelement	MUST NOT be present when <i>Layout/@Automated</i> = " <i>false</i> ". Specifies automated fit policy for the page cell described by the <i>PlacedObject</i> . When present, <i>PlacedObject/@TrimSize</i> MUST also be present in the <i>PlacedObject</i> , and represents the cell size for this <i>PlacedObject</i> .

7.2.109.8 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 7-247: ContentObject Element

Name	Data Type	Description
DocOrd? New in JDF 1.1	integer	Reference to an index of an Instance Document in the content RunList . This references an Instance Document with an index module. Layout/@MaxDocOrd equals <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 1.4	integer	A 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". If Layout/@Automated = "true", <i>Ord</i> MAY be a negative integer in a ContentObject . In this case, the explicit <i>Ord</i> for each iteration of the automated Layout is calculated by subtracting the appropriate number of <i>Ord</i> values from the back of the document. For details on automated Layout , see Section 6.4.17, "Imposition" on page 284.
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 PlacedObject . Value format is from: Section 7.2.109.12.5, "Using Expressions in the OrdExpression Attribute" on page 611.
SetOrd? New in JDF 1.1	integer	A non-negative, zero-based reference to an index of a Document Set in the content RunList . This references an Instance Document with an index module. Layout/@MaxSetOrd = <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).

7.2.109.9 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 7-248: MarkObject Element (Section 1 of 3)

Name	Data Type	Description
ContentRef? New in JDF 1.4	IDREF	<i>ContentRef</i> refers to the ContentObject that this MarkObject is related to. <i>ContentRef</i> is used to define the object that metadata for generating dynamic marks MAY be extracted from.
LayoutElementPageNum? New in JDF 1.1 Modified in JDF 1.3 Deprecated in JDF 1.4	integer	Page number to use from the PDL file described by LayoutElement . Modification note: starting with JDF 1.3, the default value of "0" is removed. Deprecation note: starting with JDF 1.4, PDL for Marks MUST be referenced via RunList (<i>Marks</i>).

Table 7-248: MarkObject Element (Section 2 of 3)

Name	Data Type	Description
Ord? Modified in JDF 1.4	integer	A non-negative reference to an index in the RunList (<i>Marks</i>). The index is incremented for every page of the RunList with <i>IsPage = "true"</i> . The first page of a RunList has the value 0. Modification note: starting with JDF 1.4, at most one of <i>Ord</i> or DeviceMark MUST be specified. For JDF 1.3 only, at most one of LayoutElement , <i>Ord</i> or JobField MUST be specified.
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 1.4	refelement	DeviceMark specifies all formatting options for dynamic Marks JobField/DeviceMark specifies the formatting parameters of the JobField and all other dynamically generated marks are positioned with <i>CTM</i> . Constraint: at most one of <i>Ord</i> or DeviceMark MUST be specified. Creation note: starting with JDF 1.4, DeviceMark is back after being deprecated in JDF 1.3.
DynamicField *	element	Definition of text replacement for a MarkObject . MarkObject/DynamicField specifies text replacement within an existing PDL mark.
IdentificationField *	refelement	Specific information about this kind of mark object. See below for information regarding dynamically generated marks.
JobField ? New in JDF 1.1 Modified in JDF 1.4	refelement	JobField specifies the metadata of a given dynamic slug line. Modification note: starting with JDF 1.4, the maximum number of JobField Elements per MarkObject is limited to 1; previously, there was no limit. For JDF 1.3 only, at most one of LayoutElement , <i>Ord</i> or JobField SHOULD be specified.
LayoutElement ? Deprecated in JDF 1.4	refelement	PDL description of the mark. The LayoutElement and <i>Ord</i> are mutually exclusive within one MarkObject . Modification note: for JDF 1.3 only, at most one of LayoutElement , <i>Ord</i> or JobField MUST be specified. Deprecation note: starting with JDF 1.4, PDL for Marks MUST be referenced via RunList (<i>Marks</i>).
MarkActivation * New in JDF 1.4	element	Rules about when to apply the mark in an automated Layout . If no MarkActivation is specified, the MarkObject is unconditionally active. If multiple MarkActivation Elements are specified, all conditions MUST be met for the mark to be active. MarkActivation MUST NOT be specified unless Layout/@Automated = "true" .

Table 7-248: MarkObject Element (Section 3 of 3)

Name	Data Type	Description
RefAnchor ? New in JDF 1.4	element	Details of the coordinate system that this mark is placed relative to. This MAY be either the sheet coordinate system or the coordinate system of a referenced PlacedObject. If the anchor point in the referenced object (PlacedObject or Sheet surface) is modified, e.g. due to a change in <i>TrimSize</i> , the CTM of the placed object of this DeviceMark MUST be modified accordingly. Note: RefAnchor does NOT modify the origin of the CTM of this PlacedObject. It is only used to recalculate relative shifts.
RegisterMark * Modified in JDF 1.1	refelement	Specific information about this kind of mark object. See below for information regarding dynamically generated marks.
ScavengerArea * New in JDF 1.1	refelement	Specific information about this kind of mark object. See below for information regarding dynamically generated marks.

7.2.109.10 Element: MarkActivation[New in JDF 1.4](#)

MarkActivation specifies condition when to apply the mark in an automated Layout.

Table 7-249: MarkActivation Element

Name	Data Type	Description
<i>Context</i>	NMTOKEN	The context in which the iteration is counted. Values include: <i>CollectSheetIndex</i> – a parameter maintained by the imposition engine to count sheets (e.g. in the context of a signature). Its value starts at 0 and is incremented by one for each sheet. If <i>Layout/@MaxCollect</i> is specified, its maximum value is one less than <i>Layout/@MaxCollect</i> . Otherwise, it continues to increment per sheet until completion of the page-pool/page-pool-list processing through the Imposition Template. See Section 6.4.17, "Imposition" on page 284. <i>DocIndex</i> – a Partition Key. <i>SetDocIndex</i> – a Partition Key. <i>SetIndex</i> – a Partition Key. <i>SheetIndex</i> – a Partition Key. <i>SubDocIndex0, ...</i> – a parameter maintained by the imposition engine. See Section 6.4.17, "Imposition" on page 284.
<i>Index</i>	IntegerRangeList	The enclosing MarkObject is active and its specified Mark MUST be imaged if the value of the variable specified by <i>Context</i> is equal to one of the values of this Attribute.

7.2.109.10.1 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.

7.2.109.11 Element: DynamicField

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.

Table 7-250: DynamicField Element

Name	Data Type	Description
<i>Format</i>	string	Format string in C printf format that defines the replacement. Values are from: "Generating strings with Format and Template" on page 909
<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> ? Deprecated in JDF 1.4	integer	Reference to an index in the Contents RunList that contains DynamicInput Elements. Constraint: at most one of <i>Ord</i> or <i>OrdExpression</i> MUST be specified. Deprecation note: starting with JDF 1.4, <i>Ord</i> MUST be specified in the parent MarkObject Element.
<i>OrdExpression</i> ? Deprecated in JDF 1.4	string	Expression to calculate the reference to an index in the Contents RunList that contains DynamicInput fields. Values include those from: <i>ContentObject/@OrdExpression</i> Constraint: 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 <i>MarkObject/Ord</i> , <i>MarkObject/OrdExpression</i> or <i>MarkObject/LayoutElement</i> . 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> . Values are from: "Generating strings with Format and Template" on page 909. Deprecation note: starting with JDF 1.4, <i>RunList/DynamicInput/@Name</i> (mentioned here in JDF 1.3) no longer defines further variables because DynamicInput has been deprecated.
<i>DeviceMark</i> ? New in JDF 1.1 Deprecated in JDF 1.4	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.

Example 7-24: Layout: DynamicField Element

In this example, the text “__xxx__” in the file MyReplace.pdf would be replaced by the sentence “Replacement Text for Joe and John go in here at 14:00 on Mar-31-2000”. MyReplace.pdf is placed at the position defined by the *CTM* of the *MarkObject* and Variable.pdf is placed at the position defined by the *CTM* of the *ContentObject*.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <RunList Class="Parameter" ID="L3" PartIDKeys="Run" Status="Available">
      <MetadataMap DataType="String" Name="i1" ValueFormat="%s"
        ValueTemplate="s1">
        <!--This expression maps the value of /Dokument/Rezipient/@Name to a
          variable "s1"-->
        <Expr Name="s1" Path="/Dokument/Rezipient/@Name"/>
      </MetadataMap>
      <LayoutElement ElementType="Graphic">
        <FileSpec URL="File:///Variable.pdf"/>
      </LayoutElement>
    </RunList>
    <Layout Class="Parameter" ID="Link0003" Status="Available">
      <!--The MarkObject in the Layout hierarchy: -->
      <ContentObject CTM="1 0 0 1 0 0" Ord="0"/>
      <MarkObject CTM="1 0 0 1 10 10">
        <LayoutElement ElementType="Graphic">
          <FileSpec URL="File:///MyReplace.pdf"/>
        </LayoutElement>
        <DynamicField
          Format="Replacement Text for %s goes in here at %s on %s"
          Ord="0" ReplaceField="__xxx__" Template="i1,Time,Date"/>
        <DynamicField Format="More Replacement Text for %s go in here"
          Ord="0" ReplaceField="__yyy__" Template="SignatureName"/>
      </MarkObject>
    </Layout>
  </ResourcePool>
  <ResourceLinkPool>
    <LayoutLink Usage="Input" rRef="Link0003"/>
    <RunListLink Usage="Input" rRef="L3"/>
  </ResourceLinkPool>
</JDF>
```

7.2.109.12 More about Layout

7.2.109.12.1 Migrating from a Pre-JDF 1.3 Layout to a Partitioned Layout

[New in JDF 1.3](#)

The **Layout** Resource was significantly modified in JDF 1.3. This section describes how a pre-JDF 1.3 **Layout** can be transformed into a JDF 1.3 **Layout** and what restrictions MAY be applied to a JDF 1.3 **Layout** so that it can be easily transformed into a pre-JDF 1.3 **Layout** or a PJTF Layout.

Note: this section is not applicable when *Layout/@Automated* = "true" for any Partitions.

7.2.109.12.1.1 Partition Key restrictions:

If *SignatureName* *SheetName* or *Side* are specified in *PartIDKeys*, the order MUST be specified as *SignatureName SheetName Side*.

Only a **Layout** with exactly *PartIDKeys* = "*SignatureName SheetName Side*" can be translated into a JDF 1.2 **Layout** or a PJTF. Thus, it is highly RECOMMENDED to use exactly this Partitioning of the **Layout** in JDF 1.3 whenever possible. Any other Partitioning will make consumption by existing products very unlikely.

7.2.109.12.1.2 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.

Example 7-25: Invalid MarkObject

The following INVALID example is correct according to "Subelements in Partitioned Resources" on page 94. 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">:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Layout Class="Parameter" ID="L3" Status="Available"
      PartIDKeys="SignatureName SheetName Side">
      <!-- INVALID, this PlacedObject is not in a leaf partition and not used -->
      <!-- since it is overwritten by <MarkObject Ord="1"> -->
      <MarkObject Ord="0" CTM="0.0 1.0 -1.0 0.0 176.69 23.62" >
        <RegisterMark Center="0.0 0.0" MarkType="Cross" MarkUsage="PaperPath" />
      </MarkObject>
      <Layout SignatureName="Sig00">
        <!-- INVALID, this PlacedObject is not in a leaf partition -->
        <MarkObject Ord="1" CTM="0.0 1.0 -1.0 0.0 176.69 23.62" >
          <RegisterMark Center="0.0 0.0" MarkType="Cross"
            MarkUsage="PaperPath" />
        </MarkObject>
        <Layout SheetName="Sheet00">
          <Layout Side="Front">
            <MarkObject Ord="2" CTM="0.0 1.0 -1.0 0.0 176.69 23.62" >
              <RegisterMark Center="0.0 0.0" MarkType="Arc"
                MarkUsage="PaperPath" />
            </MarkObject>
            <ContentObject CTM="0.0 1.0 -1.0 0.0 176.69 23.62" Ord="0" />
          </Layout>
        </Layout>
        <Layout SheetName="Sheet01">
          <Layout Side="Front">
            <!-- Not clear whether this is layered over or under
              <MarkObject Ord="0">
            -->
            <ContentObject CTM="0.0 1.0 -1.0 0.0 176.69 23.62" Ord="0" />
          </Layout>
        </Layout>
      </Layout>
    </Layout>
  </ResourcePool>
  <ResourceLinkPool>
    <LayoutLink Usage="Input" rRef="L3"/>
  </ResourceLinkPool>
</JDF>
```

Example 7-26: MarkObject

This VALID example contains the same PlacedObject Elements as the previous example but they are correctly specified in the leaves of the Partitioned Layout.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Layout Class="Parameter" ID="L3" Status="Available"
      PartIDKeys="SignatureName SheetName Side">
      <Layout SignatureName="Sig00">
      <Layout SheetName="Sheet00">
      <Layout Side="Front">
```

```

    <MarkObject Ord="2" CTM="0.0 1.0 -1.0 0.0 176.69 23.62" >
      <RegisterMark Center="0.0 0.0" MarkType="Arc"
        MarkUsage="PaperPath" />
    </MarkObject>
    <ContentObject CTM="0.0 1.0 -1.0 0.0 176.69 23.62" Ord="0" />
  </Layout>
</Layout>
<Layout SheetName="Sheet01">
  <Layout Side="Front">
    <MarkObject Ord="1" CTM="0.0 1.0 -1.0 0.0 176.69 23.62" >
      <RegisterMark Center="0.0 0.0" MarkType="Cross"
        MarkUsage="PaperPath" />
    </MarkObject>
    <ContentObject CTM="0.0 1.0 -1.0 0.0 176.69 23.62" Ord="0" />
  </Layout>
</Layout>
</Layout>
</ResourcePool>
<ResourceLinkPool>
  <LayoutLink Usage="Input" rRef="L3" />
</ResourceLinkPool>
</JDF>

```

7.2.109.12.2 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.

7.2.109.12.3 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**

7.2.109.12.4 Using Ord to Reference Elements in RunList Resources

[New in JDF 1.1A](#)

The `Ord` Attribute in `ContentObject` or `MarkObject` Elements represents a reference to a *logical* element in a `RunList`. The index is incremented for every page of the `RunList` with `IsPage = "true"`. The reference is not

changed by repartitioning the **RunList**. The content and marks **RunList** are referenced independently. The following examples illustrate the usage of *Ord*.

Example 7-27: RunList: Simple Multi-File Unseparated RunList

This example specifies all pages contained in File1.pdf and File2.pdf. File 1 has 6 pages, file 2 has an unknown number of pages.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool> >
    <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>
  </ResourcePool>
  <ResourceLinkPool>
    <RunListLink Usage="Input" rRef="L3"/>
  </ResourceLinkPool>
</JDF>
```

Table 7-251: Example (1) of Ord Attribute in PlacedObject Elements

Ord	File	Page	Ord	File	Page
0	File1	0	1	File1	1
2	File1	2	3	File1	3
4	File1	4	5	File1	5
6	File2	0	7	File2	1
8	File2	2	(n)	File2	(n - 6)

Example 7-28: RunList: Simple Multi-File Separated RunList

This example specifies two pages contained in Presep.pdf and following that, pages 1, 3 and 5 of each preprepared file.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool> >
    <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>
    </ResourcePool>
  </JDF>
```

```

</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>
</ResourcePool>
<ResourceLinkPool>
  <RunListLink Usage="Input" rRef="Link0003" />
</ResourceLinkPool>
</JDF>

```

Table 7-252: 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

7.2.109.12.5 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.

Example 7-29: OrdExpression

Saddle stitched booklet for variable page length documents.

The following describes the OrdExpressions for a booklet with varying page lengths. The example page assignments are for a book of 13-16 pages.

```

Front:
OrdExpression = i2*s 0 2 4 6
OrdExpression = i4*((n+3)/4) n(s*2)-11513119
Back:
OrdExpression = i2*s+1 1 3 5 7

```

OrdExpression = i4*((n+3)/4) n(s*2)-21412108

Example 7-30: DocOrd Usage

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
```

7.2.109.12.5.1 Structure of Signature Subelement (Deprecated)

[Deprecated in JDF 1.3](#)

The table defining the deprecated *Signature Subelement* has been moved to "Layout Deprecated Subelement" on page 1037. All Attributes that were defined in *Signature* have been moved into **Layout**.

7.2.110 LayoutElement

This Resource is needed for **LayoutElementProduction**. It describes some text, an image, one or more pages or anything else that is used in the production of the layout of a product.

Resource Properties

Resource Class: Parameter

Resource referenced by: **LayoutElement/Dependencies**, **LayoutElementProductionParams/**
LayoutElementPart, **RunList**

Example Partition: *PageNumber*

Input of Processes: *DBDocTemplateLayout*, *DBTemplateMerging*,
LayoutElementProduction, *ShapeDefProduction*

Output of Processes: *DBDocTemplateLayout*, *LayoutElementProduction*

Table 7-253: LayoutElement Resource (Section 1 of 3)

Name	Data Type	Description
<i>ClipPath</i> ? 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.
<i>ContentDataRefs</i> ? New in JDF 1.4	IDREFS	IDs of ContentData Elements in the referenced ContentList . ContentData Elements provide Metadata related to the product to be published. <i>ContentDataRefs</i> MUST NOT be specified if no ContentList is specified.
<i>ElementType</i> ? Modified in JDF 1.3	enumeration	Describes the content type for this LayoutElement . Values are from: Table 7-254, "ElementType Attribute Values" on page 614

Table 7-253: LayoutElement Resource (Section 2 of 3)

Name	Data Type	Description
<i>HasBleeds</i> ? 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.
<i>IgnorePDLCopies</i> = " <i>false</i> " New in JDF 1.1	boolean	If " <i>true</i> ", any PDL defined copy count MUST be ignored.
<i>IgnorePDLImposition</i> = " <i>true</i> " New in JDF 1.1	boolean	If " <i>true</i> ", any PDL defined imposition definition MUST be ignored. Examples are PDF with embedded PJTF or PPML with a PRINT_LAYOUT. If <i>IgnorePDLImposition</i> = " <i>false</i> " 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 " <i>true</i> ", 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> = " <i>false</i> " specifies a blank page.
<i>IsPrintable</i> ? Modified in JDF 1.2	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, 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 " <i>true</i> ", 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>SourceMediaBox</i> ? New in JDF 1.4	rectangle	The MediaBox of the LayoutElement .

Table 7-253: LayoutElement Resource (Section 3 of 3)

Name	Data Type	Description
SourceTrimBox ? Modified in JDF 1.2	rectangle	A rectangle that describes the intended trimmed size of the element to be included. This rectangle is expressed in the source coordinate system of the object. If not specified, the set of values of PageList/PageData/@SourceTrimBox selected by <i>PageListIndex</i> is applied.
Template ? 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.
ContentList ? New in JDF 1.4	refelement	ContentList with additional metadata. Constraint: at most one of ContentList and PageList MUST be specified.
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 . Constraint: at most one of ContentList and PageList MUST be specified.
ScreeningParams ? New in JDF 1.2	refelement	Specification of the screening properties. If not specified, the value of PageList/PageData/ScreeningParams is applied.
SeparationSpec * Modified in JDF 1.2	element	List of used separation names. If not specified, the value of PageList/PageData/SeparationSpec is applied.

— Attribute: ElementType

Table 7-254: ElementType Attribute 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).
Barcode New in JDF 1.3	A barcode.
<i>Composed</i>	Combination of elements that define an element that is not bound to a document page.

Table 7-254: ElementType Attribute Values (Section 2 of 2)

Value	Description
<i>Document</i>	An ordered set of one or more pages.
<i>Graphic</i>	Line art.
<i>IdentificationField</i> New in JDF 1.3	A general identification field excluding bar codes.
<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, document breaks and sub document breaks (e.g. PPML, PPML/VDX, ISO 16612-2 PDF/VT) Modification note: starting with JDF 1.4, 3 kinds of breaks are added.
<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.
<i>Unknown</i> Deprecated in JDF 1.2	

7.2.110.1 Element: Dependencies

[New in JDF 1.2](#)

This Element provides a container for dependent references of the **LayoutElement**.

Table 7-255: Dependencies Element

Name	Data Type	Description
LayoutElement *	refelement	Description of dependent elements, (e.g., fonts, images, etc.).

7.2.111 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 has been added in JDF 1.4.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>LayoutElementProduction</i>
Output of Processes:	—

Table 7-256: LayoutElementProductionParams Resource

Name	Data Type	Description
ActionPool ? New in JDF 1.4	element	A pool of Action Elements that describe the restrictions that are applied to the created output
LayoutElementPart *	element	Description of the specific parameters for generating a LayoutElement.
ShapeDef ? New in JDF 1.4	refelement	A resource describing the shape of the LayoutElement to be produced.
TestPool ? New in JDF 1.4	element	Container for zero or more Test elements that are referenced from Action Elements in the ActionPool. TestPool MUST be supplied if ActionPool is present.

Example 7-31: LayoutElementProductionParams: Page Shape

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Page Shape Sample
      Date: Aug 2, 2007 Version: 2
      A page with a certain size -->
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n001"
      Type="LayoutElementProduction"
      Status="Waiting" DescriptiveName="Page sample for shape"
      JobPartID="ID34" Version="1.4">
  <ResourcePool>
    <LayoutElementProductionParams Class="Parameter" ID="LEPParams"
      Status="Available" />
    <LayoutElement Class="Parameter" ID="LayElOut" Status="Unavailable"
      SourceMediaBox="0 0 595.27 822.05"
      SourceTrimBox="28.34 28.34 566.93 793.71"/>
  </ResourcePool>
  <ResourceLinkPool>
    <LayoutElementProductionParamsLink rRef="LEPParams" Usage="Input" />
    <LayoutElementLink rRef="LayElOut" Usage="Output" />
  </ResourceLinkPool>
  <AuditPool>
    <Created Author="XYZ Corporation" TimeStamp="2006-01-09T09:00:00+01:00"/>
  </AuditPool>
</JDF>
```

Example 7-32: LayoutElementProductionParams: Label Shape

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Shape Sample for a label with a cut line
      Date: Jan 9, 2005 Version: 1.00
      A page with a certain size -->
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n001"
      Type="LayoutElementProduction"
      Status="Waiting" DescriptiveName="Page sample for shape"
      JobPartID="ID400" Version="1.4">
  <ResourcePool>
    <LayoutElementProductionParams Class="Parameter" ID="LEPParams"
      Status="Available" >
      <ShapeDef>
        <Shape ShapeType="Path" DDESCutType="101" CutPath="..." />
      </ShapeDef>
    </LayoutElementProductionParams>
    <LayoutElement Class="Parameter" ID="LayElOut" Status="Unavailable"
      SourceMediaBox="0 0 595.27 822.05"
      SourceTrimBox="28.34 28.34 566.93 793.71"/>
  </ResourcePool>
```

```

<ResourceLinkPool>
  <LayoutElementProductionParamsLink rRef="LEPParams" Usage="Input" />
  <LayoutElementLink rRef="LayElOut" Usage="Output" />
</ResourceLinkPool>
<AuditPool>
  <Created Author="ABC-Corporation" TimeStamp="2006-01-09T09:00:00+01:00" />
</AuditPool>
</JDF>

```

Example 7-33: LayoutElementProductionParams: Box Shape

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Shape Sample for a box defined by a CAD file
      Date: Jan 9, 2005 Version: 1.00
      A page with a certain size -->
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n001"
      Type="LayoutElementProduction"
      Status="Waiting" JobPartID="ID100"
      DescriptiveName="Page sample for shape" Version="1.4">
  <ResourcePool>
    <LayoutElementProductionParams Class="Parameter" ID="LEPParams"
      Status="Available">
      <ShapeDef>
        <FileSpec URL="file://myserver/myshare/olive.dd3" />
      </ShapeDef>
    </LayoutElementProductionParams>
    <LayoutElement Class="Parameter" ID="LayElOut" Status="Unavailable" />
  </ResourcePool>
  <ResourceLinkPool>
    <LayoutElementProductionParamsLink rRef="LEPParams" Usage="Input" />
    <LayoutElementLink rRef="LayElOut" Usage="Output" />
  </ResourceLinkPool>
  <AuditPool>
    <Created Author="ZYX Corporation" TimeStamp="2006-01-09T09:00:00+01:00" />
  </AuditPool>
</JDF>

```

7.2.111.1 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 7-257: LayoutElementPart Element

Name	Data Type	Description
<i>ID</i> ? New in JDF 1.4	ID	ID of the LayoutElementPart.
BarcodeProductionParams ?	element	Description of the specific parameters for barcode production.
LayoutElement ? New in JDF 1.4	refelement	Specification of an existing LayoutElement that is used to initially populate this LayoutElementPart. Any LayoutElement Resources that are specified here MUST also be specified as Input Resources to the <i>LayoutElementProduction</i> Process.
PositionObj ? New in JDF 1.4	element	Definition of the size and position of this LayoutElementPart

7.2.111.2 Element: BarcodeProductionParams

BarcodeProductionParams describes of the specific parameters for barcode production.

Table 7-258: 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.

7.2.111.3 Element: PositionObj[New in JDF 1.4](#)

PositionObj describes the size and position of the LayoutElementPart.

Table 7-259: PositionObj Element

Name	Data Type	Description
<i>Anchor ?</i>	Anchor	<i>Anchor</i> specifies the origin (0,0) of the coordinate system in the unrotated LayoutElementPart.
<i>CTM ?</i>	matrix	Transformation matrix of the origin of LayoutElementPart as specified by @Anchor. Not that this is not necessarily the actual CTM that will position a given LayoutElementPart. The actual CTM MUST be recalculated based on the values of <i>Anchor</i> and <i>Size</i> .
<i>PageRange ?</i>	Integer-RangeList	Reader Page index in the PageList.
<i>PositionPolicy ?</i>	enumeration	Specifies the level of freedom when applying the values specified in PositionObj. Values are: <i>Exact</i> – The values MUST be followed precisely. <i>Free</i> – The values are used as guidance and MAY be modified by the designer.
<i>RelativeSize ?</i>	XYPair	Specifies the size of the unrotated and unscaled object, relative to the parent specified in RefAnchor.
<i>RotationPolicy ?</i>	enumeration	Specifies the level of freedom when applying the values specified in PositionObj. Values are: <i>Exact</i> – The values MUST be followed precisely. <i>Free</i> – The values are used as guidance and MAY be modified by the designer.
<i>Size ?</i>	XYPair	Specifies the size of the unrotated and unscaled object, in points.
<i>SizePolicy ?</i>	enumeration	Specifies the level of freedom when applying the values specified in PositionObj. Values are: <i>Exact</i> – The values MUST be followed precisely. <i>Free</i> – The values are used as guidance and MAY be modified by the designer.
RefAnchor ?	element	Reference to a LayoutElementPart that this LayoutElementPart is positioned relative to. If RefAnchor is not specified, PositionObj refers to the lower left of the first page specified in page Range.

Example 7-34: LayoutElementProductionParams: PositionObj

```

<?xml version="1.0" encoding="UTF-8"?>
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n000002"
  JobPartID="n000002" Status="Waiting" Type="LayoutElementProduction"
  Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="LayoutElementProduction">
  <!--Generated by the CIP4 Java open source JDF Library version :
    CIP4 JDF Writer Java 1.3 BLD 46-->
  <AuditPool>
    <Created AgentName="CIP4 JDF Writer Java" AgentVersion="1.3 BLD 46"
      Author="CIP4 JDF Writer Java 1.3 BLD 46" ID="a000003"
      TimeStamp="2007-09-05T18:20:31+02:00"/>
  </AuditPool>
  <ResourcePool>
    <RunList Class="Parameter" ID="r000004" Status="Unavailable">
      <LayoutElement Class="Parameter">
        <FileSpec Class="Parameter" MimeType="application/pdf" URL="output.pdf"/>
      </LayoutElement>
    </RunList>
    <LayoutElementProductionParams Class="Parameter" ID="r000005"
      Status="Unavailable">
      <!--This is a "well placed" CTM defined mark
        The anchor defines the 0,0 point to be transformed
        The element to be placed is referenced by LayoutElement/FileSpec/URL
      -->
      <LayoutElementPart>
        <PositionObj Anchor="BottomLeft" CTM="1 0 0 1 0 0" PageRange="0"
          PositionPolicy="Exact">
          <RefAnchor Anchor="BottomLeft" AnchorType="Parent"/>
        </PositionObj>
        <LayoutElement Class="Parameter">
          <FileSpec Class="Parameter" MimeType="application/pdf"
            URL="bkg.pdf"/>
        </LayoutElement>
      </LayoutElementPart>
      <!--This is a "roughly placed" reservation in the middle of the page-->
      <LayoutElementPart ID="l000006">
        <PositionObj Anchor="CenterCenter" PageRange="0" PositionPolicy="Free">
          <RefAnchor Anchor="CenterCenter" AnchorType="Parent"/>
        </PositionObj>
        <LayoutElement Class="Parameter" ElementType="Image">
          <Comment ID="c000007">
            Please add an image of a palm tree on a beach here!
          </Comment>
        </LayoutElement>
      </LayoutElementPart>
      <!--This is a "roughly placed" reservation 36 points below the previous
        image; NextPosition points from Anchor on this to NextAnchor on next,
        i.e. a positive vector specifies that next is shifted in the positive
        direction in the parent (in this case page) coordinate system
      -->
      <LayoutElementPart>
        <PositionObj Anchor="TopCenter" CTM="1 0 0 1 0 36"
          PageRange="0" PositionPolicy="Free">
          <RefAnchor Anchor="BottomCenter" AnchorType="Sibling"
            rRef="l000006"/>
        </PositionObj>
        <LayoutElement Class="Parameter" ElementType="Image">
          <Comment ID="c000008">
            Please add an image of a beach ball below the palm tree!
          </Comment>
        </LayoutElement>
      </LayoutElementPart>

```

```

<!--This is a "well placed" CTM defined mark. The anchor defines the
      0,0 point used as the RefAnchor for the element to be transformed
-->
<LayoutElementPart>
  <PositionObj Anchor="LowLeft" CTM="1 0 0 1 2 3" PageRange="0"
    PositionPolicy="Exact">
    <RefAnchor Anchor="BottomLeft" AnchorType="Parent"/>
  </PositionObj>
  <BarcodeProductionParams>
    <!--barcode details here-->
  </BarcodeProductionParams>
</LayoutElementPart>
<LayoutElementPart>
  <PositionObj Anchor="TopRight" PageRange="0" PositionPolicy="Exact">
    <RefAnchor Anchor="TopRight" AnchorType="Parent"/>
    <!--This is a "roughly placed" mark.
      The anchor at top right is placed at the right (=1.0) top(=1.0)
      position of the page. No rotation is specified
-->
  </PositionObj>
  <BarcodeProductionParams>
    <!--barcode details here-->
  </BarcodeProductionParams>
</LayoutElementPart>
<!--This is a "roughly placed" container for marks
      The anchor at top left is defined in the !Unrotated! orientation.
      It is placed at the left (=0.0) bottom(=0.0) position of the page.
      The text flows bottom to top (=Rotate 90 = counterclockwise)
      do we need margins?
-->
<LayoutElementPart ID="1000009">
  <PositionObj Anchor="TopLeft" CTM="0 1 -1 0 0 0"
    PageRange="1" PositionPolicy="Free">
    <RefAnchor Anchor="BottomCenter" AnchorType="Parent"/>
  </PositionObj>
</LayoutElementPart>
<!--This is a barcode inside the previous container
      The anchor at bottom left is defined in the !Unrotated! orientation.
      It is placed at the left (=0.0) bottom(=0.0) position of the container.
-->
<LayoutElementPart ID="1000010">
  <PositionObj Anchor="BottomLeft" CTM="1 0 0 1 0 0">
    <RefAnchor Anchor="BottomLeft" AnchorType="Parent" rRef="1000009"/>
  </PositionObj>
  <BarcodeProductionParams>
    <!--barcode details here-->
  </BarcodeProductionParams>
</LayoutElementPart>
<!--This is a disclaimer text inside the previous container
      The anchor at top left is defined in the !Unrotated! orientation.
      The barcode and text are justified with their top margins and spaced
      by 72 points which corresponds to the left of the page because the
      container is rotated 90° AbsoluteSize specifies the size of the
      object in points
-->
<LayoutElementPart>
  <PositionObj AbsoluteSize="300 200" Anchor="TopLeft" CTM="1 0 0 1 -72 0">
    <RefAnchor Anchor="TopRight" AnchorType="Sibling" rRef="1000010"/>
  </PositionObj>
  <LayoutElement Class="Parameter" ElementType="Text">
    <FileSpec Class="Parameter"
      URL="file://myServer/disclaimers/de/aspirin.txt"/>
  </LayoutElement>
</LayoutElementPart>

```

```

<!--This is a "VERY roughly placed" piece of text somewhere on pages 2-3
  RelativeSize specifies the size of the object as a ratio of the size
  of the container
-->
<LayoutElementPart>
  <PositionObj PageRange="1 ~ 2" RelativeSize="0.8 0.5"/>
  <LayoutElement Class="Parameter" ElementType="Text">
    <Comment ID="c000011" Name="Instructions">
      Please add some text about
      the image of a palm tree on a beach here!
    </Comment>
  </LayoutElement>
</LayoutElementPart>
<!--This is another "VERY roughly placed" piece of text somewhere on
  pages 2-3; the text source is the JDF-->
<LayoutElementPart>
  <PositionObj PageRange="1 ~ 2"/>
  <LayoutElement Class="Parameter" ElementType="Text">
    <Comment ID="c000012" Name="TextInput">
      Laurum Ipsum Blah blah blah!
      btw. this is unformatted plain text and nothing else!
    </Comment>
  </LayoutElement>
</LayoutElementPart>
</LayoutElementProductionParams>
</ResourcePool>
<ResourceLinkPool>
  <RunListLink Usage="Output" rRef="r000004"/>
  <LayoutElementProductionParamsLink Usage="Input" rRef="r000005"/>
</ResourceLinkPool>
</JDF>

```

Example 7-35: LayoutElementProductionParams: Preflight

```

<?xml version="1.0" encoding="UTF-8"?>
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n000002"
  JobPartID="n000002" Status="Completed" Type="LayoutElementProduction"
  Version="1.4" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="LayoutElementProduction">
  <!--Generated by the CIP4 Java open source JDF Library version : CIP4 JDF
    Writer Java 1.3 BLD 47-->
  <AuditPool>
    <Created AgentName="CIP4 JDF Writer Java" AgentVersion="1.3 BLD 47"
      Author="CIP4 JDF Writer Java 1.3 BLD 47" ID="a000003"
      TimeStamp="2007-10-11T20:23:18+02:00"/>
    <PhaseTime AgentName="CIP4 JDF Writer Java"
      AgentVersion="1.3 BLD 47" Author="CIP4 JDF Writer Java 1.3 BLD 47"
      End="2007-10-11T20:23:23+02:00" ID="a000020"
      Start="2007-10-11T20:23:21+02:00" Status="InProgress"
      StatusDetails="Creative Work" TimeStamp="2007-10-11T20:23:21+02:00"/>
    <ProcessRun AgentName="CIP4 JDF Writer Java"
      AgentVersion="1.3 BLD 47" Author="CIP4 JDF Writer Java 1.3 BLD 47"
      Duration="PT2S" End="2007-10-11T20:23:23+02:00"
      EndStatus="Completed" ID="a000024"
      Start="2007-10-11T20:23:21+02:00" TimeStamp="2007-10-11T20:23:23+02:00"/>
  </AuditPool>
  <ResourcePool>
    <RunList Class="Parameter" ID="r000004" Status="Unavailable">
      <LayoutElement Class="Parameter">
        <FileSpec Class="Parameter" MimeType="application/pdf" URL="output.pdf"/>
      </LayoutElement>
    </RunList>
    <LayoutElementProductionParams Class="Parameter" ID="r000005"
      Status="Unavailable">

```

```

<Comment ID="c000006" Name="Instruction">
  Add any human readable instructions here
</Comment>
<ActionPool>
  <Action DescriptiveName="set number of pages to 4" ID="A000007"
    Severity="Error" TestRef="T000008"/>
  <Action
    DescriptiveName="set number of separations to 6 on page 0 and 3"
    ID="A000009" Severity="Error" TestRef="T000010">
    <PreflightAction SetRef="T000011"/>
  </Action>
  <Action
    DescriptiveName="separation to black only on page 1 and 2"
    ID="A000012" Severity="Error" TestRef="T000013">
    <PreflightAction SetRef="T000014"/>
  </Action>
  <Action DescriptiveName="set TrimBox to 8.5*11 Method 2"
    ID="A000015" Severity="Error" TestRef="T000016">
    <PreflightAction SetRef="T000017"/>
  </Action>
  <Action
    DescriptiveName="Warn when effective resolution<300 dpi"
    ID="A000018" Severity="Warning" TestRef="T000019"/>
</ActionPool>
<TestPool>
  <Test ID="T000008">
    <not>
      <IntegerEvaluation ValueList="4">
        <BasicPreflightTest Name="NumberOfPages"/>
      </IntegerEvaluation>
    </not>
  </Test>
  <Test ID="T000010">
    <not>
      <StringEvaluation>
        <BasicPreflightTest ListType="UniqueList" MaxOccurs="6"
          MinOccurs="6" Name="SeparationList"/>
      </StringEvaluation>
    </not>
  </Test>
  <Test ID="T000011">
    <IntegerEvaluation ValueList="0 3">
      <BasicPreflightTest Name="PageNumber"/>
    </IntegerEvaluation>
  </Test>
  <Test ID="T000013">
    <not>
      <StringEvaluation>
        <BasicPreflightTest Name="SeparationList"/>
        <Value Value="Black"/>
      </StringEvaluation>
    </not>
  </Test>
  <Test ID="T000014">
    <IntegerEvaluation ValueList="1 ~ 2">
      <BasicPreflightTest Name="PageNumber"/>
    </IntegerEvaluation>
  </Test>
  <Test ID="T000016">
    <not>
      <RectangleEvaluation ValueList="0 0 612 792">
        <BasicPreflightTest Name="PageBoxSize"/>
      </RectangleEvaluation>
    </not>
  </Test>

```



```

</Test>
<Test ID="T000017">
  <EnumerationEvaluation ValueList="TrimBox">
    <BasicPreflightTest Name="PageBoxName"/>
  </EnumerationEvaluation>
</Test>
<Test ID="T000019">
  <XYPairEvaluation ValueList="0 0 ~ 300 300">
    <BasicPreflightTest Name="EffectiveResolution"/>
  </XYPairEvaluation>
</Test>
</TestPool>
</LayoutElementProductionParams>
</ResourcePool>
<ResourceLinkPool>
  <RunListLink Usage="Output" rRef="r000004"/>
  <LayoutElementProductionParamsLink Usage="Input" rRef="r000005"/>
</ResourceLinkPool>
</JDF>

```

7.2.112 LayoutPreparationParams

[New in JDF 1.1](#)

Warning. *LayoutPreparationParams* MAY be deprecated in a future version.

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:	<i>LayoutPreparation</i>
Output of Processes:	—

Table 7-260: 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. Values are: <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.
<i>FinishingOrder</i> = "GatherFold"	enumeration	Specifies the order of operations for finishing a bound booklet created from multiple imposed Sheets. The <i>LayoutPreparation</i> 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. Values are: <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. <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. <i>Gather</i> – The Sheets of a document are gathered on a pile. No folding is assumed. <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.

Table 7-260: LayoutPreparationParams Resource (Section 2 of 8)

Name	Data Type	Description
<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 Figure 7-31, “Fold catalog part 1” on page 542 and Figure 7-32, “Fold catalog part 2” on page 543.</p> <p>Value format is: “<i>F_n-i</i>” where “n” is the number of finished pages and “i” is either an integer, which identifies a particular fold or the letter “X”, which identifies a generic fold. E.g., “<i>F6-2</i>” describes a Z-fold of 6 finished pages, and “<i>F6-X</i>.” describes a generic fold with 6 finished pages.</p> <p>The <i>LayoutPreparation</i> 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> = “<i>Saddle</i>” is assumed.</p>
<i>FoldCatalogOrientation</i> = “ <i>Rotate0</i> ” New in JDF 1.3	Orientation	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).
<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.</p> <p>Values include those from: Table 7-261, “FrontMarkList Attribute Values” on page 631</p>
<i>Gutter?</i> Modified in JDF 1.2	XYPair	<p>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 <i>PageCell/@TrimSize</i> 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 <i>NumberUp</i> Attribute is not specified or its explicit values are equal to one, this Attribute MUST be ignored.</p> <p>In the case where a gutter is identified as creeping by either <i>VerticalCreep</i> or <i>HorizontalCreep</i>, then the values of <i>Gutter</i> specify the initial width of explicitly creeping gutters where the gutter width may increment or decrement depending on the <i>CreepValue</i> Attribute. If a value of <i>CreepValue</i> is negative then <i>Gutter</i> 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.</p> <p><i>Gutter</i> is applied in addition to any <i>Border</i> specified in the <i>PageCell</i>.</p>
<i>GutterMinimumLimit?</i> 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.

Table 7-260: LayoutPreparationParams Resource (Section 3 of 8)

Name	Data Type	Description
<i>HorizontalCreep</i> ? Modified in JDF 1.2	IntegerList	<p>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 <i>NumberUp</i>. The value for an entry in this list MUST be between zero and two (2) less then the second value of <i>NumberUp</i>.</p> <p>If not specified, then horizontal gutters MUST NOT creep.</p> <p>Gutters identified by this Attribute are known as explicitly creeping gutters whereas those not identified are known as implicitly creeping gutters.</p> <p>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.</p>
<i>ImplicitGutter</i> ? 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 <i>CreepValue</i> (see <i>Gutter</i>).
<i>ImplicitGutterMinimumLimit</i> ? 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.
<i>NumberUp</i> ?	XYPair	<p>Specifies a regular, multi-up grid of <i>PageCell</i> 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 <i>RunList</i> 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>

Table 7-260: LayoutPreparationParams Resource (Section 4 of 8)

Name	Data Type	Description
<i>PageDistributionScheme</i> = "Sequential"	NMTOKEN	<p>Specifies how finished pages are to be distributed onto a multi-up grid of finished <i>PageCell</i> Elements defined by the values of the <i>NumberUp</i> Attribute.</p> <p>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>
<i>PageOrder</i> = "Reader"	NMTOKEN	<p>The assumed ordering of the finished pages in the RunList.</p> <p>Values include:</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>

Table 7-260: LayoutPreparationParams Resource (Section 5 of 8)

Name	Data Type	Description															
<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.</p> <p>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. 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> <ul style="list-style-type: none"> • X – Specifies flowing left to right across a Sheet surface. • x – Specifies flowing right to left across a Sheet surface. • Y – Specifies flowing bottom to top vertically across a Sheet surface. • y – Specifies flowing top to bottom vertically across a Sheet surface. • Z – Specifies flowing bottom of stack to top of it through the stack. • z – Specifies flowing top of stack to bottom of it through the stack. <p>Examples: 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 1396 1422 1505"> <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 7-260: 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.</p> <p>Values are:</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>Note: For details of orthogonal rotations, refer to Table 2-4, “Matrices and Orientation values for describing the orientation of a Component” on page 30. If a <i>RotatePolicy</i> value other than "NoRotate" is specified in <i>FitPolicy</i>, the actual rotation specified in <i>Rotate</i> MAY 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 <i>RunList</i> 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.</p> <p>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 “<i>WorkAndTumble</i>” 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 “<i>WorkAndTurn</i>” 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 “<i>WorkAndTumble</i>”.</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 “<i>WorkAndTurn</i>”.</p>
<i>StackDepth</i> ?	integer	<p>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.</p>

Table 7-260: LayoutPreparationParams Resource (Section 7 of 8)

Name	Data Type	Description																							
StepDocs? Modified in JDF 1.2	XYPair	<p>A list of two integers that species how to impose multiple Instance Documents on one Sheet. The first value specifies the document repeats along the X axis, the second value specifies the repeats along the Y axis. Each entry of <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 1" 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 predicable placement of content. If this Attribute is not supplied, a rectangle with the origin at "0 0" and an extent that MAY be dependent on the dimensions of the Media is implied.</p>																							

Table 7-260: 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 than 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.
<i>DeviceMark ?</i>	refelement	Details how Device-dependent marks are to be generated. If not specified, the marks are Device-dependent.
<i>ExternalImpositionTemplate ?</i> 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 .
<i>FitPolicy ?</i>	refelement	Details how to fit the grid of PageCell Elements onto the <i>SurfaceContentsBox</i> .
<i>ImageShift ?</i>	element	Details how to place the grid of PageCell Elements into the <i>SurfaceContentsBox</i> . <i>ImageShift</i> MUST be applied before any transformations of the grid of PageCell Elements as specified by <i>Rotate</i> or <i>FitPolicy</i> . 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.
<i>InsertSheet *</i>	refelement	Additional Sheets to be inserted before, after or within a Job.
<i>JobField *</i>	refelement	Specific information about this kind of mark object.
<i>Media ?</i>	refelement	Specific information about the media.
<i>PageCell ?</i> Modified in JDF 1.1A	element	PageCell Elements describe how page contents will be imaged onto individual page cells. At most one PageCell MUST be specified and it is applied to all page cells on both surfaces of a Sheet.

— Attribute: FrontMarkList

Table 7-261: FrontMarkList Attribute Values (Section 1 of 2)

Value	Description	Value	Description
<i>CIELABMeasuringField</i>		<i>IdentificationField</i>	
<i>ColorControlStrip</i>		<i>JobField</i>	

Table 7-261: FrontMarkList Attribute Values (Section 2 of 2)

Value	Description	Value	Description
<i>ColorRegisterMark</i>		<i>PaperPathRegisterMark</i>	
<i>CutMark</i>		<i>RegisterMark</i>	
<i>DensityMeasuringField</i>		<i>ScavengerArea</i>	

7.2.112.1 Element: PageCell

Table 7-262: 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.</p> <p>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 7-262: 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. Values are: <i>Rotate0</i> <i>Rotate90</i> – 90° counterclockwise rotation. <i>Rotate180</i> – 180° rotation. <i>Rotate270</i> – 90° clockwise rotation. Note: for details of orthogonal rotation, refer to Table 2-4, “Matrices and Orientation values for describing the orientation of a Component” on page 30. If a <i>RotatePolicy</i> value other than "NoRotate" is specified in <i>FitPolicy</i> , the actual rotation specified in <i>Rotate</i> MAY be modified accordingly.
<i>TrimSize</i> ? 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 <i>LayoutPreparationParams/@SurfaceContentsBox</i> and dividing by the appropriate <i>NumberUp</i> value.
<i>Color</i> ?	refelement	Color of the border.
<i>DeviceMark</i> ?	refelement	Details how Device dependent marks are to be generated. Defaults to the value of <i>DeviceMark</i> in the parent <i>LayoutPreparationParams</i> .
<i>FitPolicy</i> ?	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.
<i>ImageShift</i> ?	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 <i>PageCell</i> .

7.2.112.2 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 7-263: 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 7-263: 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.

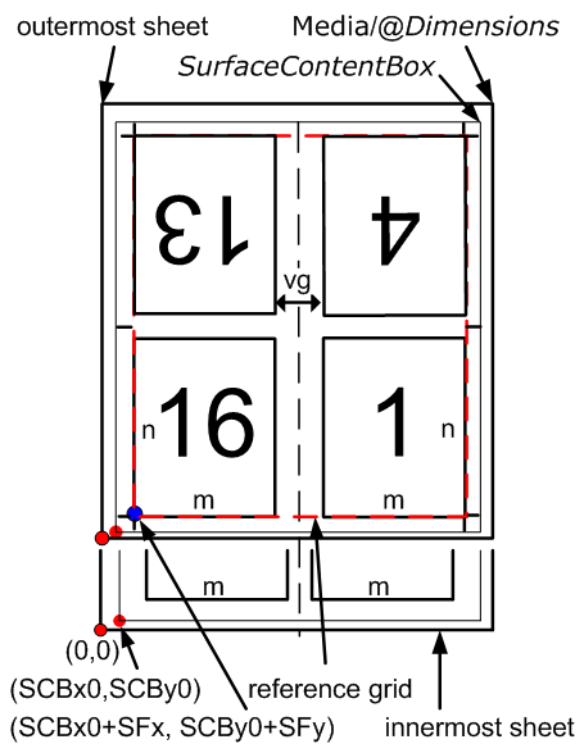


Figure 7-39: Diagram of a 4-up cross-folded saddle-stitch imposition with vertical gutter creep

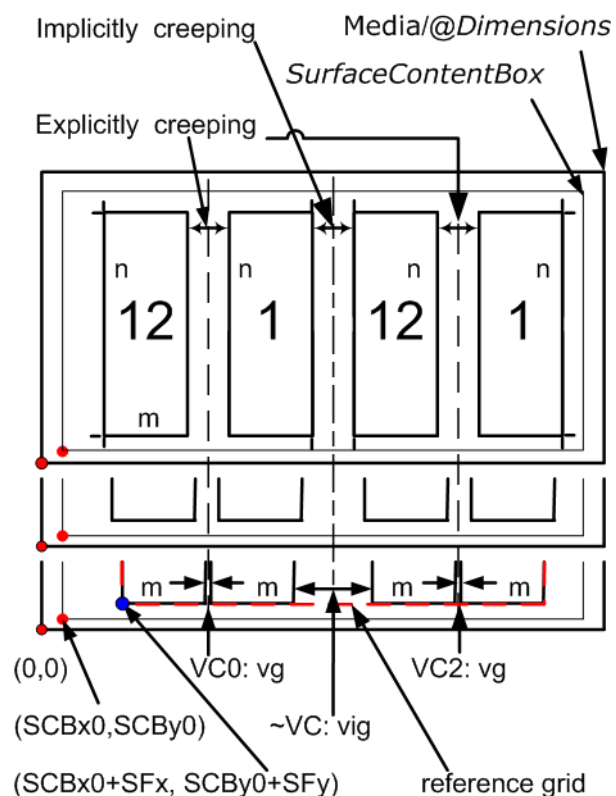


Figure 7-40: Diagram of a step-and-repeat 2-up saddle-stitch imposition with vertical spine gutter creep

Description for Figure 7-39

The following terms are used in Figure 7-39

- **reference grid** in Figure 7-39 refers to the dashed red box around page cells of outermost Sheet, which indicates the size of the reference grid used in calculating grid placement relative to the *SurfaceContentsBox* origin using *LayoutPreparationParams/@ImageShift*
- **SCBx0, SCBy0, SFx, SFy, m, n** and **vg** are used in the JDF below.

Figure 7-39 illustrates the JDF below. The JDF assumes that the dimensions of the **RunList** page's trim rectangle matches *PageCell/@TrimSize*, whose dimensions are m by n (width and height) in the JDF example below. The Sheet with the widest creep gutter is on the top of the logical Sheet stack.

Example 7-36: LayoutPreparationParams: JDF for Figure 7-39

```
<JDF ID="ID" Type="ProcessGroup" xmlns="http://www.CIP4.org/JDFSchema_1_1"
  Status="Waiting" JobPartID="ID34" Version="1.4">
  <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="0 0 612 792" BindingEdge="Left"
      VerticalCreep="0" GutterMinimumLimit="5 5" CreepValue="0 -5"
      Gutter="20 20" FinishingOrder="FoldCollect" FrontMarkList="CutMark">
      <!-- Note: the value of some attributes in LayoutPreparationParams and
        subElements relate to symbols in the above Figure:
          SurfaceContentsBox="SCBx0 SCBy0 SCBx1 SCBy1"
          GutterMinimumLimit="hml vml"
          CreepValue="0 -vc"
          Gutter="hg vg"
          TrimSize="m n"
          ShiftFront="SFx SFy"
        -->
      <PageCell TrimSize="612 792">
        <ImageShift PositionX="Spine" PositionY="Center" />
      </PageCell>
      <ImageShift PositionY="Bottom" PositionX="Left" ShiftFront="20 20"/>
    </LayoutPreparationParams>
  </ResourcePool>
  <ResourceLinkPool>
    <LayoutPreparationParamsLink Usage="Input" rRef="LPP_2" />
  </ResourceLinkPool>
</JDF>
```

Description for Figure 7-40

The following terms are used in Figure 7-40

- **reference grid** in Figure 7-40 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 **vg** are used in the JDF below.

Figure 7-40 illustrates the JDF below. The JDF assumes that the dimensions of source content page rectangle matches *PageCell/@TrimSize*, whose dimensions are m by n (width and height) in the JDF example below.

Example 7-37: LayoutPreparationParams: JDF for Figure 7-40

```
<JDF ID="ID" Type="ProcessGroup" xmlns="http://www.CIP4.org/JDFSchema_1_1"
  Status="Waiting" JobPartID="ID34" Version="1.4">
  <ResourcePool>
    <LayoutPreparationParams Class="Parameter" ID="LPP_1" Status="Available"
      NumberUp="4 1" PageDistributionScheme="Saddle" FoldCatalog="F4-1"
      FoldCatalogOrientation="Flip0" Sides="TwoSidedFlipY" StepRepeat="2 1 1"
    >
```

```

SurfaceContentsBox="0 0 612 792" VerticalCreep="0 2"
ImplicitGutter="0 30" ImplicitGutterMinimumLimit="0 20" CreepValue="0 5"
Gutter="0 10" FinishingOrder="GatherFold" FrontMarkList="CutMark">
<!--Note: folding pattern F4-1 applies to each of the two 2x1
signatures
Note: step and repeat by two in X direction logically divides grid
into two 2x1 signatures
Note: first (VC0) and third (VC2) vertical gutters are explicitly
creeping and the rest (~VC) are implicitly creeping
Note: Positive vertical creep value indicates initial gutter
Widths of inner most Sheet
Note: cut marks are located relative to largest page cell grid
trim box
Note: the value of some attributes in LayoutPreparationParams and
subElements relate to symbols in the above Figure:
SurfaceContentsBox="SCBx0 SCBx1 SCBy0 SCBy1"
ImplicitGutter="0 vig"
ImplicitGutterMinimumLimit="0 vigl"
CreepValue="0 +vc"
Gutter="0 vg"
TrimSize="m n"
ShiftFront="SFx SFy"
-->
<PageCell TrimSize="612 792">
  <ImageShift PositionX="Spine" PositionY="Bottom"/>
</PageCell>
  <ImageShift PositionY="Bottom" PositionX="Left" ShiftFront="20 20"/>
</LayoutPreparationParams>
</ResourcePool>
<ResourceLinkPool>
  <LayoutPreparationParamsLink Usage="Input" rRef="LPP_1"/>
</ResourceLinkPool>
</JDF>

```

7.2.113 LayoutShift

[New in JDF 1.4.](#)

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>LayoutShifting</i>
Output of Processes:	—

Table 7-264: LayoutShift Resource

Name	Data Type	Description
ShiftPoint	element	Description of separation dependent transformations for a given point on the Layout.

7.2.113.1 Element: ShiftPoint

Table 7-265: ShiftPoint Element (Section 1 of 2)

Name	Data Type	Description
<i>CTM</i>	matrix	<i>CTM</i> that MUST be applied to the Separation after all other transformations.

Table 7-265: ShiftPoint Element (Section 2 of 2)

Name	Data Type	Description
<i>Position</i>	XYPair	Point that this ShiftPoint applies to. Note: the interpolation algorithm between ShiftPoint positions is implementation dependent.

Example 7-38: LayoutShift[New in JDF 1.4.](#)Example of absolute positions with *@Position*

```
<JDF ID="ID" Type="ProcessGroup" xmlns="http://www.CIP4.org/JDFSchema_1_1"
  Status="Waiting" JobPartID="ID34" Version="1.4">
  <ResourcePool>
    <!--LayoutShift SHOULD be partitioned: at least Side and Separation
      will make sense -->
    <LayoutShift ID="r000005" Class="Parameter" Status="Unavailable"
      PartIDKeys="Side Separation" >
      <!--LayoutShift SHOULD be partitioned: at least Side and Separation
        will make sense-->
      <!--Note that the interpolation algorithm between positions is
        implementation dependent-->
      <LayoutShift Side="Front">
        <LayoutShift Separation="Cyan">
          <ShiftPoint CTM="1 0 0 1 0 0" Position="360 500"/>
          <ShiftPoint CTM="1 0 0 1 0 2" Position="1800 500"/>
          <ShiftPoint CTM="1 0 0 1 1 0" Position="360 1500"/>
          <ShiftPoint CTM="1 0 0 1 1 2" Position="1800 1500"/>
          <ShiftPoint CTM="1 0 0 1 2 0" Position="360 2500"/>
          <ShiftPoint CTM="1 0 0 1 2 2" Position="1800 2500"/>
          <ShiftPoint CTM="1 0 0 1 3 0" Position="360 3500"/>
          <ShiftPoint CTM="1 0 0 1 3 2" Position="1800 3500"/>
        </LayoutShift>
        <LayoutShift Separation="Magenta">
          <ShiftPoint CTM="1 0 0 1 1 1" Position="360 500"/>
          <ShiftPoint CTM="1 0 0 1 1 3" Position="1800 500"/>
          <ShiftPoint CTM="1 0 0 1 2 1" Position="360 1500"/>
          <ShiftPoint CTM="1 0 0 1 2 3" Position="1800 1500"/>
          <ShiftPoint CTM="1 0 0 1 3 1" Position="360 2500"/>
          <ShiftPoint CTM="1 0 0 1 3 3" Position="1800 2500"/>
          <ShiftPoint CTM="1 0 0 1 4 1" Position="360 3500"/>
          <ShiftPoint CTM="1 0 0 1 4 3" Position="1800 3500"/>
        </LayoutShift>
        <LayoutShift Separation="Yellow">
          <ShiftPoint CTM="1 0 0 1 2 2" Position="360 500"/>
          <ShiftPoint CTM="1 0 0 1 2 4" Position="1800 500"/>
          <ShiftPoint CTM="1 0 0 1 3 2" Position="360 1500"/>
          <ShiftPoint CTM="1 0 0 1 3 4" Position="1800 1500"/>
          <ShiftPoint CTM="1 0 0 1 4 2" Position="360 2500"/>
          <ShiftPoint CTM="1 0 0 1 4 4" Position="1800 2500"/>
          <ShiftPoint CTM="1 0 0 1 5 2" Position="360 3500"/>
          <ShiftPoint CTM="1 0 0 1 5 4" Position="1800 3500"/>
        </LayoutShift>
        <LayoutShift Separation="Black">
          <ShiftPoint CTM="1 0 0 1 3 3" Position="360 500"/>
          <ShiftPoint CTM="1 0 0 1 3 5" Position="1800 500"/>
          <ShiftPoint CTM="1 0 0 1 4 3" Position="360 1500"/>
          <ShiftPoint CTM="1 0 0 1 4 5" Position="1800 1500"/>
          <ShiftPoint CTM="1 0 0 1 5 3" Position="360 2500"/>
          <ShiftPoint CTM="1 0 0 1 5 5" Position="1800 2500"/>
          <ShiftPoint CTM="1 0 0 1 6 3" Position="360 3500"/>
          <ShiftPoint CTM="1 0 0 1 6 5" Position="1800 3500"/>
        </LayoutShift>
      </LayoutShift>
    </ResourcePool>
  </JDF>
```

```

    </LayoutShift>
  </LayoutShift>
</ResourcePool>
<ResourceLinkPool>
  <LayoutShiftLink Usage="Input" rRef="r000005" />
</ResourceLinkPool>
</JDF>

```

7.2.114 LongitudinalRibbonOperationParams

[Deprecated in JDF 1.1.](#)

See "LongitudinalRibbonOperationParams" on page 1037 for details of this deprecated Resource.

7.2.115 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:	<i>ManualLabor</i>
Output of Processes:	—

Table 7-266: ManualLaborParams Resource

Name	Data Type	Description
<i>LaborType</i> Modified in JDF 1.4	NMTOKEN	Type of manual labor that is performed. Values include: <i>CreateCoatingForm</i> – create a form to apply coatings during or after printing <i>EditArt</i> – Unspecific Art editing (for work on specific files <i>LayoutElementProduction</i> is to be used) <i>EditMarks</i> – Marks editing <i>EditTraps</i> – Traps editing <i>ManageJob</i> – General work on the Job. <i>PhoneCallToCustomer</i> – Phone calls to ask/inform the Customer. <i>SeparateBlanks</i> – Manual separation of blanks from a sheet after die cutting. New in JDF 1.4 Modification note: starting with JDF 1.3, the data type is changed from the erroneous NMTOKENS.

7.2.116 Media

This Resource describes a physical element that represents a raw, unexposed printable surface such as Sheet, film or plate. *Gloss*, *MediaColorName* and *Opacity* Attributes provide media characteristics pertinent to color management.

Resource Properties

Resource Class:	Consumable
Resource referenced by:	Color/PrintConditionColor , Component , DieLayout , DigitalPrintingParams , EmbossingParams/Emboss , ExposedMedia , FeedingParams/Feeder , FeedingParams/CollatingItem ,

ImageSetterParams, InterpretingParams, Layout, LayoutPreparationParams, Media/MediaLayers, RasterReadingParams, ShapeDef, StrippingParams, Tile
Location, SheetName, Side, SignatureName, TileID, WebName

Example Partition:
Input of Processes: *Bending, BoxPacking, Bundling, CaseMaking, ConventionalPrinting, ContactCopying, Cutting, DigitalPrinting, Embossing, Feeding, ImageSetting, Laminating, Varnishing, Wrapping*

Output of Processes: *Cutting, Feeding*

Table 7-267: Media Resource (Section 1 of 8)

Name	Data Type	Description
<i>BackCoatingDetail?</i> New in JDF 1.4	NMTOKEN	Identical to <i>FrontCoatingDetail</i> (see below), but applied to the back surface of the media. Default value is from: @ <i>FrontCoatingDetail</i> . Values are from: @ <i>FrontCoatingDetail</i>
<i>BackCoatings?</i>	enumeration	Identical to <i>FrontCoatings</i> (see below), but applied to the back surface of the media. Default value is from: @ <i>FrontCoatings</i> . Values are from: @ <i>FrontCoatings</i>
<i>BackGlossValue?</i> New in JDF 1.2	double	Gloss of the back surface of the media in gloss units as defined by [ISO2470:1999]. When not known, <i>BackGlossValue</i> defaults to the value of <i>FrontGlossValue</i> .
<i>Brightness?</i>	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, <i>Brightness</i> 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> .
<i>CIEtint?</i> New in JDF 1.2	double	Average CIE tint value. Average CIE tint is calculated according to equations given in [TAPPI T560].
<i>CIEWhiteness?</i> New in JDF 1.2	double	Average CIE whiteness value. Average CIE whiteness is calculated according to equations given in [TAPPI T560].
<i>ColorName?</i> 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 Color defined in a ColorPool Resource that is linked to the Process using this Media Resource. Deprecation note: starting with JDF 1.2, use <i>MediaColorName</i> and <i>MediaColorNameDetails</i> .
<i>CoreWeight?</i> New in JDF 1.3	double	Weight of the core of a Roll, in grams [g]

Table 7-267: Media Resource (Section 2 of 8)

Name	Data Type	Description
<p><i>Dimension</i> ? Modified in JDF 1.4</p>	XYPair	<p>The X and Y dimensions of the chosen medium, measured in points. <i>Dimension</i> specifies the outer bounding box of the Media. 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).</p> <p>Values include those from: Table G-1, "Media Sizes" on page 899 New in JDF 1.4</p> <p>Modification note: starting with JDF 1.4, the description states that <i>Dimension</i> specifies the outer bounding box of the Media and new values are specified.</p>
<p><i>Flute</i> ? New in JDF 1.3</p>	NMTOKEN	<p>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.</p> <p>Values include:</p> <p>A B C</p>
<p><i>FluteDirection</i> ? New in JDF 1.3</p>	enumeration	<p>Direction of the fluting.</p> <p>Values are:</p> <p><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</p>
<p><i>FrontCoatingDetail</i> ? New in JDF 1.4</p>	NMTOKEN	<p>Describes (beyond <i>FrontCoatings</i>) the coating to the front surface of the media and possibly the technology used to apply the coating.</p> <p>Values include:</p> <p><i>Cast</i></p>

Table 7-267: Media Resource (Section 3 of 8)

Name	Data Type	Description
FrontCoatings? Modified in JDF 1.4	enumeration	<p>What preprocess coating has been applied to the front surface of the media.</p> <p>Values are:</p> <p><i>None</i> – No coating.</p> <p><i>Coated</i> – A coating of a system-specified type. New in JDF 1.2</p> <p><i>Glossy</i></p> <p><i>HighGloss</i></p> <p><i>InkJet</i> – A coating intended for use with inkjet technology. Deprecation note: use <i>PrintingTechnology</i> = "InkJet" New in JDF 1.2 Deprecated in JDF 1.4</p> <p><i>Matte</i></p> <p><i>Polymer</i> – Coating for a photo polymer process New in JDF 1.3</p> <p><i>Silver</i> – Coating for a silver halide process New in JDF 1.3</p> <p><i>Satin</i></p> <p><i>Semigloss</i></p>
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	<p>The <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".</p> <p><i>Grade</i> of paper material is defined in accordance with the paper "types" set forth in [ISO12647-2:2004].</p> <p>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).</p> <p>The values define offset printing paper types</p> <p>Values are:</p> <p>1 – Gloss-coated paper.</p> <p>2 – Matt-coated paper.</p> <p>3 – Gloss-coated, Web paper.</p> <p>4 – Uncoated, white paper.</p> <p>5 – Uncoated, yellowish paper.</p>
GrainDirection? New in JDF 1.1 Modified in JDF 1.3	enumeration	<p>Direction of the grain in the coordinate system defined by <i>Dimension</i>.</p> <p>Values are:</p> <p><i>LongEdge</i> – Along the longer axis as defined by <i>Dimension</i>.</p> <p><i>ShortEdge</i> – Along the shorter axis as defined by <i>Dimension</i>.</p> <p><i>XDirection</i> – Along the X-axis of the Media coordinate system. New in JDF 1.3</p> <p><i>YDirection</i> – Along the Y-axis of the Media coordinate system. New in JDF 1.3</p>
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> , Hole or HoleLine , which includes the number of holes.

Table 7-267: Media Resource (Section 4 of 8)

Name	Data Type	Description
<i>HoleType</i> = "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 929. Values are: <i>None</i> – No holes. <i>Explicit</i> – Holes are defined in a HoleList . Values are from: "JDF/CIP4 Hole Pattern Catalog" on page 929
<i>ImagableSide</i> ?	enumeration	Side of the chosen medium that are to be marked. Values are: <i>Front</i> <i>Back</i> <i>Both</i> <i>Neither</i>
<i>InnerCoreDiameter</i> ? New in JDF 1.4	double	Specifies the inner diameter of the core of a Roll, in points. See also <i>OuterCoreDiameter</i> and <i>RollDiameter</i> .
<i>InsideLoss</i> ? 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.
<i>LabColorValue</i> ? New in JDF 1.2	LabColor	<i>LabColorValue</i> is the CIELAB color value of the media, computed as specified in [TAPPI T527].
<i>MediaColorName</i> ? Modified in JDF 1.1	NamedColor	A name for the color. Allowed values are defined in Section A.3.3.3, NamedColor. If more specific, specialized or site-defined media color names are needed, use <i>MediaColorNameDetails</i> .
<i>MediaColorNameDetails</i> ? 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> SHOULD also be supplied.
<i>MediaQuality</i> ? New in JDF 1.4	string	Named quality description of the media. For folding carton quality, multiple named quality description systems are in use. E.g. GC1, SBB, etc. For an overview see http://www.procarton.com/files/fact_file_6.pdf
<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>MediaTypeDetails</i> is "PreCutTabs", a <i>MediaSetCount</i> of "5" would indicate that each set includes five tab Sheets.

Table 7-267: Media Resource (Section 5 of 8)

Name	Data Type	Description
<p><i>MediaType?</i> Modified in JDF 1.4</p>	enumeration	<p>Describes the medium being employed.</p> <p>Values are:</p> <p><i>CorrugatedBoard</i> New in JDF 1.3</p> <p><i>Disc</i> – CD or DVD disc to be printed on.</p> <p><i>EndBoard</i> – end board used in the <i>Bundling</i> Process.</p> <p><i>EmbossingFoil</i></p> <p><i>Film</i></p> <p><i>Foil</i></p> <p><i>GravureCylinder</i> – gravure cylinder. New in JDF 1.3</p> <p><i>ImagingCylinder</i> – reusable direct imaging cylinder in a press. New in JDF 1.3</p> <p><i>LaminatingFoil</i></p> <p><i>MountingTape</i> – for flexo plate mounting tape. New in JDF 1.4</p> <p><i>Other</i> – not one of the defined values.</p> <p><i>Paper</i></p> <p><i>Plate</i></p> <p><i>Screen</i> – used for Screen Printing. New in JDF 1.4</p> <p><i>SelfAdhesive</i> – New in JDF 1.3</p> <p><i>Sleeve</i> – for flexo sleeves New in JDF 1.4</p> <p><i>ShrinkFoil</i></p> <p><i>Transparency</i></p> <p><i>Unknown</i> – Deprecated in JDF 1.2</p>
<p><i>MediaTypeDetails?</i></p>	NMTOKEN	<p>Additional details of the chosen medium.</p> <p>Constraint: If <i>MediaTypeDetails</i> is specified, <i>MediaType</i> MUST be specified.</p> <p>Values include those from: Table 7-268, “MediaTypeDetails Attribute Values” on page 646</p>
<p><i>MediaUnit = "Sheet"</i> Modified in JDF 1.2</p>	enumeration	<p>Describes the format of the media as it is delivered to the Device.</p> <p>Values are:</p> <p><i>Continuous</i> – Continuously connected Sheets which can be fan folded. New in JDF 1.2</p> <p><i>Roll</i></p> <p><i>Sheet</i> – Individual cut Sheets.</p>
<p><i>Opacity?</i> Modified in JDF 1.2</p>	enumeration	<p>The opacity of the media. See <i>OpacityLevel</i> to specify the degree of opacity for any of these values.</p> <p>Values are:</p> <p><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.</p> <p><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</p> <p><i>Transparent</i> – The media is transparent.</p>

Table 7-267: Media Resource (Section 6 of 8)

Name	Data Type	Description
<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. See also <i>InnerCoreDiameter</i> and <i>RollDiameter</i> .
<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 Modified in JDF 1.4	enumeration	Exposure technology of the plates. Values are: <i>FlexoAnalogSolvent</i> New in JDF 1.4 <i>FlexoAnalogThermal</i> New in JDF 1.4 <i>FlexoDigitalSolvent</i> New in JDF 1.4 <i>FlexoDigitalThermal</i> New in JDF 1.4 <i>FlexoDirectEngraving</i> New in JDF 1.4 <i>InkJet</i> – Exposure with inkjet technology. Note that <i>FrontCoatings</i> = " <i>Inkjet</i> " 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. Values are: <i>Positive</i> <i>Negative</i>
<i>PrePrinted</i> = " <i>false</i> "	boolean	Indicates whether the media is preprinted.
<i>PrintingTechnology?</i> New in JDF 1.4	NMTOKEN	Describes the printing technology that the media or coatings on the media are intended for or optimized for. Values include: <i>DyeSublimation</i> <i>Electrostatic</i> <i>InkJet</i> <i>Laser</i> <i>Offset</i> <i>Thermal</i>
<i>Recycled?</i> Deprecated in JDF 1.2	boolean	If " <i>true</i> ", 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>ReliefThickness?</i> New in JDF 1.4	double	The thickness of the relief, measured in microns [µm]. The floor thickness can be calculated as (<i>Thickness</i> - <i>ReliefThickness</i>). See Figure 7-42.

Table 7-267: Media Resource (Section 7 of 8)

Name	Data Type	Description
<i>RollDiameter?</i>	double	Specifies diameter of a Roll, in points. See also <i>InnerCoreDiameter</i> and <i>OuterCoreDiameter</i> .
<i>ShrinkIndex?</i> 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.
<i>SleeveInterlock?</i> New in JDF 1.4	NMTOKEN	The type of interlock (or notch) to use for a flexo sleeve. Values include: <i>Type01</i> – see Figure 7-43. ... <i>Type20</i> – see Figure 7-43.
<i>StockType?</i> New in JDF 1.1 Modified in JDF 1.4	NMTOKEN	Strings describing the available stock. <i>StockType</i> defines the base size when calculating North American or Japanese paper weights. See “North American and Japanese Media Weight Explained” on page 895. New in JDF 1.4 Values with Kanji names support Japanese media. Values include: <i>Bond</i> <i>Bristol</i> <i>Cover</i> <i>Index</i> <i>Newsprint</i> <i>Offset</i> – This includes book stock. <i>Tag</i> <i>Text</i> <i>Aatoposutoshi</i> – アートポスト紙 ("art-post paper") is cover stock coated on one side. New in JDF 1.4 <i>Aatoshi</i> – アート紙 ("art paper") is machine coated paper, available in top quality and medium quality (Joushitsu and Chuushitsu). New in JDF 1.4 <i>Chuushitsushi</i> – 中質紙 ("medium-quality paper") contains a minimum of 70% chemical pulp. New in JDF 1.4 <i>Joushitsushi</i> – 上質紙 ("top-quality paper") contains 100% chemical pulp. New in JDF 1.4 <i>Mashinkootoshi</i> – マシンコート紙 ("machine coated paper"), also called Kootoshi (コート紙), is machine coated paper given only a thin coat of clay. New in JDF 1.4

Table 7-267: Media Resource (Section 8 of 8)

Name	Data Type	Description
<i>Texture ?</i> New in JDF 1.1 Modified in JDF 1.2	NMTOKEN	The intended texture of the media. Values 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.
<i>Thickness ?</i>	double	The thickness of the chosen medium, measured in microns [μm]. Note: Thickness is often referred to as caliper.
<i>UserMediaType ?</i> Deprecated in JDF 1.1	NMTOKEN	A human-readable description of the type of media. The value can be used by an operator to select the correct media to load. The semantics of the values will be site-specific. Deprecation note: starting with JDF 1.1, <i>UserMediaType</i> has been merged into <i>MediaTypeDetails</i> .
<i>Weight ?</i>	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 895 for details on converting North American paper weights to g/m ² .
<i>WrapperWeight ?</i> New in JDF 1.3	double	Weight of the wrapper of a Roll, in grams [g]
<i>Color ?</i> Deprecated in JDF 1.1	refelement	A Color Resource that provides the color of the chosen medium.
<i>ColorMeasurementConditions ?</i> New in JDF 1.2	refelement	Detailed description of the measurement conditions for color measurements used to measure <i>LabColorValue</i> .
<i>HoleList ?</i> New in JDF 1.3	refelement	Explicit list of holes. HoleList MUST be specified if <i>HoleType</i> = " <i>Explicit</i> ".
<i>MediaLayers ?</i> New in JDF 1.3	element	Subelement describing the layer structure of media such as corrugated or self adhesive materials.
<i>TabDimensions ?</i> New in JDF 1.4	element	Specifies the dimensions of the tabs when <i>MediaTypeDetails</i> = " <i>TabStock</i> ", " <i>PreCutTabs</i> " or " <i>FullCutTabs</i> ". Note: see BindingIntent/Tabs (Table 7-31, "Tabs Element" on page 363) (rather than MediaIntent) for how tabbed media is specified in Product Intent.

— Attribute: **MediaTypeDetails**

Table 7-268: MediaTypeDetails Attribute Values (Section 1 of 3)

Value	Description
<i>Aluminum</i> Modified in JDF 1.3	Conventional or CtP press plate.
<i>Cardboard</i>	

Table 7-268: MediaTypeDetails Attribute Values (Section 2 of 3)

Value	Description
<i>CD</i> New in JDF 1.3	CD disc to be printed on.
<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> Deprecated in JDF 1.3	Visible light CtP plate with photo polymer process.
<i>CtPVisibleSilver</i> Deprecated in JDF 1.3	Visible light CtP plate with silver halide process.
<i>CtPThermal</i> Deprecated in JDF 1.3	Thermal CtP plate.
<i>DoubleWall</i> New in JDF 1.3	Double wall corrugated board
<i>DVD</i> New in JDF 1.3	DVD disc to be printed on.
<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>FlexoBase</i> New in JDF 1.4	For the base layer of flexo plates.
<i>FlexoPhotoPolymer</i> New in JDF 1.4	For the photopolymer layer of flexo plates.
<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> Deprecated in JDF 1.3	Press plate for the UV process.
<i>Polyester</i> Modified in JDF 1.3	Conventional or CtP press plate.
<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.

Table 7-268: MediaTypeDetails Attribute Values (Section 3 of 3)

Value	Description
<i>SingleFace</i> New in JDF 1.3	Single face corrugated board.
<i>SingleWall</i> New in JDF 1.3	Single wall corrugated board.
<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> New in JDF 1.3	Triple wall corrugated board
<i>WetFilm</i>	Conventional photographic film.

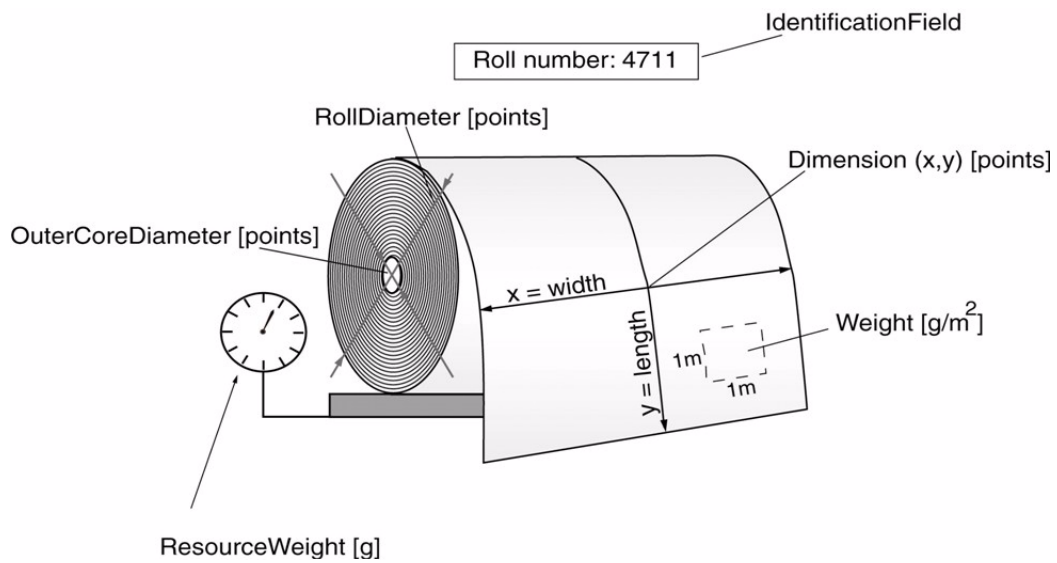


Figure 7-41: a paper Roll with some roll-specific information

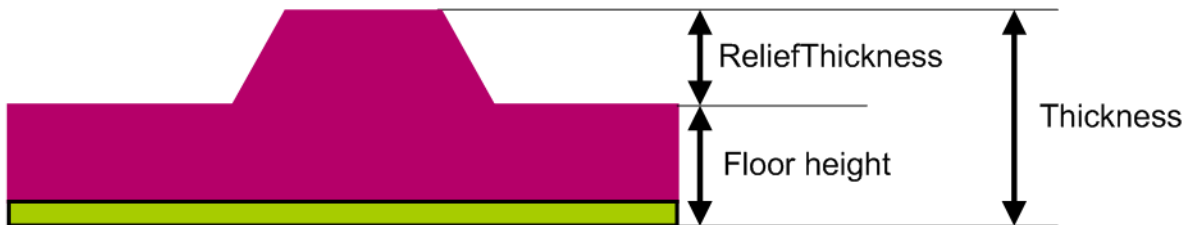


Figure 7-42: Relief and Floor Thickness for a Flexo Plate or Flexo Sleeve

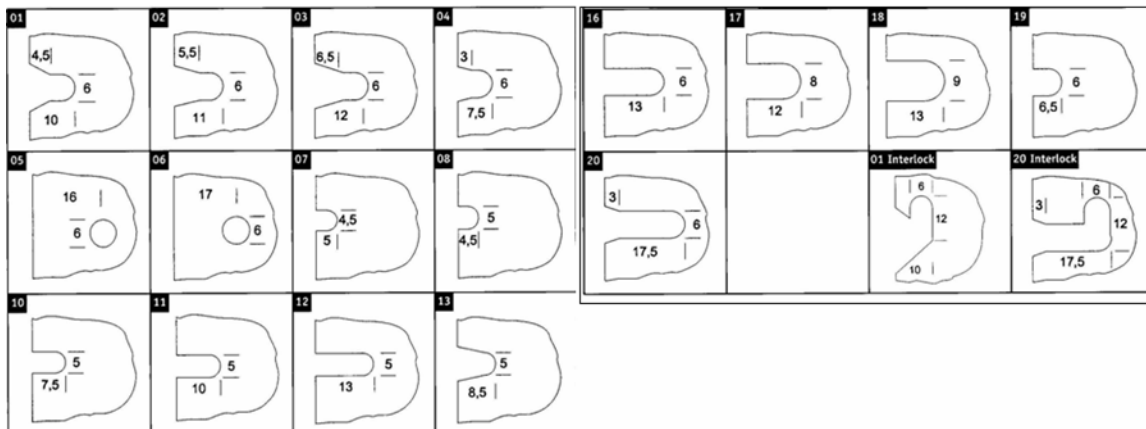


Figure 7-43: Types of Interlocks for Flexo Sleeve

7.2.116.1 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 7-269: MediaLayers Element

Name	Data Type	Description
GlueLine *	refelement	GlueLine Resource describing a glue layer of a layered Media Resource. Each GlueLine Resource MUST have <code>GlueLine/@AreaGlue = "true"</code> .
Media *	refelement	Media Resources describing a layer of a layered Media Resources.

7.2.116.2 Element: TabDimensions

[New in JDF 1.4](#)

Specifies the size and placement of tabs in a bank and in a set of tab stock.

Table 7-270: TabDimensions Element (Section 1 of 2)

Name	Data Type	Description
TabEdge?	enumeration	Indicates which edge of the media has tabs. Sets the coordinate system for <code>TabOffset</code> , <code>TabExtensionDistance</code> , and <code>TabWidth</code> . Values are: <i>Left</i> <i>Top</i> <i>Right</i> <i>Bottom</i>

Table 7-270: TabDimensions Element (Section 2 of 2)

Name	Data Type	Description
<i>TabExtensionDistance?</i>	double	The positive distance in points that the tab extends beyond the body of the other media. Note: same as BindingIntent/Tabs/@TabExtensionDistance . Note: This value is always included in the value of the overall extent of the Media defined by Media/@Dimension . See Figure 7-45, “Diagram of a Single Bank of Tabs” on page 652.
<i>TabOffset?</i>	double	Specifies the magnitude of the distance in points from the two corners to the edge of the first “tab pitch” point of the first tab in the bank along the <i>TabEdge</i> . This distance is the same on both ends of the bank of tabs. See Figure 7-45, “Diagram of a Single Bank of Tabs” on page 652.
<i>TabsPerBank?</i>	integer	Specifies the number of equal-sized tabs in a single bank if all positions were filled. Note: banks can have tabs only in some of the possible positions. Note: same as BindingIntent/Tabs/@TabsPerBank . Media/@MediaSetCount specifies the number of tabs per set. A set can consist of one or more banks. If Media/@MediaSetCount is not an even multiple of <i>TabsPerBank</i> , the last bank in each set is partially filled.
<i>TabSetCollationOrder?</i>	NMTOKEN	Collation order of media provided in sets. Applicable to sets of pre-cut tabs. See Figure 7-44, “TabSetCollationOrder Attribute Values” on page 651. Values include: <i>Forward</i> – first tab is towards top of stack <i>Reverse</i> – first tab is toward bottom of stack.
<i>TabWidth?</i>	double	The width along the <i>TabEdge</i> of each tab as measured along the mid-line of the tab. Each tab is centered within a space called the “tab pitch”. See Figure 7-45, “Diagram of a Single Bank of Tabs” on page 652.

TabSetCollationOrder = "Forward"

TabSetCollationOrder = "Reverse"

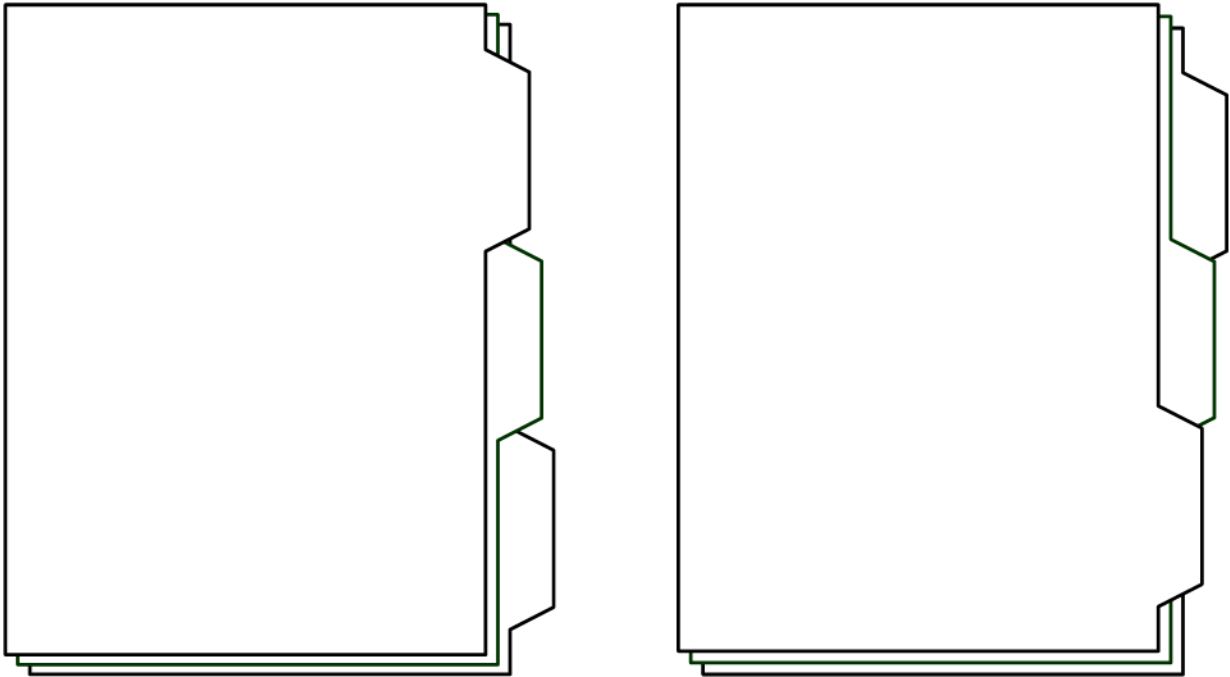


Figure 7-44: TabSetCollationOrder Attribute Values

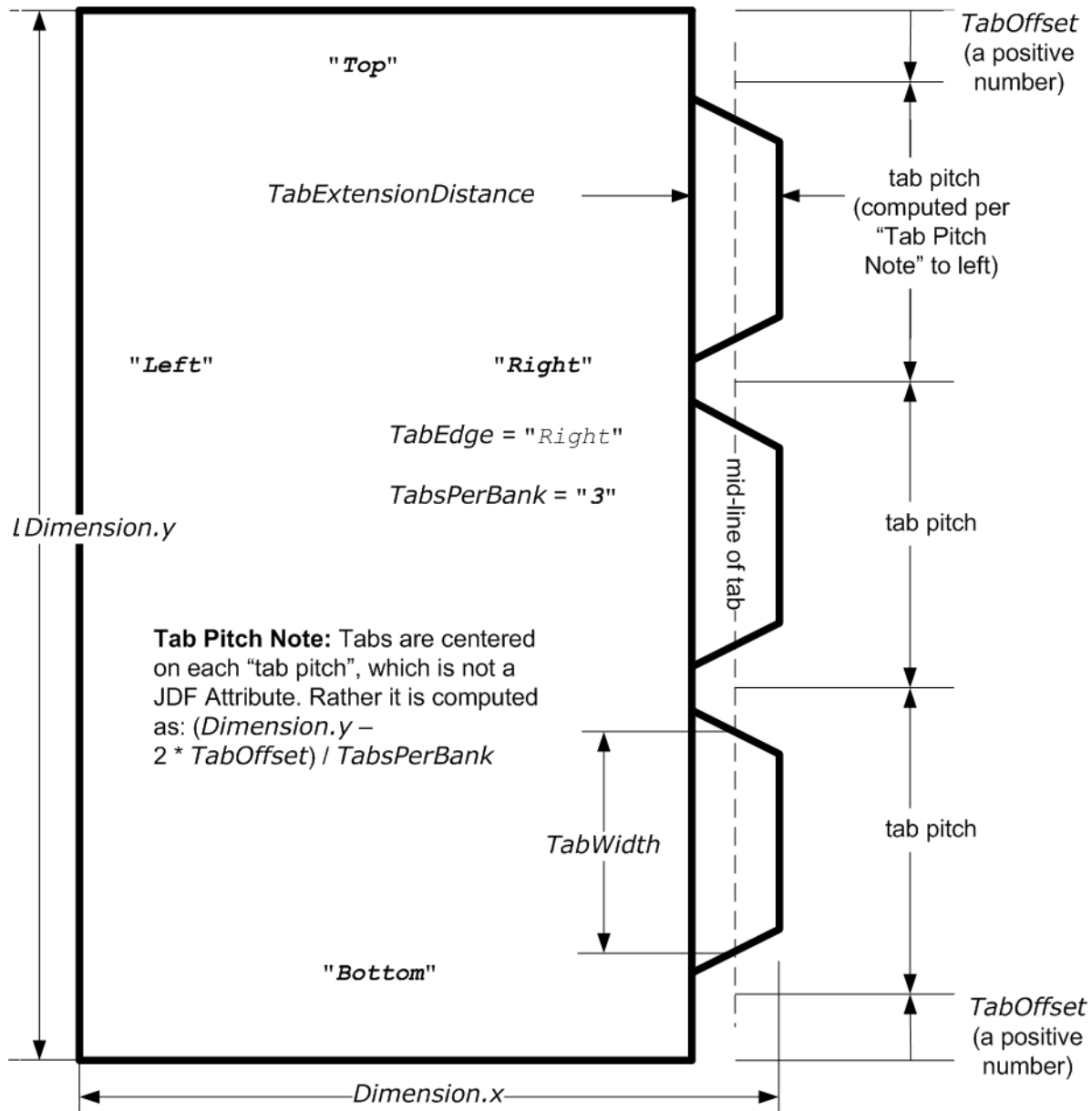


Figure 7-45: Diagram of a Single Bank of Tabs

7.2.116.3 More about Media

7.2.116.3.1 Inside Loss and Outside Gain

Inside loss and outside gain: dimensional values used in the mechanical design phase of a box. (Note: IL + OG is not exactly equal to thickness. Thickness is most often referred to as caliper.)

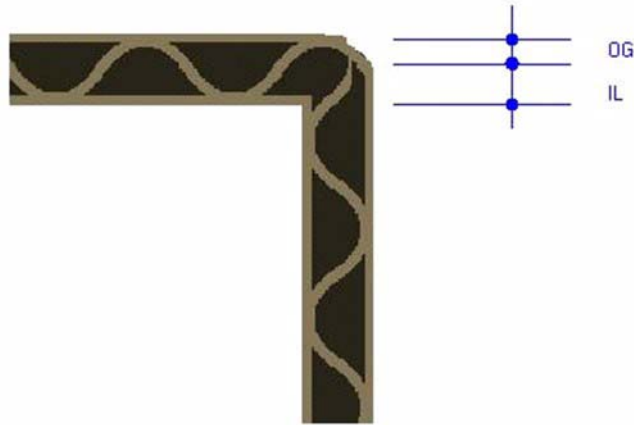


Figure 7-46: Inside Loss, Outside Gain

7.2.116.3.2 Corrugated Media:

Corrugated material consists of multiple Sheets of paper (called liners) with fluted material in between. For background information on Corrugated Media, see <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>

Example 7-39: Media: Corrugated

```
<JDF ID="ID" Type="ProcessGroup" xmlns="http://www.CIP4.org/JDFSchema_1_1"
  Status="Waiting" JobPartID="ID34" Version="1.4">
  <ResourcePool>
    <Media Class="Consumable" ID="M123456" ProductID="B190Y180D1050x120"
      Status="Available" DescriptiveName="B Flute 190Y 180D 1050x1210"
      Dimension="1050.0 120.0" MediaType="CorrugatedBoard"
      MediaTypeDetails="SingleWall" MediaUnit="Sheet" Thickness="2382.0"
      InsideLoss="1000.0" OutsideGain="1380.0" Weight="600">
      <MediaLayers>
        <!-- FrontLiner -->
        <Media DescriptiveName="190gsm clay coated" MediaType="Paper"
          Weight="190" FrontCoatings="Coated"/>
        <!-- Flute -->
        <Media DescriptiveName="Flute" MediaType="Paper" Weight="180"
          FluteDirection="ShortEdge" Flute="B" MediaTypeDetails="Flute"/>
        <!-- BackLiner -->
        <Media DescriptiveName="180gsm white top" MediaType="Paper"
          Weight="180"/>
      </MediaLayers>
    </Media>
  </ResourcePool>
  <ResourceLinkPool>
    <MediaLink Usage="Input" rRef="M123456"/>
  </ResourceLinkPool>
</JDF>
```

7.2.116.3.3 Self adhesive Media

Self adhesive media is described as **MediaLayers** Elements with nested **Media** and **GlueLine** Elements.

Example 7-40: Media: Self Adhesive

```
<JDF ID="ID" Type="ProcessGroup" xmlns="http://www.CIP4.org/JDFSchema_1_1"
  Status="Waiting" JobPartID="ID34" Version="1.4">
  <ResourcePool>
    <Media Class="Consumable" ID="M123456" ProductID="7890123" Status="Available"
      DescriptiveName="40# Fasson coated label stock" Dimension="1134.0 0"
      MediaType="SelfAdhesive" MediaUnit="Roll" Thickness="1000.0"
      Weight="150">
      <MediaLayers>
        <!-- Front -->
        <Media DescriptiveName="Antique Cream Smooth WS IL" MediaType="Paper"
          Weight="90"/>
        <!-- Glue -->
        <GlueLine DescriptiveName="Permanent 91A" AreaGlue="true"
          GlueType="Hotmelt" GlueBrand="Uhu"/>
        <!-- Back -->
        <Media DescriptiveName="Blue Glassine 50" MediaType="Paper" Weight="50"/>
      </MediaLayers>
    </Media>
  </ResourcePool>
  <ResourceLinkPool>
    <MediaLink Usage="Input" rRef="M123456"/>
  </ResourceLinkPool>
</JDF>
```

7.2.116.3.4 Flexo Plate Media

A sample of a flexo plate with dimensions of 900 mm x 1200 mm, a base of 177 microns and a total thickness of 1143 microns.

A raw plate can contain several separations from multiple jobs. The real printing dimensions can only be determined when all elements of the mounting process are known: circumference of the sleeve on which the flat plate will be mounted, thickness of the mounting tape, thickness of base and thickness of the photopolymer.

Example 7-41: Media: Flat Plate

```
<JDF ID="ID" Type="ProcessGroup" xmlns="http://www.CIP4.org/JDFSchema_1_1"
  Status="Waiting" JobPartID="ID34" Version="1.4">
  <ResourcePool>
    <Media Class="Consumable" ID="M123456" ProductID="FlexoPlate"
      Status="Available" DescriptiveName="" Dimension="2551.181 3401.574"
      MediaType="Plate" PlateTechnology="FlexoDigitalThermal"
      Manufacturer="PlateManufacturerA" Brand="BrandB" BatchID="Batch 12345"
      Thickness="1143" ReliefThickness="500">
      <!--MediaLayers contains 2 items: the base layer and the
        photopolymer layer of the flexo plate
      -->
      <MediaLayers>
        <Media DescriptiveName="Base" MediaType="Plate"
          MediaTypeDetails="FlexoPlateBase" Thickness="177"/>
        <Media DescriptiveName="Photopolymer Layer" MediaType="Plate"
          MediaTypeDetails="FlexoPlatePhotopolymer" Thickness="966"/>
      </MediaLayers>
    </Media>
  </ResourcePool>
  <ResourceLinkPool>
    <MediaLink Usage="Input" rRef="M123456"/>
  </ResourceLinkPool>
</JDF>
```

 </JDF>

7.2.116.3.5 Flexo Sleeve Media

The flexo sleeve has dimensions of 500 x 250 mm, a base of 1249 microns and a total thickness of 2810 microns. The sleeve dimensions are identical to printing dimensions (no distortion).

Example 7-42: Media: Flexo Sleeve

```
<JDF ID="ID" Type="ProcessGroup" xmlns="http://www.CIP4.org/JDFSchema_1_1"
  Status="Waiting" JobPartID="ID34" Version="1.4">
  <ResourcePool>
    <Media Class="Consumable" ID="M123456" ProductID="FlexoSleeve"
      Status="Available"
      DescriptiveName="Sleeve" Dimension="1417.32 750.0" MediaType="Sleeve"
      PlateTechnology="FlexoDigitalSolvent" Manufacturer="PlateManufacturerB"
      Brand="BrandB" BatchID="Batch 6789" Thickness="2810"
      ReliefThickness="500">
      <!--MediaLayers contains 2 items: the base layer and the
        photopolymer layer of the flexo plate
      -->
      <MediaLayers>
        <Media DescriptiveName="Base" MediaType="Plate"
          MediaTypeDetails="FlexoPlateBase" Thickness="1249"/>
        <Media DescriptiveName="Photopolymer Layer" MediaType="Plate"
          MediaTypeDetails="FlexoPhotopolymer" Thickness="1570"/>
      </MediaLayers>
    </Media>
  </ResourcePool>
  <ResourceLinkPool>
    <MediaLink Usage="Input" rRef="M123456"/>
  </ResourceLinkPool>
</JDF>
```

7.2.117 MediaSource

[Deprecated in JDF 1.1](#)

See "MediaSource" on page 1039 for details of this deprecated Resource.

7.2.118 MiscConsumable

[New in JDF 1.3](#)

The **MiscConsumable** Resource is intended for cost accounting, inventory control and availability scheduling of supplies used in the production workflow where a more detailed parameterization of the Resource is not necessary. **MiscConsumable** is limited to modeling consumables not already more specifically defined in JDF (**Ink**, **Media**, **Pallet**, **RegisterRibbon**, **Strap** or **UsageCounter**).

MiscConsumable Resources MAY appear as inputs to any JDF Process. The default Unit for Amounts of **MiscConsumable** is Countable Objects.

Certain types of **MiscConsumable** Elements such as **MiscConsumable**[*@ConsumableType* = "WasteContainer"] are typically "consumed" by being filled. The sense of the *Amount* Attribute for such Resources shall be the quantity of unused or empty waste containers that are available. If *Unit* is a volume, distance or weight instead of Countable Objects, such *Amount* will still represent the remaining unused capacity of the waste container.

Resource Properties

Resource Class:	Consumable
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>Any Process</i>

Output of Processes: —

Table 7-271: MiscConsumable Resource

Name	Data Type	Description
<i>ConsumableType</i> ? Modified in JDF 1.4	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> . Values include: <i>Developer</i> – Starter <i>FuserOil</i> – Silicon Oil <i>Glue</i> <i>Headband</i> – New in JDF 1.4 <i>Paperclips</i> <i>Staples</i> <i>WasteContainer</i> – Waste Toner Bottle. <i>Wire</i> – bulk wire used for forming staples or other binding.

7.2.119 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: PhaseTime, ResourceAudit, ResourceCmdParams, ResourceInfo, ResourcePullParams, JobPhase, **NodeInfo**

Example Partition: —

Input of Processes: —

Output of Processes: —

Table 7-272: MISDetails Element (Section 1 of 2)

Name	Data Type	Description
<i>Complexity</i> ? New in JDF 1.4	double	Complexity of the task specified by this JDF Node in a range from 0.0 to 1.0. Note: the interpretation of values is implementation dependant. Values include: <i>0.0</i> – The job is simple and therefore reduced setup and waste or higher speeds are possible. <i>0.5</i> – The job is of standard complexity and therefore standard setup and waste or normal speeds are possible. <i>1.0</i> – The job is complex and therefore more setup and waste or lower speeds are possible.
<i>CostType</i> ?	enumeration	Whether or not this MISDetails is chargeable to the customer or not. Values are: <i>Chargeable</i> <i>NonChargeable</i>

Table 7-272: MISDetails Element (Section 2 of 2)

Name	Data Type	Description
<i>DeviceOperationMode?</i>	enumeration	<p><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.</p> <p>Values are:</p> <p><i>Productive</i> – The Device is used to produce good product. Any times recorded in this mode are to be allocated against the Job.</p> <p><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.</p> <p><i>Maintenance</i> – The Device is used without the intention to produce good product, e.g., to perform (preventive) maintenance.</p>
<i>WorkType?</i>	enumeration	<p>Definition of the work type for this MISDetails, (i.e., whether or not this MISDetails relates to originally planned work, an alteration or rework).</p> <p>Values are:</p> <p><i>Original</i> – Standard work that was originally planned for the Job.</p> <p><i>Alteration</i> – Work done to accommodate change made to the Job at the request of the customer.</p> <p><i>Rework</i> – Work done due to unforeseen problem with original work (bad plate, Resource damaged, etc.)</p>
<i>WorkTypeDetails?</i>	string	<p>Definition of the details of the work type for this MISDetails, (i.e., why the work was done).</p> <p>Values include:</p> <p><i>CustomerRequest</i> – The customer requested change(s) requiring the work.</p> <p><i>EquipmentMalfunction</i> – Equipment used to produce the Resource malfunctioned; Resource needs to be created again.</p> <p><i>InternalChange</i> – Change was made for production efficiency or other internal reason.</p> <p><i>ResourceDamaged</i> – A Resource needs to be created again to account for a damaged Resource (damaged plate, etc.).</p> <p><i>UserError</i> – Incorrect operation of equipment or incorrect creation of Resource requires creating the Resource again.</p>

7.2.120 NodeInfo

The **NodeInfo** Resource contains information about planned scheduling and message routing. It allows MIS to plan, schedule and invoice Jobs or Job Parts. Prior to JDF 1.3, **NodeInfo** was a direct Subelement of the JDF Node and not a Resource.

Modification note: starting with JDF 1.3, **NodeInfo** is a Resource that MUST be linked (via NodeInfoLink) like any other Resource; there is no “inheritance”. However, a Node MAY link to the same **NodeInfo** Resource as its parent

Note: the NORMATIVE **NodeInfo** is specified by a linked Resource. An Informative **NodeInfo** MAY be retrieved by searching the **NodeInfo** of parent Nodes or Ancestor Elements.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	Ancestor
Example Partition:	—
Input of Processes:	<i>Any Process</i>
Output of Processes:	—

Table 7-273: NodeInfo Resource (Section 1 of 3)

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. 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	integer	The scheduling priority for the Node where 100 is the highest and 0 is the lowest. Amongst the Nodes that can be processed in the JDF Instance, all higher priority Nodes are to be processed before any lower priority ones. If one or more of the deadline oriented Attributes (e.g., <i>FirstStart</i> or <i>LastEnd</i>) is specified, such attribute(s) MUST be honored before considering <i>JobPriority</i> . The priority from JMF (QueueSubmissionParams/@Priority or QueueEntryPriParams/@Priority) takes precedence over NodeInfo /@JobPriority. Modification note: starting with JDF 1.4, scheduling priority in the first paragraph is described in terms of the Node rather than the job
<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. Default value is from: JDF/@Status. Values are from: JDF/@Status (Table 3-5, “JDF Node” on page 47)
<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> . Default value is from: JDF/@StatusDetails Values include those from: "StatusDetails Supported Strings" on page 875

Table 7-273: NodeInfo Resource (Section 2 of 3)

Name	Data Type	Description
<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.
<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 <i>ResourceRef</i> 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 MUST be sent to the processor default output URL. If <i>TargetRoute</i> specifies a file-schemed URL, it MUST be the exact file name and NOT just the directory of the resulting JDF. <i>JMF/QueueSubmissionParams/@ReturnURL</i> 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 <i>BusinessInfo</i> .
<i>WorkStepID</i> ? New in JDF 1.4	string	ID of an individual work step, e.g. a Press Run. If NodeInfo is not Partitioned, or all Partitions are executed simultaneously, <i>WorkStepID</i> corresponds to <i>JobPartID</i> .
<i>Employee</i> ?	refelement	The internal administrator or supervisor that is responsible for the product or Process defined in this Node.

Table 7-273: NodeInfo Resource (Section 3 of 3)

Name	Data Type	Description
JMF *	element	Represents JMF Query Messages that set up a persistent channel, as described in Section 5.4.3, 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 Signal Messages). The JMF specified in this NodeInfo MUST be restricted in scope to the containing JDF Element. Typically this will be achieved by explicitly stating <i>JobID</i> in the appropriate <i>QueryTypeObj</i> .
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 5.8.1, Events.

7.2.121 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:	<i>Numbering</i>
Output of Processes:	—

Table 7-274: NumberingParams Resource

Name	Data Type	Description
NumberingParam *	element	Set of parameters for one numbering machine

7.2.121.1 Element: NumberingParam

Table 7-275: 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.

7.2.122 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 7-276: 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>ObjectTags</i> ? New in JDF 1.4	NMTOKENS	Tags associated with individual objects that this ObjectResolution MUST be applied to. Each tag specified in <i>ObjectTags</i> is logically anded with the object type(s) specified by <i>SourceObjects</i> , enabling first qualification by object type (such as image), and then tags associated with those objects. The values of <i>ObjectTags</i> depends on the PDL that the ObjectResolution is applied to.
<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. 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>

7.2.123 OrderingParams

Attributes of the *Ordering* Process, which results in an acquisition.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>Ordering</i>
Output of Processes:	—

Table 7-277: 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	element	OrderingParams require a Comment Element that contains a human-readable description of what to order.
Company ? Deprecated in JDF 1.1	refelement	Address and further information of the Company responsible for this order. Replaced with Contact/Company in JDF 1.1.
Contact * New in JDF 1.1	refelement	Address and further information of the Contact responsible for this order.

7.2.124 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 1039 for details of this deprecated Resource.

7.2.125 PageAssignParams

[New in JDF 1.4](#)

This Resource is an empty container for future extensions

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>PageAssigning</i>
Output of Processes:	—

Table 7-278: PageAssignParams Resource

Name	Data Type	Description

7.2.126 PageList

[New in JDF 1.2](#)

PageList defines the additional metadata of individual finished pages such as pagination details. **PageList** references the finished page regardless of the page's position in a PDL file or **RunList**.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	Assembly, Component, ExposedMedia, LayoutElement, RunList
Example Partition:	<i>PartVersion</i>
Input of Processes:	—
Output of Processes:	—

Table 7-279: 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, AssemblySection Elements or StrippingParams [@ <i>BinderySignatureName</i>] that the finished pages specified by this PageList belong to.
<i>HasBleeds</i> ?	boolean	If " <i>true</i> ", the file has bleeds.
<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.
<i>Assembly</i> ? 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> .
<i>ColorPool</i> ?	refelement	Definition of the color details.
<i>ContentList</i> ? New in JDF 1.3	refelement	List of ContentData Elements that describe individual pieces of content on the pages.

Table 7-279: PageList Resource (Section 2 of 2)

Name	Data Type	Description
ElementColorParams ?	refelement	Color details of the page list.
ImageCompressionParams ?	refelement	Specification of the image compression properties.
PageData *	element	Details of the individual finished page. The PageData Elements are referred to by the values of PageData/@PageIndex (if present), or otherwise, their index in the PageList. In the latter case, the PageData Elements SHOULD, therefore, not be removed or inserted in a position other than the end of the list Modification note: see the Modification note in the section on the PageData Element below.
ScreeningParams ?	refelement	Specification of the screening properties.
SeparationSpec *	element	List of separation names defined in the PageList.

7.2.126.1 Element: PageData

PageData defines the additional metadata of individual finished pages or sets of finished pages with common properties, such as pagination details.

If *PageIndex* is not present in PageData Elements, PageData Elements are referred to by index of the PageData in the PageList. If *PageIndex* is present, it explicitly specifies the indices within the PageList. Either all or no PageData Elements in a PageList MUST have *PageIndex*. If a Page is not represented by a PageData, the Attributes of the PageList itself apply.

Modification note: starting with JDF 1.4, PageData/@PageIndex is added. It allows PageData to describe multiple finished pages and to explicitly specify the index of a PageData Element within a PageList. The explicit index allows a PageList to contain a PageData for a particular index (e.g. 100) without the need for PageData Elements for all indices that are lower (e.g. 0 to 99). Without *PageIndex*, the position of PageData within PageList implicitly specifies its index.

If the PageList is Partitioned, the index refers to PageData Elements in the respective leaves of the Partitioned PageList. The index restarts at 0 with each Partitioned leaf.

Table 7-280: PageData Element (Section 1 of 3)

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. Default value is from: PageList/@AssemblyID
<i>AssemblyIDs</i> ? New in JDF 1.3	NMTOKENS	IDs of the Assembly Elements, AssemblySection Elements or StrippingParams [@BinderySignatureName] that this finished page belongs to. Default value is from: PageList/@AssemblyIDs.
<i>CatalogID</i> ?	string	Identification of the Resource, (e.g., in a catalog environment). Default value is from: PageList/@CatalogID.
<i>CatalogDetails</i> ?	string	Additional details of a Resource in a catalog environment. Default value is from: PageList/@CatalogDetails.

Table 7-280: PageData Element (Section 2 of 3)

Name	Data Type	Description
<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 . Default behavior: PageData does not describe a part of a foldout.
<i>HasBleeds</i> ?	boolean	If " <i>true</i> ", the file has bleeds. Default value is from: PageList/@HasBleeds .
<i>IsBlank</i> ?	boolean	If " <i>true</i> ", the PageData 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. Default value is from: PageList/@IsBlank .
<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> ". Default value is from: PageList/@IsPrintable .
<i>IsTrapped</i> ?	boolean	If " <i>true</i> ", the file has been trapped. Default value is from: PageList/@IsTrapped .
<i>JobID</i> ?	string	ID of the Job that this finished page belongs to. Default value is from: PageList/@JobID .
<i>PageFormat</i> ? New in JDF 1.3	NMTOKEN	Defines the format of the page in a production 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. Note: the values are for a newspaper workflow.
<i>PageIndex</i> ? New in JDF 1.4	IntegerRange-List	List of pages the PageData Element represents. A Page number MUST NOT appear more than once in the PageList .
<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 " <i>1</i> ", " <i>iv</i> " or " <i>C-1</i> ". Note that this might be different from the position of the page in the finished document.

Table 7-280: PageData Element (Section 3 of 3)

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. Default value is from: 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. Default value is from: PageList/@PageLabelSuffix
<i>PageStatus</i> ? New in JDF 1.3	NMTOKENS	Status of a single PageData Element. Values include those from: Table C-20, "MessageEvents and MilestoneType Values" on page 886
<i>ProductID</i> ?	string	An ID of the page as defined in the MIS system. Default value is from: 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. Default value is from: 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. Default value is from: 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. Default value is from: 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. Default value is from: PageList/@Template
<i>ElementColorParams</i> ?	refelement	Color details of the PageData Element. Default value is from: PageList/ElementColorParams
<i>ImageCompressionParams</i> ?	refelement	Specification of the image compression properties. Default value is from: PageList/ImageCompressionParams
<i>PageElement</i> * 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.
<i>ScreeningParams</i> ?	refelement	Specification of the screening properties. Default value is from: PageList/ScreeningParams
<i>SeparationSpec</i> *	element	List of separation names defined in the Element. Default value is from: PageList/SeparationSpec

7.2.126.2 Element: PageElement

[New in JDF 1.3](#)

PageElement defines the additional metadata of individual Elements within a page.

Table 7-281: PageElement Element

Name	Data Type	Description
<i>ContentDataRefs</i> ? New in JDF 1.4	IDREFS	ID of the <i>ContentData</i> Elements in the referenced ContentList . <i>ContentData</i> Elements provide Metadata related to this <i>PageData</i> . <i>ContentDataRefs</i> MUST NOT be specified if no ContentList is specified in the grand-parent PageList Element.
<i>ContentListIndex</i> ? Deprecated in JDF 1.4	integer	Index into a ContentList/ContentData Element. If neither <i>ContentListIndex</i> nor <i>PageElement</i> are specified, this <i>PageElement</i> is a reservation. Deprecation note: starting with JDF 1.4, use <i>ContentDataRefs</i> .
<i>ElementPages</i> ?	Integer-RangeList	List of Pages that this <i>PageElement</i> traverses, e.g., fold out pages or multi-page ads.
<i>ContentType</i> ?	NMTOKEN	Type of content that is placed in this <i>PageElement</i> . Values include those from: ContentList/ContentData/@ContentType
<i>RelativeBox</i> ?	Rectangle	Position of the <i>PageElement</i> in the coordinate system of the parent <i>PageElement</i> or PageList .
<i>PageElement</i> *	element	Further sub-page Elements that comprise this <i>PageElement</i> .

7.2.127 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:	<i>Palletizing</i>
Output of Processes:	—

Table 7-282: Pallet Resource

Name	Data Type	Description
<i>PalletType</i> Modified in JDF 1.4	NMTOKEN	Type of pallet used. Values include: <i>2Way</i> – Two-way entry. <i>4Way</i> – Four-way entry. <i>Euro</i> – Standard 1*1 m Euro pallet. Deprecated in JDF 1.4 <i>Euro800x600</i> – 800x600mm according to 15146-4 (equals half Euro pallet). New in JDF 1.4 <i>Euro800x1200</i> – 800x1200mm according to DIN EN 13698-1 (equals Euro pallet). New in JDF 1.4 <i>Euro1000x1200</i> – 1000x1200mm according to DIN EN 13698-2 (flat pallet). New in JDF 1.4 <i>Euro1200x1200</i> – 1200x1200mm no norm, but in use in the field. New in JDF 1.4
<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> .

7.2.128 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:	<i>Palletizing</i>
Output of Processes:	—

Table 7-283: PalletizingParams Resource

Name	Data Type	Description
LayerAmount? New in JDF 1.4	IntegerList	Ordered number of input components in a layer. The first number is the first layer on the bottom. If there are more layers than entries in the list, counting restarts at the first entry.
MaxHeight?	double	Maximum height of a loaded pallet in points.
MaxWeight?	double	Maximum weight of a loaded pallet in grams.
Overhang? New in JDF 1.4	XYPair	Overhang in x and y direction on each side
OverhangOffset? New in JDF 1.4	XYPair	Overhang offset if overhang is not centered
Pattern?	string	Name of the palletizing pattern. Used to store a predefined pattern that defines the layers and positioning of individual component on the pallet.
Bundle? New in JDF 1.4	refelement	Describes additional properties, such as the number of individual products, and describes the list of the individual products on the pallet

7.2.129 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:	PDLCreationParams
Example Partition:	<i>DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName</i>
Input of Processes:	<i>PDFToPSConversion</i>
Output of Processes:	—

Table 7-284: PDFToPSConversionParams Resource (Section 1 of 3)

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.
<i>IgnoreDeviceExtGState?</i> 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.

Table 7-284: PDFToPSConversionParams Resource (Section 2 of 3)

Name	Data Type	Description
<i>IgnorePageRotation</i> = "false"	boolean	If "true", ignores a "concatenation" provided at the beginning of each page that orients the page so that it is properly rotated. Used when emitting EPS.
<i>IgnoreRawData</i> = "false"	boolean	If "true", no unnecessary filters are to be added when emitting image data.
<i>IgnoreSeparableImages Only</i> = "false"	boolean	If "true", and if emitting EPS, ignores only CMYK and gray images.
<i>IgnoreShowPage</i> = "false"	boolean	If "true", ignores save-and-restore showpage in PostScript files
<i>IgnoreTransfers</i> = "true" New in JDF 1.1	boolean	Ignores TR , TR2 parameters in a source PDF ExtGState dictionary, settransfer and setcolortransfer in a source PostScript file, etc.
<i>IgnoreTTFontsFirst</i> = "false"	boolean	If "true", ignores TrueType fonts before any other fonts.
<i>IgnoreUCR</i> = "true" 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</i> = "IncludeNever"	enumeration	Determines when to embed the base fonts. 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> . Values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeCIDFonts</i> = "IncludeOncePerDoc"	enumeration	Determines when to embed CID fonts. Values are from: <i>IncludeBaseFonts</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. Values are from: <i>IncludeBaseFonts</i>
<i>IncludeOtherResources</i> = "IncludeOncePerDoc"	enumeration	Determines when to include all other types of Resources in the file. Values are from: <i>IncludeBaseFonts</i>
<i>IncludeProcSets</i> = "IncludeOncePerDoc"	enumeration	Determines when to include ProcSets in the file. Values are from: <i>IncludeBaseFonts</i>
<i>IncludeTrueTypeFonts</i> = "IncludeOncePerDoc"	enumeration	Determines when to embed TrueType fonts. Values are from: <i>IncludeBaseFonts</i>
<i>IncludeType1Fonts</i> = "IncludeOncePerDoc"	enumeration	Determines when to embed Type-1 fonts. Values are from: <i>IncludeBaseFonts</i>
<i>IncludeType3Fonts</i> = "IncludeOncePerPage"	enumeration	Determines when to embed Type-3 fonts. It is included here to complete the precedence hierarchy. It has only one value. Values are: <i>IncludeOncePerPage</i>

Table 7-284: PDFToPSConversionParams Resource (Section 3 of 3)

Name	Data Type	Description
<i>OutputType</i> = "PostScript"	enumeration	Describes the kind of output to be generated. 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 <i>MediaBox</i> for outputting PostScript files and <i>CropBox</i> for EPS.
<i>SetupProcsets</i> = "true"	boolean	If "true", indicates that if ProcSets are included, the init/term code is also included.
<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.

7.2.130 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:	<i>PDLCreation</i>
Output of Processes:	—

Table 7-285: PDLCreationParams Resource

Name	Data Type	Description
<i>MimeType</i>	string	This Resource identifies the MIME type associated with this output file format. For example " <i>application/pdf</i> ".
PDFToPSConversionParams ?	refelement	Postscript specific Parameter Resource for the output. MUST NOT be specified unless <i>MimeType</i> = " <i>application/postscript</i> "
PSToPDFConversionParams ?	refelement	PDF specific Parameter Resource for the output. It MUST NOT be specified unless <i>MimeType</i> = " <i>application/pdf</i> "

7.2.131 PDLResourceAlias

This Resource provides a mechanism for referencing Resources that occur in files, or that are expected to be provided by Devices. Prepress and printing Processes have traditionally used the word “Resource” to refer to reusable data structures that are needed to perform Processes. Examples of such Resources include fonts, halftones and functions. The formats of these Resources are defined within PDLs, and instances of these Resources can occur within PDL files or can be provided by Devices.

JDF does not provide a syntax for defining such Resources directly within a Job. Instead, Resources continue to occur within PDL files and continue to be provided by Devices. However, since it is necessary to be able to refer to these Resources from JDF Jobs, the **PDLResourceAlias** Resource is provided to fulfill this need.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	ColorantControl/ColorSpaceSubstitute
Example Partition:	—
Input of Processes:	<i>Interpreting</i>
Output of Processes:	—

Table 7-286: 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 ?	refelement	Location of the file containing the PDL Resource. If FileSpec is absent, the Device is expected to provide the Resource defined by this PDLResourceAlias Resource.

7.2.132 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:	<i>Perforating</i>

Output of Processes: —

Table 7-287: PerforatingParams Resource

Name	Data Type	Description
Perforate *	element	Defines one or more Perforate lines.

7.2.132.1 Element: Perforate

Perforate describes one perforated line.

Table 7-288: Perforate Element

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.
<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. 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.

7.2.133 Person

This Resource provides detailed information about a person. It also has the ability to specify different communication channels to this person. Use *@ProductID* when a unique identifier for the **Person** is required. The structure of the 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.

Modification note: starting with JDF 1.4, a rule about using *@ProductID* is added

Resource Properties

Resource Class:	Parameter
Resource referenced by:	Contact, Employee
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-289: Person Resource

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>Languages</i> ? New in JDF 1.4	languages	List of languages related to the person, ordered by decreasing preference
<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).
<i>Address</i> ? New in JDF 1.2	refelement	Address of the person.
<i>ComChannel</i> *	refelement	Communication channels to the person.

7.2.134 PlaceholderResource

This Resource is used to link Process Group Nodes when the exact nature of interchange Resources is still unknown. In this way, a skeleton of Process networks can be constructed, with the **PlaceholderResource** Resources serving as place holders in lieu of the appropriate Resources. This Resource needs no structure besides that provided in an Abstract Resource Element as it has no inherent value except as a stand-in for other Resources.

Resource Properties

Resource Class:	Placeholder
Resource referenced by:	—
Example Partition:	—
Input of Processes:	Process Group Nodes
Output of Processes:	Process Group Nodes

Resource Structure

The Resource has no additional structure.

7.2.135 PlasticCombBindingParams

This Resource describes the details of the *PlasticCombBinding* Process.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>PlasticCombBinding</i>

Output of Processes: —

Table 7-290: 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>ColorDetails?</i> New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If <i>ColorDetails</i> is supplied, <i>Color</i> SHOULD also be supplied.
<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 929 exist: Values are: <i>P12m-rect-02</i> – Distance = 12 mm; Holes = 7 mm x 3 mm Deprecated in JDF 1.2 <i>P16-9i-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. Deprecation note: starting with JDF 1.2, use the value implied by <i>HoleMakingParams/@HoleType</i> .
<i>HoleMakingParams?</i>	refelement	Details of the holes to be made. Note that <i>HoleMakingParams/@Shape</i> is always rectangular by design of the plastic combs.

7.2.136 PlateCopyParams

[Deprecated in JDF 1.1](#)

See "PlateCopyParams" on page 1041 for details of this deprecated Resource.

7.2.137 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 1012.

7.2.138 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 1014.

7.2.139 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 780, which defines a list of reporting actions to have for given document object tests defined into a Test. (See "TestPool" on page 807.) This section makes use of Elements and Attributes defined in "Device Capability Definitions" on page 776. It is suggested that readers familiarize themselves with that section and "Concept of the Preflight Process" on page 825.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	PreflightReport
Example Partition:	—
Input of Processes:	<i>Preflight</i>
Output of Processes:	—

Table 7-291: PreflightParams Resource

Name	Data Type	Description
ActionPool +	element	A set of ActionPool Elements. Multiple ActionPoolElements 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 780, 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.

7.2.139.1 Element: PreflightAction

Table 7-292: PreflightAction Element

Name	Data Type	Description
SetRef?	IDREF	A reference to a preflight Test ID used to filter a set of objects before applying the tests referenced by preflight Action. When SetRef is not defined, the Test is applied to all the objects.
SetSplitBy= "RunList"	enumeration	This is used to group objects in different ways. Values are: <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. Note: SetSplitBy is only used when SetRef is defined in order to create sets on a page-per-page or document-per-document basis. For instance, if you want to get the list of separations per page, SetSplitBy is set to "Page". In such a case, the report's content (as long as the PRItem is defined properly for the Action) will be grouped by page.

Test Elements make use of Evaluation Subelements that define various basic preflight testing functions that can be combined together in order to build preflight test. In order to specify basic preflight tests using Evaluation, the Sub-

element `BasicPreflightTest` is used. **Note:** The `BasicPreflightTest` includes a `PreflightArgument` Subelement that is defined below.

7.2.139.2 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 7-293: `BasicPreflightTest` Element

Name	Data Type	Description
<code>Classes?</code> New in JDF 1.4	NMTOKENS	List of object classes that the test MUST be applied to. It is strongly recommended to supply <i>Classes</i>
<code>ClassName?</code> New in JDF 1.4	NMTOKEN	This tag can be used to directly command the test to specifically apply on a given class of object. The two purposes of this change are 1) to simplify preflight engine processors, and 2) to simplify Test rules. Values are from: Table 7-463, "Object Classes for a Document" on page 826
<code>DevNS = "http://www.CIP4.org/JDFSchema_1_1"</code>	URI	Namespace of the test that is described by <i>Name</i> in this <code>BasicPreflightTest</code> Element.
<code>ListType = "SingleValue"</code> Modified in JDF 1.4	enumeration	Specifies what type of list or object the basic preflight test describes. Values are from: <code>State/@ListType</code> (Table 7.3.7, "State" on page 787) Modification note: starting with JDF 1.4, <i>ListType</i> has a specified default value.
<code>MaxOccurs = "1"</code>	integer	Maximum number of elements in the list described by this <code>BasicPreflightTest</code> , (e.g., the maximum number of integers in an integer list). If <i>MaxOccurs</i> is not "1", the <code>BasicPreflightTest</code> Element refers to a list or <code>RangeList</code> of values, (e.g., a <code>NameEvaluation</code> will allow a list of NMTOKENS).
<code>MinOccurs = "1"</code>	integer	Minimum number of elements in the list described by this <code>BasicPreflightTest</code> . Default = "1", (i.e., it is an individual value). If <i>MinOccurs</i> is not "1", the <code>BasicPreflightTest</code> Element refers to a list or <code>RangeList</code> of values, (e.g., a <code>NameEvaluation</code> will allow a list of NMTOKENS).
<code>Name</code> Modified in JDF 1.4	NMTOKEN	Local name of the preflight constraint that is evaluated by this <code>BasicPreflightTest</code> . A valid <i>Name</i> value for the JDF namespace is any property name defined in any of the <code>Properties</code> tables in Section 7.4.2, "Properties" on page 829. Preflight tests are defined through the use of constraints. Modification note: starting with JDF 1.4, <i>Name</i> is no longer optional.
<code>PreflightArgument?</code>	element	Additional arguments for the preflight test. For details see "PreflightParams" on page 676 for the definition of <code>PreflightArgument</code> and constraints upon which preflight tests are defined.

7.2.139.3 Element: `PreflightArgument`

This Subelement is used by `BasicPreflightTest` when additional data are needed to determine object property.

Table 7-294: PreflightArgument Element

Name	Data Type	Description
BoxArgument ?	element	Used if BasicPreflightTest/@Name has a value of either "InsideBox" and "OutsideBox". Used for tests with the same two names.
BoxToBoxDifference ?	element	Used by the BoxToBoxDifference test.

7.2.139.4 Element: BoxArgument

Table 7-295: BoxArgument Element

Name	Data Type	Description
Box	enumeration	The box type used to verify inclusion or exclusion. Values are from: Table 7-296, "Box Attribute Values" on page 678
MirrorMargins ?	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. Default behavior: the value of <i>Offset</i> is not changed (if unspecified). Values are: <i>Vertical</i> – turns [l b r t] into [r b l t]. <i>Horizontal</i> – turns [l b r t] into [l t r b].
Offset?	rectangle	The offset to build real rectangle to which test is made.
Overlap="false"	boolean	Explains if overlap is allowed to check inclusion or exclusion.

— Attribute: Box Attribute

Table 7-296: Box Attribute Values (Section 1 of 2)

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.

Table 7-296: Box Attribute Values (Section 2 of 2)

Box Type	Description
<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.

7.2.139.5 Element: BoxToBoxDifference

Table 7-297: BoxToBoxDifference Element

Name	Data Type	Description
<i>FromBox?</i>	enumeration	The "From" box used for BoxToBoxDifference calculation. Values are from: BoxArgument/@Box
<i>ToBox?</i>	enumeration	The "To" box used for BoxToBoxDifference calculation. Values are from: BoxArgument/@Box

Example 7-43: Test with InsideBox and a BoxArgument Subelement

The following is an example of Test using *InsideBox* and a *BoxArgument* Subelement:

```
<JDF ID="ID" Type="ProcessGroup" xmlns="http://www.CIP4.org/JDFSchema_1_1"
  Status="Waiting" JobPartID="ID34" Version="1.4">
  <ResourcePool>
    <PreflightParams Class="Parameter" ID="PP001" Status="Available">
      <TestPool>
        <Test ID="PT01">
          <BooleanEvaluation ValueList="true">
            <BasicPreflightTest Name="InsideBox">
              <PreflightArgument>
                <BoxArgument Box="TrimBox" Overlap="true"/>
              </PreflightArgument>
            </BasicPreflightTest>
          </BooleanEvaluation>
        </Test>
      </TestPool>
    </PreflightParams>
  </ResourcePool>
  <ResourceLinkPool>
    <PreflightParamsLink Usage="Input" rRef="PP001"/>
  </ResourceLinkPool>
</JDF>
```

7.2.140 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 1015.

7.2.141 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 776. It is suggested that reader's familiarize themselves with that section and "Concept of the Preflight Process" on page 825.

Resource Properties

Resource Class: Parameter

Resource referenced by: —
 Example Partition: —
 Input of Processes: *Any Process*
 Output of Processes: *Preflight*

Table 7-298: PreflightReport Resource

Name	Data Type	Description
<i>ErrorCount</i>	integer	The count of errors that were encountered while preflighting the Job.
<i>ErrorState?</i>	enumerations	Describes the type of errors that occurred during preflighting when the <i>Preflight</i> Process does not understand certain preflight tests or cannot apply them to the given objects. Default behavior: no errors occurred (if not specified). Values are: <i>TestNotSupported</i> <i>TestWrongPDL</i>
<i>WarningCount</i>	integer	The count of warnings that were encountered while preflighting the Job.
PreflightParams	refelement	References the PreflightParams that was used to create this report.
PreflightReportRulePool	refelement	References the PreflightReportRulePool that was used to create this report.
PRItem *	element	Describes the Action Elements that produced an error or a warning.
RunList	refelement	References the RunList that was used to create this report.

7.2.141.1 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 7-299: 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.

7.2.141.2 Element: PRError

The PRError structure is used to describe generic errors that occurred while evaluating an object property while executing a Test.

Table 7-300: PRError Element

Name	Data Type	Description
<i>ErrorType</i>	enumeration	Values are: <i>TestWrongPDL</i> <i>TestNotSupported</i>
<i>Value</i>	NMTOKEN	The name of the object property that was being tested when the Process error occurred.

7.2.141.3 Element: PRGroup

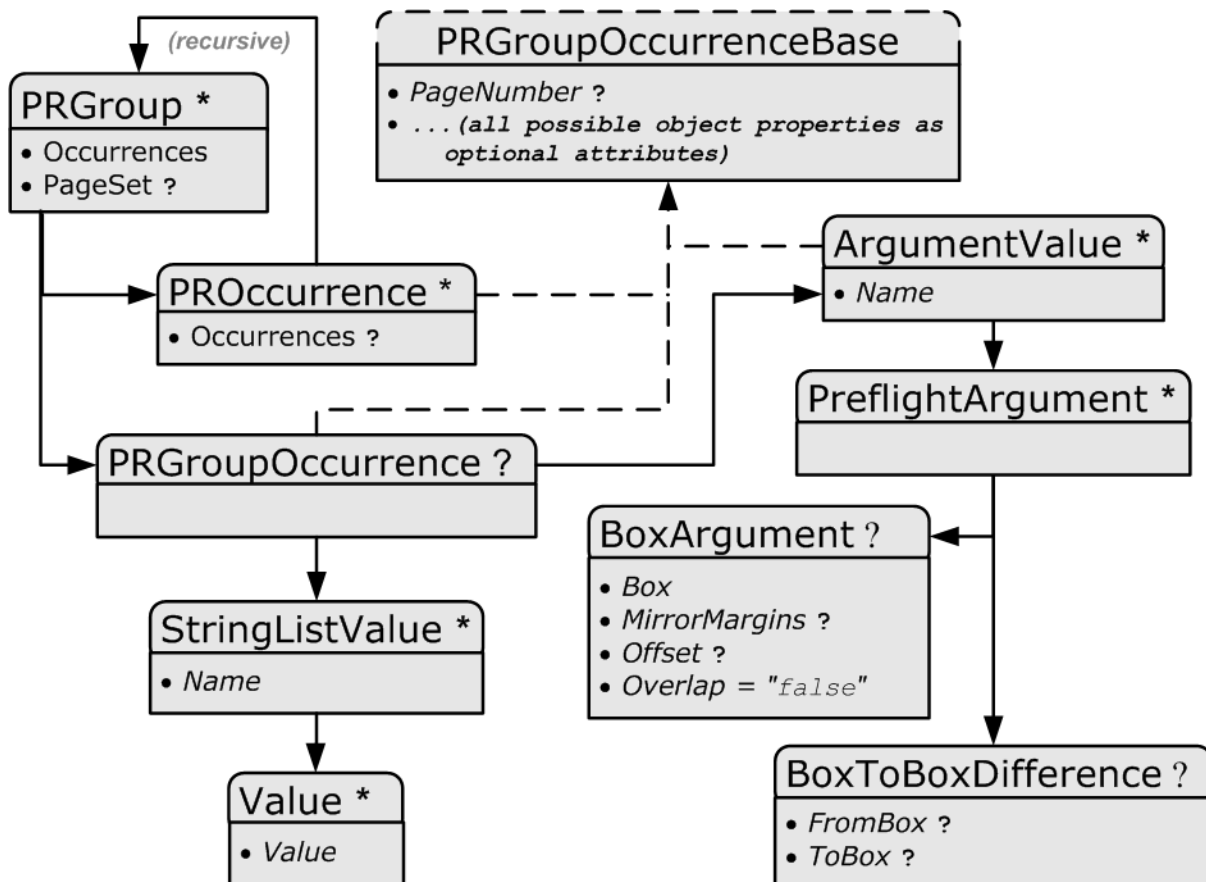


Figure 7-47: PRGroup – a diagram of its structure

The PRGroup structure is used to describe a group of document objects that share common properties and that failed the Action.

Table 7-301: 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.

Table 7-301: PRGroup Element (Section 2 of 2)

Name	Data Type	Description
<i>PageSet</i> ?	IntegerRange-List	All run indices where there is an object of this group that gives an error on that page.
PRGroupOccurrence ?	element	The properties that are shared by all Elements of the group as defined by PreflightReportRulePool/PRRule/@GroupBy .
PROccurrence *	element	An object that failed the Action.

Depending on the test in the Action, the PRGroup is used in two different ways:

- When the test is not a set-test, there will be one level of PRGroup and PROccurrence Elements. These are used to describe all the document objects that failed the preflight test. The PROccurrence describes the actual object while PRGroup is used to group those objects that share common properties.
- When the test is a set-test, there will be two levels of PRGroup and PROccurrence Elements whereby the second level occurs as a child Element of PROccurrence.
 - The top level describes the set objects that failed the preflight test. Just as in the non-set-test case, PROccurrence describes the actual set-objects while PRGroup is used to group those sets that share common properties. In the example below there are four page sets that failed the test, (e.g., pages 1, 4, 8 and 12).
 - The second level, which is a child Element of the top level PROccurrence, describes the document objects that are part of the set. These document objects are grouped as well. In the example below page one consists of 20 objects: five text objects and 15 image objects.

Example 7-44: PRItem

```
<JDF ID="ID" Type="ProcessGroup" xmlns="http://www.CIP4.org/JDFSchema_1_1"
  Status="Waiting" JobPartID="ID34" Version="1.4">
  <ResourcePool>
    <PreflightReport Class="Parameter" ID="PP001" Status="Available"
      ErrorCount="0" WarningCount="0" >
      <PRItem Occurrences="4" ActionRef="A001">
        <PRGroup Occurrences="1">
          <PRGroupOccurrence PageNumber="1"/>
          <PROccurrence Occurrences="20">
            <PRGroup Occurrences="5">
              <PRGroupOccurrence/>
              <PROccurrence TextSize="12"/>
            </PRGroup>
            <PRGroup Occurrences="15">
              <PRGroupOccurrence/>
              <PROccurrence EffectiveResolution="300 300"/>
            </PRGroup>
          </PROccurrence>
        </PRGroup>
        <PRGroup Occurrences="1">
          <PRGroupOccurrence PageNumber="4"/>
          <PROccurrence Occurrences="20">
            <PRGroup Occurrences="7">
              <PRGroupOccurrence/>
              <PROccurrence NumberOfPathPoints="4"/>
            </PRGroup>
            <PRGroup Occurrences="13">
              <PRGroupOccurrence/>
              <PROccurrence EffectiveResolution="300 300"/>
            </PRGroup>
          </PROccurrence>
        </PRGroup>
        <PRGroup Occurrences="1">
          <PRGroupOccurrence PageNumber="8"/>
        </PRGroup>
      </PRItem>
    </PreflightReport>
  </ResourcePool>
</JDF>
```

```

    <PRGroup Occurrences="1">
      <PRGroupOccurrence PageNumber="12"/>
    </PRGroup>
  </PRItem>
  <PreflightParams>
    <TestPool>
      <Test ID="T001">
        <BooleanEvaluation ValueList="true"/>
      </Test>
    </TestPool>
    <ActionPool>
      <Action ID="A001" TestRef="T001"/>
    </ActionPool>
  </PreflightParams>
  <PreflightReportRulePool/>
  <RunList/>
</PreflightReport>
</ResourcePool>
<ResourceLinkPool>
  <PreflightReportLink Usage="Input" rRef="PP001"/>
</ResourceLinkPool>
</JDF>

```

7.2.141.4 Element: Abstract PRGroupOccurrenceBase

Abstract PRGroupOccurrenceBase is an Abstract Element that serves as container for properties that were evaluated during the *Preflight* Process.

Table 7-302: 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 above. See also section "Properties" on page 829 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.

The following Elements are derived from the Abstract PRGroupOccurrenceBase Element

Table 7-303: List of PRGroupOccurrenceBase Elements

Name	Page	Description
ArgumentValue	page 683	For additional arguments for a PRGroupOccurrence.
PRGroupOccurrence	page 684	Specifies shared Properties of all PROccurrence Elements in a PRGroup
PROccurrence	page 684	Describes an individual occurrence of a preflight action failure

7.2.141.5 Element: ArgumentValue

ArgumentValue specifies a value that is specified with additional arguments. ArgumentValue is derived from Abstract PRGroupOccurrenceBase:

Table 7-304: ArgumentValue Element (Section 1 of 2)

Name	Data Type	Description
<i>Name</i>	NMTOKEN	The name of the subject property.

Table 7-304: ArgumentValue Element (Section 2 of 2)

Name	Data Type	Description
PreflightArgument	element	The argument that was used to evaluate this property. This is a PreflightArgument Element. (See “PreflightArgument” on page 677.)

7.2.141.6 Element: PRGroupOccurrence

PRGroupOccurrence specifies the shared properties of all PROccurrence Elements in a PRGroup. When the object does not support a certain property, the corresponding Attribute MUST NOT be specified in PRGroupOccurrence. PRGroupOccurrence is derived from Abstract PRGroupOccurrenceBase.

Table 7-305: 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.

7.2.141.7 Element: StringListValue

StringListValue specifies a type that returns a set of strings.

Table 7-306: StringListValue Element

Name	Data Type	Description
<i>Name</i>	NMTOKEN	The name of the subject property.
Value *	element	Element of type StringEvaluation/Value. (See “StringEvaluation” on page 817.)

7.2.141.8 Element: PROccurrence

PROccurrence describes an individual occurrence of a preflight action failure. When the object does not support a certain property, the corresponding Attribute MUST NOT be specified in PROccurrence. PROccurrence is derived from Abstract PRGroupOccurrenceBase.

Table 7-307: 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.

7.2.142 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 776. It is suggested that reader's familiarize themselves with that section and "Concept of the Preflight Process" on page 825.

Resource Properties

Resource Class: Parameter
Resource referenced by: PreflightReport
Example Partition: —
Input of Processes: Preflight

Output of Processes: —

Table 7-308: 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> Deprecated in JDF 1.3	IDREFS	References the <i>ActionPool</i> whose reporting are defined by this rule. Deprecation note: starting with JDF 1.3 Errata, <i>ActionPools</i> is deprecated because <i>PRRule/ActionRefs</i> has the same role.
<i>PRRule</i> *	element	A list of available <i>PRRule</i> Elements.
<i>PRRuleAttr</i> ?	element	Defines the default behavior of all <i>PRRule</i> when not defined inside of a <i>PRRule</i> Subelement.

7.2.142.1 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 7-309: PRRule Element

Name	Data Type	Description
<i>ActionRefs</i> +	IDREFS	References the action for which the report behavior is defined in <i>PRRule</i> .
<i>PRRuleAttr</i>	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).

7.2.142.2 Element: PRRuleAttr

Table 7-310: PRRuleAttr Element (Section 1 of 2)

Name	Data Type	Description
<i>GroupBy</i> = "Tested"	NMTOKENS	Group objects having the same N-pair of Attributes listed here. Values include those from: <i>@ReportAttr</i>
<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. Values include those from: <i>Table 7-311, "ReportAttr Attribute Values" on page 686</i>
<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 825.)
<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> .

Table 7-310: PRRuleAttr Element (Section 2 of 2)

Name	Data Type	Description
<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> .

— Attribute: ReportAttr

Table 7-311: ReportAttr Attribute Values

Value	Description
< <i>Property Attribute</i> >	An object-specific Attribute, (e.g., <i>ColorSpace</i> , <i>FontName</i> , etc.). At the time that we define the <i>Test</i> , we will almost automatically define these Attributes.
<i>Tested</i>	Refers to all the Attributes that are referred to in the <i>Test</i> Element(s) used by the <i>Action</i> Element(s) listed in the <i>ActionRefs</i> .
<i>TestRelated</i>	Refers to all the Attributes referred in the <i>Test</i> Element(s) used by the <i>Action</i> Element(s) listed in <i>ActionRefs</i> and the ones that belong to the group of properties in which the tested property was found. For instance, if the <i>Creator</i> basic test was made, then all other document properties will be reported as well.
<i>VerboseAppSpecific</i>	Refers to a large list of Attributes that the preflight agent (with preflight agent-specific logic) finds interesting for the <i>Test</i> Element(s) used by the <i>Action</i> Element(s) listed in <i>ActionRefs</i> .
<i>BriefAppSpecific</i>	Refers to a small list of Attributes that the preflight agent (with preflight agent-specific logic) finds interesting for the <i>Test</i> Element(s) used by the <i>Action</i> Element(s) listed in <i>ActionRefs</i> .

When the report is generated, the "*Tested*", "*VerboseAppSpecific*" and "*BriefAppSpecific*" terms are expanded depending on the context (i.e., the specific test and the specific preflight agent) so that the list of Attributes only contain object specific Attributes.

Note: The "*VerboseAppSpecific*" and "*BriefAppSpecific*" tokens can be dependent on the context of a specific test. It is expected that a preflight agent will have a default list of tokens that will always be added (e.g., "*PageNumber*"). In addition it is expected that a preflight agent will define separate lists for specific domains, (e.g., color, font). When a specific test covers some of these specific domains, the Attributes of these lists are also added. When *ReportAttr* = "*Tested BriefAppSpecific PageNumber*", the Attributes that are reported are dependent on the *Test* Element(s) used by the *Action* Element(s) and on the preflight agent as demonstrated in the table below.

Table 7-312: Contingent Report Behavior

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
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/>`.

7.2.143 Preview

The preview of the content of a surface. It can be used for the calculation of the ink coverage (`PreviewUsage = "Separation"`) or as a preview of what is currently processed in a Device (`PreviewUsage = "Viewable"` or `PreviewUsage = "ThumbNail"`). When the preview is of `PreviewUsage = "Separation"` or `PreviewUsage = "SeparationRaw"`, a gray value of "0" represents full ink, while a value of "255" represents no ink (for more information, see `DeviceGray` color model chapter 4.8.2 of the *PostScript Language Reference Manual*) [PS].

Resource Properties

Resource Class:	Parameter
Resource referenced by:	Any Element (generic content) , <code>QueueEntry</code>
Example Partition:	<code>PreviewType</code> , <code>Separation</code> , <code>SheetName</code> , <code>Side</code> , <code>TileID</code> , <code>WebName</code> , <code>RibbonName</code>
Input of Processes:	<code>InkZoneCalculation</code> , <code>PreviewGeneration</code>
Output of Processes:	<code>PreviewGeneration</code>

Table 7-313: Preview Resource (Section 1 of 3)

Name	Data Type	Description
Compensation? Modified in JDF 1.2	enumeration	Compensation of the image to reflect the application of transfer curves to the image. Values are: <code>Unknown</code> – Deprecated in JDF 1.2 <code>None</code> – No compensation. <code>Film</code> – Compensated until film exposure. <code>Plate</code> – Compensated until plate exposure. <code>Press</code> – Compensated until press.
CTM? New in JDF 1.1 Modified in JDF 1.3	matrix	Orientation of the Preview with respect to the Layout coordinate system. CTM is applied after any transformation defined within the referenced image file, (for example: the transformation defined in the CIP3PreviewImageMatrix of a PPF file). In case of PPF, <code>CTM</code> is applied to the native Postscript coordinate system of the preview. In case of PNG, the origin of the object is defined as the lower left corner of the image.
Directory? New in JDF 1.1	URL	Defines a base URL for the files that represent this Preview . If <code>Directory</code> 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 913 and [FileURL] for examples.
MimeTypeDetails? New in JDF 1.4	string	Specifies additional details of the preview's MIME type in case the value of <code>PreviewFileType</code> is a MIME type.

Table 7-313: Preview Resource (Section 2 of 3)

Name	Data Type	Description
<p><i>PreviewFileType</i> = "PNG" New in JDF 1.2 Modified in JDF 1.4</p>	string	<p>The file type of the preview.</p> <p>Values include:</p> <p><i>PNG</i> – The Portable Network Graphics format.</p> <p><i>CIP3Multiple</i> – The format as defined in the CIP3 PPF specification. One or more previews per CIP3 file are supported.</p> <p><i>CIP3Single</i> – The format as defined in the CIP3 PPF specification. Only one preview per CIP3 file is supported.</p> <p>Values are also: any MIME media type. See Appendix H, "MimeType and MimeTypeVersion Attributes" on page 903. New in JDF 1.4</p> <p>Note: 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 <i>InkZoneCalculation</i>.</p> <p>Modification note: starting with JDF 1.4, the Data Type is changed from enumeration to string because MIME media types are added as values.</p>
<p><i>PreviewType</i> ? Deprecated in JDF 1.2</p>	enumeration	<p>Type of the preview.</p> <p>Values are:</p> <p><i>Separation</i> – Separated preview in medium resolution.</p> <p><i>SeparationRaw</i> – Separated preview in medium resolution.</p> <p><i>SeparatedThumbNail</i> – Very low resolution separated preview.</p> <p><i>ThumbNail</i> – Very low resolution RGB preview.</p> <p><i>Viewable</i> – RGB preview in medium resolution.</p> <p>Deprecation note: starting with JDF 1.2, <i>PreviewType</i> is still a Partition Key and MUST be used only as such — as an Attribute of Preview, <i>PreviewUsage</i> (below) replaces <i>PreviewType</i>.</p>
<p><i>PreviewUsage</i> = "Separation" New in JDF 1.2 Modified in JDF 1.4</p>	enumeration	<p>The kind of the preview.</p> <p><i>PreviewUsage</i> defines the semantics of the preview.</p> <p>Constraint: If both <i>PreviewType</i> as a Partition Key and <i>PreviewUsage</i> are specified, they MUST match.</p> <p>Values are:</p> <p><i>3D</i> – static 3D model New in JDF 1.4</p> <p><i>Animation</i> – animated previews for 3D display. New in JDF 1.4</p> <p><i>Separation</i> – Separated preview in medium resolution. Separation is generally used in <i>InkZoneCalculation</i>.</p> <p><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.</p> <p><i>SeparatedThumbNail</i> – Very low resolution separated preview.</p> <p><i>ThumbNail</i> – Very low resolution RGB preview.</p> <p><i>Viewable</i> – RGB preview in medium resolution.</p>

Table 7-313: Preview Resource (Section 3 of 3)

Name	Data Type	Description
URL Modified in JDF 1.2	URL	<p>URL identifying any preview file, (e.g., the PNG image or CIP3 PPF file that represents this Preview).</p> <p>See [RFC3986] and "Resolving RunList/@Directory and FileSpec/@URL URI references" on page 913 and "FileSpec Attributes and Container Subelement" on page 939 for the syntax and examples. For the "file:" URL scheme see also [RFC1738] and [FileURL].</p> <p>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> partion keys used in the JDF.</p>

7.2.144 PreviewGenerationParams

Parameters specifying the size and the type of the preview.

Resource Properties

Resource Class: Parameter

Resource referenced by: —

Example Partition: *PreviewType, Separation, SheetName, Side, TileID, WebName, RibbonName*

Input of Processes: *PreviewGeneration*

Output of Processes: —

Table 7-314: PreviewGenerationParams Resource (Section 1 of 3)

Name	Data Type	Description
AspectRatio = "Ignore" New in JDF 1.1	enumeration	<p>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.</p> <p>Values are:</p> <p><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.</p> <p><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.</p> <p><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>.</p> <p><i>Expand</i> – Keep the aspect ratio, and modify the preview size so that the smaller image dimension is defined by <i>Size</i>.</p> <p><i>Ignore</i> – Fill the preview completely, keeping <i>Size</i>, even if this requires modifying the aspect ratio.</p>

Table 7-314: PreviewGenerationParams Resource (Section 2 of 3)

Name	Data Type	Description
<i>Compensation</i> ? Modified in JDF 1.2	enumeration	Compensation of the image to reflect the application of transfer curves to the image. 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. 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. Note: 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 <i>InkZoneCalculation</i> .
<i>PreviewType</i> ? Deprecated in JDF 1.1	enumeration	The kind of preview to be generated. Values are: <i>Separation</i> <i>Viewable</i> Deprecation note: starting with JDF 1.1, <i>PreviewType</i> is still a Partition Key and MUST be used only as such — as an Attribute of Preview , <i>PreviewUsage</i> (below) replaces <i>PreviewType</i> .
<i>PreviewUsage</i> = "Separation" New in JDF 1.1 Modified in JDF 1.2	enumeration	The kind of preview to be generated. 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. Constraint: <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.
<i>Resolution</i> ?	XYPair	Resolution of the preview, in dpi. If <i>PreviewUsage</i> = "Separation", the default is "50.8 50.8".
<i>Size</i> ?	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.

Table 7-314: PreviewGenerationParams Resource (Section 3 of 3)

Name	Data Type	Description
ImageSetterParams ? New in JDF 1.1	refelement	Details of the <i>ImageSetting</i> Process. Needed for accessing information about coordinate transformations that are performed by the imagesetter hardware.

7.2.145 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:	<i>ConventionalPrinting, DigitalPrinting</i>
Output of Processes:	—

Table 7-315: PrintCondition Resource

Name	Data Type	Description
<i>AimCurve</i> ?	Transfer-Function	Describes the desired tone-value increase function. If not specified, it defaults to the media and printing machine-specific values
<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 7-45: PrintCondition

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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>
</ResourcePool>
<ResourceLinkPool>
  <PrintConditionLink Usage="Input" rRef="PC"/>
</ResourceLinkPool>
</JDF>

```

7.2.146 PrintRollingParams

[New in JDF 1.2](#)

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>PrintRolling</i>
Output of Processes:	—

Table 7-316: 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.

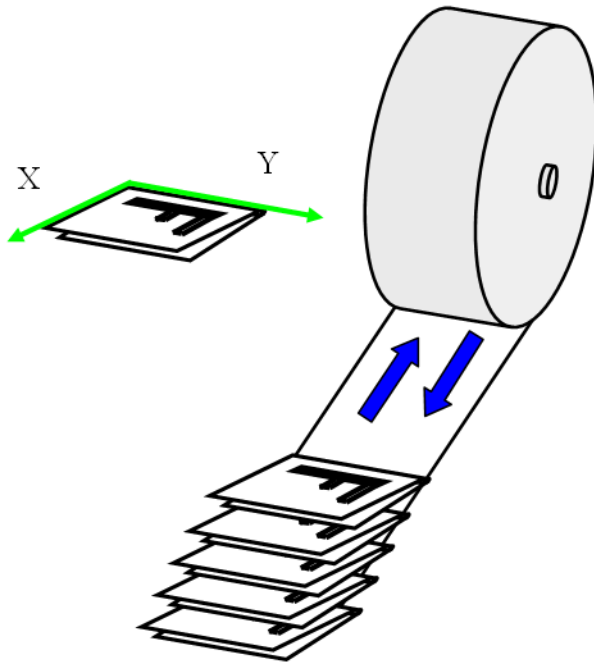


Figure 7-48: PrintRollingParams Coordinate System

7.2.147 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 7-317: ProductionPath Resource (Section 1 of 2)

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.

Table 7-317: ProductionPath Resource (Section 2 of 2)

Name	Data Type	Description
PrintingUnitWebPath ?	element	Describes the path through the printing units. If not specified, <i>ProductionPathID</i> MUST be specified.

7.2.147.1 Element: FolderSuperstructureWebPath

This is a placeholder that might be filled with additional information in future versions of JDF. In JDF 1.3, paths are identified by ID only.

Table 7-318: FolderSuperstructureWebPath Element

Name	Data Type	Description
<i>ProductionPathID</i> ?	string	Unique identification of the part of the production path specified in this Element.

7.2.147.2 Element: PostPressComponentPath

This is a placeholder that might be filled with additional information in future versions of JDF. In JDF 1.3, paths are identified by ID only.

Table 7-319: PostPressComponentPath Element

Name	Data Type	Description
<i>ProductionPathID</i> ?	string	Unique identification of the part of the production path specified in this Element.

7.2.147.3 Element: PrintingUnitWebPath

This is a placeholder that might be filled with additional information in future versions of JDF. In JDF 1.3, paths are identified by ID only.

Table 7-320: PrintingUnitWebPath Element

Name	Data Type	Description
<i>ProductionPathID</i> ?	string	Unique identification of the part of the production path specified in this Element.

Example 7-46: ProductionPath: on Path Level:

This example and the next illustrate the different Web path description levels:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <ProductionPath Class="Parameter" ID="F1" Status="Available"
      ProductionPathID="ID_2webproduction_64pages" />
  </ResourcePool>
  <ResourceLinkPool>
    <ProductionPathLink Usage="Input" rRef="F1" />
  </ResourceLinkPool>
</JDF>
```

Example 7-47: ProductionPath: on Part Path Level:

This example and the previous illustrate the different Web path description levels:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
```



```

<ProductionPath Class="Parameter" ID="F1" Status="Available"
  PartIDKeys="WebName">
  <ProductionPath WebName="1">
    <PrintingUnitWebPath ProductionPathID="ID_PrintingUnitWebPath"/>
    <FolderSuperstructureWebPath ProductionPathID="abcd"/>
    <PostPressComponentPath ProductionPathID="xyz"/>
  </ProductionPath>
</ProductionPath>
</ResourcePool>
<ResourceLinkPool>
  <ProductionPathLink Usage="Input" rRef="F1"/>
</ResourceLinkPool>
</JDF>

```

7.2.148 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 1041.

7.2.149 PSToPDFConversionParams

This Resource contains the parameters that control the conversion any PDL to PDF documents. Prior to JDF 1.3, **PSToPDFConversionParams** was used only for converting PostScript streams to PDF. The name "PSToPDFConversionParams" was retained for backwards compatibility, although most parameters apply to PDF conversion from any source format.

Some descriptions below mention Attributes or structures in specific source formats, such as PostScript. Appropriate equivalent actions should be taken when converting from other source formats that have equivalent Attributes or structures. A small number of parameters apply only to PostScript sources.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	PDLCreationParams
Example Partition:	<i>DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName</i>
Input of Processes:	<i>PSToPDFConversion</i>
Output of Processes:	—

Table 7-321: PSToPDFConversionParams Resource (Section 1 of 3)

Name	Data Type	Description
<i>AllowJBIG2Globals = "false"</i>	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 = "false"</i>	boolean	If "true", binary streams (e.g., page contents streams, sampled images, and embedded fonts) are ASCII85-encoded, resulting in a PDF file that is almost pure ASCII. If "false", they are not, resulting in a PDF file that can contain substantial amounts of binary data.

Table 7-321: PStoPDFConversionParams Resource (Section 2 of 3)

Name	Data Type	Description
<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”. 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 = "Left"</i>	enumeration	Determines how the printed pages would be bound. Values are: <i>Left</i> – for left binding. <i>Right</i> – 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.
<i>DefaultRenderingIntent?</i> Modified in JDF 1.2	enumeration	Selects the rendering intent for the current Job. Values are: <i>Default</i> Deprecated in JDF 1.2 <i>Perceptual</i> <i>Saturation</i> <i>RelativeColorimetric</i> <i>AbsoluteColorimetric</i> Note: See the <i>Portable Document Format Reference Manual</i> for more information on rendering intent.
<i>DetectBlend = "true"</i>	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 = "true"</i>	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.

Table 7-321: PStoPDFConversionParams Resource (Section 3 of 3)

Name	Data Type	Description
<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 <i>InitialPageSize</i>).
<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. 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 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> .
<i>AdvancedParams</i> ?	element	Advanced parameters which control how certain features of PDF are handled.
<i>PDFXParams</i> ? New in JDF 1.2	element	PDF/X parameters.
<i>ThinPDFParams</i> ?	element	Parameters that control the optional content or form of PDF files that will be created.

7.2.149.1 Element: AdvancedParams

Table 7-322: 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)

Table 7-322: AdvancedParams Element (Section 2 of 3)

Name	Data Type	Description
<code>LockDistillerParams = "true"</code>	boolean	If <code>"true"</code> , any <code>PSToPDFConversionParams</code> settings configured by the source content (e.g., with <code>setdistillerparams</code> in a PostScript source document) are ignored. If <code>"false"</code> , 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 <code>LockDistillerParams</code> was accidentally inverted. It is now consistent with the PostScript <code>setdistillerparams</code> operator.
<code>ParseDSCComments = "true"</code>	boolean	If <code>"true"</code> , 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 <code>"false"</code> , the Process treats the DSC comments as pure PS comments and ignores them. (PostScript source only)
<code>ParseDSCCommentForDocInfo = "true"</code>	boolean	If <code>"true"</code> , 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.
<code>PassThroughJPEGImages = "false"</code> New in JDF 1.2	boolean	If <code>"true"</code> , JPEG images are passed through without recompressing them.
<code>PreserveCopyPage = "true"</code>	boolean	If <code>"true"</code> , the <code>copypage</code> operator of PostScript Level 2 is maintained. If <code>"false"</code> , the PostScript Level 3 definition of <code>copypage</code> operator is used. In PostScript Levels 1 and 2, the <code>copypage</code> operator transmits the page contents to the current output Device (similar to <code>showpage</code>). However, <code>copypage</code> does not perform many of the re-initializations that <code>showpage</code> does. Many PostScript Level 1 and 2 programs used the <code>copypage</code> operator to perform such operations as printing multiple copies and implementing forms. These programs produce incorrect results when interpreted using the Level 3 <code>copypage</code> semantics. This Attribute provides a mechanism to retain Level 2 compatibility for this operator. (PostScript source only)
<code>PreserveEPSInfo = "true"</code>	boolean	If <code>"true"</code> , preserves the EPS information in a PostScript source file and stores it in the resulting PDF file. (PostScript source only)
<code>PreserveHalftoneInfo = "false"</code> New in JDF 1.1	boolean	If <code>"true"</code> , passes halftone screen information (frequency, angle and spot function) into the PDF file. If <code>"false"</code> , halftone information is not passed in.
<code>PreserveOverprintSettings = "true"</code> New in JDF 1.1	boolean	If <code>"true"</code> , passes the value of the <code>setoverprint</code> operator through to the PDF file. Otherwise, overprint is ignored.

Table 7-322: AdvancedParams Element (Section 3 of 3)

Name	Data Type	Description
<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. 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. Values are: <i>Preserve</i> – The arguments are passed into the PDF file. <i>Remove</i> – The arguments are ignored.
<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)

7.2.149.2 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 7-323: 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.

Table 7-323: PDFXParams Element (Section 2 of 2)

Name	Data Type	Description
<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. Values include those from: Table 7-324, “PDFXOutputIntentProfile Attribute Values”.
<i>PDFXNoTrimBoxError="true"</i>	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="true"</i>	boolean	If "true" and the BleedBox entry is not specified in the page object of the source document, BleedBox is set to MediaBox .
<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. Values are: <i>Unknown</i> <i>false</i> <i>true</i> Note: <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 7-324: PDFXOutputIntentProfile Attribute Values (Section 1 of 2)

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>	

Table 7-324: PDFXOutputIntentProfile Attribute Values (Section 2 of 2)

Value	Description
<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>	

7.2.149.3 Element: ThinPDFParams**Table 7-325: ThinPDFParams Element**

Name	Data Type	Description
<i>FilePerPage</i> = "false"	boolean	If "true", the Process generates 1 PDF file per page.
<i>SidelineEPS</i> = "false" New in JDF 1.2	boolean	If "true", embedded EPS files in PostScript source documents are not converted but are stored in external files in the same location as the PDF itself. (PostScript source only)
<i>SidelineFonts</i> = "false"	boolean	If "true", font data are stored in external files during PDF generation.
<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.

7.2.150 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:	<i>QualityControl</i>
Output of Processes:	—

Table 7-326: 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.

7.2.150.1 Element: BindingQualityParams**Table 7-327: 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].

7.2.151 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:	Abstract Resource
Example Partition:	—
Input of Processes:	—
Output of Processes:	<i>QualityControl</i>

Table 7-328: QualityControlResult Resource

Name	Data Type	Description
<i>Failed?</i>	integer	Total number of failed measurements.
<i>Passed?</i>	integer	Total number of passed measurements.
<i>BindingQualityParams?</i>	element	Reference to the measurement setup definition.
<i>FileSpec?</i>	refelement	Location of an external file that contains details of the quality control measurement.
<i>QualityMeasurement*</i>	element	One individual measurement result.

7.2.151.1 Element: QualityMeasurement

QualityMeasurement Elements describe an individual measurement.

Table 7-329: 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> ". Values include: <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.
<i>BindingQualityMeasurement?</i>	element	Details of the <i>BindingQualityMeasurement</i> .

7.2.151.2 Element: BindingQualityMeasurement

Table 7-330: 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].

7.2.152 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:	<i>RasterReading</i>
Output of Processes:	—

Table 7-331: 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 / <i>@SizePolicy</i> = "ClipToMaxPage" and clipping is requested.
<i>MirrorAround</i> = "None"	enumeration	This Attribute specifies the axis around which a raster reader is to mirror an image. 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. 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 Resource MUST be ignored in a Combined Process with <i>LayoutPreparation</i> .

Table 7-331: RasterReadingParams Resource (Section 2 of 2)

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 <i>RasterReading</i> . 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.

7.2.153 RefAnchor

[New in JDF 1.4](#)

RefAnchor describes the relative position with respect to a related element in a layout. Depending on the value of *AnchorType*, it specifies either a parent Element or a sibling Element.

Resource Properties

Resource Class: ResourceElement

Resource referenced by: **Layout/MarkObject, LayoutElementProductionParams/**
LayoutElementPart/PositionObj, StrippingParams/StripMark

Example Partition: —

Input of Processes: —

Output of Processes: —

Table 7-332: RefAnchor Element

Name	Data Type	Description
<i>Anchor?</i>	Anchor	<i>Anchor</i> specifies the origin (0,0) of the vector specified in the rotated coordinate system of the related layout element.
<i>AnchorType?</i>	enumeration	Role of this RefAnchor . Values are: <i>Parent</i> – The layout element referenced by this RefAnchor is a parent. This layout element is transformed with the parent. <i>Sibling</i> – The layout element referenced by this RefAnchor is a sibling. Both layout elements share a common parent. The parent of this layout element is specified as the RefAnchor of the first child in the chain of siblings.
<i>rRef?</i>	IDREF	Reference to a layout element that this layout element is positioned by. if <i>rRef</i> is not specified, the page or sheet defined by the layout element is the parent container. <i>rRef</i> MUST be specified if <i>AnchorType = "Sibling"</i> .

7.2.154 RegisterMark

Defines a register mark, which can be used for setting up and monitoring color registration in a printing Process. It can also be used to synchronize the Sheet position in a paper path. The position and rotation of each register mark can be specified with the help of the following Attributes. It is important that the register marks are defined in such a way that their centers are on the point of origin of the coordinate system, as otherwise they are not positioned properly.

Resource Properties

Resource Class: Parameter

Resource referenced by: HoleMakingParams, Layout/MarkObject
Example Partition: —
Input of Processes: —
Output of Processes: —

Table 7-333: 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> ? Modified in JDF 1.4	NMTOKENS	Type of RegisterMark. Values include: <i>Arc</i> <i>Circle</i> <i>Cross</i> Modification note: starting with JDF 1.4, the data type changes from NMTOKEN to NMTOKENS.
<i>MarkUsage</i> ? New in JDF 1.1 Modified in JDF 1.4	enumerations	Specifies the usage of the RegisterMark. 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>Tile</i> – The mark is used to mark the position of tiles in Tiling. New in JDF 1.4
<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	element	Set of separations to which the register mark is bound.

7.2.155 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 7-49, “RegisterRibbon lengths and coordinate system for BlockPreparation” on page 706:

Resource Properties

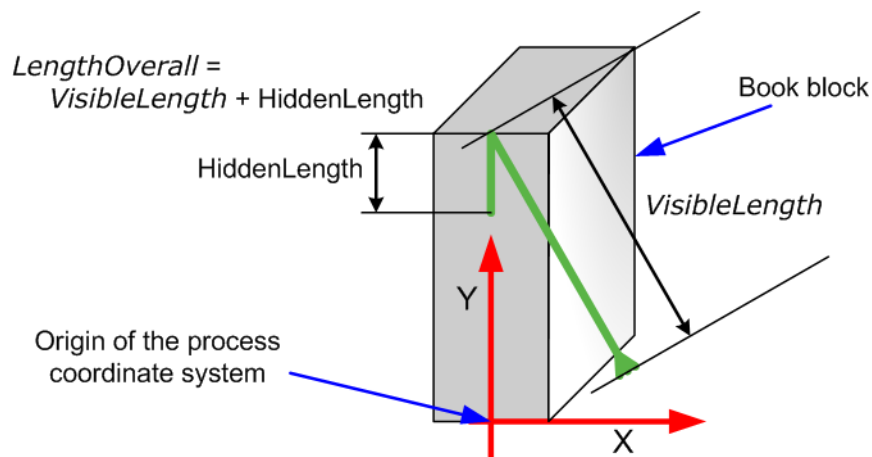
Resource Class: Consumable
Resource referenced by: BindingIntent/HardCoverBinding, BindingIntent/BindList/BindItem/HardCoverBinding, BlockPreparationParams
Example Partition: —
Input of Processes: —
Output of Processes: —

Table 7-334: RegisterRibbon Resource (Section 1 of 2)

Name	Data Type	Description
<i>LengthOverall</i> ? Modified in JDF 1.4	double	Overall length of the register ribbon, (i.e., <i>VisibleLength</i> + <i>HiddenLength</i> in Figure 7-49). Note “ <i>HiddenLength</i> ” is not an Attribute Modification note: starting with JDF 1.4, <i>LengthOverall</i> is optional.
<i>Material</i> ?	string	Material of the register ribbon.
<i>RibbonColor</i> ?	NamedColor	Color of the ribbon.

Table 7-334: RegisterRibbon Resource (Section 2 of 2)

Name	Data Type	Description
<i>RibbonColorDetails</i> ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If <i>RibbonColorDetails</i> is supplied, <i>RibbonColor</i> SHOULD also be supplied.
<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> ? Modified in JDF 1.4	double	Length of the register ribbon which will be seen when opening the book, (See Figure 7-49). Modification note: starting with JDF 1.4, <i>VisibleLength</i> is optional.

**Figure 7-49: RegisterRibbon lengths and coordinate system for BlockPreparation**

7.2.156 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 7-335: RenderingParams Resource

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. 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.
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 <i>Rendering</i> . In JDF 1.2 and beyond, a RIP is to obtain Media information from InterpretingParams/Media .

7.2.157 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:	<i>ResourceDefinition</i>
Output of Processes:	—

Table 7-336: ResourceDefinitionParams Resource (Section 1 of 2)

Name	Data Type	Description
<i>DefaultID</i> ? 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.
<i>DefaultJDF</i> ?	URL	Link to a JDF Resource that defines preset values.

Table 7-336: ResourceDefinitionParams Resource (Section 2 of 2)

Name	Data Type	Description
<i>DefaultPriority</i> = "DefaultJDF"	enumeration	Defines whether preset values of the application or of the Resource specified in <i>DefaultJDF</i> have priority. 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.

7.2.157.1 Element: ResourceParam[New in JDF 1.1](#)

Table 7-337: ResourceParam Element

Name	Data Type	Description
<i>DefaultID</i> ?	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.
<i>DefaultJDF</i> ?	URL	Link to a JDF Resource that defines preset values. Defaults to the <i>DefaultJDF</i> specified in ResourceDefinitionParams .
<i>DefaultPriority</i> ?	enumeration	Defines whether preset values of the application or of the Resource specified in <i>DefaultJDF</i> have priority. Default value is from: parent's ResourceDefinitionParams/@DefaultPriority . Values are: <i>Application</i> <i>DefaultJDF</i>

7.2.158 RingBindingParamsThis Resource describes the details of the *RingBinding* Process.**Resource Properties**

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>RingBinding</i>
Output of Processes:	—

Table 7-338: RingBindingParams Resource (Section 1 of 2)

Name	Data Type	Description
<i>BinderColor</i> ?	NamedColor	Color of the ring binder.
<i>BinderColorDetails</i> ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If <i>BinderColorDetails</i> is supplied, <i>BinderColor</i> SHOULD also be supplied.

Table 7-338: RingBindingParams Resource (Section 2 of 2)

Name	Data Type	Description
<i>BinderMaterial</i> ?	NMTOKEN	The following describe <i>RingBinding</i> 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	<i>RingBinding</i> values: Values include: <i>Round</i> <i>Oval</i> <i>D-shape</i> <i>SlantD</i>
<i>RingSystem</i> ? Deprecated in JDF 1.1	enumeration	Ring binding systems Values are: <i>2HoleEuro</i> – In Europe <i>3HoleUS</i> – In North America <i>4HoleEuro</i> – In Europe Deprecation note: starting with JDF 1.2, use the value implied by <i>HoleMakingParams/@HoleType</i> .
<i>RivetsExposed</i> ?	boolean	The following <i>RingBinding</i> 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>SpineColorDetails</i> ? New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If <i>SpineColorDetails</i> is supplied, <i>SpineColor</i> SHOULD also be supplied.
<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	For <i>RingBinding</i> clear vinyl outer-wrap types on top of a colored base wrap: Values include: <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.
<i>HoleMakingParams</i> ? New in JDF 1.2	refelement	Details of the holes in <i>RingBinding</i> .

7.2.159 RollStand

[New in JDF 1.2](#)

Resource Properties

Resource Class:	Handling
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>PrintRolling</i>
Output of Processes:	—

Table 7-339: 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 .

7.2.160 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 3.10.5, 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 whose *ElementType* is either "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*.

In essence, a **RunList** is a virtual document or set of documents. It allows a document to either be physically spread over multiple files, or multiple documents to be contained within a single file (e.g. PPML, PDF/VT). It retains the same properties as the original documents, e.g. the pages of a document that is described by a **RunList** are ordered.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	ArtDeliveryIntent/ArtDelivery, DigitalMedia, Layout/PageCondition, Layout/SheetCondition, PreflightReport
Example Partition:	<i>PartVersion, Run, RunPage, RunSet, Separation, WebProduct</i>
Input of Processes:	<i>AssetListCreation, ColorCorrection, ColorSpaceConversion, ContoneCalibration, CylinderLayoutPreparation, DigitalDelivery, DigitalPrinting, FormatConversion, ImageReplacement, ImageSetting, Imposition, Interpreting, LayoutPreparation, LayoutShifting, PageAssigning, PDFToPSConversion, PDLCreation, Preflight, PreviewGeneration, PSToPDFConversion, RasterReading, Rendering, Screening, Separation, Stripping, Tiling, Trapping</i>
Output of Processes:	<i>AssetListCreation, ColorCorrection, ColorSpaceConversion, ContoneCalibration, DBTemplateMerging, DigitalDelivery, FormatConversion, ImageReplacement, Imposition, Interpreting, LayoutElementProduction, LayoutPreparation, LayoutShifting, PageAssigning, PDFToPSConversion, PDLCreation, PSToPDFConversion, RasterReading, Rendering, Scanning, Screening, Separation, Stripping, Tiling, Trapping</i>

Table 7-340: RunList Resource (Section 1 of 6)

Name	Data Type	Description
<i>ComponentGranularity</i> = "Document" New in JDF 1.2 Deprecated in JDF 1.4	enumeration	Specifies which grouping of input LayoutElement PDL pages define the equivalent of an individual output Component instance for processing in a multi-document print Job, e.g., in a variable data Job. For instance, all pages defined between end-of-set markers would be stitched in a Combined Process Node with <i>DigitalPrinting</i> and <i>Stitching</i> Processes if <i>ComponentGranularity</i> = "Set". Values are: <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 913 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. If <i>DocCopies</i> is > 1 for an automated imposition job, the imposition engine places the equivalent <i>DocCopies</i> Attribute into the RunList (<i>Surfaces</i>) Resource generated by the <i>Imposition</i> Process. An exception is cut-and-stack imposition, where <i>DocCopies</i> is applied by the imposition engine itself, and not placed into the RunList (<i>Surfaces</i>). 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>	NameRangeList	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 Resource.

Table 7-340: RunList Resource (Section 2 of 6)

Name	Data Type	Description
<i>EndOfBundleItem?</i> New in JDF 1.2	boolean	If " <i>true</i> ", the last page in the RunList is the last page of a BundleItem . The implied default value of <i>EndOfBundleItem</i> = " <i>false</i> ", except for the last RunList Partition, which always has an implied default value of <i>EndOfBundleItem</i> = " <i>true</i> ". Modification note: starting with JDF 1.4, this Attribute no longer depends on the deprecated <i>ComponentGranularity</i> .
<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 pre-separated files.
<i>IgnoreContext?</i> New in JDF 1.4	enumerations	Specifies the <i>PartIDKeys</i> values that do not affect the context in which this RunList is processed. Typically used when the ResourceLink is Partitioned to re-order a content RunList . For the keys specified in this list, processing the RunList MUST operate as if the identified parts represent the entire RunList . If Partition Keys are not specified, processing the RunList MUST operate as if the entire RunList Resource was processed, and all results removed except for those identified by the ResourceLink (e.g. for reprinting or recreating sheets with processing order-sensitive content - <i>SheetIndex</i> has whatever value it would have had if sheets were generated using the entire, original RunList). See example just below this table.
<i>IsPage = "true"</i>	boolean	If " <i>true</i> ", the individual RunList Resource defines one or more page slots, (e.g., for filling PlacedObject Elements). If " <i>false</i> ", the first parent Partitioned RunList Resource with <i>IsPage</i> = " <i>true</i> " defines the page level. In general, <i>IsPage</i> = " <i>false</i> " for separations of a pre-separated RunList .

Table 7-340: RunList Resource (Section 3 of 6)

Name	Data Type	Description
<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 Resource 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> = "true") 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.
<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 PageData 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.

Table 7-340: RunList Resource (Section 4 of 6)

Name	Data Type	Description
<i>Pages</i> ?	IntegerRangeList	Zero-based list of indices in the documents specified by the LayoutElement Resource 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 (i.e., "0 ~ -1") referred to by the RunList are selected. Modification note: before JDF 1.4, LayoutElement appeared in place of RunList in the preceding sentence.
<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. If <i>SetCopies</i> is > 1 for an automated imposition Job, the imposition engine places the equivalent <i>SetCopies</i> Attribute into the RunList (<i>Surfaces</i>) Resource generated by the Imposition Process. An exception is cut-and-stack imposition, where <i>SetCopies</i> is applied by the imposition engine itself, and not placed into the RunList (<i>Surfaces</i>). Note: 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 / <i>@ElementType</i> = "MultiSet".
<i>Sets</i> ? New in JDF 1.1	IntegerRangeList	Zero-based list of Document Set indices in a multi-Document Sets file specified by the LayoutElement Resource. If not present, all Document Sets are selected. <i>Sets</i> is only valid if LayoutElement / <i>@ElementType</i> = "MultiSet".

Table 7-340: RunList Resource (Section 5 of 6)

Name	Data Type	Description
SheetSides ? New in JDF 1.4	enumeration	Specifies the binding of surfaces referenced by this RunList to sheets. MUST only be specified in RunList (Surfaces) . Values are: <i>Front</i> – all surfaces referenced from a RunList leaf Partition describe one or more front sides of successive sheets, with implicit back blank sides. <i>Back</i> – all surfaces referenced from a RunList leaf Partition describe one or more back sides of successive sheets, with implicit front blank sides. <i>FrontBack</i> – all surfaces referenced from a RunList leaf Partition describe a succession of sheets, where for each sheet a front is followed by a back surface. <i>BackFront</i> – all surfaces referenced from a RunList leaf Partition describe a succession of sheets , where for each sheet a back is followed by a front surface.
SkipPage ?	integer	Used when the RunList comprises every Nth page of the file. <i>SkipPage</i> indicates the number of finished pages to be skipped between each of the pages that comprise the RunList Resource. This is generally used to describe pre-separated files, or to select only even or odd pages. Note that <i>SkipPage</i> 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 RunList is retained indefinitely. RunList/ LayoutElement/ FileSpec/ Disposition takes precedence over RunList/ Disposition . Modification note: starting with JDF 1.4, “this RunList ” above replaces “this FileSpec ”.
DynamicInput * Deprecated in JDF 1.4	element	Replacement text for a DynamicField Element. This information defines the contents of a dynamic mark on the automated page layout (see Section 7.2.109.10.1, 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 Name Attribute that, when linked to the <i>ReplaceField</i> Attribute of the DynamicField Element, defines the string that is to be replaced. Deprecation note: starting with JDF 1.4, metadata should be extracted from the PDL itself or from other sources, but not from the RunList . DynamicInput was designed to associates metadata with RunList Elements.

Table 7-340: RunList Resource (Section 6 of 6)

Name	Data Type	Description
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 Resource.
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.
MetadataMap * New in JDF 1.4	element	Describes the mapping of Metadata in a RunList to <i>PartIDKeys</i>
PageList ?	refelement	Specification of page metadata for pages described by this RunList .

Example 7-48: Marks and Reordering of Content using RunList/@IgnoreContext[New in JDF 1.4](#)

Assume that a VDP job consists of sets where each set contains a Cover Letter, Brochure, and Postcard document types. Production needs all of each document type for all sets printed first, and the imposition includes dynamic marks where some of the marking uses *SheetIndex*. The **RunListLink** parameterizes the processing such that all Cover Letter sheets for all sets are processed first, followed by the Brochure sheets for all sets, and finally, the Postcard sheets for all sets. The **RunList** then specifies **@IgnoreContext = "SheetIndex"**, which forces the *SheetIndex* to be calculated in the order in which sheets are produced by the processing of the reordered "virtual" **RunList**.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <RunList Class="Parameter" ID="MyVDPRunList" Status="Available"
      PartIDKeys="DocTags" IgnoreContext="SheetIndex" >
      <!-- additional attributes and elements -->
      <RunList DocTags="CoverLetter" />
      <RunList DocTags="Brochure" />
      <RunList DocTags="Postcard" />
    </RunList>
  </ResourcePool>
  <ResourceLinkPool>
    <RunListLink Usage="Input" rRef="MyVDPRunList" >
      <Part DocTags="CoverLetter" />
      <Part DocTags="Brochure" />
      <Part DocTags="Postcard" />
    </RunListLink>
  </ResourceLinkPool>
</JDF>
```

To enable later reprinting of part of the **RunList**, the **RunList** then might also specify a **MetadataMap** Element that extracts the value of a RecordNumber metadata key and assigns the value to *Metadata0*. Subsequently, if record # 12 needs reprinting, the **RunListLink** can be modified to appear as:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
```

```

<RunList Class="Parameter" ID="MyVDPRunList" Status="Available"
  PartIDKeys="DocTags" IgnoreContext="SheetIndex" >
  <!-- additional attributes and elements -->
  <RunList DocTags="CoverLetter"/>
  <RunList DocTags="Brochure"/>
  <RunList DocTags="Postcard"/>
</RunList>
</ResourcePool>
<ResourceLinkPool>
  <RunListLink Usage="Input" rRef="MyVDPRunList" ProcessUsage="Document">
    <Part DocTags="CoverLetter" Metadata0="12"/>
    <Part DocTags="Brochure" Metadata0="12"/>
    <Part DocTags="Postcard" Metadata0="12"/>
  </RunListLink>
</ResourceLinkPool>
</JDF>

```

7.2.160.1 Element: DynamicInput

[Deprecated in JDF 1.4](#)

Table 7-341: DynamicInput Element

Name	Data Type	Description
<i>Name</i> ?	string	Label that MUST match the <i>ReplaceField</i> Attribute of the appropriate <i>DynamicField</i> Element
—	text	Defines the text string that is to be inserted as a replacement for the text defined in <i>ReplaceField</i> of a <i>DynamicField</i> Element.

7.2.160.2 Element: MetadataMap

[New in JDF 1.4](#)

The *MetadataMap* in *RunList* allows metadata embedded in PDL files to be assigned to Partition Key values, certain *RunList* Attributes, or Attributes created using *GeneralID*. During the mapping of PDL data to the JDF document structure (see the definition in the glossary or the discussion in the *Imposition* Process), each *MetadataMap* Element will be evaluated for each node (Set, Document, Page, etc.) of the PDL document structure. For XML based PDL files an XPath expression will be evaluated relative to the XML node that defines each node in the document hierarchy. For non-XML based PDLs a PDL specific mapping of the XPath to the PDL document structure is used instead and the value assignment is performed on the derived XML for the PDL file. If the path specified by the XPath does not exist in the PDL, then the associated metadata value is undefined, otherwise the metadata value will be set to the conversion of the node list to a string.

Table 7-342: MetadataMap Element (Section 1 of 3)

Name	Data Type	Description
<i>Context</i> = "PagePool"	enumeration	<p>Specifies the node context in which the XPaths specified in this MetadataMap Element are to be evaluated.</p> <p>Values are:</p> <p><i>Set</i> – evaluated relative to the current set node.</p> <p><i>Document</i> – evaluated relative to the current document node.</p> <p><i>SubDoc0</i> – evaluated relative to the current subdocument immediately below the Document level.</p> <p><i>SubDoc1</i> – evaluated relative to the current subdocument immediately below <i>SubDoc0</i> level.</p> <p><i>SubDoc2</i> – see <i>SubDoc1</i>, but relative to <i>SubDoc1</i> level.</p> <p><i>SubDoc3</i> – see <i>SubDoc1</i>, but relative to <i>SubDoc2</i> level.</p> <p><i>SubDoc4</i> – see <i>SubDoc1</i>, but relative to <i>SubDoc3</i> level.</p> <p><i>SubDoc5</i> – see <i>SubDoc1</i>, but relative to <i>SubDoc4</i> level.</p> <p><i>SubDoc6</i> – see <i>SubDoc1</i>, but relative to <i>SubDoc5</i> level.</p> <p><i>SubDoc7</i> – see <i>SubDoc1</i>, but relative to <i>SubDoc6</i> level.</p> <p><i>SubDoc8</i> – see <i>SubDoc1</i>, but relative to <i>SubDoc7</i> level.</p> <p><i>SubDoc9</i> – see <i>SubDoc1</i>, but relative to <i>SubDoc8</i> level.</p> <p><i>Object</i> – evaluated for each unique object on each page.</p> <p><i>PagePool</i> – evaluated relative to the current Page Pool.</p> <p><i>Page</i> – evaluated relative to the current page.</p>
<i>DataType</i>	enumeration	<p>Expected data type of the metadata value.</p> <p>Values are:</p> <p><i>PartIDKeys</i> – with this value, <i>@Name</i> MUST match a Partition Key.</p> <p>Values are also from: <i>GeneralID/@DataType</i>.</p>

Table 7-342: MetadataMap Element (Section 2 of 3)

Name	Data Type	Description
<i>Name</i>	NMTOKEN	<p>The name of the metadata.</p> <p>If <i>DataType</i> = "PartIDKeys", the value of <i>Name</i> MUST be a <i>PartIDKeys</i> value. See @PartIDKeys in Table 3-27, "Partitionable Resource Element" on page 103.</p> <p>If <i>Name</i> = "ObjectTags", then values are added to a logical pool of tag values associated with each object being processed. This pool of object tags is referenced from: ColorSpaceConversionParams/ColorSpaceConversionOp/@ObjectTags, ScreeningParams/ScreenSelector/@ObjectTags, ObjectResolution/@ObjectTags, ColorCorrectionParams/ColorCorrectionOp/@ObjectTags.</p> <p>Otherwise, <i>Name</i> specifies the value of an implied variable (e.g. for use in GeneralID/@IDUsage, RunList/@EndOfSet, RunList/@SetCopies, RunList/@PageCopies, or RunList/@DocCopies).</p> <p>If @<i>DataType</i> is not "PartIDKeys" or a RunList implied variable name (e.g. RunList/@DocCopies), then the MetadataMap Element is equivalent to explicitly defining a GeneralID Element with the value being assigned by MetadataMap/@ValueFormat. The following example counts the number of Page Elements within all DocPart Elements.</p> <pre><MetadataMap DataType="Integer" Name="NumPages" ValueFormat="%d" ValueTemplate="npages"> <Expr Name="ncopies" Path="count(.. /DocPart/Page)" > </MetadataMap></pre> <p>If multiple MetadataMap Elements specify the same name, then the specified key has the value from the last MetadataMap Element to assign a value to that key.</p> <p>If the specified <i>Name</i> sets the value for a <i>PartIDKeys</i> or RunList variable, where a RunList Attribute also supplies a value (e.g. RunList/@RunTag, RunList/@DocCopies, ...), the value supplied by the RunList Attribute shall be replaced by the value supplied by the MetadataMap.</p>
<i>ValueFormat</i>	string	<p>Formatting value for combining extracted values from the Expr Elements.</p> <p>Values are from: "Generating strings with Format and Template" on page 909.</p>
<i>ValueTemplate</i>	string	<p>Arguments for combining extracted values from the Expr Elements. The argument names MUST match the values of Expr/@<i>Name</i>.</p> <p>Values are from: "Generating strings with Format and Template" on page 909.</p>

Table 7-342: MetadataMap Element (Section 3 of 3)

Name	Data Type	Description
Expr +	element	Each Expr Element describes a Term expression (see Section 7.3.13, "Term" on page 807. See also Section 7.3, "Device Capability Definitions" on page 776.) evaluating metadata values in the PDL. If Expr/Term is not specified, or if the Term expression returns true, then the value specified by the Expr element is assigned to the key specified by MetadataMap/@Name. Expr Elements are evaluated in the XML order specified. Expr Elements with identical @Name Attributes where a previous Expr Element with that @Name has already evaluated to true MUST NOT be processed. If any name specified in MetadataMap/@ValueTemplate is unassigned, then the key specified by MetadataMap/@Name is undefined.

7.2.160.3 Element: Expr[New in JDF 1.4](#)**Table 7-343: Expr Element**

Name	Data Type	Description
Name	NMTOKEN	Name of this Expr. The value (as specified by @Value or extracted from @Path) MUST be used to evaluate the parent @ValueTemplate.
Path ?	XPath	If specified, and either the value returned by the Term Element (if present) is true or no Term Element is specified, then the value specified by this path is assigned to Expr/@Name. If the Path does not evaluate to a value, then this Expr Element fails and any subsequent Expr Elements are evaluated. Constraint: exactly one of Path or Value MUST be specified in an Expr Element.
Value ?	string	If specified, and either the value returned by the Term Element (if present) is true or no Term Element is specified, then the value of this Attribute is assigned to Expr/@Name. Constraint: exactly one of Path or Value MUST be specified in an Expr element.
Term ?	element	Evaluates one or more metadata values from the PDL, and returns a true or false result. Evaluation/@XPath MUST be specified for all Evaluation Elements in the Term hierarchy.

For PPML the XPath expression will be relative to the JOB, DOCUMENT or PAGE element. Example XPath expressions:

- "METADATA/DATUM[@key='Gender']" will extract the value of the Gender metadata for each JDF set, document and page.
- "count(PAGE)" will count the pages within a given document (only works for JDF document level Nodes).
- "count(PAGE/METADATA/DATUM[@key='special'])" will count the number of pages that have a Special metadata defined for it.

MetadataMap may also be used to set the value of certain RunList Attributes. These Attributes are EndOfSet, EndOfDocument, PageCopies, DocCopies and SetCopies. The values set will be instantiated as if actually present in a Partitioned RunList for the current page or Page Pool being processed. Care should be taken to ensure their consistency across Page Pools within a document or set.

Example 7-49: MetadataMap: Setting Attributes

This example extracts the value of the *Copies* Attribute as specified by the *Path*, and sets the value of *RunList/@DocCopies*.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <RunList Class="Parameter" ID="r000004" Status="Available">
      <MetadataMap DataType="Integer" Name="DocCopies" ValueFormat="%d"
        ValueTemplate="ncopies">
        <Expr Name="ncopies" Path="//record/document/@Copies"/>
        <Expr Name="ncopies" Value="1"/>
      </MetadataMap>
    </RunList>
  </ResourcePool>
  <ResourceLinkPool>
    <RunListLink Usage="Input" rRef="r000004"/>
  </ResourceLinkPool>
</JDF>
```

Example 7-50: RunList/MetadataMap

[New in JDF 1.4](#)

In the following example, the *MetadataMap* Element maps arbitrary tags in the document to a structural *RunTag* Partition Key. Note that any Partition Key may be mapped. Note also that although an XPath syntax is used, this may be mapped to any hierarchical structure including but not limited to XML. Finally, note that if */Dokument/@Sektion* is a value other than *"Einband"* or *"HauptTeil"*, then the *Expr* Elements assigning values to section will all fail, resulting in *RunTags* being undefined.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">

  <ResourcePool>
    <!--this runlist points to a structured pdl with arbitrary structural
      tagging-->
    <RunList Class="Parameter" ID="r000004" Status="Available">
      <MetadataMap DataType="PartIDKey" Name="RunTags"
        ValueFormat="%s%s" ValueTemplate="sex,section">
        <!--This expression maps the value of /Dokument/Rezipient/@Sex
          to a variable "sex"-->
        <Expr Name="sex" Path="/Dokument/Rezipient/@Sex"/>
        <!--Maps all elements with /Dokument/@Sektion=Einband to Cover-->
        <Expr Name="section" Value="Cover">
          <NameEvaluation Path="/Dokument/@Sektion" RegExp="Einband"/>
        </Expr>
        <!--Maps all elements with /Dokument/@Sektion=HauptTeil and >50 pages
          to BigBody-->
        <Expr Name="section" Value="BigBody">
          <and>
            <NameEvaluation Path="/Dokument/@Sektion" RegExp="HauptTeil"/>
            <IntegerEvaluation Path="count(PAGE)" ValueList="51 ~ INF"/>
          </and>
        </Expr>
        <!--Maps all elements with /Dokument/Sektion=HauptTeil and <=50 pages
          to SmallBody-->
        <Expr Name="section" Value="SmallBody">
          <and>
            <NameEvaluation Path="/Dokument/Sektion" RegExp="HauptTeil"/>
            <IntegerEvaluation Path="count(PAGE)" ValueList="0 ~ 50"/>
          </and>
        </Expr>
      </MetadataMap>
    <LayoutElement Class="Parameter">
```

```

        <FileSpec Class="Parameter"
            MimeType="application/vnd.foobar+xml" URL="bigVariable.foo"/>
    </LayoutElement>
</RunList>
<!--Layout for versioned product-->
<Layout Class="Parameter" ID="r000005" PartIDKeys="RunTags" Status="Available">
    <Layout RunTags="MaleCover">
        <MediaRef rRef="r000006">
            <Part RunTags="MaleCover"/>
        </MediaRef>
    </Layout>
    <Layout RunTags="FemaleCover">
        <MediaRef rRef="r000006">
            <Part RunTags="FemaleCover"/>
        </MediaRef>
    </Layout>
    <Layout RunTags="MaleBigBody FemaleBigBody">
        <MediaRef rRef="r000006">
            <Part RunTags="MaleBigBody MaleSmallBody FemaleBigBody
                FemaleSmallBody"/>
        </MediaRef>
    </Layout>
    <Layout RunTags="MaleSmallBody FemaleSmallBody">
        <MediaRef rRef="r000006">
            <Part RunTags="MaleBigBody MaleSmallBody FemaleBigBody
                FemaleSmallBody"/>
        </MediaRef>
    </Layout>
</Layout>
<Media Class="Consumable" ID="r000006" PartIDKeys="RunTags"
    PartUsage="Implicit" Status="Available">
    <Media RunTags="MaleCover"/>
    <Media RunTags="FemaleCover"/>
    <Media RunTags="MaleBigBody MaleSmallBody FemaleBigBody FemaleSmallBody"/>
</Media>
</ResourcePool>
<ResourceLinkPool>
    <RunListLink Usage="Input" rRef="r000004"/>
    <LayoutLink Usage="Input" rRef="r000005"/>
    <MediaLink Usage="Input" rRef="r000006"/>
</ResourceLinkPool>
</JDF>

```

Example 7-51: RunList: Unstructured Single-File RunList

The following five examples illustrate how a **RunList** can be structured using Partitioning mechanisms. Note that the Partitioning of a **RunList** often generates the values necessary to evaluate the Partitioning of other Resources, (e.g., the *RunIndex* into the **RunList**). Thus, the order in which the **RunList** Elements appear in the XML document is significant. Note that the *Run* Partition Key has a string value, which MAY be non-numeric. Below is an example of simple unstructured single-file **RunList**. This example specifies all pages contained in “/in/colortest.pdf”.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
    Type="ProcessGroup" JobPartID="ID300" Version="1.4">
    <ResourcePool>
        <RunList Class="Parameter" ID="Link0003" Pages="0 ~ -1" Status="Available">
            <LayoutElement>
                <FileSpec URL="File:///in/colortest.pdf"/>
            </LayoutElement>
        </RunList>
    </ResourcePool>
    <ResourceLinkPool>
        <RunListLink Usage="Input" rRef="Link0003"/>
    </ResourceLinkPool>
</JDF>

```

```

    </ResourceLinkPool>
  </JDF>

```

Example 7-52: RunList: Multi-File Unseparated RunList

Example of simple multi-file unseparated **RunList** using **RunList/@Directory**. This example specifies all pages contained in File1.pdf and File2.pdf, which are located in the directory “///Dir/” that is specified in **RunList/@Directory**.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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>
  </ResourcePool>
  <ResourceLinkPool>
    <RunListLink Usage="Input" rRef="Link0003" />
  </ResourceLinkPool>
</JDF>

```

Example 7-53: RunList: Multi-File Unseparated RunList with Spawning

Example of simple multi-file unseparated **RunList** with independent spawning. This example specifies the first five pages contained in File1.pdf and File2.pdf. File2.pdf has been spawned and is being processed individually.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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>
  </ResourcePool>
  <ResourceLinkPool>
    <RunListLink Usage="Input" rRef="Link0003" />
  </ResourceLinkPool>
</JDF>

```

Example 7-54: RunList: Spawned RunList

This is the corresponding spawned **RunList**. Note the *LogicalPage* Attribute, which specifies the number of skipped pages.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">

```

```

<ResourcePool>
  <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>
</ResourcePool>
<ResourceLinkPool>
  <RunListLink Usage="Input" rRef="Link0003"/>
</ResourceLinkPool>
</JDF>

```

Example 7-55: RunList: 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.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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>
  </ResourcePool>
  <ResourceLinkPool>
    <RunListLink Usage="Input" rRef="Link0003"/>
  </ResourceLinkPool>
</JDF>

```

7.2.161 SaddleStitchingParams

[Deprecated in JDF 1.1](#)

See "SaddleStitchingParams" on page 1042 for details of this deprecated Resource.

7.2.162 ScanParams

This Resource provides the parameters for the *Scanning* Process.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	ArtDeliveryIntent/ArtDelivery
Example Partition:	<i>RunIndex</i>
Input of Processes:	<i>Scanning</i>
Output of Processes:	—

Table 7-344: ScanParams Resource (Section 1 of 2)

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. Values are: <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 = "1 1"</i>	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. 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.

Table 7-344: ScanParams Resource (Section 2 of 2)

Name	Data Type	Description
<i>OutputColorSpace</i>	enumeration	Color space of the output images. 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>)?	reference	A FileSpec Resource pointing to an ICC profile that describes color corrections.
FileSpec (<i>TargetProfile</i>)?	reference	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>) ?	reference	A FileSpec Resource pointing to an ICC profile that describes the scanner.

7.2.163 ScavengerArea

[New in JDF 1.1](#)

This Resource describes a scavenger area for removing excess ink from printed Sheets. It is defined within a MarkObject of a surface.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	Layout/MarkObject
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-345: 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.

7.2.164 ScreeningParams

This Resource specifies the parameter of the *Screening* Process. Since screening is, in most cases, very OEM specific, the following parameters are generic enough that they can be mapped onto a number of OEM controls.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	ContactCopyParams, ContentList/ContentData, ExposedMedia, LayoutElement, PageList, PageList/PageData
Example Partition:	Separation, SheetName, Side, SignatureName
Input of Processes:	ContoneCalibration, Screening
Output of Processes:	—

Table 7-346: ScreeningParams Resource

Name	Data Type	Description
<i>IgnoreSourceFile</i> = "true"	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 *	element	List of screen selectors. A screen selector is included for each separation, including a default specification. ScreenSelector MUST contain the complete set of Parameters for a given screening operation. For instance, it is invalid to specify one ScreenSelector for a given @ObjectTags and another ScreenSelector for a given @SourceObjects.

7.2.164.1 Element: ScreenSelector

Description of screening for a selection of source object types and separations.

Table 7-347: ScreenSelector Element (Section 1 of 3)

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> .

Table 7-347: ScreenSelector Element (Section 2 of 3)

Name	Data Type	Description
<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, it specifies the angle of the separation defined by <i>AngleMap</i> prior to that separation being mapped. Note that, in general, the known default will be a CMYK process color, but it can also be another process color, (e.g., HexaChrome™). The following example specifies that <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>ObjectTags</i> ? New in JDF 1.4	NMTOKENS	Tags associated with individual objects that this <i>ScreenSelector</i> MUST be applied to. Each tag specified in <i>ObjectTags</i> is logically anded with the object type(s) specified by <i>SourceObjects</i> , enabling first qualification by object type (such as image), and then tags associated with those objects. The values of <i>ObjectTags</i> depends on the PDL that the <i>ScreenSelector</i> is applied to.
<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. 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. Values include: <i>All</i>

Table 7-347: ScreenSelector Element (Section 3 of 3)

Name	Data Type	Description
<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> .
<i>SourceObjects = "All"</i>	enumerations	Identifies the class(es) of incoming graphical objects on which to use the selected screen. 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>
<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. 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>

7.2.165 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:	<i>Separation</i>
Output of Processes:	—

Table 7-348: 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.

7.2.166 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:	ColorIntent/ColorsUsed, NumberingIntent/NumberItem, ProofingIntent/ProofItem, ColorantAlias, ColorantControl/ColorantConvertProcess, ColorantControl/ColorantOrder, ColorantControl/ColorantParams, ColorantControl/DeviceColorantOrder, ColorantControl/ColorSpaceSubstitute, ColorControlStrip, ColorSpaceConversionOp, ContentList/ContentData, DeviceNSpace, LayoutElement, PageList, PageList/PageData, RegisterMark, ScavengerArea, TrappingDetails/TrappingOrder
Example Partition:	—
Input of Processes:	—
Output of Processes:	—

Table 7-349: SeparationSpec Resource

Name	Data Type	Description
<i>Name ?</i>	string	Name of one specific separation. If <i>Name</i> is not specified, this SeparationSpec consumes a slot in a separation order without setting a separation, for instance when specifying modules to skip on a press or color fields to leave blank in a ColorControlStrip . Modification note: starting with JDF 1.4, <i>Name</i> is optional.

7.2.167 Shape**Resource Properties**

Resource Class:	Parameter
Resource referenced by:	ShapeCuttingParams, ShapeDef
Example Partition:	—
Input of Processes:	—

Output of Processes: —

Table 7-350: Shape Resource

Name	Data Type	Description
<i>CutBox</i> ?	rectangle	Specification of a rectangular window.
<i>CutOut</i> = "false"	boolean	If "true", the inside of a specified shape will be removed. If "false", the outside of a specified shape will be removed. An example of an inside shape is a window. An example of an outside shape is a shaped greeting card.
<i>CutPath</i> ?	PDFPath	Specification of a complex path. This MAY be an open path in the case of a single line.
<i>CutType</i> = "Cut" Deprecated in JDF 1.4	enumeration	Type of cut or perforation used. Values are: <i>Cut</i> – Full cut. <i>Perforate</i> – Interrupted perforation that does not span the entire Sheet
<i>DDESCutType</i> = "101" New in JDF 1.4	integer	Type of cut or perforation used. Values include: a number between "0" and "999" corresponding to a line type as defined in DDES3 standard (ANSI® IT8.6-2002). Note: the default value 101 corresponds to a cut line.
<i>Material</i> ?	string	Transparent material that fills a shape (e.g., an envelope window) that was cut out when <i>CutOut</i> = "true".
<i>ShapeDepth</i> ?	double	Depth of the shape cut, measured in microns [µm]. If not specified, the shape is completely cut.
<i>ShapeType</i>	enumeration	Describes any precision cutting other than hole making. Values are: <i>Path</i> <i>Rectangular</i> <i>Round</i> <i>RoundedRectangle</i> – Rectangle with rounded corners. New in JDF 1.3
<i>StationName</i> ? New in JDF 1.3 Deprecated in JDF 1.4	string	The name of the 1-up design in the die layout. Used to match DieLayout /Station Elements with Shape Elements.
<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 (n > 1000).

7.2.168 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 7-351: ShapeCuttingParams Resource

Name	Data Type	Description
<i>DeliveryMode</i> ? New in JDF 1.3	enumeration	Values are: <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.
<i>SheetLay</i> ? New in JDF 1.3	enumeration	Lay of input media. Reference edge of where paper is placed in the feeder. Values are: <i>Center</i> <i>Left</i> <i>Right</i>
<i>DieLayout</i> ? New in JDF 1.3	reference	A Resource containing the reference of an external file describing the cutting and other paths.
<i>Shape</i> *	reference	Details of each individual cut shape

7.2.169 ShapeDef

[New in JDF 1.4](#)

A structural design describing a 2D surface with paths that describe different finishing operations like cutting, creasing, perforation, etc. In the case of box production this resource is a description of the unprinted blank box as it will be available after die cutting and blanking and before folding. A **ShapeDef** is defined either by an external file (**FileSpec**) describing the structural design or a collection of PDFPaths contained in Shape elements. In case this description is stored in a file, the format of this file may be a vendor specific format, a standard DDES3 file (ANSI® IT8.6-2002), or less well specified but commonly used formats like CFF2 or DXF or even a PDF or EPS file.

Resource Properties

Resource Class: Parameter
Resource referenced by: , **DieLayout/Station**, **LayoutElementProductionParams**
Example Partition: —
Input of Processes: *DieLayoutProduction*
Output of Processes: *ShapeDefProduction*

Table 7-352: ShapeDef Resource (Section 1 of 2)

Name	Data Type	Description
<i>Area</i> ?	double	The net area of the shape after cutting. (m2)
<i>CutBox</i> ?	rectangle	A rectangle describing the bounding box of all cut lines. This is sometimes referred to as the knife to knife dimensions of the BlankBox. This Attribute is usually only valid after the generation of the structural design.
<i>Dimensions</i> ?	shape	Width <i>x</i> , height <i>y</i> and depth <i>z</i> coordinates of the 3D shape. For a box, these are the outer dimensions e.g. for palletizing.

Table 7-352: ShapeDef Resource (Section 2 of 2)

Name	Data Type	Description
<i>FluteDirection</i> ?	enumeration	Intended direction of the flute for this design in the coordinate system defined by <i>CutBox</i> . This information needs to be taken into account by the <i>DieLayoutProduction</i> Process to give the <i>ShapeDef</i> the correct orientation on the <i>Media</i> . Values are: <i>XDirection</i> – Along the X-axis of the <i>CutBox</i> coordinate system. <i>YDirection</i> – Along the Y-axis of the <i>CutBox</i> coordinate system.
<i>GrainDirection</i> ?	enumeration	Intended direction of the grain for this design in the coordinate system defined by <i>CutBox</i> . This information needs to be taken into account by the <i>DieLayoutProduction</i> Process to give the <i>ShapeDef</i> the correct orientation on the <i>Media</i> . Values are: <i>XDirection</i> – Along the X-axis of the <i>CutBox</i> coordinate system. <i>YDirection</i> – Along the Y-axis of the <i>CutBox</i> coordinate system. <i>Both</i> – Both orientations are acceptable.
<i>MediaSide</i> ?	enumeration	Determines the printing side for which the structural design is made. Values are: <i>Front</i> – for a box this corresponds to the outside of a box. <i>Back</i> - for a box this corresponds to the inside of a box. Note: folding carton is usually cut from the outside (Front), corrugated from the inside (Back).
<i>ResourceWeight</i> ?	double	The weight of the shape after cutting (g).
<i>FileSpec</i> ?	refelement	The <i>FileSpec</i> of the structural design file. The format of this file may be a vendor specific format, a standard DDES3 file (ANSI® IT8.6-2002), less well specified but commonly used formats like CFF2 or DXF or even a PDF or EPS file. <i>FileSpec</i> and <i>Shape</i> are mutually exclusive
<i>Media</i> ?	refelement	<i>Media</i> for which this structural design was intended for. The <i>Media</i> description defines important design parameters as the type of <i>Media</i> , thickness, inside loss, outside gain, etc. <i>Media/@GrainDirection</i> and <i>Media/@FluteDirection</i> do not have any significance.
<i>Shape</i> *	refelement	The shape is defined by a collection of <i>Shape</i> Elements. <i>Shape</i> and <i>FileSpec</i> are mutually exclusive.

7.2.170 ShapeDefProductionParams

[New in JDF 1.4](#)

Parameters for the structural design.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>ShapeDefProduction</i>
Output of Processes:	—

Table 7-353: ShapeDefProductionParams Resource

Name	Data Type	Description
ObjectModel *	element	A 3D model of the object that needs to be packed.
ShapeTemplate ?	element	A structural template sometimes called a parametric structural design. Given a set of parametric values a structural template can be instantiated to an actual structural design.

7.2.170.1 Element: ObjectModel

[New in JDF 1.4!](#)

Table 7-354: ObjectModel Element

Name	Data Type	Description
<i>Dimensions ?</i>	shape	Width <i>x</i> , height <i>y</i> and depth <i>z</i> values for the bounding box of the object.
FileSpec ?	refelement	The FileSpec of the 3D model of the objects that needs to be packed. The format of this file may be a vendor specific format or a standard 3D format like VRML or PDF (U3D).

7.2.170.2 Element: ShapeTemplate

[New in JDF 1.4](#)

Additional parametric values **MUST** be specified with GeneralID Elements. GeneralID/@IDUsage **MUST** be set to the name of the Parameter. GeneralID/@DataType **MUST** be set to "double". GeneralID/@IDValue **MUST** be set to value of the Parameter.

Table 7-355: ShapeTemplate Element

Name	Data Type	Description
<i>InnerDimensions ?</i>	shape	Width <i>x</i> , height <i>y</i> and depth <i>z</i> coordinates of the 3D shape. For a box these are the inner dimensions.
<i>Name ?</i>	string	The name of a parametric structural design or CAD template.
<i>Standard ?</i>	string	The name of the standard this template belongs to e.g. FEFCO, ECMA or the name of a company internal standard.
FileSpec ?	refelement	The FileSpec of the parametric structural design.

The three Figures below show shapes specified by a ShapeTemplate with each named variable represented by a GeneralID that specifies the name and value of the variable. The ShapeTemplate for the diagram below might be:

Example 7-56: ShapeTemplate for Figure 7-50

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <ShapeDefProductionParams Class="Parameter" ID="Link0003"
      Status="Available">
      <ShapeTemplate>
        <GeneralID IDUsage="L" DataType="double" IDValue="1440.0"/>
        <GeneralID IDUsage="W" DataType="double" IDValue="720.0"/>
        <GeneralID IDUsage="D" DataType="double" IDValue="1440.0"/>
      </ShapeTemplate>
    </ShapeDefProductionParams>
  </ResourcePool>
  <ResourceLinkPool>
    <ShapeDefProductionParamsLink Usage="Input" rRef="Link0003"/>
  </ResourceLinkPool>
</JDF>
```

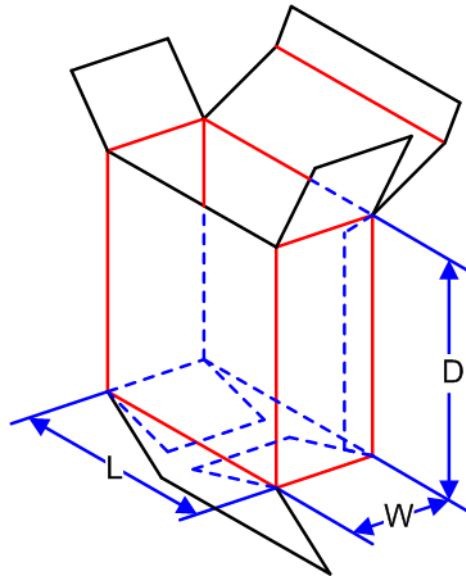



Figure 7-50: ShapeTemplate Example 1

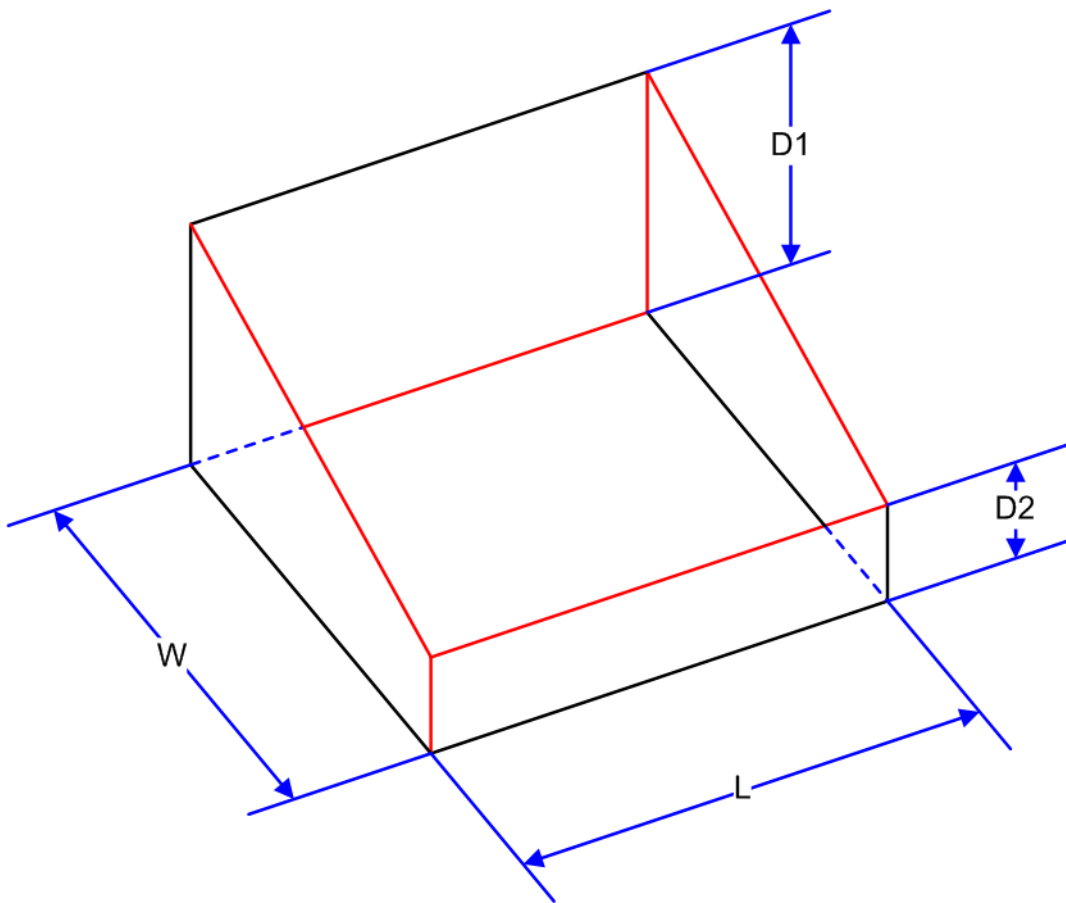


Figure 7-51: ShapeTemplate Example 2

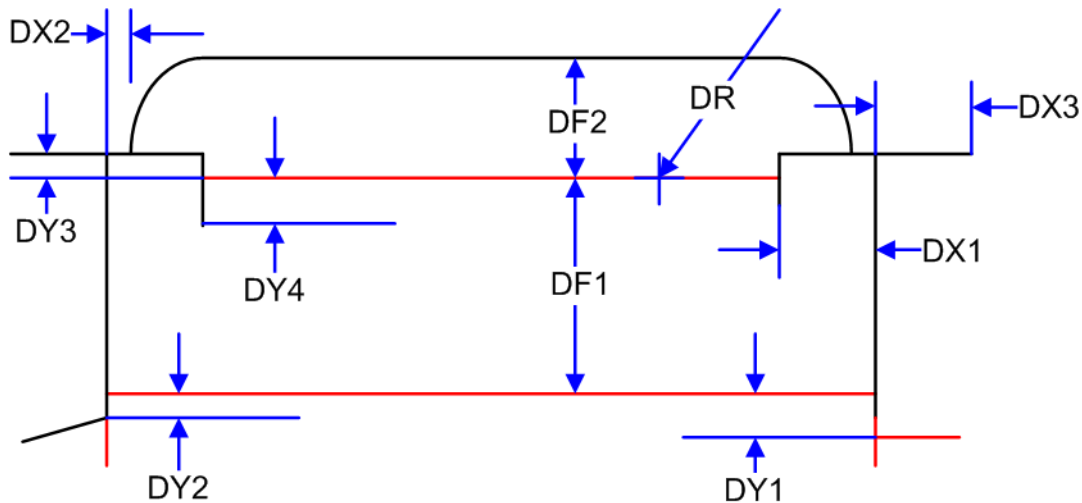


Figure 7-52: ShapeTemplate Example 3

7.2.171 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 592.

7.2.172 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 7-356: ShrinkingParams Resource

Name	Data Type	Description
<i>Duration?</i>	duration	Shrinking time.
<i>ShrinkingMethod="ShrinkHot"</i>	enumeration	Specifics of the shrinking method for shrink wrapping. Values are: <i>ShrinkCool</i> <i>ShrinkHot</i>
<i>Temperature?</i>	double	Oven temperature in ° Centigrade.

7.2.173 SideSewingParams

[Deprecated in JDF 1.1](#)

See "SideSewingParams" on page 1044 for details of this deprecated Resource.

7.2.174 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 7-357: SpinePreparationParams Resource

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 701 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. Values include those from: Table 7-358, "Operations Attribute Values"
<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 701 for details.
<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 7-358: Operations Attribute Values (Section 1 of 2)

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.

Table 7-358: Operations Attribute Values (Section 2 of 2)

Value	Description
<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.

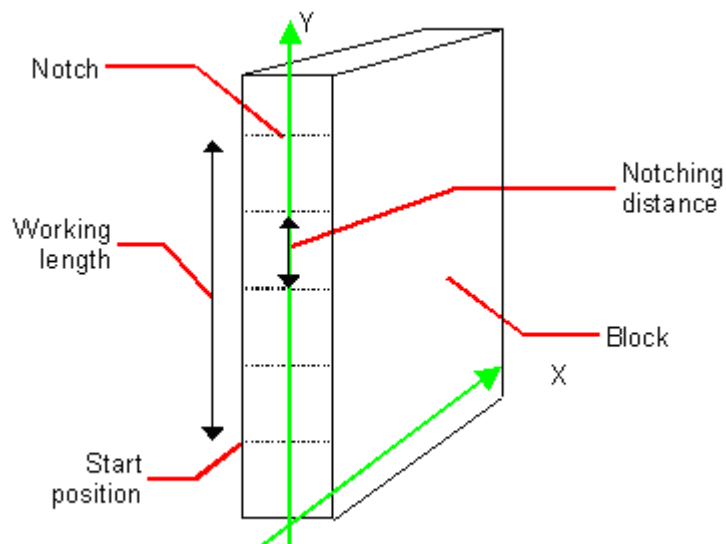


Figure 7-53: Parameters and coordinate systems for the SpinePreparation Process

7.2.175 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:	<i>SpineTaping</i>
Output of Processes:	—

Table 7-359: SpineTapingParams Resource (Section 1 of 2)

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>HorizontalExcessBack?</i> New in JDF 1.4	double	Horizontal excess of back if tape is not centered
<i>StripBrand?</i>	string	Strip brand.
<i>StripColor?</i>	NamedColor	Color of the strip.
<i>StripColorDetails?</i> New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If <i>StripColorDetails</i> is supplied, <i>StripColor</i> SHOULD also be supplied.

Table 7-359: SpineTapingParams Resource (Section 2 of 2)

Name	Data Type	Description
<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. 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.

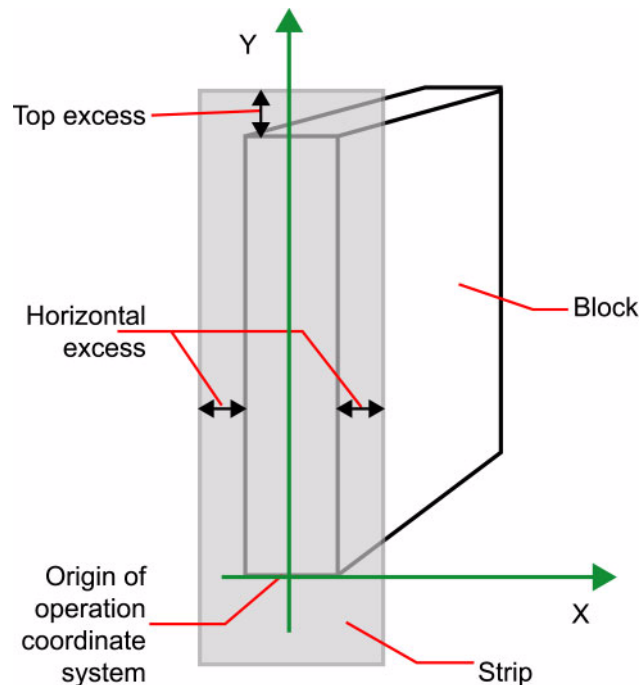


Figure 7-54: Parameters and coordinate system for the SpineTaping Process

7.2.176 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 7-55).

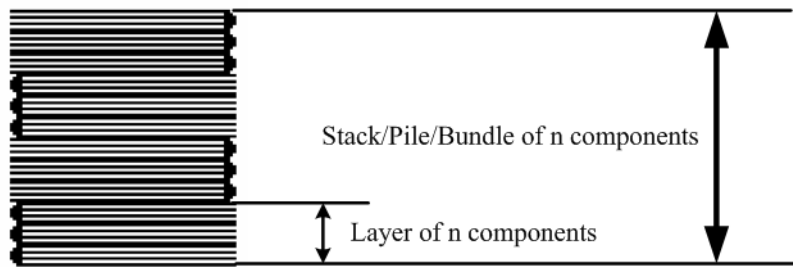


Figure 7-55: Stacking Layers

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 7-56).

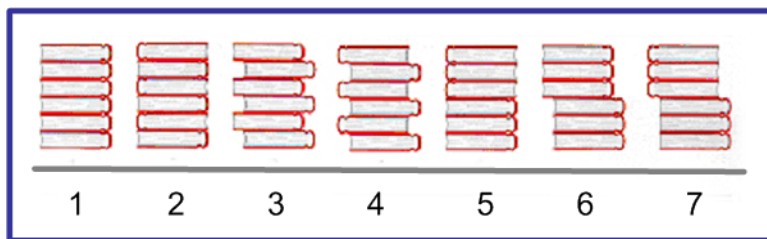


Figure 7-56: Pile Patterns

Table 7-360: Parameters in Stacking

Pile Pattern	StandardAmount	LayerAmount (Default = StandardAmount)	Compensate (Default = true)	Disjointing/ @Offset
1	6	6	true	0 0
2	6	1	true	0 0
3	6	1	false	x 0
4	6	1	true	x 0
5	6	3	true	0 0
6	6	3	false	x 0
7	6	3	true	x 0

If the number of components is not evenly divisible by standard stack size (*StandardAmount*) or the number of components in a bundle is not evenly divisible by layer size (*LayerAmount*), there will be a remainder, yielding one or more odd-count stacks or layers. By default, the odd-count stack or layer size can contain as few as one component. This might exceed equipment cycle times, and flimsy components (newspapers) might cause problems with downstream equipment such as strappers. The *MinAmount* and *MaxAmount* control the minimum and maximum size of odd-count stacks and layers. The following figures show the odd count handling for bundles and layers.

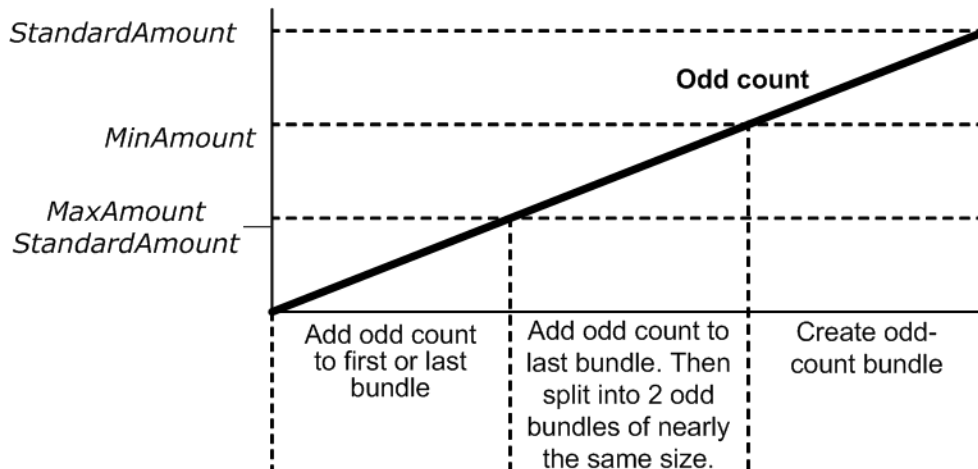


Figure 7-57: Odd count handling for a Bundle

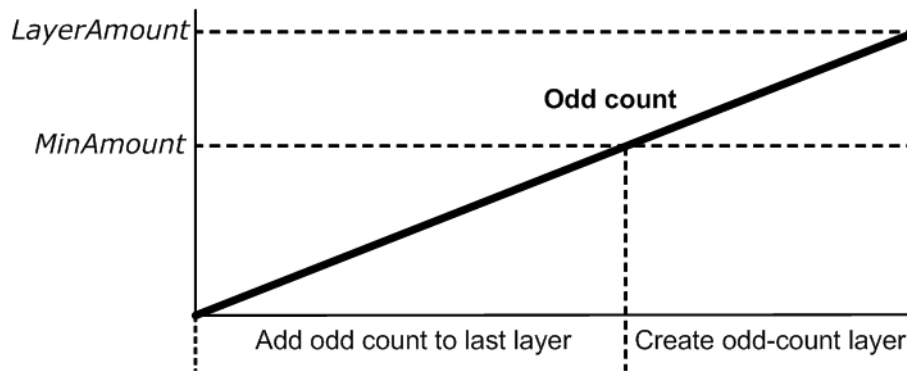


Figure 7-58: Odd count handling for a Layer

Resource Properties

Resource Class: Parameter
 Resource referenced by: —
 Example Partition: —
 Input of Processes: *Stacking*
 Output of Processes: —

Table 7-361: StackingParams Resource (Section 1 of 2)

Name	Data Type	Description
<i>BundleDepth</i> = "0" New in JDF 1.4	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 to allow a level of de-stacking. 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>BundleDepth</i> = "0" addresses the entire Component , <i>BundleDepth</i> = "1" addresses the bundle in the Component and so on.

Table 7-361: StackingParams Resource (Section 2 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>LayerLift</i> ? New in JDF 1.4	boolean	If true layer is lifted to reduce height.
<i>LayerCompression</i> ? New in JDF 1.4	boolean	If true layer is compressed before next layer is started.
<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> .
<i>MinAmount</i> ?	integer	Minimum number of products in a stack or layer, (<i>MaxAmount</i> – <i>StandardAmount</i>) <= <i>MinAmount</i> < <i>StandardAmount</i> and <i>MinAmount</i> < <i>LayerAmount</i> . Where not known, the default case is to use a value equivalent to <i>MaxAmount</i> – <i>StandardAmount</i> .
<i>MaxHeight</i> ? New in JDF 1.4	integer	Max height of the stack
<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>PreStackAmount</i> ? New in JDF 1.4	integer	Amount that is gathered at first
<i>PreStackMethod</i> ? New in JDF 1.4	enumeration	Values are: <i>All</i> – all layers are pre-stacked <i>First</i> – only first layer is pre-stacked <i>None</i> – no pre-stacking
<i>StackCompression</i> ? New in JDF 1.4	boolean	If true stack is compressed before push out
<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.

7.2.177 StaticBlockingParams

[New in JDF 1.4](#)

StaticBlockingParams defines the details of *StaticBlocking*.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>StaticBlocking</i>
Output of Processes:	—

Table 7-362: StrappingParams Resource

Name	Data Type	Description
		No attributes defined

7.2.178 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.

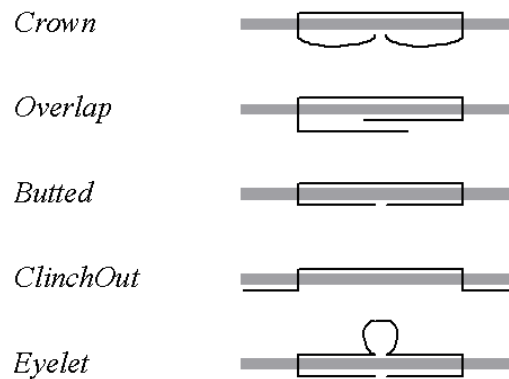


Figure 7-59: Staple shapes

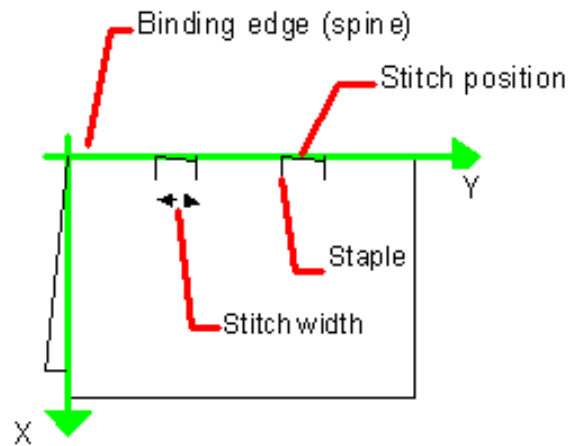


Figure 7-60: Parameters and coordinate system used for saddle stitching

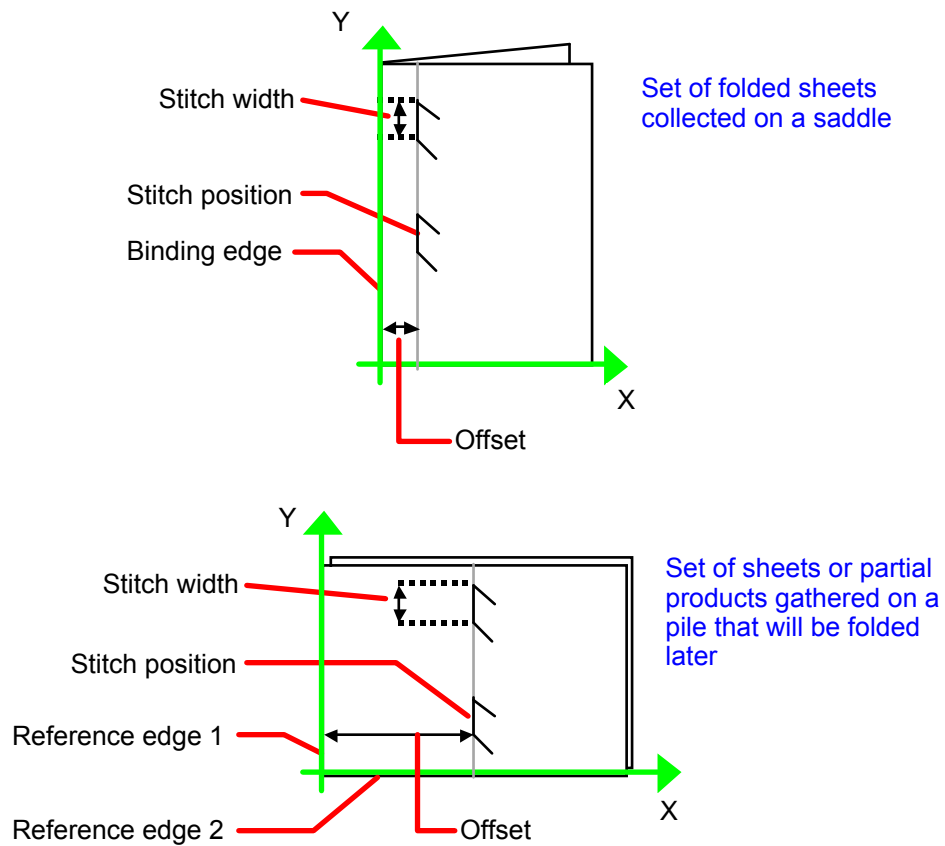


Figure 7-61: Parameters and coordinate system used for Stitching

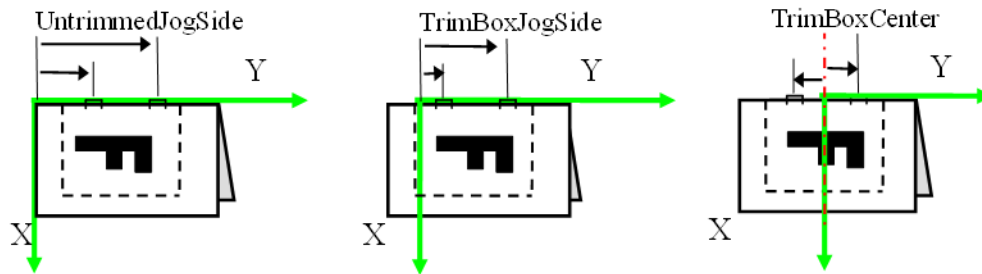


Figure 7-62: Stitching Coordinate System for StitchOrigin Values

Resource Properties

Resource Class: Parameter
 Resource referenced by: —
 Example Partition: *ProductionRun, SubRun, WebProduct*
 Input of Processes: *Stitching*
 Output of Processes: —

Table 7-363: 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 sticher without inserting any stitches. Use "NoOp" to bypass the sticher 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 <i>Stitching</i> Process is part of a Combined Process with other Processes (e.g., <i>DigitalPrinting</i>) where, when combined, there is no input Component to be stitched. Values are: <i>Top</i> <i>Left</i> <i>Right</i> <i>Bottom</i> Deprecation note: starting with JDF 1.2, use 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.

Table 7-363: **StitchingParams Resource (Section 2 of 2)**

Name	Data Type	Description
<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. Values are: <i>Crown</i> <i>Overlap</i> <i>Butted</i> <i>ClinchOut</i> <i>Eyelet</i> Note: representations of the values are displayed in Figure 7-59.
<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>StitchOrigin</i> = "UntrimmedJogSide" New in JDF 1.4	enumeration	Defines the origin of <i>StitchPositions</i> . For an illustration of the values, see Figure 7-62. Values are: <i>TrimBoxCenter</i> <i>TrimBoxJogSide</i> <i>UntrimmedJogSide</i>
<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> .
<i>StitchType</i> ? Modified in JDF 1.2	enumeration	Specifies the type of the Stitching operation. Values are: <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 <i>ComponentLink/@Orientation</i> = "Rotate90". <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.

7.2.179 Strap

[New in JDF 1.1](#)

Resource Properties

Resource Class: Consumable

Resource referenced by:

Example Partition: —
Input of Processes: *Strapping*
Output of Processes: —

Table 7-364: Strap Resource

Name	Data Type	Description
<i>StrapColor?</i>	NamedColor	Color of the string or strap.
<i>StrapColorDetails?</i> New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If <i>StrapColorDetails</i> is supplied, <i>StrapColor</i> SHOULD also be supplied.
<i>Material</i>	enumeration	Strap material. Values are: <i>AdhesiveTape</i> <i>Strap</i> <i>String</i>

7.2.180 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 7-365: StrappingParams Resource

Name	Data Type	Description
<i>StrappingType</i>	enumeration	Strapping pattern. 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 minimum to maximum 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 7-63 and Figure 7-64). 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.

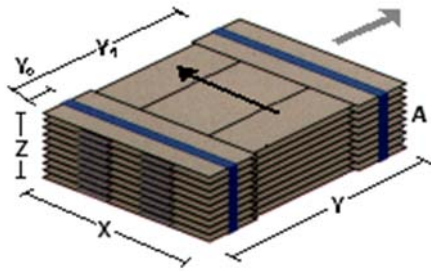


Figure 7-63: Strapped Bundle

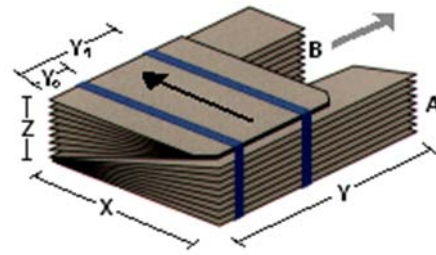


Figure 7-64: Strapped Bundle with Sub-bundles

7.2.181 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 7-366: StripBindingParams Resource

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 <i>HoleMakingParams/@HoleType</i> .
<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.
<i>StripColorDetails?</i> New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If <i>StripColorDetails</i> is supplied, <i>StripColor</i> SHOULD also be supplied.
<i>HoleMakingParams?</i> New in JDF 1.2	refelement	Details of the holes in <i>StripBinding</i> .

7.2.182 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 **BinderySignature** Elements of a Job are placed onto press Sheets and also gives concrete values for the various **StripCellParams** defined by the **BinderySignature**.

The Partitioning of **StrippingParams** defines the structure of the finished product and the structure of the **Layout** Resource that is produced by the **Stripping** Process. It is therefore RECOMMENDED to Partition the **StrippingParams** Resource by *SheetName*. Note that some Attributes and Elements 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:	<i>SignatureName, SheetName, BinderySignatureName, PartVersion, SectionIndex, CellIndex</i>
Input of Processes:	<i>Stripping</i>
Output of Processes:	—

Table 7-367: StrippingParams Resource (Section 1 of 3)

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	IDs of the Assembly Elements, AssemblySection Elements or StrippingParams [@ <i>BinderySignatureName</i>] to which the StrippingParams or Partition belongs.
<i>InnermostShingling</i> ? New in JDF 1.4	double	Percentage (1.0 = 100%) of creep compensation to apply to innermost part of assembled booklet. Shingling is perpendicular to the spine. Negative values go towards the spine. Values for pages between inner and outer are interpolated. Actual values of shingling are calculated by the system or operator. See Figure 7-65, “Shingling for Stripping” on page 751, and Figure 7-66, “Shingling for Stripping – Details” on page 751.
<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>OutermostShingling</i> ? New in JDF 1.4	double	Percentage (1.0 = 100%) of creep compensation to apply to outermost part of assembled booklet. Shingling is perpendicular to the spine. Negative values go towards the spine. Values for pages between inner and outer are interpolated. Actual values of shingling is calculated by the system or operator. See Figure 7-65, “Shingling for Stripping” on page 751, and Figure 7-66, “Shingling for Stripping – Details” on page 751.
<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 SignatureCell Elements with <i>SectionIndex</i> = "0"; the section that matches the second entry is flowed into SignatureCell Elements with <i>SectionIndex</i> = "1"; and so forth. <i>SectionList</i> MUST NOT be specified at the <i>CellIndex</i> Partition level.

Table 7-367: StrippingParams Resource (Section 2 of 3)

Name	Data Type	Description
<i>SheetNameFormat</i> ? New in JDF 1.4	string	Formatting value for identifying individual parts of the Layout . Values are from: "Generating strings with Format and Template" on page 909.
<i>SheetNameTemplate</i> ? New in JDF 1.4	string	Arguments for combining extracted values for identifying individual parts of the Layout . Values are from: "Generating strings with Format and Template" on page 909.
<i>StackDepth</i> ? New in JDF 1.4	integer	If specified, this Attribute describes cut-and-stack imposition. The order of stacks is defined by the order of StrippingParams Partitions. <i>StackDepth</i> MUST NOT be specified in Partitions lower than the Sheet level.
<i>WorkStyle</i> ?	WorkStyle	The direction in which to turn the press Sheet. Constraint: <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 SignatureCells, then, by default, the front pages are placed on the front side of the Sheet and the back pages are placed on the back side of the Sheet. Using the <i>Orientation</i> Attribute one can influence this default behavior. When the BinderySignature is defined by <i>FoldCatalog</i> or <i>Fold</i> Elements, 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.

Table 7-367: StrippingParams Resource (Section 3 of 3)

Name	Data Type	Description
StripMark * New in JDF 1.3 Modified in JDF 1.4	element	Indicates areas on the StrippingParams reserved for Marks. Modification note: Starting with JDF 1.4, the following constraint is removed: a StripMark MUST NOT be specified at Partition levels that are more granular than <i>SheetName</i> .

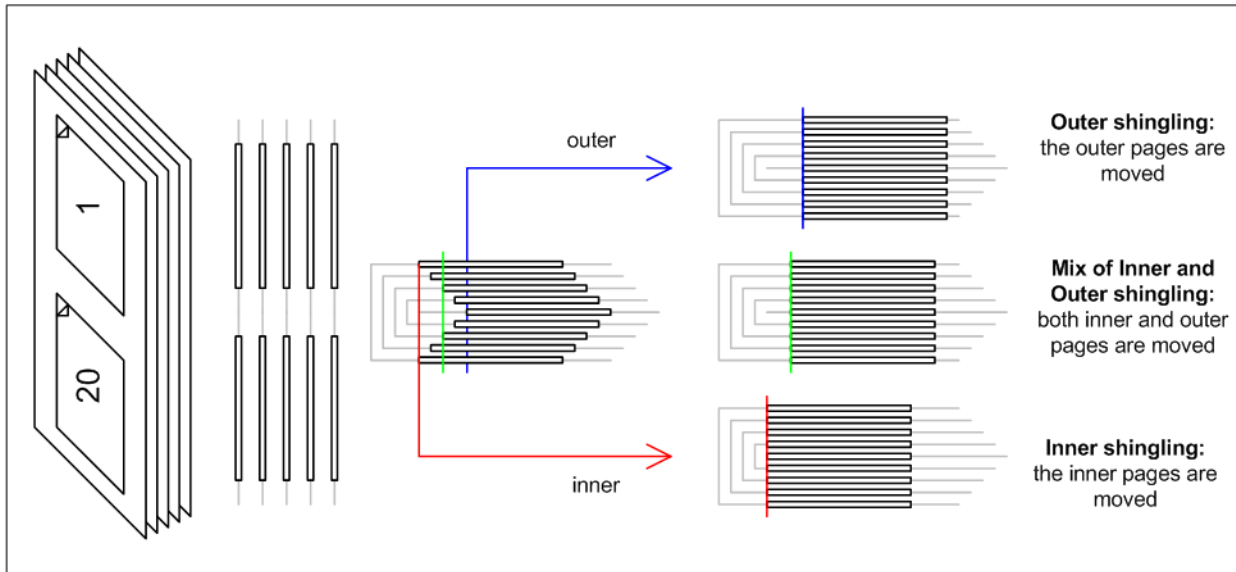


Figure 7-65: Shingling for Stripping

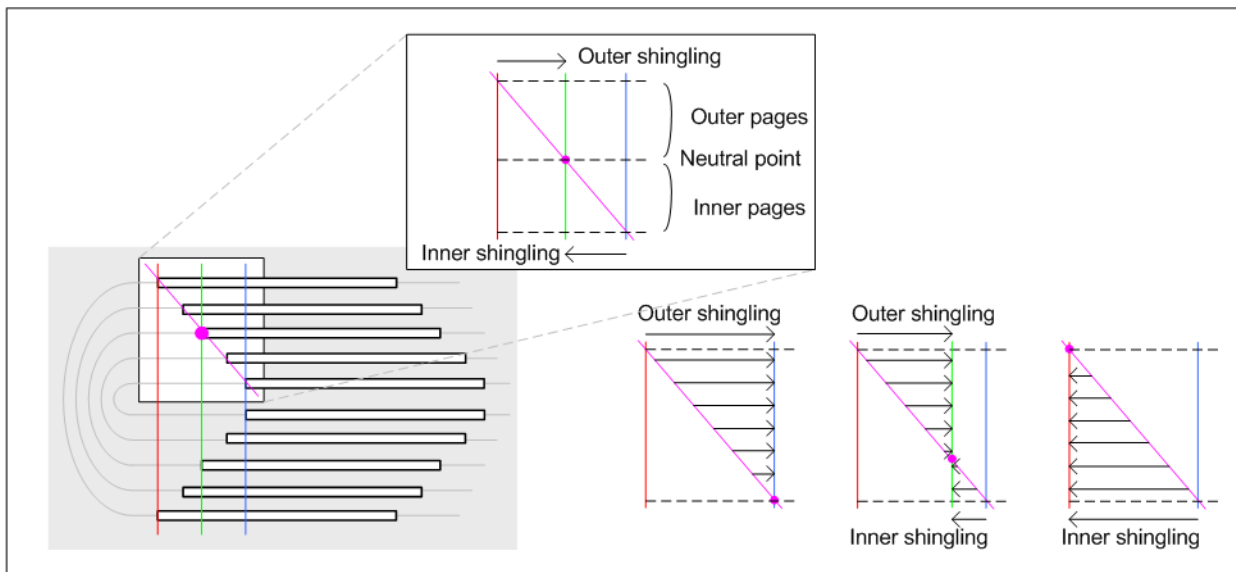


Figure 7-66: Shingling for Stripping – Details

7.2.182.1 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.

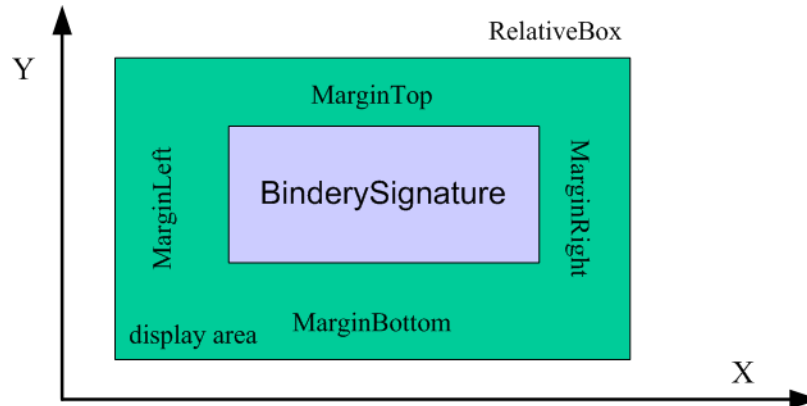


Figure 7-67: RelativeBox including margins

Table 7-368: Position Element (Section 1 of 2)

Name	Data Type	Description
<i>AbsoluteBox</i> ? New in JDF 1.3	Rectangle	Absolute position, in points, of the display area of this BinderySignature or StripMark on the front side of the StrippingParams . The BinderySignature is placed onto the display area after applying the <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 .

Table 7-368: Position Element (Section 2 of 2)

Name	Data Type	Description
<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 30.
<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.

7.2.182.2 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 7-369: 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.

Table 7-369: StripCellParams Element (Section 2 of 2)

Name	Data Type	Description
<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.
<i>Mask?</i> New in JDF 1.3	enumeration	The definition of the clipping mask for the placed graphics. 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 <i>MaskSeparation</i> 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.
<i>MaskBleed?</i> New in JDF 1.3	double	The distance over which to expand the mask in points.
<i>MaskSeparation?</i> New in JDF 1.3	string	Color/@Name of the separation that specifies <i>Mask</i> . <i>MaskSeparation</i> MUST be specified if and only if <i>Mask</i> = "PDL". Color/@ColorType of this separation MUST be <i>DieLine</i> .
<i>Sides?</i>	enumeration	Indicates whether contents are to be printed on one or both sides of the media. 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.

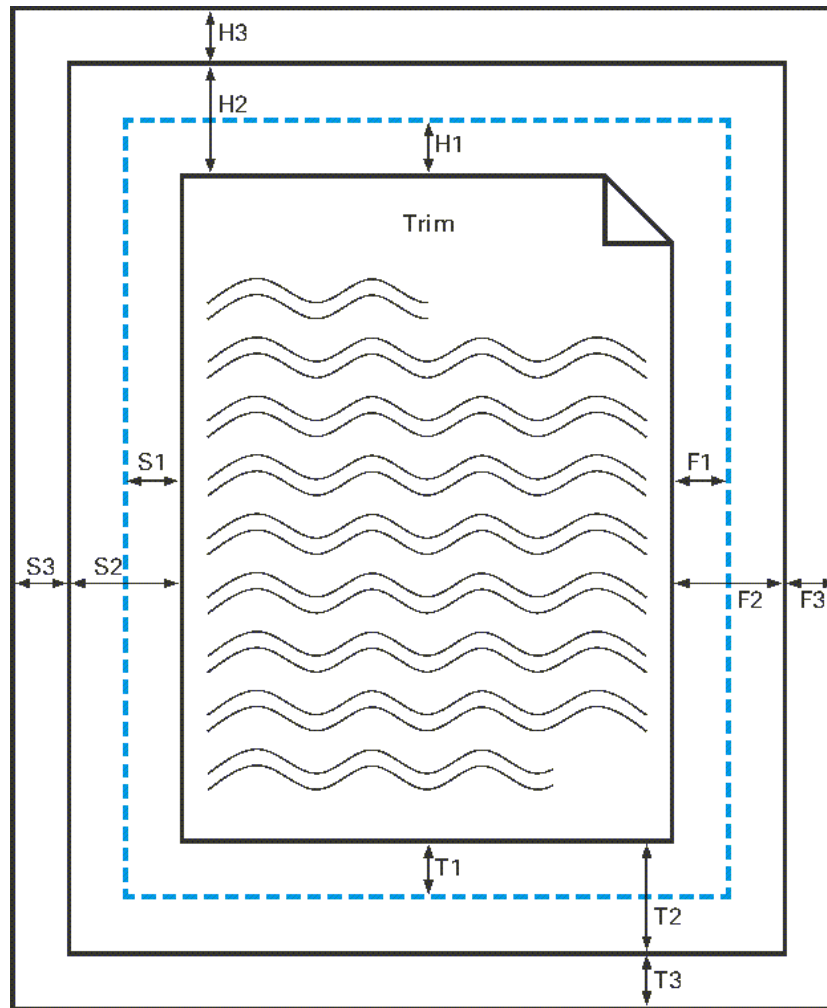


Figure 7-68: Definition of margins in StripCellParams

7.2.182.3 Element: StripMark

[New in JDF 1.3](#)

The StripMark Element specifies Marks to be placed on the Sheet.

Table 7-370: StripMark Element (Section 1 of 3)

Name	Data Type	Description
<i>AbsoluteHeight?</i> New in JDF 1.4	double	Absolute height in points.
<i>AbsoluteWidth?</i> New in JDF 1.4	double	Absolute width in points.
<i>Anchor?</i> New in JDF 1.4	Anchor	Origin of the mark coordinate system.

Table 7-370: StripMark Element (Section 2 of 3)

Name	Data Type	Description
<i>HorizontalFitPolicy</i> ? New in JDF 1.4	enumeration	How to fit the mark in the size. Values are: <i>NoRepeat</i> – The mark is neither resized nor repeated horizontally, but it is clipped if it is bigger than size. <i>StretchToFit</i> – The mark is resized horizontally to fill the allocated space. Aspect of the mark may get distorted. <i>UndistortedScaleToFit</i> – The mark is resized to maximum size that can fit in allocated space, without affecting aspect ratio. <i>RepeatToFill</i> – The mark is placed in requested position, then it is repeated horizontally, allowing clipping to occur so that all allocated space is entirely covered. <i>RepeatUnclipped</i> – The mark is placed in requested position, then it is repeated horizontally to fit as much unclipped copies as possible.
<i>ID</i> ? New in JDF 1.4	ID	Used as reference for <i>rRef</i> (mark that is relative to another mark)
<i>MarkContext</i> ? New in JDF 1.4	enumeration	<i>MarkContext</i> specifies context where a Mark must be applied. MUST NOT specify a <i>MarkContext</i> value that has a higher level than the Partitioning level where StripMark Elements resides Values are: <i>Sheet</i> – The mark belongs to a press sheet <i>BinderySignature</i> – The mark belongs to a BinderySignature and must be repeated for each StrippingParams/Position Element. <i>Cell</i> – The mark belongs to a page cell and must be repeated for each pagecell. <i>CellPair</i> – The mark belongs to a bound pair of Sheets repeated for each pair of page cells.
<i>MarkName</i> ?	NMTOKEN	Mark that is to be marked on the StrippingParams . Values include those from: Table 7-371, “MarkName Attribute Values”.
<i>MarkSide</i> ?	enumeration	Side and alignment of the marks. Values are from: Table 7-372, “MarkSide Attribute Values” on page 758
<i>Offset</i> ? New in JDF 1.4	XYPair	Position of the Anchor of this StripMark relative to RefAnchor/@Anchor as defined by @Anchor , RefAnchor/@Anchor and @MarkContext
<i>Ord</i> ? New in JDF 1.4	integer	Specifies an index into the Input RunList (Marks) for Stripping.
<i>Orientation</i> ? New in JDF 1.4	Orientation	Orientation of the mark in the coordinate system of the parent
<i>RelativeHeight</i> ? New in JDF 1.4	double	Height relative to the size of the parent specified by @MarkContext .
<i>RelativeWidth</i> ? New in JDF 1.4	double	Width relative to the size of the parent specified by @MarkContext .

Table 7-370: StripMark Element (Section 3 of 3)

Name	Data Type	Description
<i>StripMarkDetails</i> ? Modified in JDF 1.4	string	More detailed information about the <i>StripMark</i> . If <i>MarkName</i> = "Set" then <i>StripMarkDetails</i> is a name to refer to a private set of marks.
<i>VerticalFitPolicy</i> ? New in JDF 1.4	enumeration	How to fit the mark in the size. Values are: <i>NoRepeat</i> – The mark is not resized nor repeated vertically, but it is clipped if bigger than size. <i>StretchToFit</i> – The mark is resized vertically to fill the allocated space. Aspect of the mark may get distorted. <i>UndistortedScaleToFit</i> – The mark is placed once, resized to maximum size that can fit in allocated space without affecting aspect ratio. <i>RepeatToFill</i> – The mark is placed in requested position. Then it is repeated vertically, allowing clipping to occur so that all allocated space is entirely covered. If <i>@HorizontalFitPolicy</i> is set to "RepeatToFill" or "RepeatUnclipped", horizontal repetition is performed first. Then the resulting row is repeated vertically as requested. <i>RepeatUnclipped</i> – The mark is placed in requested position, then it is repeated vertically to fit as much unclipped copies as possible. If <i>@HorizontalFitPolicy</i> is set to "RepeatToFill" or "RepeatUnclipped", horizontal repetition is performed first. Then the resulting row is repeated vertically as requested.
Position? Deprecated in JDF 1.4	element	Specifies where to place the <i>StripMark</i> on the StrippingParams . Deprecation note: starting with JDF 1.4, the position of the Anchor of this <i>StripMark</i> is relative to RefAnchor/@Anchor as defined by @Anchor , RefAnchor/@Anchor and @MarkContext
<i>JobField</i> ?	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.
<i>RefAnchor</i> ? New in JDF 1.4	element	Details of the coordinate system that this mark is placed relative to. This MAY be either the parent coordinate system or the coordinate system of a referenced <i>StripMark</i> .

— Attribute: MarkName

Table 7-371: MarkName Attribute Values (Section 1 of 2)

Value	Description
<i>BleedMark</i> New in JDF 1.4	
<i>CenterMark</i> New in JDF 1.4	
<i>CIELABMeasuringField</i>	
<i>CollationMark</i> New in JDF 1.4	
<i>ColorControlStrip</i>	

Table 7-371: MarkName Attribute Values (Section 2 of 2)

Value	Description
<i>ColorRegisterMark</i>	
<i>CutMark</i>	
<i>DensityMeasuringField</i>	
<i>FoldMark</i> New in JDF 1.4	
<i>IdentificationField</i>	
<i>JobField</i>	
<i>PaperPathRegisterMark</i>	
<i>RegisterMark</i>	
<i>ScavengerArea</i>	
<i>Set</i> New in JDF 1.4	Specifies to use a MarkSet (file containing multiple marks). The name of the MarkSet MAY be passed in <i>StripMarkDetails</i>
<i>TrimMark</i> New in JDF 1.4	

— Attribute: MarkSide

Table 7-372: MarkSide Attribute 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.

7.2.183 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 592.

7.2.184 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:	<i>ThreadSealing</i>
Output of Processes:	—

Table 7-373: ThreadSealingParams Resource

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. 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.
<i>ThreadLength?</i> Modified in JDF 1.2	double	Length of one thread.
<i>ThreadStitchWidth?</i> Modified in JDF 1.2	double	Width of one stitch.
<i>SealingTemperature?</i>	integer	Temperature needed for sealing thread and Sheets together, in degrees centi-grade.

7.2.185 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).

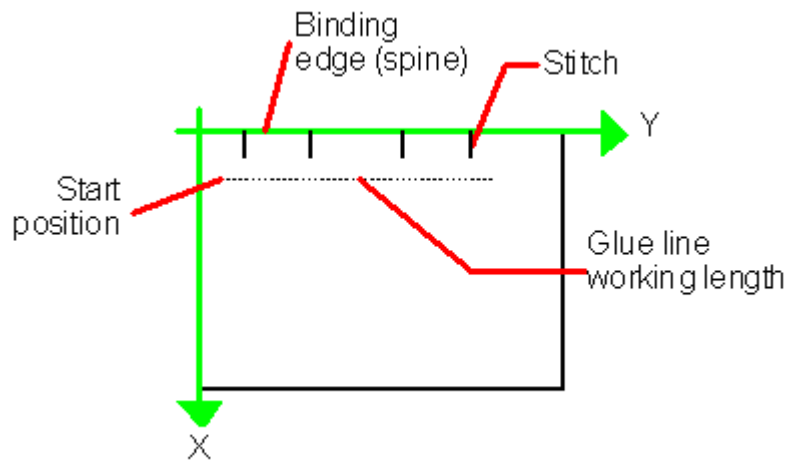


Figure 7-69: Parameters and coordinate system used for thread sewing

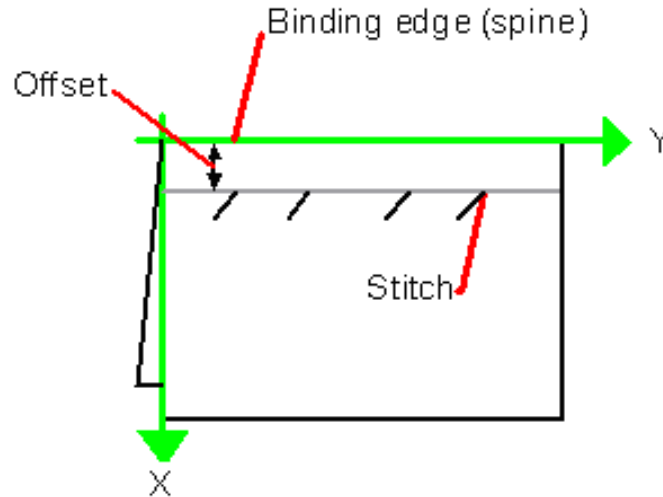


Figure 7-70: Parameters and coordinate system used for side sewing

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>ThreadSewing</i>
Output of Processes:	—

Table 7-374: ThreadSewingParams Resource (Section 1 of 2)

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. 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. 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 GlueLine 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.

Table 7-374: ThreadSewingParams Resource (Section 2 of 2)

Name	Data Type	Description
<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. 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.

7.2.186 Tile

Each **Tile** Resource defines how content from a surface Resource will be imaged onto a piece of media that is smaller than the designated surface. Tiling occurs in some production environments when pages are imaged on to an intermediate medium, and the resulting image of the surface is larger than the media. In this case, instructions are needed to determine how the intermediate media (tiles) will be assembled to achieve the desired output, (e.g., a single plate for the surface). For example, a Device might require that four pieces of film be assembled to create the image for the plate.

In general, a **Tile** Resource will be Partitioned (see Section 3.10.5, Description of Partitioned Resources) by *TileID*. Individual tiles are selected and matched by specifying the appropriate *TileID* Attribute, which is described in Table 3-28, "Part Element" on page 104.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	<i>TileID</i>
Input of Processes:	<i>Tiling</i>
Output of Processes:	—

Table 7-375: 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.
<u>MarkObject</u> * New in JDF 1.4	element	List of marks that are placed on the tile. <i>MarkObject/@CTM</i> applies to the coordinate system of the Tile .
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

7.2.187 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:	ArtDeliveryIntent/ArtDelivery, EmbossingParams/Emboss
Example Partition:	—
Input of Processes:	<i>Any Process, Embossing, ShapeCutting</i>
Output of Processes:	<i>DieMaking</i>

Table 7-376: Tool Resource

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). Deprecation note: starting with JDF1.3, use DieLayout to describe the number of cut forms in a cutting die.
<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 <i>Resource/@ProductID</i> in JDF 1.3
<i>ToolType?</i> Modified in JDF 1.4	NMTOKEN	Type of the tool. Values include: <i>Braille</i> – embossing tool for blind script. New in JDF 1.4 <i>CentralStripper</i> – The center tool of the stripper tool set. Stripping means removing small parts of waste in between blanks. <i>ChangingCuttingBlock</i> – a changeable part for a tool set (CutDie). Used for cutting of optional shapes like windows, stars, etc. It is not a part of tool set. Described in MIS with an own <i>ProductID</i> . New in JDF 1.4 <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 <i>ToolID</i> refers not to a single tool, but to a set of matching tools (i.e. tool set) that are used in the Process. E.g., when <i>ProductID</i> is a single stock item number in the MIS for a tool 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

7.2.188 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, Color/PrintConditionColor, TransferCurvePool/TransferCurveSet
Example Partition:	<i>RibbonName, SheetName, Side, WebName</i>
Input of Processes:	—
Output of Processes:	—

Table 7-377: 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. Values include: <i>All</i>

7.2.189 TransferCurvePool

A transfer curve pool is a collection of **TransferCurveSet** Elements that each contains information about a **TransferCurve**. Multiple **TransferCurveSet** Elements MAY exist at one time. For example, one MAY exist for the laser calibration of the imagesetter, one for the **ContactCopying** Process and one for the printing Process. Each **TransferCurveSet** consists of one or more **TransferCurve** Elements. A **TransferCurve** Resource is applied to the appropriate *Separation*, or to all *Separations* when *Separation* = "All". The **TransferCurveSet** Elements are concatenated in the following order:

Film -> Plate -> Press -> Paper.
and
Proof.

In addition to the **TransferCurve** Resource, the **TransferCurveSet** Elements contain Device-dependent geometrical information, (e.g., *CTM* definitions).

Resource Properties

Resource Class:	Parameter
Resource referenced by:	TransferFunctionControl, Layout
Example Partition:	—
Input of Processes:	<i>ContactCopying, ContoneCalibration, ConventionalPrinting, DigitalPrinting, ImageSetting, InkZoneCalculation, PreviewGeneration, Stripping</i>
Output of Processes:	<i>LayoutPreparation</i>

Table 7-378: TransferCurvePool Resource

Name	Data Type	Description
TransferCurveSet *	element	The set of transfer curves.

7.2.189.1 Element: TransferCurveSet

TransferCurveSet Elements describe both the characteristic curve of transfer and the relation between the various process coordinate systems.

Table 7-379: 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. Values are: <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
<i>TransferCurve</i> * Modified in JDF 1.1	refelement	List of TransferCurve entries.

7.2.190 TransferFunctionControl

Resource Properties

Resource Class:	Parameter
Resource referenced by:	SeparationControlParams
Example Partition:	—
Input of Processes:	<i>ContoneCalibration</i>
Output of Processes:	—

Table 7-380: 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. Values are: <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. Modification note: starting with JDF 1.3, the data type changes from enumeration to enumerations. If multiple values are specified, the transfer functions that are specified by the individual enumeration values are concatenated.
<i>TransferCurvePool</i> ?	refelement	Provides a set of transfer curves to be used by the Process.

7.2.191 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:	<i>DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName</i>
Input of Processes:	<i>Trapping</i>
Output of Processes:	—

Table 7-381: TrappingDetails Resource

Name	Data Type	Description
<i>DefaultTrapping</i> = "false"	boolean	If "true", pages that have no defined TrapRegion Elements are trapped using the set of TrimmingParams . The bleed box is used for the trap zone. If "false", only pages that have TrapRegion Elements are trapped.
<i>IgnoreFileParams</i> = "true" Deprecated in JDF 1.4	boolean	If "true", any detectable trapping controls (or traps) provided within any source files used by this Process are ignored. If "false", trapping controls embedded in the source files are honored. Note that if TrappingDetails (and the <i>Trapping</i> Process) is not present, then the trapping defined in PostScript MAY still be applied. Deprecation note: starting with JDF1.4, the application of trap annotations is specified in InterpretingParams/ PDFInterpretingParams/@PrintTrapAnnotations.
<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>TrappingType</i> ? Deprecated in JDF 1.2	integer	Identifies the trapping method to be used by the <i>Trapping</i> Process. The number identifies the minor (last three digits) and major (any digits prior to the last three) version of the trapping type requested.
<i>TrappingOrder</i> ?	element	<i>Trapping</i> 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>TrappingParams</i> ?	refelement	A TrappingParams Resource that is used to define the default trapping parameters when <i>DefaultTrapping</i> = "true".
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.

7.2.191.1 Element: TrappingOrder

Table 7-382: TrappingOrder Element

Name	Data Type	Description
SeparationSpec * Modified in JDF 1.2	element	An array of colorant names.

7.2.192 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:	<i>DocIndex, RunIndex, RunTags, DocTags, PageTags, SetTags, SheetName, Side, SignatureName</i>
Input of Processes:	—
Output of Processes:	—

Table 7-383: 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 " <i>true</i> ", trapping is enabled for zones that are defined with this parameter set. Use <i>NoOp</i> in JDF 1.2 and above.
<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.

Table 7-383: TrappingParams Resource (Section 2 of 4)

Name	Data Type	Description
<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 "1" are marked, samples with a value of "0" 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. 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.
<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> .

Table 7-383: TrappingParams Resource (Section 3 of 4)

Name	Data Type	Description
<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 ($low + \max(StepLimit * low, 0.05)$), then the edge is a candidate for trapping. The value 0.05 is set to avoid trapping light areas in vignettes. This entry is used when not specified explicitly by a <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.
<i>TrapEndStyle</i> = "Miter"	NMTOKEN	Instructs the trap engine how to form the end of a trap that touches another object. Values include: <i>Miter</i> <i>Overlap</i> Note: other values might be added later from 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. 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> .

Table 7-383: TrappingParams Resource (Section 4 of 4)

Name	Data Type	Description
ColorantZoneDetails *	element	ColorantZoneDetails Subelements. Entries in this dictionary reflect the results of any named colorant aliasing specified. Each entry defines parameters specific for one named colorant. If the colorant named is neither listed in the ColorantParams array nor implied by the ProcessColorModel for the ColorantControl object in effect when these TrappingParams are applied, the entry is not used for trapping.

7.2.192.1 Element: ColorantZoneDetails

Table 7-384: ColorantZoneDetails Element

Name	Data Type	Description
Colorant	string	The colorant name that occurs in the SeparationSpec/@Name of the ColorantParams array of the ColorantControl object used by the Process.
StepLimit?	double	A number between 0 and 1. Specifies the smallest step specified in the color value of a colorant to trigger trapping at a given boundary. If the higher color value at the boundary exceeds the lower value by an amount that is equal or greater than the larger of 0.05 or StepLimit times the lower value ($\text{low} + \max(\text{StepLimit} * \text{low}, 0.05)$), then the edge is a candidate for trapping. The value 0.05 is set to avoid trapping light areas in vignettes. If omitted, the StepLimit Attribute in the TrappingParams Resource is used.
TrapColorScaling?	double	A number between 0 and 1. Specifies a scaling of the amount of color applied in traps towards the neutral density of the dark area. A value of "1" means the trap has the combined color values of the darker and the lighter area. A value of "0" means the trap colors are reduced so that the trap has the neutral density of the darker area. If omitted, the TrapColorScaling Attribute in the TrappingParams Resource is used.

7.2.193 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 7-385: TrapRegion Resource (Section 1 of 2)

Name	Data Type	Description
TrapZone?	PDFPath	Each element within TrapZone is one subpath of a complex path. The TrapZone is the area that results when the paths are filled using the non-zero winding rule. When absent, the MediaBox array for the RunList defines the TrapZone.

Table 7-385: TrapRegion Resource (Section 2 of 2)

Name	Data Type	Description
<i>Pages</i>	Integer-RangeList	Identifies a set of pages from the RunList to trap using the specified geometry and trapping style. The logical indices that <i>Pages</i> reference in a RunList are referenced in the same way as Layout/ContentObject/@Ord does. For details, see Section 7.2.109.12.4, "Using Ord to Reference Elements in RunList Resources" on page 609.
TrappingParams ?	refelement	The set of trapping parameters which will be used when trapping in this region.

7.2.194 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).

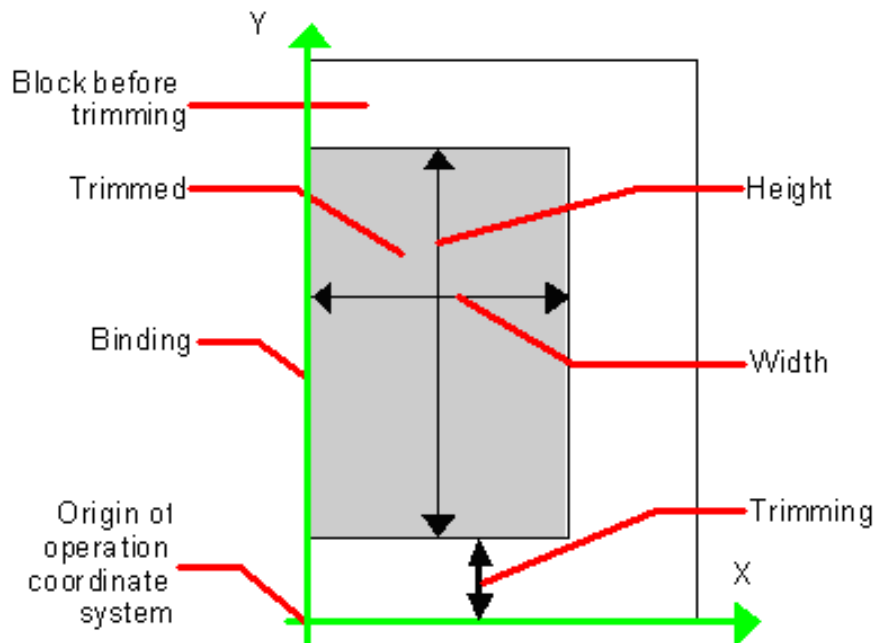


Figure 7-71: Parameters and coordinate system used for trimming

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	Trimming
Output of Processes:	—

Table 7-386: TrimmingParams Resource

Name	Data Type	Description
<i>Height?</i>	double	Height of the trimmed product.
<i>TrimCover="Both"</i> New in JDF 1.3	enumeration	Specifies the covers to be trimmed. Covers containing flaps are generally not trimmed. Values are: <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. 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.

7.2.195 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 90. The section has details on tracking *Amount* and *ActualAmount*, Default units are “countable objects”. See “Units” on page 16.

Resource Properties

Resource Class:	Consumable
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>Any Process</i>
Output of Processes:	—

Table 7-387: 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 7-387: 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.</p> <p>Values include:</p> <p><i>Insert</i> – post fuser inserter (Media Sides category)</p> <p><i>OneSided</i> – includes one sided counts (Media Sides category)</p> <p><i>TwoSided</i> – includes two sided counts (Media Sides category)</p> <p><i>NormalSize</i> – includes normal size counts (Media Size category)</p> <p><i>LargeSize</i> – includes large size counts (Media Size category)</p> <p><i>Black</i> – includes black colorant only counts (Colorant category)</p> <p><i>Color</i> – includes one or more non-black, non-highlight color colorants counts (Colorant category)</p> <p><i>Blank</i> – includes entirely blank counts (Colorant category)</p> <p><i>HighlightColor</i> – includes highlight colorant counts (Colorant category)</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. (Usage category)</p> <p><i>Auxiliary</i> – includes all counts for work not requested by the user, e.g. banner, confirmation, slip, separator, error log. (Usage category)</p>
<i>Scope</i>	enumeration	<p>The scope of this usage counter.</p> <p>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>

7.2.196 VarnishingParams

[New in JDF 1.4](#)

This Resource provides the parameters of a *Varnishing* Process.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>Varnishing</i>
Output of Processes:	—

Table 7-388: VarnishingParams Resource

Name	Data Type	Description
<i>ModuleIndex?</i>	integer	See ConventionalPrintingParams . In a Combined Process, all modules of the Device, including press modules are counted to calculate <i>ModuleIndex</i> . Only one of <i>ModuleIndex</i> or <i>ModuleType</i> MAY be specified.
<i>ModuleType?</i>	NMTOKEN	The type of module used to apply the Varnish. Only one of <i>ModuleIndex</i> or <i>ModuleType</i> MAY be specified. Values include: <i>PrintModule</i> – The Varnish is applied in a printing unit <i>CoatingModule</i> – The Varnish is applied in a specialized coating unit Values include those from: Section C.2, "ModuleType Supported Strings" on page 879.
<i>VarnishArea?</i>	enumeration	Area to be varnished. <i>VarnishArea</i> specifies the requirements for ExposedMedia . Values are: <i>Full</i> – The entire Media surface MUST be varnished. <i>Spot</i> – Only parts of the Media surface MUST be varnished.
<i>VarnishMethod?</i>	enumeration	Method used for varnishing. <i>VarnishMethod</i> specifies the requirements for ExposedMedia . Values are: <i>Blanket</i> – The Varnishing is performed in a CoatingModule. An ExposedMedia with ExposedMedia/Media/@MediaType = "Blanket" SHOULD be specified. <i>Plate</i> – The Varnishing is performed in a PrintModule or a CoatingModule. An ExposedMedia with ExposedMedia/Media/@MediaType = "Plate" SHOULD be specified. <i>Independent</i> – No additional ExposedMedia is required. This method MAY be used to specify varnishing in a digital press.

7.2.197 VerificationParams

This Resource provides the parameters of a *Verification* Process.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>Verification</i>
Output of Processes:	—

Table 7-389: VerificationParams Resource (Section 1 of 2)

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.

Table 7-389: VerificationParams Resource (Section 2 of 2)

Name	Data Type	Description
<i>InsertError?</i>	string	Database insertion statement in <i>C printf</i> format defining how information read from the Resource of the <i>Verification</i> Process is to be stored in case of verification errors. The database is defined by the DBSelection Resource of the <i>Verification</i> 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

7.2.198 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 7-390: WebInlineFinishingParams Resource

Name	Data Type	Description
<i>FolderProduction*</i>	element	Specifies the Folder setup for newspaper presses:

7.2.198.1 Element: FolderProduction

Table 7-391: FolderProduction Element (Section 1 of 2)

Name	Data Type	Description
<i>FolderModuleIndex?</i>	integer	Identifies a particular folder module to be used. <i>FolderModuleIndex</i> MUST match Device/Module/@ModuleIndex .

Table 7-391: FolderProduction Element (Section 2 of 2)

Name	Data Type	Description
<i>ProductionType</i> ="NonCollect"	enumeration	Indicates whether the product is collected or not. Values are: <i>Collect</i> <i>NonCollect</i>

7.2.199 WireCombBindingParams

This Resource describes the details of the *WireCombBinding* Process.

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>WireCombBinding</i>
Output of Processes:	—

Table 7-392: 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>ColorDetails?</i> New in JDF 1.4	string	A more specific, specialized or site-defined name for the color. If <i>ColorDetails</i> is supplied, <i>Color</i> SHOULD also be supplied.
<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 <i>HoleMakingParams/@HoleType</i> .
<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. Values are: <i>LaqueredSteel</i> <i>TinnedSteel</i> <i>ZincsSteel</i>
<i>Shape</i> = "Single"	enumeration	The shape of the wire comb binding. 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.
<i>HoleMakingParams?</i> New in JDF 1.2	refelement	Details of the holes in <i>WireCombBinding</i> .

7.2.200 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 7-393: WrappingParams Resource

Name	Data Type	Description
<i>WrappingKind</i>	enumeration	Values are: <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 531. For more details on preflight, See “Preflight” on page 300.

Capabilities descriptions that are saved in files MUST be formatted as a JMF/Signal/Response to the KnownDevices Query Message.

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.

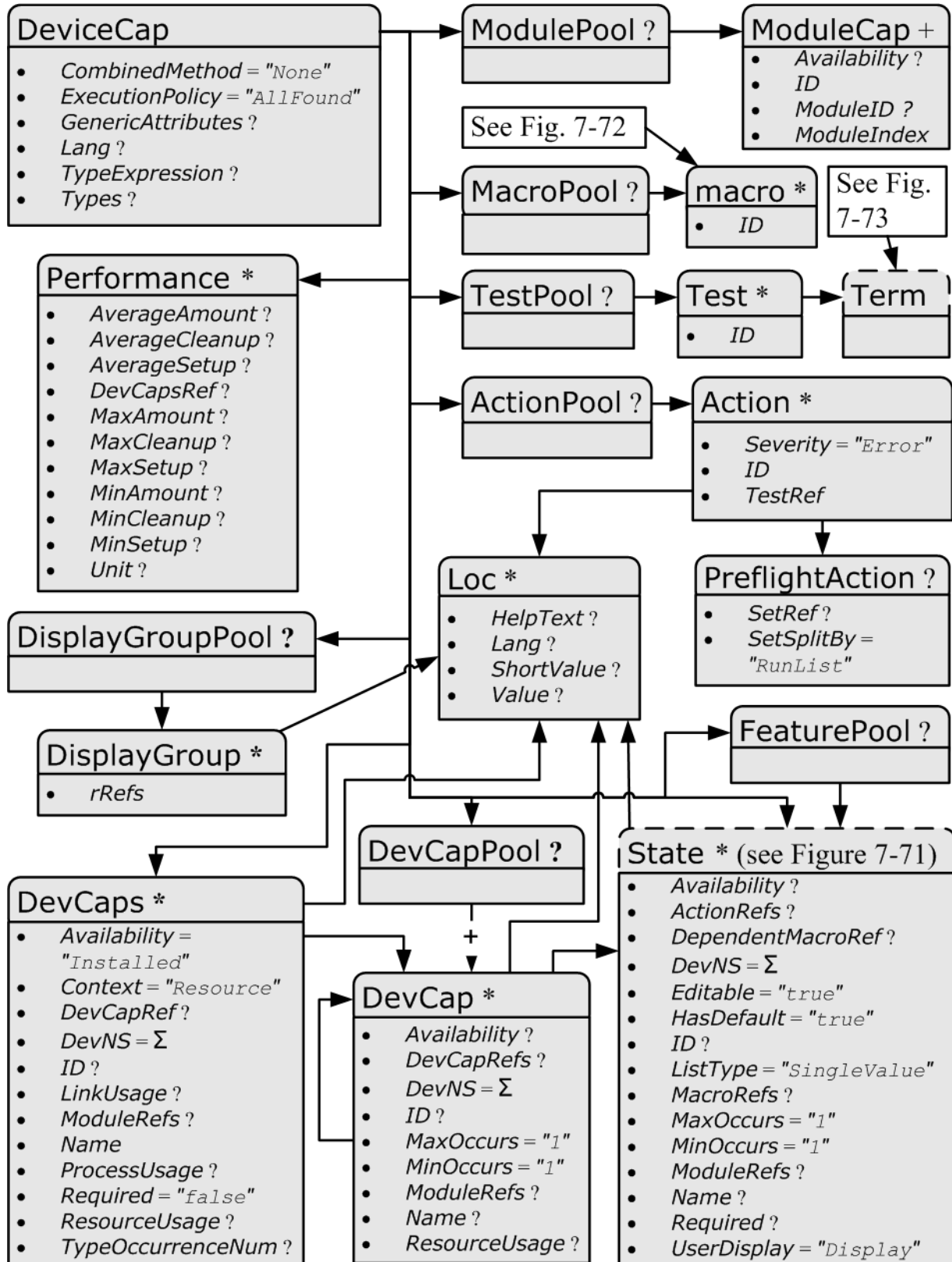


Figure 7-72: DeviceCap – a diagram of its structure

Table 7-394: DeviceCap Element (Section 1 of 3)

Name	Data Type	Description
<p><i>CombinedMethod</i> = "None" Modified in JDF 1.3</p>	enumerations	<p>Specifies how the Processes specified in <i>Types</i> are to be specified. If multiple values are specified, the structure of the JDF MUST match one of the values.</p> <p>Values are:</p> <p><i>Combined</i> – The list of Processes in <i>Types</i> MUST be specified as a Combined Process.</p> <p><i>CombinedProcessGroup</i> – The list of Processes in <i>Types</i> MUST be specified either as a Combined Process or as a <i>ProcessGroup</i> of individual Processes. In JDF 1.3 and beyond, the pair of individual tokens: "<i>CombinedProcessGroup</i>" replace this single value. Deprecated in JDF 1.3</p> <p><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</p> <p><i>ProcessGroup</i> – The list of Processes in <i>Types</i> MUST be specified as a <i>ProcessGroup</i> of individual Processes.</p> <p><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.</p>
<p><i>ExecutionPolicy</i> = "AllFound" New in JDF 1.2</p>	enumeration	<p>Describes the policy for finding and executing JDF Nodes as described in "Determining Executable Nodes" on page 147.</p> <p>Values are:</p> <p><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.</p> <p><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.</p> <p><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.</p>
<p><i>GenericAttributes</i> ?</p>	NMTOKENS	<p>List of all generic Attributes that are supported and unrestricted by the Device implementation. Descriptions of Attributes that appear in State Elements (see the following Section 7.3.7, State) overwrite the description in <i>GenericAttributes</i>.</p>
<p><i>Lang</i> ? New in JDF 1.2</p>	languages	<p>Specifies the localization(s) provided with the capabilities. If not specified, no localizations are provided.</p>
<p><i>OptionalCombinedTypes</i>? Deprecated in JDF 1.2</p>	NMTOKENS	<p>List of optional JDF Node types. The entries of the list MUST be a subset of <i>Types</i>.</p> <p>Values include those from: JDF/@<i>Types</i></p> <p>Example: a RIP with optional in-RIP trapping would specify <i>OptionalCombinedTypes</i> = "Trapping" if <i>Types</i> = "Trapping Interpreting Rendering".</p> <p>Deprecation note: starting with JDF 1.2, use <i>TypeExpression</i>.</p>

Table 7-394: DeviceCap Element (Section 2 of 3)

Name	Data Type	Description
Type ? 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. Values include those from: JDF/@ <i>Type</i> Deprecation note: starting with JDF 1.2, a single value of type is also defined in the <i>Types</i> Attribute.
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). Constraint: 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 Combined Process Nodes and Process Group Nodes. Values are: <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, 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, Screening, 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, Folding</i>). The Device can do both Processes, in any order multiple times. Deprecation note: starting with JDF 1.2, use <i>TypeExpression</i> .
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> . Constraint: in JDF 1.2 and above, one of <i>Types</i> or <i>TypeExpression</i> MUST be specified. Values include those from: JDF/@ <i>Types</i>
ActionPool ? New in JDF 1.2	element	Container for zero or more <i>Action</i> Elements for use as constraints.
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>DevCaps</i> *	element	List of definitions of the accepted Resources and Elements. The <i>DevCaps</i> Elements are combined with a logical AND, (i.e., a JDF MUST fulfill all restrictions defined by the set of <i>DevCaps</i>). Only Resources that are specified within this list are honored by the Device.
DisplayGroupPool ? New in JDF 1.2	element	List of <i>DisplayGroup</i> Subelements, which define the user interface presentation of sets of related <i>DevCap</i> Attribute Values. This is metadata to provide assistance in user interface display layout.

Table 7-394: DeviceCap Element (Section 3 of 3)

Name	Data Type	Description
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 <i>FeaturePool</i> 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 <i>macro</i> Elements, each of which contains an expression that can cause <i>State</i> Attribute Values (e.g., <i>CurrentValue</i> or <i>UserDisplay</i>) to be changed.
ModulePool ? New in JDF 1.3	element	Pool of <i>ModuleCap</i> Elements that specify the availability of a given Module.
Performance *	element	Specification of a Devices performance capabilities.
TestPool ? New in JDF 1.2	element	Container for zero or more <i>Test</i> Elements that are referenced from <i>ActionPool/Action</i> Elements.
State * New in JDF 1.3	element	Abstract <i>State</i> Elements that define the parameter space that is covered by the Device. One <i>State</i> 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 807. 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 807.

Table 7-395: ActionPool Element

Name	Data Type	Description
Action *	element	A list of independent <i>Action</i> Elements.

7.3.2.1 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 7-396: 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. Values are: <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.
Loc *	element	Text to describe an error if the Test fails. (See "Loc" on page 785.)
PreflightAction ?	element	Provides additional constraints that are specific to the <i>Preflight</i> Process. See "PreflightParams" on page 676.

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 7-397: 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 7-398: 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.

7.3.4.1 ModuleCap

[New in JDF 1.3](#)

Module elements specify features that depend on given hardware or software modules being installed. Hardware examples include duplex units for printers. Software licensing keys MAY also be modeled as modules.

Table 7-399: ModuleCap Element

Name	Data Type	Description
<i>Availability?</i>	enumeration	Specifies whether the feature described by this <i>State</i> Element is available on the Device. Values are: <i>Installed</i> – The feature is installed on the Device and is available for use. <i>Module</i> – <i>Module</i> is not to be specified recursively in a <i>ModuleCap</i> . 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 <i>ModuleCap</i> .
<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 <i>Module</i> 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 <i>Module</i> 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 Intent Resources as well as Process definition Resources.

Table 7-400: DevCaps Element (Section 1 of 4)

Name	Data Type	Description
<i>Availability</i> = "Installed" New in JDF 1.2	enumeration	Specifies whether the feature described by this <i>DevCaps</i> Element is available on the Device. 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.

Table 7-400: DevCaps Element (Section 2 of 4)

Name	Data Type	Description
<i>Context</i> = "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). Values are: <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.
<i>DevCapRef</i> ? New in JDF 1.3	IDREFS	Reference to reusable DevCap Elements that are located in DeviceCap/DevCapPool. A reference to a DeviceCap/DevCapPool/DevCap is equivalent to an inline DevCap in this DevCaps. Exactly one of <i>DevCapRef</i> or <i>DevCap</i> MUST be specified.
<i>DevNS</i> = "http://www.CIP4.org/JDFSchema_1_1"	URI	Namespace of the Resource or Message that is described.
<i>ID</i> ? New in JDF 1.2	ID	ID of this DevCaps Element. Used for reference from Performance Elements.
<i>LinkUsage</i> ? New in JDF 1.2	enumeration	Used when the <i>Context</i> of this DevCaps = "Resource" or "Link". This field qualifies whether the DevCaps describes a Resource used as an input to a Process or as the output of a Process. Default behavior: this DevCaps applies to both usages. Values are: <i>Input</i> – The DevCaps describes an Input Resource. <i>Output</i> – The DevCaps describes an Output Resource.
<i>ModuleRefs</i> ? 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> Modified in JDF 1.3	NMTOKEN	Name of the Element excluding the namespace prefix. When describing parameters 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. Modification note: starting with JDF 1.3, <i>Name</i> MUST always specify the actual Resource name. Before JDF 1.3, the <i>ResourceUsage</i> and <i>ProcessUsage</i> of a Resource are specified in this Attribute. Values include those from: <i>Section 7, Resources</i>
<i>ProcessUsage</i> ? New in JDF 1.3	NMTOKEN	ResourceLink/@ <i>ProcessUsage</i> of the link to the Resource that is described by this DevCaps. Values include those from: ResourceLink/@ <i>ProcessUsage</i>

Table 7-400: DevCaps Element (Section 3 of 4)

Name	Data Type	Description
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. Values include: <i>None</i> – <i>ResourceUpdate</i> 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/@ <i>ResourceUsage</i> of the Resource that is described by this DevCaps. Values include those from: FileSpec/@ <i>ResourceUsage</i>
TypeOccurrenceNum? New in JDF 1.2	IntegerRange-List	Specifies which occurrence(s) of the JDF Node type that is specified either within the DeviceCap/@ <i>Types</i> or by DeviceCap/@ <i>TypeExpression</i> that the Element that is defined by this DevCaps applies to. If not specified, this DevCaps Element describes Elements belonging to all JDF Nodes or Combined Process steps with a matching type that are not defined by other DevCaps entries. Note: this is an index into the list of matching <i>Type</i> values and not an index into the complete list specified by <i>Types</i> or <i>TypeExpression</i> . 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 <i>Types</i> MUST be a subset of <i>Types</i> in DeviceCap. Values include those from: JDF/@ <i>Types</i> Deprecation note: starting with JDF 1.2, use <i>TypeOccurrenceNum</i> .
DevCap* Modified in JDF 1.3	element	List of definitions of the accepted parameter space for Resources and Messages. The parameter spaces of multiple DevCap Elements are combined as a superset of the individual DevCap Elements. Only Elements that are explicitly specified as DevCap Elements within a DevCaps are supported. When a capabilities description is constructed using constraints, each DevCaps SHOULD contain only a single DevCap Element (although a DevCap Element MAY still contain multiple DevCap Subelements). Exactly one of <i>DevCapRef</i> or <i>DevCap</i> MUST be specified, though if <i>DevCap</i> is specified, it MAY occur multiple times.

Table 7-400: DevCaps Element (Section 4 of 4)

Name	Data Type	Description
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 785.)

7.3.5.1 Loc

[New in JDF 1.2](#)

Each Loc element describes a localization for some value. Note that this Subelement is used in many of the Elements subordinate to DeviceCap Elements.

Table 7-401: 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 7-402: 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. Default value is from: parent DevCaps/@Availability or DevCap/@Availability. 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.
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/@ID 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.
<i>Name</i> ?	NMTOKEN	Name of the Resource that is described. Default, if this DevCap is the direct child of a DevCaps Element: the value of the parent DevCaps/@Name. <i>Name</i> MUST be specified for all direct DevCap child Elements in DevCapPool or DevCap Elements. Modification note: starting with JDF 1.3, <i>Name</i> MUST always specify the actual Resource name. Before JDF 1.3 <i>ResourceUsage</i> of a Resource was specified in this Attribute. Values include those from: <i>Section 7, Resources</i>
ResourceUsage? New in JDF 1.3	NMTOKEN	Resource/@ResourceUsage of the Resource that is described by this DevCap. Values include those from: FileSpec/@ResourceUsage

Table 7-402: DevCap Element (Section 2 of 2)

Name	Data Type	Description
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 = <i>DevCap/@Name</i> .
Loc * New in JDF 1.2	element	The localization(s) of the Element name. (See "Loc" on page 785.)
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 <i>DeviceCap/@GenericAttributes</i> .

7.3.7 State

[New in JDF 1.1](#)

Figure 7-73 shows all State Elements.

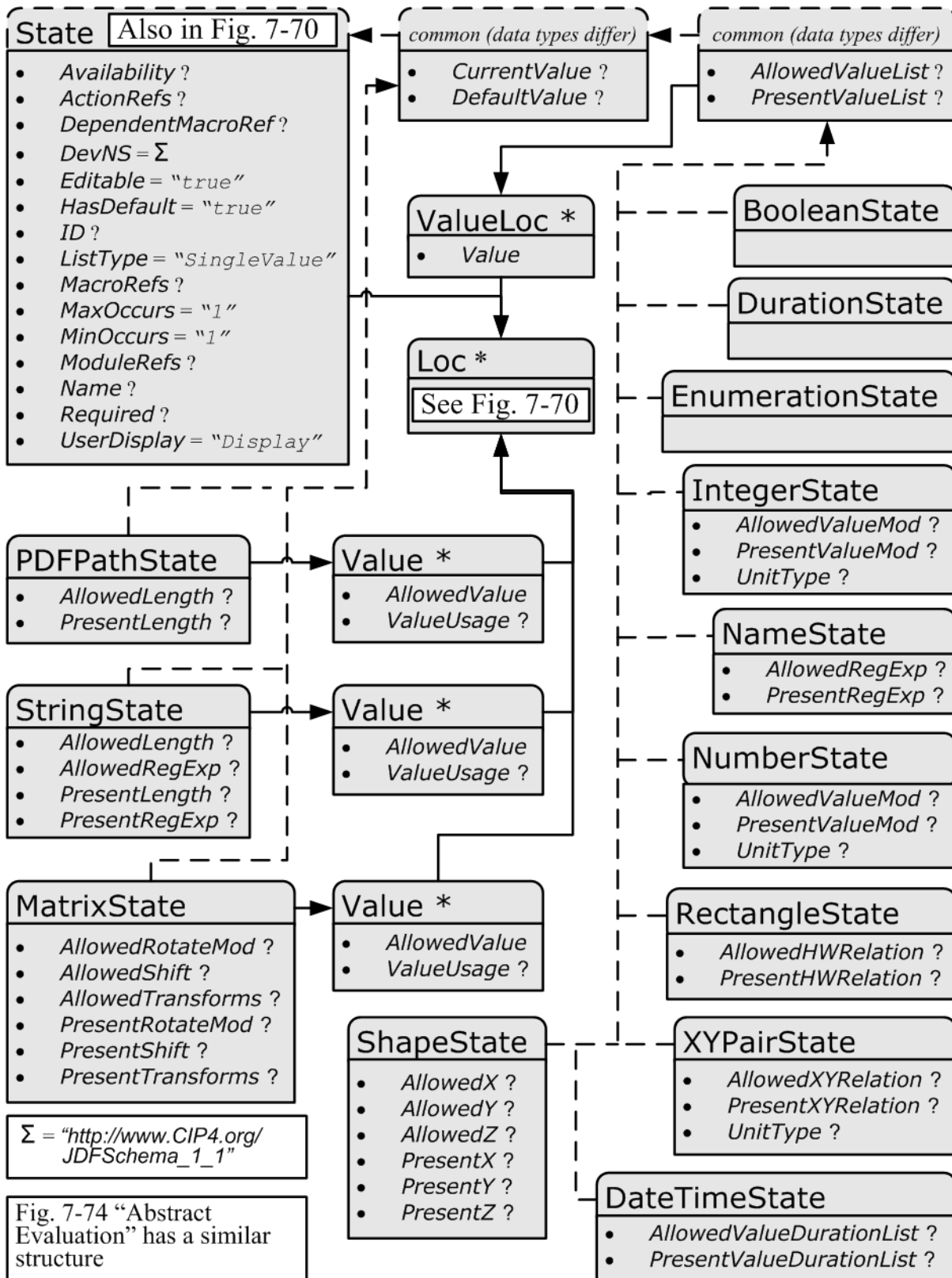


Figure 7-73: Abstract State Element – a diagram of its structure

7.3.7.1 Abstract State Element

Table 7-403 describes the common, data type-independent parameters of all State Elements. The State Elements that contain no value restriction Attributes (e.g. *AllowedValueList*) or Elements (e.g. *ValueLoc*) have no further restrictions other than the data type of their values. If value restrictions are specified in addition to a list of explicit values in *AllowedValueList*, *CurrentValue*, *Value* or *ValueLoc*, the State Element describes the union of restrictions, i.e., the State Element matches an Attribute that matches either the explicit list or the additional restrictions.

Table 7-403: Abstract State Element (Section 1 of 2)

Name	Data Type	Description
<i>Availability?</i> New in JDF 1.2	enumeration	Specifies whether the feature described by this State Element is available on the Device. Default behavior: the value specified or implied by the parent Element 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.
<i>ActionRefs?</i> 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 "false" for the value of the State Element to be valid. Any Action Elements referenced in <i>ActionRefs</i> 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.
<i>DependentMacroRef?</i> 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 "Dependent" 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 804.
<i>DevNS</i> = "http://www.CIP4.org/JDFSchema_1_1"	URI	Namespace of the Attribute that is described by this StateElement.
<i>Editable</i> = "true" New in JDF 1.2	boolean	When "true", the feature and its current value can be edited by the user. If "false", the user interface MUST NOT allow user modification of the State Element'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.
<i>ID?</i> New in JDF 1.2	ID	An identification value to allow external reference.

Table 7-403: Abstract State Element (Section 2 of 2)

Name	Data Type	Description
<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. Values are from: Table 7-404, "ListType Attribute Values" on page 791
<i>MacroRefs</i> ? New in JDF 1.2	IDREFS	Zero or more references to macro Elements that operate on the parameter. These macro Elements set other State Attribute Values as appropriate. Any macro Elements referenced in <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 macro Elements 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 804.
<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).
<i>ModuleRefs</i> ? 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).
<i>Required</i> ? 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.
<i>Span</i> ? 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. 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: this action is only taken when the user interface is first initialized.
Loc * 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 785.)

— Attribute: ListType

Table 7-404: ListType Attribute 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 exactly once and in the specified order.
<i>ContainedList</i>	The State describes a list of individual values. The State = "true" if at least one of the values occurs. This value is only expected to be used in BasicPreflightTest Elements.
<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> New in JDF 1.3	The State describes an individual Range of values.
<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 an Intent Resource.
<i>UniqueList</i>	The State describes a list of individual values. Each value MUST NOT occur more than once.
<i>UniqueRangeList</i>	The State describes a RangeList of values. Each explicit or implied value MUST NOT occur more than once.
<i>UniqueOrderedList</i>	The State describes an ordered list of individual values. Each value MUST NOT occur more than once.
<i>UniqueOrderedRangeList</i>	The State describes an ordered RangeList of individual values. Each explicit or implied value MUST NOT occur more than once.

7.3.7.2 State Elements

The following types of State Elements are defined:

Table 7-405: List of State Elements (Section 1 of 2)

Name	Page	Description
BooleanState	page 792	Describes a set of boolean values.
DateInstanceState New in JDF 1.2	page 793	Describes a set of dateTime values.
DurationState New in JDF 1.2	page 793	Describes a set of duration values.
EnumerationState	page 793	Describes a set of enumeration values.
IntegerState	page 794	Describes a numerical range of integer values.
MatrixState	page 796	Describes a range of matrices. Generally used to define valid orientations of Component Resources.
NameState	page 797	Describes a set of NMTOKEN values.
NumberState	page 797	Describes a numerical range of values.
PDFPathState New in JDF 1.2	page 798	Describes a set of PDFPaths.

Table 7-405: List of State Elements (Section 2 of 2)

Name	Page	Description
RectangleState New in JDF 1.2	page 799	Describes a set of 4 value rectangle values.
ShapeState	page 800	Describes a set of 3 value shape values.
StringState	page 800	Describes a set of string values.
XYPairState	page 801	Describes a set of XYPair values.

7.3.7.2.1 BooleanState[New in JDF 1.1](#)

This State Subelement is used to describe ranges of Boolean values. It inherits from the Abstract State Element described above.

Table 7-406: BooleanState Element

Name	Data Type	Description
<i>AllowedValueList</i> ? New in JDF 1.1A	enumerations	A list of all legal values. Values are: <i>true</i> <i>false</i>
<i>CurrentValue</i> ?	boolean	Current value for the current running Job set in the Device.
<i>DefaultValue</i> ?	boolean	Default value if not specified in a submitted JDF. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = " <i>true</i> ".
<i>PresentValueList</i> ? New in JDF 1.1A	enumerations	A list of all supported values that can be chosen without operator intervention. Default value is from: @ <i>AllowedValueList</i> Values are: <i>true</i> <i>false</i>
<i>ValueLoc</i> * 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 792.)

7.3.7.2.1.1 ValueLoc[New in JDF 1.2](#)

Each ValueLoc Element describes one or more localizations for an Attribute Value. Note that the ValueLoc Element occurs in the definition of all State Elements except MatrixState, PDFPathState and StringState.

Table 7-407: 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 785.)

7.3.7.2.2 DateTimeState

[New in JDF 1.2](#)

This State Subelement is used to describe ranges of dateTime values. It inherits from the Abstract State Element described above.

Table 7-408: DateTimeState Element

Name	Data Type	Description
<i>AllowedValueDurationList</i> ?	DurationRangeList	List of inclusive minimum and maximum allowed values relative to the current system time.
<i>AllowedValueList</i> ?	DateTimeRangeList	A list of all supported values.
<i>CurrentValue</i> ?	dateTime	Current value for the current running Job set in the Device.
<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 792.)

7.3.7.2.3 DurationState

[New in JDF 1.2](#)

This State Subelement is used to describe ranges of duration values. It inherits from the Abstract State Element described above.

Table 7-409: 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 792.)

7.3.7.2.4 EnumerationState

[New in JDF 1.1](#)

This State Subelement is used to describe ranges of enumerative values. It inherits from the Abstract State Element described above. It is identical to the NameState Element except that it describes a closed list of enumeration values.

Table 7-410: EnumerationState Element

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.
<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. Default value is from: @ <i>AllowedValueList</i>
ValueLoc * 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 792.)

7.3.7.2.5 IntegerState

[New in JDF 1.1](#)

This State Subelement is used to describe ranges of integer values. It inherits from the Abstract State Element described above.

Table 7-411: 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: $4 \bmod 3 = 4 - 3 = 1$; $17 \bmod 3 = 17 - 5 \cdot 3 = 2$; and $3 \bmod 3 = 3 - 3 = 0$.
<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.

Table 7-411: IntegerState Element (Section 2 of 2)

Name	Data Type	Description
PresentValueMax? 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.
PresentValueMin? 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.
PresentValueMod? 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.
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. 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 792.)

7.3.7.2.6 MatrixState

[New in JDF 1.1](#)

This State Subelement is used to describe ranges of matrix values. It inherits from the Abstract State Element described above. It is primarily intended to specify orientations and manipulation capabilities of Physical Resources, (e.g., in finishing Devices).

Table 7-412: MatrixState Element

Name	Data Type	Description
<i>AllowedRotateMod</i> ? 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.
<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 30 is subtracted from <i>AllowedShift</i> , thus all in-place rotations have an implied <i>AllowedShift</i> value of " <i>0 0 0 0</i> ". (No shift = " <i>0 0 0 0</i> ".) 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 30.
<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> = " <i>true</i> ".
<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: <i>360</i> – No rotation is allowed. <i>90</i> – Any orthogonal rotation. <i>0</i> – 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 30 is subtracted from <i>PresentShift</i> , thus all in-place rotations have an implied <i>PresentShift</i> value of " <i>0 0 0 0</i> ". 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 30. If not specified, the value of <i>AllowedTransforms</i> is applied.
Value *	element	A list legal values. See Section 7.3.7.2.6.1, Value.

7.3.7.2.6.1 Value

Table 7-413: MatrixState/Value Element (Section 1 of 2)

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> .

Table 7-413: MatrixState/Value Element (Section 2 of 2)

Name	Data Type	Description
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. Default behavior: valid for both <i>Present</i> and <i>Allowed</i> . Values are: <i>Present</i> – Present configuration is supported. <i>Allowed</i> – Allowed configuration is supported.
Loc * New in JDF 1.2	element	The localization(s) of the string defined in <i>AllowedValue</i> . (See “Loc” on page 785.)

7.3.7.2.7 NameState[New in JDF 1.1](#)

This State Subelement is used to describe ranges of NMTOKEN values. It inherits from the Abstract State Element described above.

Table 7-414: 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 792.)

7.3.7.2.8 NumberState[New in JDF 1.1](#)

This State Subelement is used to describe ranges of double values. It inherits from the Abstract State Element described above.

Table 7-415: 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.

Table 7-415: NumberState Element (Section 2 of 2)

Name	Data Type	Description
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: $4 \text{ mod } 3 = 4 - 3 = 1$; $17 \text{ mod } 3 = 17 - 5*3 = 2$; and $3 \text{ mod } 3 = 3 - 3 = 0$.
<i>CurrentValue?</i>	double	Current value for the current running Job set in the Device.
<i>DefaultValue?</i>	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.
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 State Element represents. Used to enable an application to localize the representation of the units. <i>UnitType</i> MUST be specified if the NumberState represents a value that has units. NumberState has the values as IntegerState plus a few more: Values include: <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. Values include those from: IntegerState/@UnitType (Table 7-411, "IntegerState Element" on page 794)
ValueLoc? New in JDF 1.2	element	Localization(s) of specific values. (See Structure of the ValueLoc Subelement under "BooleanState" on page 792.)

7.3.7.2.9 PDFPathState

[New in JDF 1.2](#)

This State Subelement is used to describe ranges of PDF paths. It inherits from the Abstract State Element described above.

Table 7-416: PDFPathState Element (Section 1 of 2)

Name	Data Type	Description
<i>AllowedLength?</i>	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.

Table 7-416: PDFPathState Element (Section 2 of 2)

Name	Data Type	Description
<i>CurrentValue?</i>	PDFPath	Current value for the current running Job set in the Device.
<i>DefaultValue?</i>	PDFPath	Default value if not specified in a submitted JDF. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = "true".
<i>PresentLength?</i>	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> . See Section 7.3.7.2.9.1, Value.

7.3.7.2.9.1 Value**Table 7-417: PDFPathState/Value Element**

Name	Data Type	Description
<i>AllowedValue</i>	PDFPath	A legal value for a matrix variable.
<i>ValueUsage?</i>	enumeration	Defines whether the value defined in <i>AllowedValue</i> means <i>Present</i> , <i>Allowed</i> or both. Default behavior: valid for both <i>Present</i> and <i>Allowed</i> . Values are: <i>Present</i> – Present configuration is supported. <i>Allowed</i> – Allowed configuration is supported.
Loc *	element	The localization(s) of the string defined in <i>AllowedValue</i> . (See "Loc" on page 785.)

7.3.7.2.10 RectangleState[New in JDF 1.2](#)

This State Subelement is used to describe ranges of rectangle values. It inherits from the Abstract State Element described above.

Table 7-418: 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 792.)

7.3.7.2.11 ShapeState

[New in JDF 1.1](#)

This State Subelement is used to describe ranges of *Shape* values. It inherits from the Abstract State Element described above.

Table 7-419: ShapeState Element

Name	Data Type	Description
AllowedValueList? Modified in JDF 1.2	ShapeRange-List	A list of values that can be chosen.
AllowedValueMax? Deprecated in JDF 1.2	shape	Inclusive maximum allowed value. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
AllowedValueMin? Deprecated in JDF 1.2	shape	Inclusive minimum allowed value. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
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 792.)

7.3.7.2.12 StringState

[New in JDF 1.1](#)

This State Subelement is used to describe ranges of string values. It inherits from the Abstract State Element described above.

Table 7-420: StringState Element

Name	Data Type	Description
<i>AllowedLength</i> ? 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”).
<i>AllowedRegExp</i> ? New in JDF 1.2	regExp	Regular expression that limits the allowed values.
<i>CurrentValue</i> ?	string	Current value for the current running Job set in the Device.
<i>DefaultValue</i> ?	string	Default value if not specified in a submitted JDF. <i>DefaultValue</i> MUST be specified if <i>HasDefault</i> = “true”.
<i>PresentLength</i> ? 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.
<i>PresentRegExp</i> ? 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. See Section 7.3.7.2.12.1, Value.

7.3.7.2.12.1 Value

[New in JDF 1.1](#)

Table 7-421: StringState/Value Element

Name	Data Type	Description
<i>AllowedValue</i>	string	A legal value for a string variable.
<i>PresentValue</i> ? 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> .
<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. Default behavior: valid for both <i>Present</i> and <i>Allowed</i> . Values are: <i>Present</i> – Present configuration is supported. <i>Allowed</i> – Allowed configuration is supported.
Loc * New in JDF 1.2	element	The localization(s) of the string defined in <i>AllowedValue</i> . (See “Loc” on page 785.)

7.3.7.2.13 XYPairState

[New in JDF 1.1](#)

This State Subelement is used to describe ranges of XYPair values. It inherits from the Abstract State Element described above.

Table 7-422: XYPairState Element (Section 1 of 2)

Name	Data Type	Description
<i>AllowedValueList</i> ? Modified in JDF 1.2	XYPair-RangeList	A list of values that can be chosen.

Table 7-422: XYPairState Element (Section 2 of 2)

Name	Data Type	Description
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.
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 State 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. Values include those from: <i>IntegerState/@UnitType</i> (Table 7-411, "IntegerState Element" on page 794)
ValueLoc* New in JDF 1.2	element	A list of supported shapes. (See Structure of the ValueLoc Subelement under "BooleanState" on page 792.)

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.

Table 7-423: DisplayGroupPool Element

Name	Data Type	Description
DisplayGroup*	element	Declares a set of references to State Elements that are intended to be displayed as a group in user interfaces.

Example 7-57: DisplayGroupPool

In this example, a single DisplayGroup is specified. This DisplayGroup declares that the State Attributes with *ID*'s "btd", "cmp", "mag", "colorspace" and "outputres" are all to be grouped together in any user interface. The English string "ScanningParameters" is associated with this DisplayGroup, though no explicit assumptions are made about how to display this group of Attributes. The DisplayGroup Element merely states that there is a user-significant relationship between the Attributes.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Device Class="Implementation" ID="Link0003" Status="Available">
      <DeviceCap>
```

```

    <DisplayGroupPool>
      <DisplayGroup rRefs="bad">
        <Loc HelpText="Parameters for scanning configuration" Lang="en"
          Value="ScanningParameters" />
      </DisplayGroup>
    </DisplayGroupPool>
  </DeviceCap>
</Device>
</ResourcePool>
<ResourceLinkPool>
  <DeviceLink Usage="Input" rRef="Link0003" />
</ResourceLinkPool>
</JDF>

```

7.3.8.1 DisplayGroup

Each `DisplayGroup` Element declares a group of features that are intended to be displayed together in user interfaces.

Table 7-424: DisplayGroup Element

Name	Data Type	Description
<i>rRefs</i>	IDREFS	References to State Elements. (See "State" on page 787 for details of the State Element.)
Loc *	element	Localized strings describing the <code>DisplayGroup</code> . See Section 7.3.5.1, "Loc" on page 785..

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 804.

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`

Table 7-425: 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 <code>macro</code> Elements. For more information on macro definitions, see "MacroPool" on page 804. For details of the State Element, see "State" on page 787.

Example 7-58: FeaturePool

In this example, `ScanMode` is a feature that doesn't map directly to any JDF Resource or Attribute, but provides a "shell" feature that allows users to control a set of JDF Resources and/or Attributes to indicate a common or preferred grouping based on the user's desired task. The actual corresponding JDF Resource Attribute Values are determined and set by the `ScanModeMacro` macro that is called when the `ScanMode` feature is manipulated.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Device Class="Implementation" ID="Link0003" Status="Available">

```

```

    <DeviceCap>
      <FeaturePool>
        <EnumerationState
          AllowedValueList="Mono ColorTransparency Photo" ID="sm"
          HasDefault="false" MacroRefs="ScanModeMac" Name="ScanMode"
          UserDisplay="Display"/>
        </FeaturePool>
      </DeviceCap>
    </Device>
  </ResourcePool>
  <ResourceLinkPool>
    <DeviceLink Usage="Input" rRef="Link0003"/>
  </ResourceLinkPool>
</JDF>

```

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 7-426: `MacroPool` Element

Name	Data Type	Description
macro *	element	A list of independent macros.

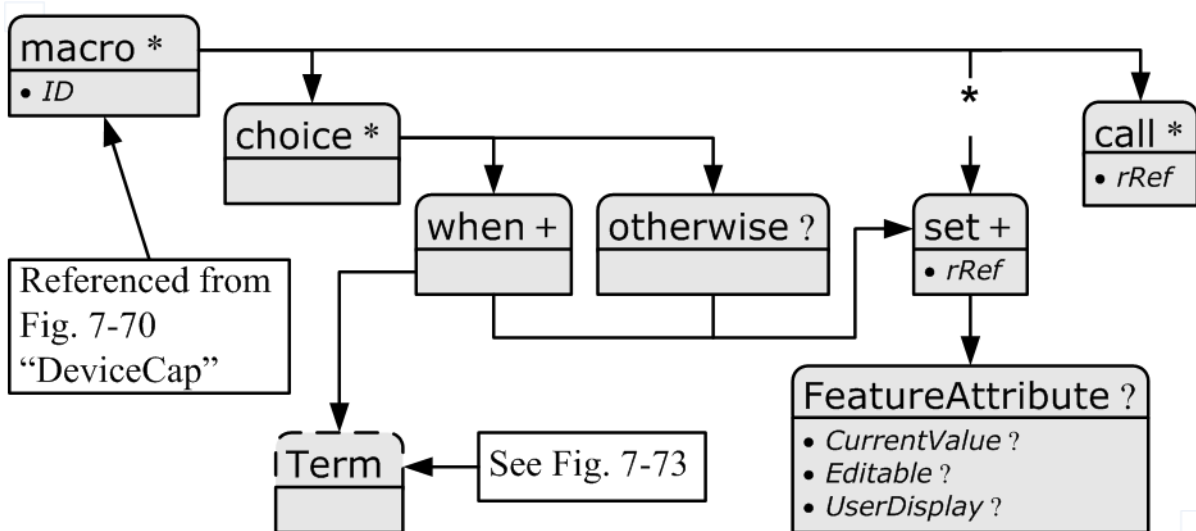


Figure 7-74: `macro` Element – a diagram of its structure

7.3.10.1 macro

[New in JDF 1.2](#)

The `macro` Subelement is used to contain a set of conditional operations that are used to change `State Element` Attribute Values. Each `macro` contains one or more of the following Elements:

- `choice` — Declares one or more `when` statements, each of which contains a Boolean expression (as defined in "Term" on page 807) 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 7-427: 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 .

7.3.10.2 choice

The **choice** Subelement is used to contain expressions that declare conditional operations that can cause **State Element Attribute Values** to be changed. The **choice** includes one or more **when** statements that are evaluated in order, each of which contains a Boolean expression (as defined in "**Term**" on page 807) 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 7-428: choice Element

Name	Data Type	Description
when +	element	A set of conditional operations that set (or not) feature values.
otherwise ?	element	An Element that sets one or more State Element Attribute Values if none of the when expressions evaluate to " true ".

7.3.10.3 otherwise

The **otherwise** Subelement sets one or more feature values if none of the **when** expressions in a **choice** Element evaluate to "**true**".

Table 7-429: otherwise Element

Name	Data Type	Description
set +	element	An element that sets one or more feature values.

7.3.10.4 when

The **when** Subelement is used to contain expressions that declare conditional operations to enforce sets of feature behaviors. The **when** Element includes a Boolean expression (as defined in "**Term**" on page 807) and a **set** Element. When the **Term** evaluates to "**true**", the action specified in the **set** Element is to be performed.

Table 7-430: 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.

7.3.10.5 set

The `set` Subelement sets one or more State Element Attribute Values.

Table 7-431: set Element

Name	Data Type	Description
<i>rRef</i>	IDREF	Reference to a State Element referring to the feature value to set
<i>FeatureAttribute?</i>	element	Specifies one or more Attributes within the State Element that are to have their value changed (along with the value they change to).

7.3.10.6 FeatureAttribute

`FeatureAttribute` specifies one or more Attributes of a State Element that are to have their value changed. The following Attributes can be changed:

Table 7-432: FeatureAttribute Element

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.
<i>UserDisplay?</i>	enumeration	Indicates under which conditions the feature is to be displayed in user interfaces. Values are from: <i>State/@UserDisplay</i>

7.3.10.7 call

The `call` Subelement is used to call other macro Elements, effectively using them as macro “templates”.

Table 7-433: 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 7-434: Performance Element (Section 1 of 2)

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 <code>DevCaps</code> Element that describes the Resource whose performance is specified by this Performance 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> .

Table 7-434: Performance Element (Section 2 of 2)

Name	Data Type	Description
<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). Deprecation note: starting with JDF 1.2, use <i>DevCapsRef</i> .
<i>Unit?</i>	NMTOKEN	Unit of measure of Resource consumption per hour. Default value is from: Resource's generic units as defined in Table 1-7, "Units used in JDF" on page 16.

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 7-435: TestPool Element

Name	Data Type	Description
Test *	element	A list of independent Test Elements.

7.3.12.1 Test

The Test Subelement is used to contain Boolean expressions that are for use only when referenced by another Test or Action and are not evaluated independently. Its purpose is to simplify the description of other Test Elements and macro Elements by representing a commonly used Boolean expression.

Table 7-436: Test Element

Name	Data Type	Description
<i>ID</i>	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

Figure 7-75 shows all Term Elements

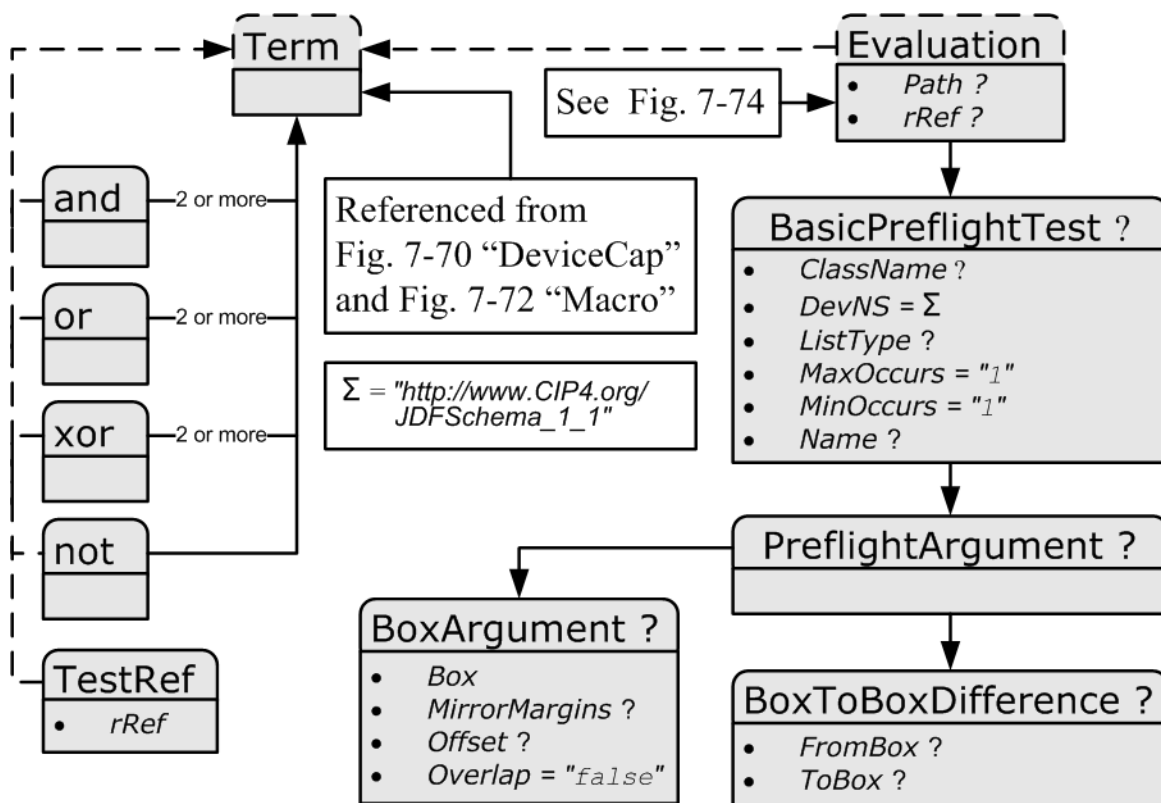


Figure 7-75: Abstract Term Element – a diagram of its structure

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 Elements in Table 7-437. The Term Elements that are Boolean operators MAY be nested. They are used both in Device capabilities and preflighting context.

Note: in the actual JDF schema, several Abstract Element definitions are used to create an appropriate inheritance structure. Rather than reproduce this here, only the actual non-Abstract Elements that will appear in JDF files will be described

Table 7-437: List of Term Elements

Name	Page	Description
and	page 809	Boolean AND operator.
not	page 810	Boolean negation.
or	page 809	Boolean OR operator.
xor	page 810	Boolean exclusive or (XOR) operator.
TestRef	page 810	Reference to a constraint Test Element to be evaluated as a nested Boolean expression inside a larger expression.
Evaluation	page 810	Elements, which evaluate a JDF State Attribute Value to create a simple Boolean expression, e.g., "Is the value of <i>BitDepth</i> equal to 8?". Each XXXExpression Element is derived from the Abstract Evaluation Element

Example 7-59: ActionPool and TestPool

Term is an Abstract Element, so it will never appear in a JDF document. In this "ctcmp" constraint example, the Term is represented by the and Element. Since the Term Element itself is Abstract, what will actually appear in constraints will be Boolean expressions. In this example, the logic is, "We can not use CCITT compression if the bit depth is not 1 bit." The check for compression type uses an EnumerationEvaluation Element, which evaluates an EnumerationState value against "CCITTFaxEncode". If the value of the EnumerationState Element referred to by "cmp" = CCITTFaxEncode, the EnumerationEvaluation evaluates as "true". The check for "btd" is accomplished through a TestRef to the "is1bit" constraint. The and and not Elements behave according to the standard semantics for Boolean combinatorial logic.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Device Class="Implementation" ID="Link0003" Status="Available">
      <DeviceCap>
        <ActionPool>
          <Action ID="MyAction" TestRef="ctcmp">
            <Loc HelpText="Only select CCITTFaxEncoding for 1 bit documents"
              Lang="en" ShortValue="Ouch!"
              Value="CCITTFaxEncoding not supported on
                grayscale images"/>
          </Action>
        </ActionPool>
        <TestPool>
          <Test ID="ctcmp">
            <!-- Can't CCITT compress anything but 1 bit grayscale -->
            <and>
              <not>
                <TestRef rRef="is1bit"/>
              </not>
              <EnumerationEvaluation ValueList="CCITTFaxEncode" rRef="cmp"/>
            </and>
          </Test>
          <Test ID="is1bit">
            <IntegerEvaluation ValueList="1" rRef="btd"/>
          </Test>
        </TestPool>
      </DeviceCap>
    </Device>
  </ResourcePool>
  <ResourceLinkPool>
    <DeviceLink Usage="Input" rRef="Link0003"/>
  </ResourceLinkPool>
</JDF>
```

7.3.13.1 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 7-438: and Element

Name	Data Type	Description
Term	element	Any Element derived from an Abstract Term.
Term +	element	Any Element derived from an Abstract Term.

7.3.13.2 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 7-439: or Element

Name	Data Type	Description
Term	element	Any Element derived from an Abstract Term.
Term +	element	Any Element derived from an Abstract Term.

7.3.13.3 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 7-440: xor Element

Name	Data Type	Description
Term	element	Any Element derived from an Abstract Term.
Term +	element	Any Element derived from an Abstract Term.

7.3.13.4 not

The `not` Subelement inverts the Boolean state of a `Term`.

Table 7-441: not Element

Name	Data Type	Description
Term	element	Any Element derived from an Abstract Term.

7.3.13.5 TestRef

The `TestRef` Element refers to another constraint that is to be evaluated as part of the parent constraint.

Table 7-442: TestRef Element

Name	Data Type	Description
<i>rRef</i>	IDREF	Reference to a <code>Test</code> to be evaluated as a nested Boolean expression inside a larger expression.

7.3.13.6 Evaluation

Figure 7-76 shows all Evaluation Elements

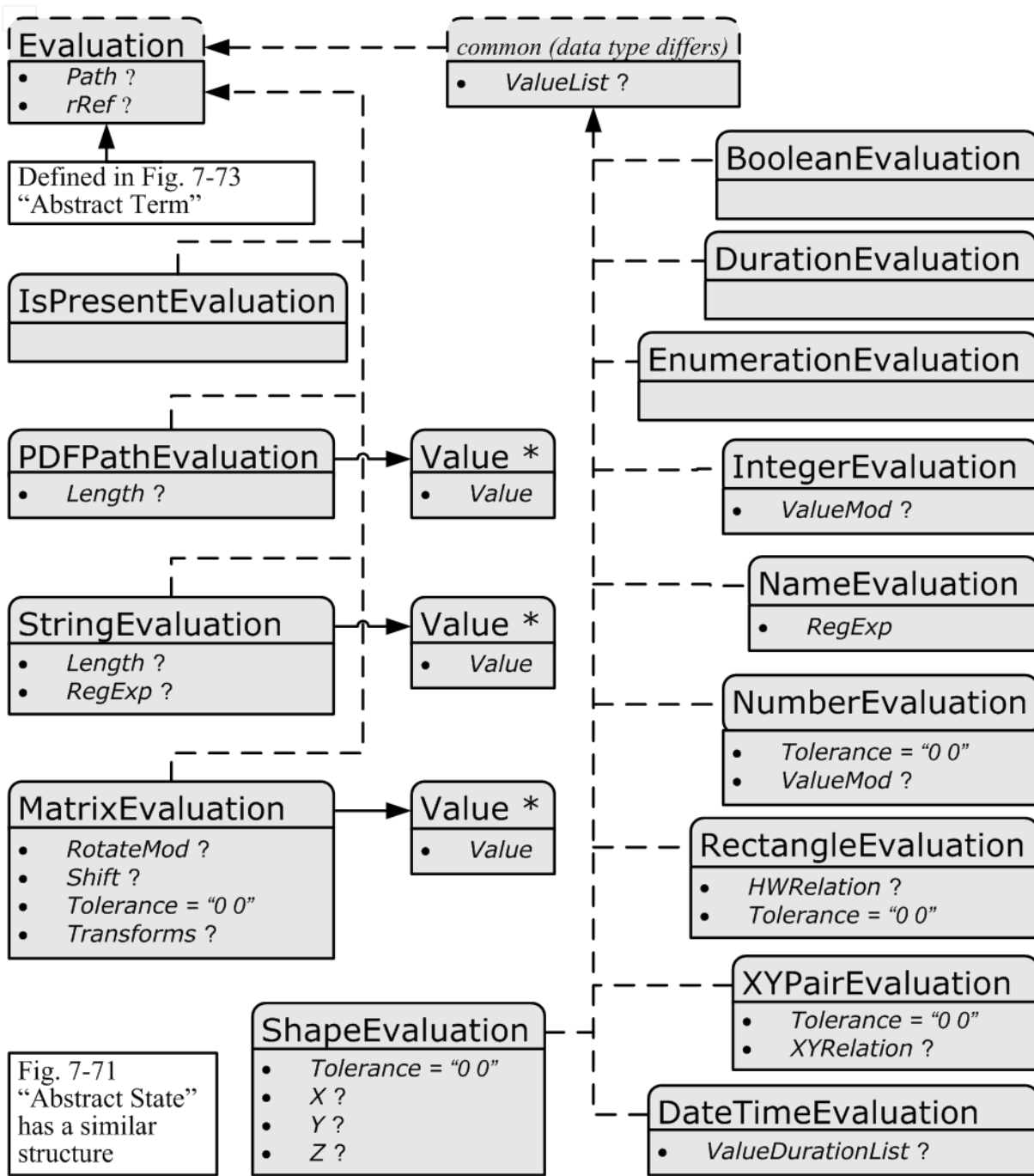


Figure 7-76: Abstract Evaluation Element – a diagram of its structure

7.3.13.6.1 Abstract Evaluation

The following table describes the common, data type-independent parameters of all Evaluation Elements.

Table 7-443: Abstract Evaluation Element

Name	Data Type	Description
<i>Path</i> ? New in JDF 1.4	XPath	When present, describes an XPath within the file where the value to be evaluated may be found. Constraint: Exactly one of <i>Path</i> , <i>rRef</i> or <i>BasicPreflightTest</i> MUST be specified.
<i>rRef</i> ? Modified in JDF 1.4	IDREF	A reference to <i>State</i> , <i>DevCap</i> , <i>DevCaps</i> or <i>Module Elements</i> when used in the context of Device capability descriptions. Constraint: Exactly one of <i>Path</i> , <i>rRef</i> or <i>BasicPreflightTest</i> MUST be specified. Modification note: starting with JDF 1.4, <i>DevCap</i> and <i>DevCaps</i> can also be referenced and <i>Path</i> added to the all constraints in this table.
<i>BasicPreflightTest</i> ?	element	Definition of the preflight basic test to which the <i>Evaluation</i> refers. <i>BasicPreflightTest</i> is only valid when <i>Evaluation Elements</i> are used in the context of preflighting. The <i>Evaluation Elements</i> in capability descriptions MUST reference the appropriate <i>State Element</i> using <i>rRef</i> . For details of the <i>BasicPreflightTest</i> , see "PreflightParams" on page 676. Constraint: Exactly one of <i>Path</i> , <i>rRef</i> or <i>BasicPreflightTest</i> MUST be specified.

7.3.13.6.2 Evaluation Elements

Evaluation Elements map generalized tests against a condition to form a true or false Boolean state that can be evaluated using the Boolean logic defined below.

Table 7-444: List of Abstract Evaluation Elements

Name	Page	Description
BooleanEvaluation	page 813	Describes operations on a set of Boolean values.
DateTimeEvaluation	page 813	Describes operations on a set of dateTime values.
DurationEvaluation	page 814	Describes operations on a set of duration values.
EnumerationEvaluation	page 814	Describes operations on a set of enumeration values.
IntegerEvaluation	page 814	Describes operations on a numerical range of integer values.
IsPresentEvaluation	page 814	Checks for the existence of a tag, Element or feature.
MatrixEvaluation	page 814	Describes operations on a range of matrices. Generally used to define valid orientations of Component Resources.
NameEvaluation	page 815	Describes operations on a set of NMTOKEN values
NumberEvaluation	page 815	Describes operations on a numerical range of values.
PDFPathEvaluation	page 816	Describes operations on PDFPath.
RectangleEvaluation	page 816	Describes operations on a set of four-value rectangle values.
ShapeEvaluation	page 816	Describes operations on a set of three-value shape values.
StringEvaluation	page 817	Describes operations on a set of string values.
XYPairEvaluation	page 817	Describes operations on a set of XYPair values.

Mapping of Evaluation Element to State Element

When used in a Device capabilities context, the Evaluation Elements map to the State Elements (i.e., BooleanState, IntegerState, etc.). These Elements each declare individual JDF Attributes for a Device capabilities description. The Evaluation Elements are instances of Term Elements that compare the value of a given State

Attribute against a condition to form a true or false Boolean statement. The form of the condition depends on the type of the Evaluation–State Element pairing — different types of pairings need different condition declarations, depending on the structure of the logic and the data type of the Evaluation and State Elements.

When used in a preflighting context, Evaluation Elements map named preflight tests against a condition to form a true or false Boolean statement.

Table 7-445: Mapping of Evaluation Element to State Element

Name	Corresponding State Element	Description
BooleanEvaluation	BooleanState	Describes operations on a set of Boolean values.
DateTimeEvaluation	DateTimeState	Describes operations on a set of dateTime values.
DurationEvaluation	DurationState	Describes operations on a set of duration values.
EnumerationEvaluation	EnumerationState	Describes operations on a set of enumeration values.
IntegerEvaluation	IntegerState	Describes operations on a numerical range of integer values.
IsPresentEvaluation	State (all)	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.

7.3.13.6.2.1 BooleanEvaluation

The BooleanEvaluation Element declares a Boolean value for comparison in an expression to a BooleanState Element in constraints. It inherits from the Abstract Evaluation Element described above.

Table 7-446: BooleanEvaluation Element

Name	Data Type	Description
<i>ValueList?</i>	enumerations	A list of all supported values. Values are: <i>true</i> <i>false</i>

7.3.13.6.2.2 DateTimeEvaluation

The DateTimeEvaluation Element declares a Boolean value for comparison in an expression to a DateTimeState Element in constraints.

Table 7-447: 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.

7.3.13.6.2.3 DurationEvaluation

The `DurationEvaluation` Element declares a Boolean value for comparison in an expression to a `DurationState` Element in constraints. It inherits from the `Abstract Evaluation` Element described above.

Table 7-448: DurationEvaluation Element

Name	Data Type	Description
<code>ValueList?</code>	<code>DurationRangeList</code>	A list of all supported values.

7.3.13.6.2.4 EnumerationEvaluation

The `EnumerationEvaluation` Element declares an enumeration value for comparison in an expression to an `EnumerationState` Element in constraints.

Table 7-449: EnumerationEvaluation Element

Name	Data Type	Description
<code>ValueList?</code>	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 <code>ValueList</code> , even when the entire range of schema-valid values is supported.

7.3.13.6.2.5 IntegerEvaluation

The `IntegerEvaluation` Element declares an integer value for comparison in an expression to a `IntegerState` Element in constraints.

Table 7-450: IntegerEvaluation Element

Name	Data Type	Description
<code>ValueList?</code>	<code>IntegerRangeList</code>	A list of all supported values.
<code>ValueMod?</code>	<code>XYPair</code>	X defines the Modulo and Y the offset of the allowed value. In other words, if <code>AllowedValueMod = "10 2"</code> , 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 $3 \bmod 3 = 3 - 3 = 0$.

7.3.13.6.2.6 IsPresentEvaluation

The `IsPresentEvaluation` Element checks for the existence of a tag, module or feature. It inherits from the `Abstract Evaluation` Element described above and has no additional Attributes. `IsPresentEvaluation/@rRef` MAY reference a `DevCap` Element in order to test for the existence of an Element.

`IsPresentEvaluation/@rRef` MAY reference a `DevCaps` Element in order to test for the existence of a Resource.

Table 7-451: IsPresentEvaluation Element

Name	Data Type	Description

7.3.13.6.2.7 MatrixEvaluation

The `MatrixEvaluation` Element declares a matrix value for comparison in an expression to a `MatrixState` Element in constraints.

Table 7-452: MatrixEvaluation Element

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 30 is subtracted from <i>Shift</i> , thus all in-place rotations have an implied Shift value of "0 0 0".
<i>Tolerance = "0 0"</i>	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.
<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 30.
<i>Value *</i>	element	A list supported values. The <i>Value/@Value</i> Attribute MUST be a representation of a matrix. See Section 7.3.13.6.2.7.1, <i>Value</i> .

7.3.13.6.2.7.1 Value**Table 7-453: MatrixEvaluation/Value Element**

Name	Data Type	Description
<i>Value</i>	matrix	A supported value for a matrix variable.

7.3.13.6.2.8 NameEvaluation

The *NameEvaluation* Element declares a NMTOKEN value for comparison in an expression to a *NameState* Element in constraints.

Table 7-454: 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.

7.3.13.6.2.9 NumberEvaluation

The *NumberEvaluation* Element declares a number value for comparison in an expression to a *NumberState* Element in constraints.

Table 7-455: NumberEvaluation Element (Section 1 of 2)

Name	Data Type	Description
<i>Tolerance = "0 0"</i>	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.

Table 7-455: NumberEvaluation Element (Section 2 of 2)

Name	Data Type	Description
<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 $3 \bmod 3 = 3 - 3 = 0$.

7.3.13.6.2.10 PDFPathEvaluation

The PDFPathEvaluation Element declares a PDF path value for comparison in an expression to a PDFPathState Element in constraints.

Table 7-456: 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. See Section 7.3.13.6.2.10.1, <i>Value</i>

7.3.13.6.2.10.1 Value**Table 7-457: PDFPathEvaluation/Value Element**

Name	Data Type	Description
<i>Value</i>	PDFPath	A supported value for a PDF path Attribute.

7.3.13.6.2.11 RectangleEvaluation

The RectangleEvaluation Element declares a Boolean value for comparison in an expression to a RectangleState Element in constraints.

Table 7-458: RectangleEvaluation Element

Name	Data Type	Description
<i>HWRelation</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.

7.3.13.6.2.12 ShapeEvaluation

The ShapeEvaluation Element declares a shape value for comparison in an expression to a ShapeState Element in constraints.

Table 7-459: ShapeEvaluation Element (Section 1 of 2)

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.

Table 7-459: ShapeEvaluation Element (Section 2 of 2)

Name	Data Type	Description
<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 .

7.3.13.6.2.13 StringEvaluation

The **StringEvaluation** Element declares a string value for comparison in an expression to a **StringState** Element in constraints.

Table 7-460: 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. See Section 7.3.13.6.2.13.1, Value .

7.3.13.6.2.13.1 Value**Table 7-461: StringEvaluation/Value Element**

Name	Data Type	Description
<i>Value</i>	string	A supported value for a string Attribute.

7.3.13.6.2.14 XYPairEvaluation

The **XYPairEvaluation** Element declares a **XYPair** value for comparison in an expression to a **XYPairState** Element in constraints.

Table 7-462: XYPairEvaluation Element

Name	Data Type	Description
<i>Tolerance = "0 0"</i>	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.

7.3.14 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.

7.3.14.1 Device Description of a Scanner

This first example shows the general structure and provides an example of user interface localization (the query requests localization for the French language, and localizations are returned for the **ScanParams** Resource).

Example 7-60: KnownDevices Query for a Scanner

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" MaxVersion="1.4" Version="1.4"
  TimeStamp="2005-04-05T16:45:43+02:00" SenderID="Controller"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <Query ID="DeviceQuery" Type="KnownDevices" xsi:type="QueryKnownDevices">
    <DeviceFilter DeviceDetails="Capability" Localization="fre"/>
  </Query>
</JMF>
```

Example 7-61: KnownDevices Response for a Scanner

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Scanner"
  TimeStamp="2005-06-05T16:45:43+02:00" MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <Response ID="xyz" Type="KnownDevices" refID="DeviceQuery"
    xsi:type="ResponseKnownDevices" >
    <DeviceList>
      <DeviceInfo DeviceStatus="Idle">
        <Device Class="Implementation" DeviceID="Joe the Drum"
          KnownLocalizations="En Fre" ModelName="Bongo" >
          <DeviceCap GenericAttributes="ID Class SettingsPolicy
            BestEffortExceptions OperatorInterventionExceptions
            MustHonorExceptions PartIDKeys DocIndex"
            Lang="Fre" Type="Scanning">
            <!-- the scanner takes a minute to set up and scans an average
              of 2 sheets a min. -->
            <Performance AverageAmount="120" AverageSetup="PT2M"
              Name="ExposedMedia"/>
            <DevCaps Name="NodeInfo">
              <DevCap>
                <!--NodeInfo only supports JobPriority and
                  TargetRoute Attributes -->
                <StringState Name="TargetRoute" HasDefault="false"/>
                <IntegerState Name="JobPriority" HasDefault="false"/>
              </DevCap>
            </DevCaps>
            <DevCaps Name="ExposedMedia">
              <DevCap>
                <!-- ExposedMedia restrictions -->
                <DevCap Name="Media">
                  <NameState DefaultValue="Sheet" Name="MediaUnit"/>
                  <XYPairState AllowedValueMax="600 1200"
                    AllowedValueMin="0 0"
                    Name="Dimension" HasDefault="false"/>
                </DevCap>
              </DevCap>
            </DevCaps>
            <DevCaps Name="ScanParams">
              <Loc HelpText="Les parametres pour commander le
                procede de balayage."
                Value="Les parametres de module de balayage"/>
              <DevCap>
                <!-- Black and white 1 bit mode -->
                <IntegerState AllowedValueMax="1" AllowedValueMin="1"
                  DefaultValue="8" Name="BitDepth"/>
                <EnumerationState AllowedValueList="CCITTFaxEncode None"
                  Name="CompressionFilter" HasDefault="false">
                  <Loc HelpText="Choisissez la compression pour reduire la
                    taille de donnees."
                    Value="La compression de donnees"/>
                  <ValueLoc Value="CCITTFaxEncode">
                    <Loc Value="Compression de CCITT Fax"/>
                  </ValueLoc>
                </DevCap>
              </DevCaps>
            </DevCaps>
          </DeviceCap>
        </DeviceInfo>
      </DeviceList>
    </ResponseKnownDevices>
  </Response>
</JMF>
```

```

    <ValueLoc Value="None">
      <Loc Value="Aucun compression"/>
    </ValueLoc>
  </EnumerationState>
</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>

```

7.3.14.2 Device Description of a Scanner #2

This second example illustrates the use of constraints, macros and `DisplayGroup` Elements in a capability response. For the sake of simplicity, the only localizations returned are for the constraints.

Example 7-62: KnownDevices Query for a Scanner #2

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Controller"
  TimeStamp="2005-04-05T16:45:43+02:00" MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <Query ID="DeviceQuery" Type="KnownDevices" xsi:type="QueryKnownDevices">
    <DeviceFilter DeviceDetails="Capability" Localization="en"/>
  </Query>
</JMF>
```

Example 7-63: KnownDevices Response for a Scanner #2

```
<JMF SenderID="Scanner" TimeStamp="2004-10-17T14:30:47Z"
  xmlns="http://www.CIP4.org/JDFSchema_1_1" MaxVersion="1.4" Version="1.4"
  DescriptiveName="Example from JDF 1.2 Spec Document"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <Response ID="xyz" Type="KnownDevices" refID="DeviceQuery" ReturnCode="0"
    Acknowledged="false" xsi:type="ResponseKnownDevices" >
    <DeviceList>
      <DeviceInfo DeviceStatus="Idle">
        <Device DeviceID="Joe the Drum" ModelName="Bongo">
          <DeviceCap GenericAttributes="ID Class SettingsPolicy
            BestEffortExceptions OperatorInterventionExceptions
            MustHonorExceptions PartIDKeys DocIndex"
            Type="Scanning" CombinedMethod="None"
            ExecutionPolicy="AllFound">
            <Performance AverageAmount="120.0" Name="ExposedMedia" />
            <FeaturePool>
              <EnumerationState MinOccurs="1"
                AllowedValueList="Mono ColorTransparency Photo"
                UserDisplay="Display" Editable="true" ID="sm"
                ListType="SingleValue" HasDefault="true" Name="ScanMode"
                DevNS="http://www.CIP4.org/JDFSchema_1_1" MaxOccurs="1"
                MacroRefs="ScanModeMacro" />
            </FeaturePool>
            <DisplayGroupPool>
              <DisplayGroup rRefs="btd cmp mag colorspace outputs">
                <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>
          </DeviceInfo>
        </Device>
      </DeviceList>
    </Response>
  </JMF>
```

```

<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>
      </when>
    </choice>
  </macro>

```



```

        </set>
        <set rRef="colorspace">
            <FeatureAttribute CurrentValue="LAB" />
        </set>
        <set rRef="outputres">
            <FeatureAttribute CurrentValue="200 200" />
        </set>
    </when>
</choice>
</macro>
</MacroPool>
<DevCaps Required="false" Context="Resource"
    DevNS="http://www.CIP4.org/JDFSchema_1_1"
    Availability="Installed"
    Name="NodeInfo" ResourceUpdate="None">
    <DevCap MinOccurs="1" Name="NodeInfo"
        DevNS="http://www.CIP4.org/JDFSchema_1_1" MaxOccurs="1">
        <StringState UserDisplay="Display"
            DevNS="http://www.CIP4.org/JDFSchema_1_1"
            Editable="true" MinOccurs="1" MaxOccurs="1"
            Name="TargetRoute" HasDefault="true"
            ListType="SingleValue" />
        <IntegerState Name="JobPriority"
            DevNS="http://www.CIP4.org/JDFSchema_1_1"
            Editable="true" MinOccurs="1" MaxOccurs="1"
            UserDisplay="Display" HasDefault="true"
            ListType="SingleValue" />
    </DevCap>
</DevCaps>
<DevCaps Required="false" ResourceUpdate="None" Context="Resource"
    Availability="Installed" Name="ExposedMedia"
    DevNS="http://www.CIP4.org/JDFSchema_1_1">
    <DevCap MinOccurs="1" Name="ExposedMedia"
        DevNS="http://www.CIP4.org/JDFSchema_1_1" MaxOccurs="1">
        <DevCap MinOccurs="1" Name="Media"
            DevNS="http://www.CIP4.org/JDFSchema_1_1"
            MaxOccurs="1">
            <NameState MinOccurs="1" DefaultValue="Sheet"
                UserDisplay="Display" Editable="true"
                ListType="SingleValue" HasDefault="true"
                Name="MediaUnit" MaxOccurs="1"
                DevNS="http://www.CIP4.org/JDFSchema_1_1"/>
            <XYPairState MinOccurs="1" UserDisplay="Display"
                Editable="true" AllowedValueMax="600.0 1200.0"
                ListType="SingleValue" HasDefault="true"
                Name="Dimension" AllowedValueMin="0.0 0.0"
                DevNS="http://www.CIP4.org/JDFSchema_1_1"
                MaxOccurs="1" />
        </DevCap>
    </DevCap>
</DevCaps>
<DevCaps Required="false" Context="Resource"
    DevNS="http://www.CIP4.org/JDFSchema_1_1"
    Availability="Installed" Name="ScanParams"
    ResourceUpdate="None">
    <DevCap MinOccurs="1" Name="ScanParams"
        DevNS="http://www.CIP4.org/JDFSchema_1_1" MaxOccurs="1">
        <IntegerState MinOccurs="1" DefaultValue="1"
            AllowedValueList="1 4 8 10 12" UserDisplay="Hide"
            ActionRefs="BD-bw ctcmp cd" Editable="true"
            ID="btd" ListType="SingleValue" HasDefault="true"
            Name="BitDepth" MaxOccurs="1"
            DevNS="http://www.CIP4.org/JDFSchema_1_1"/>
        <EnumerationState ActionRefs="BD-bw ctcmp" MinOccurs="1"

```

```

        AllowedValueList=
            "CCITTFaxEncode FlateEncode DCTEncode None"
        UserDisplay="Hide" Editable="true" ID="cmp"
        ListType="SingleValue" HasDefault="true"
        Name="CompressionFilter" MaxOccurs="1"
        DevNS="http://www.CIP4.org/JDFSchema_1_1" />
    <NumberState MinOccurs="1" UserDisplay="Display"
        Editable="true" ID="mag" ListType="SingleValue"
        HasDefault="true" AllowedValueMax="100.0"
        AllowedValueMin="0.01" MaxOccurs="1"
        Name="Magnification"
        DevNS="http://www.CIP4.org/JDFSchema_1_1"/>
    <EnumerationState ActionRefs="cd" MinOccurs="1"
        AllowedValueList="GrayScale CMYK RGB LAB"
        UserDisplay="Display" Editable="true"
        ID="colorspace" ListType="SingleValue"
        HasDefault="true" Name="OutputColorSpace"
        MaxOccurs="1"
        DevNS="http://www.CIP4.org/JDFSchema_1_1" />
    <XYPairState MinOccurs="1" DefaultValue="600.0 600.0"
        AllowedValueList="100.0 100.0 300.0 300.0 600.0 600.0
            1200.0 1200.0 2400.0 2400.0"
        UserDisplay="Display" Editable="true" ID="outputres"
        ListType="SingleValue" HasDefault="true"
        Name="OutputResolution" MaxOccurs="1"
        DevNS="http://www.CIP4.org/JDFSchema_1_1" />
    </DevCap>
</DevCaps>
</DeviceCap>
</Device>
</DeviceInfo>
</DeviceList>
</Response>
</JMF>

```

Example 7-64: JDF Accepted by Previous Scanner

Example of JDF Node that is accepted by the scanner of the previous example. All parameters of the following Scanning Node are compliant with the capabilities.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="GoodScan"
    Status="Waiting" Type="Scanning" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <ScanParams BitDepth="8" Class="Parameter" ID="Link0007"
        OutputColorSpace="RGB" OutputResolution="600. 600." Status="Available"/>
    <ExposedMedia Class="Handling" ID="Link0008" Status="Available">
      <Media Dimension="425.196850394 566.929133858"/>
    </ExposedMedia>
    <RunList Class="Parameter" ID="Link0014" Status="Available"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ScanParamsLink Usage="Input" rRef="Link0007"/>
    <ExposedMediaLink Usage="Input" rRef="Link0008"/>
    <RunListLink Usage="Output" rRef="Link0014"/>
  </ResourceLinkPool>
</JDF>

```

Example 7-65: JDF Rejected by Previous Scanner

Example of JDF Node that is rejected by the scanner of the previous example. All parameters of the following Scanning Node except **Magnification** are compliant with the Device capabilities. Therefore, the Device can not execute the Job.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="BadScan" Status="Waiting"
  Type="Scanning" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <ScanParams BitDepth="8" Class="Parameter" ID="Link0012"
      Magnification="1000. 1000."
      OutputColorSpace="RGB" OutputResolution="600. 600." Status="Available"/>
    <ExposedMedia Class="Handling" ID="Link0013" Status="Available">
      <Media Dimension="425.196850394 566.929133858"/>
    </ExposedMedia>
    <RunList Class="Parameter" ID="Link0014" Status="Available"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ScanParamsLink Usage="Input" rRef="Link0012"/>
    <ExposedMediaLink Usage="Input" rRef="Link0013"/>
    <RunListLink Usage="Output" rRef="Link0014"/>
  </ResourceLinkPool>
</JDF>

```

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 776.

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 776. 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

Table 7-463, “Object Classes for a Document” below has a list of the real objects that can be preflighted in a document. The objects are identified by their class name specified in the “Name” column:

Table 7-463: 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.

7.4.1.1 Properties Implemented by each Class of Object

Table 7-464, “Properties Implemented by each Class of Object” below, has columns of object Classes and rows of Properties Categories. An “X” in a cell means that an object of the specified Class implements the specified Properties (see Table 7-466, “List of Properties Categories”).

Table 7-464: Properties Implemented by each Class of Object

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

7.4.1.2 Checking for the Presence of a Property

In most of the *Preflight* Process, only the “values” of properties are needed. Please note that a property MAY incorporate one or more Attributes, and it is the values (e.g., string or enumeration) of these Attributes that are collectively referred to here as the “value” of the property. In some cases, it is also useful to be able to check if a property has been defined. This happens in some types of documents where the property definition is optional. Before checking its value, you just want to check that this property was defined.

For all the basic tests described in this document where it makes sense to check if they are defined, they are checked “Yes” in the **Tag** column of properties definition tables below. Use the `IsPresentEvaluation` to check for the presence of a property.

Example 7-66: Test for Existence of TrappedKey

This example checks if the *TrappedKey* is defined in a PDF document.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Device Class="Implementation" ID="Link0003" Status="Available">
      <DeviceCap>
        <TestPool>
          <Test ID="PT01">
            <IsPresentEvaluation>
```

```

        <BasicPreflightTest Name="TrappedKey" />
      </IsPresentEvaluation>
    </Test>
  </TestPool>
</DeviceCap>
</Device>
</ResourcePool>
<ResourceLinkPool>
  <DeviceLink Usage="Input" rRef="Link0003" />
</ResourceLinkPool>
</JDF>

```

Example 7-67: Test for TrappedKey Equal to “Unknown”

This example checks if the value of the *TrappedKey* = “Unknown” in a PDF document.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Device Class="Implementation" ID="Link0003" Status="Available">
      <DeviceCap>
        <TestPool>
          <Test ID="PT02">
            <EnumerationEvaluation ValueList="Unknown">
              <BasicPreflightTest Name="TrappedKey" />
            </EnumerationEvaluation>
          </Test>
        </TestPool>
      </DeviceCap>
    </Device>
  </ResourcePool>
  <ResourceLinkPool>
    <DeviceLink Usage="Input" rRef="Link0003" />
  </ResourceLinkPool>
</JDF>

```

Table 7-465: Mapping between property types (in the preflight spec) and evaluations (Section 1 of 2)

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.

Table 7-465: Mapping between property types (in the preflight spec) and evaluations (Section 2 of 2)

Property Type	Evaluation	Expected usage for BasicPreflightTest ListType
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.

7.4.1.3 Basic tests on set of objects

Some properties can be applied to more than one object and have a value when applied to a list of objects which differs from their value when applied to a single object. For instance, this allows you to make tests on the number of separations of objects included in a given area. These properties have the column “**Set**” checked with “*Yes*.” In order to define a Test using such properties, a list of objects is filtered first, before applying the test. This is achieved using the PreflightArgument Element.

7.4.2 Properties

Table 7-466, “List of Properties Categories” specifies the Properties Categories. In each of the following subsections, there is a table with a list of Attributes belonging to the specified Properties Category. Each such Attribute can be found, extracted, and evaluated from a document. The Attributes of each Properties Category apply to Objects of certain specified Classes (see Table 7-465, “Mapping between property types (in the preflight spec) and evaluations” on page 828.).

Note: each table of Properties in the subsections below has a different meaning from a table for an Element or Resource, which describes an XML element along with its member attributes or subelements. A Properties table does not describe an XML element or any other structure. Rather each table row describes an Attribute that is a potential Attribute of some Element derived from Abstract PRGroupOccurrenceBase Element (see Table 7-303, “List of PRGroupOccurrenceBase Elements” on page 683).

Note also: for each Properties tables, the “**Set**” column is described in Section 7.4.1.3, Basic tests on set of objects, and the **Tag** column is described in Section 7.4.1.2, Checking for the Presence of a Property.

Table 7-466: List of Properties Categories (Section 1 of 2)

Name	Page	Description
Annotation Properties	page 830	Describes Annotations.
Box Properties	page 830	Describes a container box
Class Properties	page 831	Describes the Class name and Property name
Colorant Properties	page 832	Describes color and separation information.
Document Properties	page 832	Describes a document.
Fill Properties	page 836	Describes fill for graphic objects
Font Properties	page 836	Describes fonts in a document:
Graphic Properties	page 838	Describes display and graphic information
Image Properties	page 839	Describes images displayed using pixels
Logical Properties	page 841	Mainly used with “ Set ” to count the number of objects
PageBox Properties	page 841	Describes virtual boxes for each page.
Pages Properties	page 842	Describes a page in a document
PDLObject Properties	page 843	Describes particular PDF objects in a document
Reference Properties	page 843	Describes references to external objects.
Shading Properties	page 844	Describes shading that is applied graphic objects.
Stroke Properties	page 844	Describes strokes applied to graphic objects with vector primitives
Text Properties	page 845	Describes text.

Table 7-466: List of Properties Categories (Section 2 of 2)

Name	Page	Description
Vector Properties	page 846	Describes graphic objects with vector primitives.

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 7-467: Annotation Properties

Name	Type	Description	Set	Tag	Documents
<i>AnnotationPrintFlag</i>	boolean	Is "true" when it will be printed on the final document.	—	—	PDF
<i>AnnotationType</i>	NMTOKEN	The type of annotations. Values include those from: Table 7-468, "AnnotationType Attribute Values"	—	—	PDF
<i>TrapnetAnnotationPDFX</i>	NMTOKENS	The PDF/X versions to which the <i>TrapNet</i> annotation complies, (e.g., "PDF/X-1a:2003").	—	—	PDF

— Attribute: AnnotationType

Table 7-468: AnnotationType Attribute 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>			

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.

Table 7-469: Box Properties (Section 1 of 2)

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	—	—

Table 7-469: Box Properties (Section 2 of 2)

Name	Type	Description	Set	Tag	Documents
<i>DifferentBoxSize</i>	enumerations	This is the list of boxes, which are different on one page from the same boxes on another page. Values are from: <i>BoxArgument/@Box</i> (Table 7-295, "BoxArgument Element" on page 678)	Only	—	—
<i>InsideBox</i>	boolean	Is " <i>true</i> " when an object is inside a given box. <i>InsideBox</i> MUST be qualified by <i>BoxArgument</i> Subelement.	—	—	—
<i>OutsideBox</i>	boolean	Is " <i>true</i> " when an object is outside a given box. <i>OutsideBox</i> MUST be qualified by <i>BoxArgument</i> Subelement.	—	—	—

7.4.2.3 Class Properties

Each object can define the name of the class of objects it belongs to:

Table 7-470: Class Properties

Name	Type	Description	Set	Tag	Documents
<i>ClassName</i>	NMTOKEN	The name of the class to which the object belongs. Values include those from: Table 7-471, "ClassName Attribute Values" on page 831	—	—	—
<i>PropertyList</i>	enumerations	The list of Properties the object has. Values are from: Table 7-472, "PropertyList Attribute Values" on page 831	—	—	—

— Attribute: ClassName**Table 7-471: ClassName Attribute 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 7-472: PropertyList Attribute Values (Section 1 of 2)**

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>	

Table 7-472: PropertyList Attribute Values (Section 2 of 2)

Value	Description	Value	Description
<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 7-473: Colorant Properties

Name	Type	Description	Set	Tag	Documents
<i>AliasSeparations</i>	boolean	Is "true" when some of the separations have different names but the same color values.	Yes	—	—
<i>AmbiguousSeparations</i>	boolean	Is "true" 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 7-474: Document Properties (Section 1 of 5)

Name	Type	Description	Set	Tag	Documents
<i>Author</i>	string	A string describing the author of the document.	—	Yes	—
<i>Binding</i>	enumeration	The binding of the document: Values are: <i>Left</i> <i>Right</i>	—	Yes	PDF
<i>CreationDate</i>	dateTime	The date when the document was created according to the file system.	—	—	—

Table 7-474: Document Properties (Section 2 of 5)

Name	Type	Description	Set	Tag	Documents
<i>CreationDateInDocument</i>		The date when the document was created according to data inside the document.	—	Yes	—
	dateTime				
<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). Values are from: @ <i>CompressionTypes</i> in Table 7-480, "Image Properties"/	—	—	—
<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. 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: Values are: <i>ASCII</i> <i>Binary</i>	—	—	PS, PDF
<i>DocumentIsGoodCompression</i>		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.	—	—	—
	boolean				
<i>EncryptedDocument</i>	boolean	Is "true" if document is encrypted.	—	—	—
<i>EncryptionFilter</i>	NMTOKEN	The Filter name of encryption for a PDF file.	—	Yes	PDF

Table 7-474: Document Properties (Section 3 of 5)

Name	Type	Description	Set	Tag	Documents
<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. Values include: <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 7-474: Document Properties (Section 4 of 5)

Name	Type	Description	Set	Tag	Documents
<i>OutputIntentColorSpace</i> = "None"	NMTOKEN	The color space belonging to the output intent of the document. 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 " <i>true</i> " 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 expressed as a MIME-type. Values include those from: Table H-1, "MimeType Attribute Values (IANA Registered)" on page 903 and Table H-2, "MimeType and File Type Combinations" on page 905 Example: <i>PDLType</i> value is " <i>application/pdf</i> ".	—	—	—
<i>PDLVersion</i>	string	The version of document according to the <i>PDLType</i> . Values include those from: Table H-1, "MimeType Attribute Values (IANA Registered)" on page 903 and Table H-2, "MimeType and File Type Combinations" on page 905.	—	—	—
<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 " <i>true</i> " 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	—

Table 7-474: Document Properties (Section 5 of 5)

Name	Type	Description	Set	Tag	Documents
<i>TrappedKey</i>	enumeration	A value explaining the use of trapping on the document. Values are: <i>true</i> <i>false</i> <i>Unknown</i> Note: the values match 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 7-475: 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. Values are from: Table 7-476, “FillColorType Attribute Values”	—	—	—
<i>HasFillColor</i>	boolean	Is “ <i>true</i> ” if the vector object is drawn with a fill color.	—	—	—

— Attribute: FillColorType**Table 7-476: FillColorType Attribute 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 7-477: Font Properties

Name	Type	Description	Set	Tag	Documents
<i>EmbeddingRestrictionFlag</i>	boolean	Is "true" if a font cannot be embedded.	—	—	—
<i>FontCorrupted</i>	boolean	Is "true" 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 "true" if a font is embedded into the document.	—	—	—
<i>FontIsStandardLatin</i>	boolean	Is "true" when all characters belong to the standard Latin character set.	—	—	—
<i>FontName</i>	string	The font name.	—	—	—
<i>FontNotUsed</i>	boolean	Is "true" if a font is not used to draw characters from the document.	—	—	—
<i>FontSubset</i>	boolean	Is "true" 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. Values are from: Table 7-478, "FontType Attribute Values"	—	—	—
<i>FontVendor</i>	string	The font vendor.	—	—	—
<i>IsDoubleByteFont</i> New in JDF 1.4	boolean	Some fonts need double-byte encoding to store characters internally	—	—	—
<i>IsFontScreenOnly</i>	boolean	Is "true" 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 7-478: FontType Attribute 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 7-479: Graphic Properties (Section 1 of 2)

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 from: @ColorSpace	—	Yes	PS, PDF
<i>BelongsToAnnotation</i>	boolean	Is "true" when this object belongs to an annotation.	—	—	—
<i>BlackGeneration</i>	enumeration	The <i>BlackGeneration</i> function of a PS or PDF object. Values are: <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. Values are: <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.	—	—	—

Table 7-479: Graphic Properties (Section 2 of 2)

Name	Type	Description	Set	Tag	Documents
<i>NumberOfColorsInLUT</i>	integer	The number of colors in the color table used to display an indexed image.	—	—	—
<i>OverPrintFlag</i>	boolean	Is " <i>true</i> " 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
<i>TransferFunction</i>	enumeration	The transfer function of a PS or PDF object. Values are: <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 are: <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 7-480: Image Properties (Section 1 of 3)

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 value, but any other application-specific value could be used. Values include: <i>Print</i>	—	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.	—	—	—

Table 7-480: Image Properties (Section 2 of 3)

Name	Type	Description	Set	Tag	Documents
<i>CompressionTypes</i>	enumerations	The type of method used to compress or encode the image. Values are: <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> Note: Where JPEG, JPEG2000 and/or JBIG2 are specified, they can be concatenated and only JPEG need be used.	—	—	—
<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. Values are: <i>None</i> <i>Horizontal</i> <i>Vertical</i>	—	—	—
<i>ImageMaskType</i>	enumeration	The type of masks used by image. Values are: <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).	—	—	—

Table 7-480: Image Properties (Section 3 of 3)

Name	Type	Description	Set	Tag	Documents
<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 {
    quality = (long) (5000.0 / xq);
}
```

The algorithm calculates the average value in the quantization matrix and then derives a quality value in the range of 0–100 from that average.

7.4.2.10 Logical Properties

The logical Properties are mainly used with “Set” to count the number of objects.

Table 7-481: 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 7-482: PageBox Properties

Name	Type	Description	Set	Tag	Documents
<i>PageBoxType</i>	enumeration	Note: when not known, the default is to leave <i>PageBoxType</i> empty. Values are from: <i>BoxArgument/@Box</i> (Table 7-295, “BoxArgument Element” on page 678)	—	—	—

7.4.2.12 Pages Properties

This is the list of Elements and Attributes related to the page object in a document.

Table 7-483: Pages Properties

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. Values are from: <i>@ColorSpace</i> in Table 7-479, “Graphic Properties”.	—	Yes	PDF
<i>PageHasOptionalContent</i> New in JDF 1.4	boolean	Detect if a PDF has optional content (commonly called PDF layers).	—	—	—
<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 .	—	—	—
<i>PageScalingFactor</i> New in JDF 1.4	double	In PDF file, one way of scaling a page is to use a page scale factor. This factor can be ambiguous because it is not always used by all applications.	—	—	—
<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 7-297, “BoxToBoxDifference Element” on page 679	—	—	—

Note that `BoxToBoxDifference` Element is always a Subelement of a `PreflightArgument`.

Example 7-68: Test with `BoxToBoxDifference` Element

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <Device Class="Implementation" ID="Link0003" Status="Available">
      <DeviceCap>
        <TestPool>
          <Test ID="PT01">
            <RectangleEvaluation ValueList="0 0 10 10">
              <BasicPreflightTest Name="BoxToBoxDifference">
                <PreflightArgument>
                  <BoxToBoxDifference FromBox="TrimBox"
                    ToBox="BleedBox"/>
                </PreflightArgument>
              </BasicPreflightTest>
            </RectangleEvaluation>
          </Test>
        </TestPool>
      </DeviceCap>
    </Device>
  </ResourcePool>
  <ResourceLinkPool>
    <DeviceLink Usage="Input" rRef="Link0003"/>
  </ResourceLinkPool>
</JDF>
```

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 7-484: PDLObject Properties

Name	Type	Description	Set	Tag	Documents
<i>PDLObjectType</i>	NMTOKEN	The type of specific PDL object. 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 836).

Table 7-485: Reference Properties (Section 1 of 2)

Name	Type	Description	Set	Tag	Documents
<i>ExternalReferenceMissing</i>	boolean	Is " <i>true</i> " when the target of an external reference is missing.	—	—	—

Table 7-485: Reference Properties (Section 2 of 2)

Name	Type	Description	Set	Tag	Documents
<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. 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 7-486: 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 7-487: 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. Values are from: @ <i>ColorSpace</i> in Table 7-479, "Graphic Properties".	—	Yes	PS, PDF
<i>StrokeColorName</i>	string	The name of the color of the stroke of the vector object.	—	—	—

Table 7-487: Stroke Properties (Section 2 of 2)

Name	Type	Description	Set	Tag	Documents
<i>StrokeColorSpace</i>	enumeration	The color space of the stroke of one object. Values are from: @ <i>ColorSpace</i> in Table 7-479, “Graphic Properties”.	—	—	PS, PDF
<i>StrokeColorType</i>	enumeration	This is an enumeration of known colors used to draw stroke. Values are from: @ <i>FillColorType</i> in Table 7-475, “Fill Properties”.	—	—	—
<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 7-488: Text Properties (Section 1 of 2)

Name	Type	Description	Set	Tag	Documents
<i>CharacterProblem</i>	enumeration	Problem encountered to render character. 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 7-488: Text Properties (Section 2 of 2)

Name	Type	Description	Set	Tag	Documents
<i>UseArtificialTextEffect</i>	enumerations	<p>The artificial text effects list used to draw a character.</p> <p>Values are:</p> <p><i>Bold</i></p> <p><i>Italic</i></p> <p><i>Outline</i></p> <p><i>Shadow</i></p> <p><i>Underline</i></p> <p>Note: The authoring applications can apply the text effect directly, whereas in PS or PDF, the effect will be calculated.</p>	—	—	—

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 7-489: 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 147.

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 a Hot Folder

If a JDF (or a MIME package containing a JDF) is placed into a hot folder, this is an implicit submission to the Queue that watches the hot folder. See `SubmitQueueEntry` and related JMF Messages for details on queues.

8.2.1.2 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 Messages or Command Messages. This is done with a standard HTTP Post request. The JMF is the body of the HTTP post message. The Response Message is the body of the response to the initiated HTTP post. Signal and Acknowledge Messages are also implemented as HTTP post messages. The body of the HTTP response to these Messages 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 required in an individual installation).

Support for JMF over HTTPS does not require the implementation of any additional JMF messages, though the RequestForAuthentication Message (which is new in 1.4) may be used to exchange certificates and establish a secure connection.

On a web server, the server provides its certificate to you. The client decides whether to accept communication. With two-way authentication client authentication is required.

8.2.3.2 Certificates

JMF over HTTPS requires both parties to provide exchange and validate certificates. The certificates **MUST** contain the core four fields of the X.509 format and the UserID. Any additional fields are **OPTIONAL**. These fields are:

- Common Name (Abbreviation CN) (i.e. hostname which could be an IP address or DNS name by which the receiver knows the sender),
- Organization Unit (Abbreviation OU)

- Organization (Abbreviation O)
- Country (Abbreviation C)
- UserID (Abbreviation UID) - this **MUST** be the SenderID that messages from the sender will be identified by. This would be the client's SenderID for commands, queries, and signals, and the server's SenderID for responses and acknowledgements
- givenName? - The vendor name, product name, and any other information about the product **MAY** be optionally included in the certificate using the givenName certificate field.

Example for XYZ Software's XYZImpose product:

```
CN=impose7.printinginc.internal OU=Prepress O=Printing, Inc.  
C=US UID=XYZImpose7 givenName=XYZ Software XYZImpose v7.0
```

More information can be found at <http://www.rsasecurity.com/rsalabs/node.asp?id=2307>

Certificates can be generated by any certificate generation tool such as Sun Keytool. See Section 8.2.3.5.2, Example of Sun Keytool Usage.

The certificates should be self-signed to remove the need to access third-party Certificate Authorities.

8.2.3.2.1 Verification of Certificates

Certificates should be verified against the hostname of the machine. Therefore certificates should reference the machine and may need to be generated on site.

Note: The difference between the hostname and the IP address is that if the IP address changes, this will effectively revoke the certificate. However, if the hostname is used, then the name **MUST** be resolvable by the receiver using either DNS or local name resolution.

8.2.3.3 Exchange of Certificates

Certificates may be exchanged and authenticated by the following sequence which makes use of the RequestForAuthentication Message.

The RequestForAuthentication Message includes a requirement that the recipient return its appropriate certificate on receipt of the sender's certificate, based on the value of the *AuthenticationType* Attribute.

The likely sequence of events between two parties, A and B, can be summarized as follows:

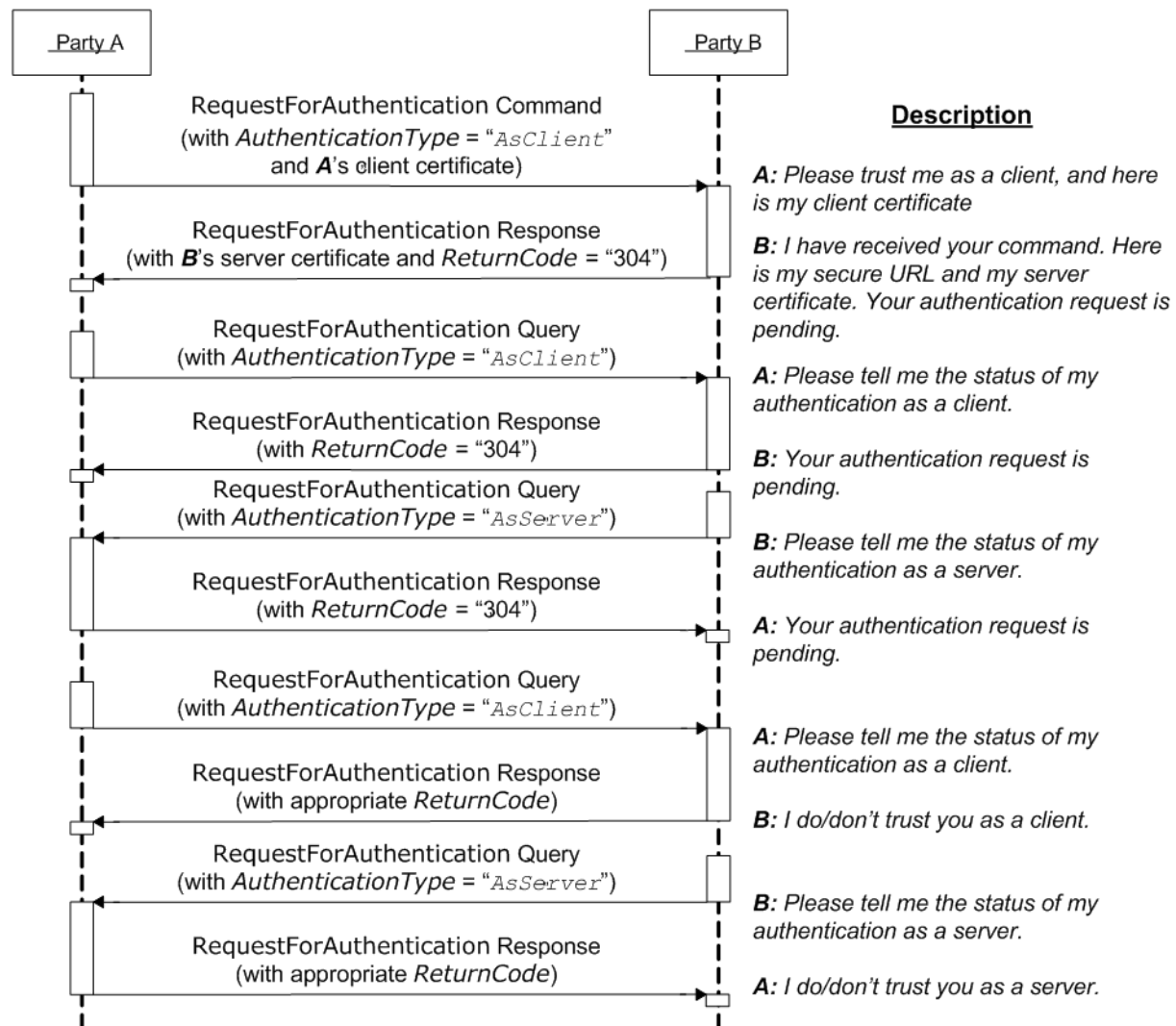


Figure 8-1: Example of Exchange of Certificates

We now have 2 way authentication in one direction with *A* as the client, and *B* as the server. To complete the other direction, there are two possibilities:

- The Process is repeated with *B* sending a *RequestForAuthentication* to *A* using the same steps.
- *A* to initiate the same steps, but sets the *AuthenticationType* Attribute to "AsServer", and provides its Secure URL in the *AuthenticationCmdParams* Element.

If the certificate received by *A* in the response from *B* is bad, then *B*'s trust of *A* must be manually deleted. Then *A* can repeat the above steps.

If the certificate received by *A* at some later time goes bad, then *A* can repeat the above steps over a secure channel, with the *Reason* Attribute set appropriately to indicate a problem. Effectively it is saying "I'm serving you notice that your certificate is bad; send me a new one". *B*'s response will be to present a certificate that should be different to the one previously sent.

If party *B* realizes that it needs to re-issue it's server certificate, it MAY send a *RequestForAuthentication* Command to party *A*'s secure URL, with the *AuthenticationType* Attribute set to "AsServer". *A* should then respond appropriately.

Reconnection: if certificates have been exchanged, but the secure URL has been lost, reconnection can be facilitated by sending a `KnownControllers Query` to the system whose URL has been lost. If the signed certificate has been lost, then the existing trust relationships **MUST** be manually deleted, then a repeat of the above steps.

8.2.3.4 Standards

See [SSL3] and [X.509].

8.2.3.5 Implementation

If a client communicates with a server over an HTTPS connection and at some point the client receives a “permission denied” HTTP response, this indicated that the secure connection has been revoked and that the client needs to resubmit the `RequestForAuthentication` message.

8.2.3.5.1 Discovery Messages

The `KnownDevices` Message has been extended so that `Device` Resource has a new Attribute `SecureJMFURL`.

The `KnownMessages` Message has been extended to indicate which Messages are supported under which protocols, by adding the `URLSchemes` Attribute to the `MessageService` Element. The `KnownControllers` Message has been extended so that the `JDFController` Element has a new Attribute `URLType`, allowing discovery of a Device or Controller's various URL's, and to be able to filter the response using a `ControllerFilter` that is new in JDF 1.4. The filter may specify a `ControllerID` or `DeviceID` in `ControllerFilter/@ControllerID`, and/or the filter may specify one or more `JDFController/@URLType` values in `ControllerFilter/@URLTypes`.

8.2.3.5.2 Example of Sun Keytool Usage

A command line example of using the Java keytool:

- 1 Use Java keytool to generate a public/private key pair and wrap the public key into an X.509 v1 self-signed certificate. The private key and certificate are stored in a JKS key store.


```
keytool -genkey -alias impose7 -dname "CN=xyzimpose7.myCompany.internal
      OU=Prepress O=Printing, Inc. C=US UID=XYZImpose7
      givenName=XYZ Software XYZImpose" -validity 365 -keystore keystore.jks
```
- 2 Export the self-signed certificate to the base64 encoded PEM format:


```
keytool -export -keystore keystore.jks -rfc -alias impose7 -file impose7.cer
```

For full documentation, see <http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/keytool.html>

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;

Example 8-1: Packaging of Individual JDF/JMF files in MIME

[New in JDF 1.2](#)

The following example displays MIME packaging of a JDF file as an individual MIME object:

```
MIME-Version: 1.0
Content-Type: multipart/related; boundary=abcdefg0123456789
Content-Transfer-Encoding: 8bit
--abcdefg0123456789
Content-Type: application/vnd.cip4-jdf+xml
<JDF ... >
  <PreviewImage Separation = "PANTONE 128" URL="cid:123456.png" />
</JDF>
--abcdefg0123456789--
```

8.3.2.2 CID URL Scheme

[New in JDF 1.2](#)

One of the benefits of the MIME Multipart/Related *MediaType* is the ability of a URL in one body part to refer to the content of another body part. This is done by using a `"cid"` scheme in a URL, specified in [RFC2392]. Please look at the example to see how it is used.

Example 8-2: CID URL Scheme

```
MIME-Version: 1.0
Content-Type: multipart/related; boundary=abcdefg0123456789
Content-Transfer-Encoding: 8bit
--abcdefg0123456789
Content-Type: application/vnd.cip4-jdf+xml
```

```

<JDF ... >
  <PreviewImage Separation = "PANTONE 128" URL="cid:123456.png@cip4.org" />
</JDF>

--abcdefg0123456789
Content-Type: image/png
Content-Transfer-Encoding: base64
Content-ID: <123456.png@cip4.org>

BASE64DATA
BASE64DATA

--abcdefg0123456789--

```

Note: [RFC2392] *requires* that the value of the Content-ID be enclosed in angle brackets (<>). Also the characters that [RFC2392] allows in Content-ID include characters that [RFC3986] does not permit in URLs; any such character (such as "+" or "&") **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.

8.3.2.3 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 8-3: MIME Multipart/Related

A Multipart/Related message is received that contains:

- Message.jmf
 - Ticket01.jdf
- Pages.pdf

```

MIME-Version: 1.0
Content-Type: multipart/related; boundary=unique-boundary

--unique-boundary
Content-Type: application/vnd.cip4-jmf+xml
Content-Transfer-Encoding: 8bit
...
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFClient"
  Timestamp="2005-07-07T13:15:56+01:00" Version="1.4">
  <Command ID="C0001" Type="SubmitQueueEntry" xsi:type="CommandSubmitQueueEntry">
    <QueueSubmissionParams Hold="true" URL="cid:JDF1@hostname.com"/>
  </Command>
</JMF>

--unique-boundary
Content-Type: application/vnd.cip4-jdf+xml
Content-Transfer-Encoding: 8bit
Content-ID: <JDF1@hostname.com>
Content-Disposition: attachment; filename="Ticket01.jdf";

<?xml version="1.0" encoding="UTF-8" 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="1.4" JobPartID="345">

```



```

<ResourceLinkPool>
  <ComponentLink Usage="Output" rRef="ID125"/>
</ResourceLinkPool>
<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>
  <Component ID="ID125" Class="Quantity" Status="Unavailable"
    ComponentType="Sheet" />
</ResourcePool>
<JDF ID="JDF-3" Status="Waiting" Type="DigitalPrinting" JobPartID="400">
  <ResourceLinkPool>
    <DigitalPrintingParamsLink Usage="Input" rRef="ID123"/>
    <RunListLink Usage="Input" rRef="RunList4"/>
    <ComponentLink Usage="Output" rRef="ID125"/>
  </ResourceLinkPool>
  <ResourcePool>
    <DigitalPrintingParams ID="ID123" Class="Parameter" Status="Available" />
  </ResourcePool>
</JDF>
</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 A-1: boolean

```
<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 A-2: CMYKColor

```
<Color cmyk = "0.3 0.6 0.8 0.1"/><!--brick red-->
```

A.2.3 date

A calendar date, it represents a time period that starts at midnight on a specified day and lasts for 24 hours. Based on [ISO8601:2004].

Encoding

It is represented identically to the XML Schema type: *date*

Example A-3: date

```
<Example StartDate="1999-05-31" />
```

A.2.4 dateTime

Represents a specific instant of time. It **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*.

Note that [ISO8601:2004] allows a wider range of time zone specifications than XML. *dateTime* **MUST** adhere to the stricter limitations defined in [XMLSchema]. For instance the colon ‘:’ in the time zone field **MUST** be present when writing time zones in the format *hh:mm*.

Example A-4: dateTime

```
<Example Start="1999-05-31T18:20:00Z" />
<Example Start="1999-05-31T13:20:00-05:00" />
```

A.2.5 DateTimeRange

[New in JDF 1.2](#)

XML Attributes of type *DateTimeRange* are used to describe a range of points in time. More specifically, it describes a time span that has an absolute start and end. Unbounded ranges can use the infinity value tokens *INF* and *-INF*

Encoding

A *DateTimeRange* is represented by two *dateTime* or infinity tokens separated by the whitespace “~” whitespace sequence.

Example A-5: DateTimeRange

```
<XXX range="1999-05-31T18:20:00Z ~ 1999-05-31T18:20:00Z" />
<XXX range="1999-05-31T18:20:00Z ~ INF" />
<XXX range="-INF ~ 1999-05-31T18:20:00Z" />
```

A.2.6 DateTimeRangeList

[New in JDF 1.2](#)

XML Attributes of type *DateTimeRangeList* are used to describe a list of ranges of points in time. More specifically, it describes a list of time spans, which each have a relative start and end.

Encoding

A *DateTimeRangeList* is represented by sequence of either *DateTimeRange* values (See 1.5), separated by whitespace or *dateTime* values.

Example A-6: DateTimeRangeList

```
<xxx RangeList=
  "1999-05-31T18:20:00Z ~ 1999-05-31T18:20:00Z 1999-05-31T13:20:00-05:00 ~ INF"/>
```

A.2.7 double

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 A-7: double

```
<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 A-8: DoubleList

```
<xxx list="3.14 1 .6"/>
```

A.2.9 DoubleRange

[New in JDF 1.2](#)

XML Attributes of type DoubleRange are used to describe a range of numbers (as doubles.) Mathematically spoken, the two numbers define a closed interval.

Encoding

A DoubleRange is represented by two double values separated by a “~” (tilde) character and OPTIONAL additional whitespace. Note: It is now RECOMMENDED that the ‘~’ is surrounded by whitespace to aid validation and parsing.

Example A-9: DoubleRange

```
<xxx range="-3.14 ~ 5.13"/>
<xxx range="0 ~ INF"/>
```

A.2.10 DoubleRangeList

[New in JDF 1.2](#)

XML Attributes of type DoubleRangeList are used to describe a list of DoubleRange values and/or enumerated numbers as doubles).

Encoding

A DoubleRangeList is a sequence of DoubleRange values and single double values separated by whitespace.

Example A-10: DoubleRangeList

```
<xxx list="-1 ~ -6 3.14 ~ 5.13 7 9 ~ 128 131 255 ~ INF"/>
```

A.2.11 duration

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*

Note that [XMLSchema] explicitly allows negative durations. Thus a value of *-PT15M* is valid and describes a negative duration of 15 minutes in the past.

Example A-11: duration

```
<Example Duration= "P1Y2M3DT10H30M" />
```

A.2.12 DurationRange

XML Attributes of type *DurationRange* are used to describe a range of time durations. More specifically, it describes a time span that has a relative start and end.

Encoding

A *DurationRange* is represented by two duration values, separated by the “~” (tilde) character and OPTIONAL additional whitespace. Note: It is now RECOMMENDED that the ‘~’ is surrounded by whitespace to aid validation and parsing.

Example A-12: DurationRange

```
<XXX range="P1Y2M3DT10H30M ~ P1Y2M3DT10H35M" />
<XXX range="P1Y2M3DT10H30M ~ INF" />
```

A.2.13 DurationRangeList

[New in JDF 1.2](#)

XML Attributes of type *DurationRangeList* are used to describe a list of ranges of time durations. More specifically, it describes a list of time spans that have a relative start and end.

Encoding:

A *DurationRangeList* is represented by sequence of *DurationRange* values and durations, separated by whitespace.

Example A-13: DurationRangeList

```
<XXX RangeList="P1Y2M3DT10H30M ~ P1Y2M3DT10H35M P1Y3M2DT10H30M" />
```

A.2.14 gYearMonth

Represents a specific Gregorian month in a specific Gregorian year. Based on [ISO8601:2004].

Encoding

It is represented identically to the XML Schema type: *gYearMonth*

Example A-14: gYearMonth

```
<Example Month="2002-11" />
```

A.2.15 hexBinary

Represents arbitrary hex encoded binary data.

Encoding

It is represented identically to the XML Schema type: *hexBinary*

Example A-15: hexBinary

```
<Example Hex="0A1C" />
```

A.2.16 ID

[Modified in JDF 1.3](#)

Represents the *ID* Attribute from [XMLSchema]. It represents a name or string that contains no space characters and starts with a letter, or ‘_’. Each ID value MUST be unique within a JDF document and thus uniquely identify the elements that bear them.

Note that the *ID* Attribute definition in [XMLSchema] is more restrictive than the *ID* Attribute definition in [XML]. [XMLSchema] explicitly forbids the use of ‘:’ in ID.

Encoding

It is represented identically to the XML Schema type: *ID*

Example A-16: ID

```
<Example ID="R-16" />
```

A.2.17 IDREF

IDREF Represents the IDREF Attribute from [XMLSchema]. For a valid XML-document, an element with the ID value specified in IDREF MUST be present in the scope of the document.

Encoding

It is represented identically to the XML Schema type: *IDREF*

Example A-17: IDREF

```
<Example IDREF="R-16" />
```

A.2.18 IDREFS

IDREFS Represents the IDREFS Attribute from [XMLSchema]. More specifically, this is a whitespace-separated list of IDREF values.

Encoding

It is represented identically to the XML Schema type: *IDREFS*

Example A-18: IDREFS

```
<Example IDREFS="R-12 R-16" />
```

A.2.19 integer

Represents numerical integer values with tokens for representing infinity limits.

Implementation note: Except where explicitly noted otherwise, integers are not expected to exceed a value that can be represented as signed 32 bits.

Encoding

It is represented as a union of the XML Schema type: *integer* and the infinity value tokens *INF* and *-INF*

Example A-19: integer

```
<Example Copies="36" />
```

A.2.20 IntegerList

XML Attributes of type IntegerList are used to describe a variable length list of integer values.

Encoding

An IntegerList is encoded as a string of integers separated by whitespace.

Example A-20: IntegerList

```
<xxx list="-INF 0 1 2 3 4 INF 1 3 0"/>
```

A.2.21 IntegerRange

XML Attributes of type IntegerRange are used to describe a range of integers. In some cases, ranges are defined for an unknown number of objects. In these cases, a negative value denotes a number counted from the end. For example, -1 is the last object, -2 the second to last and so on. IntegerRanges that follow this convention are marked in the respective Attribute descriptions.

If the first element of an IntegerRange specifies an element that is behind the second element, the Range specifies a list of integers in reverse order, counting backwards. For example “6 ~ 4” = “6 5 4” and “-1 ~ 0” = “last... 2 1 0”.

Encoding

An IntegerRange is represented by two integers, separated by a “~” (tilde) character and OPTIONAL additional whitespace. Note: It is now RECOMMENDED that the ‘~’ is surrounded by whitespace to aid validation and parsing.

Example A-21: IntegerRange

```
<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 A-22: IntegerRangeList

```
<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 A-23: LabColor

```
<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 A-24: language

```
<Example Language="de"/> <!-- German -->
<Example Language="de-CH"/> <!-- Swiss German -->
<Example Language="en"/> <!-- English -->
<Example Language="en-GB"/> <!-- British English -->
```

A.2.25 languages

[New in JDF 1.4](#)

XML Attributes of type languages are used to describe a variable length list of language values.

Encoding

A languages value is encoded as a string of languages, each language separated by whitespace.

Example A-25: languages

```
<Example Languages="de-CH de en-GB en"/>
```

A.2.26 matrix

Coordinate transformation matrices are widely used throughout the whole printing Process, especially in **Layout Resources**. They represent two dimensional transformations as defined by [PS] and [PDF1.6]. For more information, refer to the respective reference manuals, and look for “Coordinate Systems and Transformations.” The “identity matrix”, which is “1 0 0 1 0 0”, is often used as a default throughout this specification. When another matrix is factored against a matrix with the identity matrix value, the result is that the original matrix remains unchanged.

Encoding

Coordinate transformation matrices are primitive data types and are encoded as a list of six numbers (as doubles), separated by whitespace: “a b c d Tx Ty”. The variables *Tx* and *Ty* describe distances and are defined in points.

Example A-26: matrix

```
<ContentObject CTM="1 0 0 1 3.14 21631.3" />
```

A.2.27 NameRange

XML Attributes of type NameRange are used to describe a range of NMTOKEN data that are acquired from a list of named elements, such as named pages in a PDL file. It depends on the ordering of the targeted list, which names are assumed to be included in the NameRange. The following two possibilities exist:

- 1 There is no explicit ordering. In this case, case sensitive alphabetical ordering [Unicode5.0] is implied. This behavior is the default unless called out explicitly in the specification.
- 2 There is explicit ordering, such as in a list of named pages in a **RunList**. In this case, the ordering of the **RunList** defines the order and all pages between the end pages are included in the NameRange.

Modification note: starting with JDF 1.4, the first item is specified as the default behavior.

Encoding

A NameRange typed Attribute is represented by two NMTOKEN values separated by a “~” (tilde) character and OPTIONAL additional whitespace. Note: It is now RECOMMENDED that the ‘~’ is surrounded by whitespace to aid validation and parsing.

Example A-27: NameRange

```
<XXX NameRange="Jack ~ Jill"/>
```

A.2.28 NameRangeList

XML Attributes of type NameRangeList are used to describe a list of NameRanges.

Encoding

A NameRangeList is represented by a sequence of NameRanges and NMTOKEN, separated by whitespace.

Example A-28: NameRangeList

```
<xxx list="A brian ~ fred x z"/>
```

A.2.29 NMTOKEN

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 A-29: NMTOKEN

```
<Example Alias="ABC_6"/>
```

A.2.30 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 A-30: NMTOKENS

```
<Example AliasList="ABC_6 ABCD_3 DEGF"/>
```

A.2.31 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 A-31: PDFPath

```
<ElementWithPath path="0 0 m 10 10 l 20 20 l"/>
```

A.2.32 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 A-32: rectangle

```
<ContentObject ClipBox="0 0 3.14 21631.3" />
```

Implementation Remark

Since all numbers are real numbers, any comparison of boxes SHOULD take into account certain rounding errors. For example, different XYPair values MAY be considered equal when all numbers are the same within a range of 1 point.

A.2.33 RectangleRange

[New in JDF 1.2](#)

XML Attributes of type RectangleRange are used to describe a range of rectangles.

Encoding

A RectangleRange is represented by one or two Rectangles, separated by a “~” (tilde) character and OPTIONAL additional whitespace. Note: It is now RECOMMENDED that the ‘~’ is surrounded by whitespace to aid validation and parsing.

Example A-33: RectangleRange

```
<XXX range="1 2 3 4 ~ 5 6 7 8"/>
<XXX range="-INF -INF 3 4 ~ 0 1 INF INF"/>
```

A.2.34 RectangleRangeList

[New in JDF 1.2](#)

XML Attributes of type RectangleRangeList are used to describe a list of rectangle ranges.

Encoding

A RectangleRangeList is represented by sequence of RectangleRange values and Rectangle values, separated by whitespace.

Example A-34: RectangleRangeList

```
<XXX RectangleRangeList="1 2 3 4 ~ 5 6 7 8 9 10 11 12 13 14 15 16"/>
```

A.2.35 regExp

Represents a regular expression as defined in [XMLSchema].

Encoding

It is represented identically to the XML Schema type: *normalizeString*

Example A-35: regExp

```
<Example expression="Foo({1|2}*)" />
```

A.2.36 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 A-36: shape

```
<XXX Dimensions="10 20 40" />
```

A.2.37 ShapeRange

XML Attributes of type ShapeRange are used to describe a range of shapes (three dimensional boxes). The range " $x1\ y1\ z1\ \sim\ x2\ y2\ z2$ " 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 A-37: ShapeRange

```
<xxx Shaperange="1 2 3 ~ 4 5 6"/>
<xxx Shaperange="1 2 3 ~ 4 INF 6"/>
```

A.2.38 ShapeRangeList

XML Attributes of type ShapeRangeList are used to describe a list of ShapeRange and/or shapes.

Encoding

A ShapeRangeList is a sequence of ShapeRange and shapes separated by whitespace.

Example A-38: ShapeRangeList

The brackets below the example illustrate the grouping of shapes and ShapeRange values.

```
<xxx Shapelist="100 200 300 ~ 110 220 330 150 300 150 2 3 0 ~ 3 4 5"/>
          [                               ] [           ] [           ]
```

A.2.39 sRGBColor

XML Attributes of type sRGBColors are used to specify sRGB colors.

Encoding

sRGBColors are primitive data types and are encoded as a string of three numbers in the range of [0...1.0] separated by whitespace. A value of 0 specifies no intensity (black) and a value of 1 specifies full intensity. The sequence is defined as: "r g b"

Example A-39: sRGBColor

```
<Color sRGB="0.3 0.6 0.8" />
```

A.2.40 string

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 A-40: string

```
<Example Name="Test With Space"/>
```

A.2.41 TimeRange

[Deprecated in JDF 1.2](#)

A.2.42 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 A-41: TransferFunction

```
<someElementWithTransferCurve someCurve="0 0 .1 .2 .5 .6 .8 .9 1 1"/>
```

A.2.43 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

A URI is represented identically to the XML Schema type: *anyURI*.

Example A-42: URI

```
<Example URI="http://www.w3.org/1999/XMLSchema"/>
```

A.2.44 URL

Short for URL-reference. Represents a Uniform Resource Locator (URL) Reference as defined in [RFC3986]. In JDF 1.3 and above, the URL data typed is represented as an Internationalized Resource Identifier (IRI) as defined in [RFC3987].

Encoding

A URL is represented identically to the XML Schema type: *anyURI*.

Some characters in a URL MUST be escaped and all characters MAY be escaped by encoding their UTF-8 representation into a ‘%’ followed by the double digit hex representation of the character. The list of characters that MUST be encoded is dependent on the URL scheme. Non-escaped characters MUST be encoded in the encoding of the containing JDF document.

Example A-43: URL[New in JDF 1.4](#)

A UNC path to be displayed as a URL:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
"\\\\myHost\\a\\c äöü.pdf"
```

Example A-44: URL: UTF-8[New in JDF 1.4](#)

The UNC path encoded as an IRL with internationalized characters in UTF-8:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Example URL="file://myHost/a/c%20äöü%25.pdf" />
```

Example A-45: URL: Windows Locale 1252[New in JDF 1.4](#)

The same UNC path encoded as an IRL with internationalized characters in UTF-8 viewed in a Windows locale 1252:

```
<Example URL="file://myHost/a/c%20ÄöÜ%25.pdf" />
```

Example A-46: URL: Escaped Characters[New in JDF 1.4](#)

The same UNC path encoded as an IRL with internationalized characters escaped:

```
<Example URL="file://myHost/a/c%20%c3%a4%c3%b6%c3%bc%25.pdf" />
```

A.2.45 XPath[New in JDF 1.2](#)

Represents an XPath expression. [XPath]

EncodingIt is represented identically to the XML Schema type: *token***Example A-47: XPath**

```
<Example xpath="JDF/AuditPool/Created/@TimeStamp" />
```

A.2.46 XYPairXML 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 A-48: XYPair**

```
<CutBlock BlockSize="612 792" />
```

Implementation Remark

Since all numbers are real numbers, comparison of XYPair values SHOULD take into account certain rounding errors. For example, different XYPair values MAY be considered equal when all numbers are the same within a range of 1 point.

A.2.47 XYPairRange

XML Attributes of type XYPairRange are used to describe a range of XYPair values. The range " $x1\ y1 \sim x2\ y2$ " describes the area $x1 \leq x \leq x2$ and $y1 \leq y \leq y2$. 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 A-49: XYPairRange

```
<XXX XYrange="1 2 ~ 3 4" />
<XXX XYrange=" -INF 2 ~ 3 INF" />
```

A.2.48 XYPairRangeList

XML Attributes of type XYPairRangeList are used to describe a list of XYPairRange and/or XYPair values.

Encoding

A XYPairRangeList is a sequence of XYPairRange and XYPair values separated by whitespace.

Example A-50: XYPairRangeList

The brackets below the example illustrate the grouping of XYPair values and XYPairRange values.

```
<XXX xylist="100 200 ~ 110 220 150 300 150 350 200 300 ~ INF INF" />
           [                ] [                ] [                ] [                ]
```

A.3 Enumerations 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 A-51: enumeration

```
<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 A-52: enumerations

```
<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 Anchor

[New in JDF 1.4](#)

Attributes with a data type of Anchor describe the 9 anchor points of a rectangle.

Table A-1: Anchor Enumeration Values

Enumeration Value	Comment
<i>TopLeft</i>	
<i>TopCenter</i>	
<i>TopRight</i>	
<i>CenterLeft</i>	
<i>Center</i>	
<i>CenterRight</i>	
<i>BottomLeft</i>	
<i>BottomCenter</i>	
<i>BottomRight</i>	

A.3.3.2 JDFJMFVersion

Describes the schema version of a JDF or JMF instance.

Table A-2: JDFJMFVersion Enumeration Values

Enumeration Value	Comment
<i>1.1</i>	JDF 1.1
<i>1.2</i>	JDF 1.2
<i>1.3</i>	JDF 1.3
<i>1.4</i>	JDF 1.4

A.3.3.3 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-3: NamedColor Enumeration Values (Section 1 of 2)

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>	—

Table A-3: NamedColor Enumeration Values (Section 2 of 2)

Color name/ Enumeration Value	Comment	Color name/ Enumeration Value	Comment
<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.3.3.4 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 30.

Table A-4: Orientation Enumeration Values

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.3.3.5 WorkStyle

Table A-5: WorkStyle Enumeration Values (Section 1 of 2)

Enumeration Value	Comment
<i>Simplex</i>	No turning.
<i>Perfecting</i>	Many Sheet-Fed printing presses have perfecting cylinder(s) built in. The leading edge of the print Sheet changes as the Sheet is turned by the perfecting cylinder, but the side lays remain unaltered. In this regard, this <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.

Table A-5: WorkStyle Enumeration Values (Section 2 of 2)

Enumeration Value	Comment
<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 pre-validation 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 Intent Nodes, Combined Process Nodes and so on.) General definitions with the appropriate name are provided to enable consistent use of *xsi:type*.

The JDF Schema defines types for JDF Process Nodes and JMF Messages. It is RECOMMENDED that these types are used with *xsi:type*.

Example B-1: JDF Nodes: xsi:type

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="BackCover"
  Status="InProgress"
  Type="DigitalPrinting" Version="1.4" JobPartID="345"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.CIP4.org/Schema/JDFSchema_1_4/JDF.xsd"
  xsi:type="DigitalPrinting">
  <ResourceLinkPool>
    <DigitalPrintingParamsLink Usage="Input" rRef="ID123" />
    <RunListLink Usage="Input" rRef="ID124" />
    <ComponentLink Usage="Output" rRef="ID125" />
  </ResourceLinkPool>
  <ResourcePool>
    <DigitalPrintingParams ID="ID123" Class="Parameter" Status="Available" />
    <RunList ID="ID124" Class="Parameter" Status="Available" />
    <Component ID="ID125" Class="Quantity" Status="Unavailable"
      ComponentType="Sheet" />
  </ResourcePool>
</JDF>
```



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.

Example B-2: JDF Nodes: xsi:type (not in Default Namespace)

If the JDF is not in the default namespace then the type name needs to be altered accordingly:

```
<jdf:JDF xmlns:jdf="http://www.CIP4.org/JDFSchema_1_1" ID="BackCover"
  Status="InProgress"
  Type="DigitalPrinting" Version="1.4" JobPartID="345"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="jdf:DigitalPrinting">
  <jdf:ResourceLinkPool>
    <jdf:DigitalPrintingParamsLink Usage="Input" rRef="ID123"/>
    <jdf:RunListLink Usage="Input" rRef="ID124"/>
    <jdf:ComponentLink Usage="Output" rRef="ID125"/>
  </jdf:ResourceLinkPool>
  <jdf:ResourcePool>
    <jdf:DigitalPrintingParams ID="ID123" Class="Parameter"
      Status="Available" />
    <jdf:RunList ID="ID124" Class="Parameter" Status="Available" />
    <jdf:Component ID="ID125" Class="Quantity" Status="Unavailable"
      ComponentType="Sheet" />
  </jdf:ResourcePool>
</jdf:JDF>
```

B.1.2 Using xsi:type with JMF Messages

[New in JDF 1.2](#)

JMF Messages are organized into families — Command, Acknowledge, etc. (See “JMF Message Families” on page 174.)— 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`”.

Note JMF Messages also do not have to be in the default namespace as in the JDF Node example below.

Example B-3: JMF: xsi:type

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="TestSender"
  Timestamp="2003-11-07T12:15:56Z" MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <Query ID="Message_001Q" Type="Events" xsi:type="QueryEvents">
    <NotificationFilter/>
  </Query>
  <Response ID="Message_001R" Type="Events" refID="Q001"
    xsi:type="ResponseEvents">
    <NotificationDef Classes="Error" Type="Barcode"/>
  </Response>
</JMF>
```

Appendix C 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 Mapping for Generic Devices (Section 1 of 3)

StatusDetails	JDF/@Status	DeviceStatus	Description
<i>AbortedBySystem</i> New in JDF 1.3	<i>Aborted</i>	<i>Stopped</i>	The Job is being or has been aborted by the Device
<i>BreakDown</i>	<i>Stopped</i>	<i>Down</i>	Breakdown of the device, repair needed.
<i>Calibrating</i>	<i>Setup</i>	<i>Setup</i>	The Device is calibrating, either manually or automatically.
<i>ControlDeferred</i> Modified in JDF 1.4	-	<i>Unknown</i>	The Machine is not accessible by the Device. Note: <i>JDF/@Status</i> is unknown if the device is not accessible. Modification note: starting with JDF 1.4, the <i>DeviceStatus</i> value changed from "Stopped" to "Unknown"
<i>CoverOpen</i> New in JDF 1.3	<i>Stopped</i>	<i>Stopped</i>	One or more covers on the Device are open.
<i>DocumentAccessError</i> New in JDF 1.3	<i>Aborted</i>	<i>Stopped</i>	The Device could not access one or more documents passed by reference.
<i>DoorOpen</i> New in JDF 1.3	<i>Stopped</i>	<i>Stopped</i>	One or more doors on the Device are open.
<i>Failure</i>	<i>Stopped</i>	<i>Stopped</i>	Failure of the device. Requires some maintenance in order to restart the device. "Failure" has specialized subcategories: "PaperJam", "DoubleFeed", "BadFeed", "BadTrim", "ObliqueSheet", "IncorrectComponent", "IncorrectThickness".
<i>Good</i>	<i>InProgress</i>	<i>Running</i>	Production of products in progress, good copy counter is on, waste copy counter is off
<i>Idling</i>	<i>Stopped</i>	<i>Running</i>	Device is running, but no products are produced or consumed. Good and waste copy counter are off.
<i>InputTrayMissing</i> New in JDF 1.3	<i>Stopped</i>	<i>Stopped</i>	One or more input trays are not in the Device
<i>InterlockOpen</i> New in JDF 1.3	<i>Stopped</i>	<i>Stopped</i>	One or more interlock devices on the printer are unlocked.

Table C-1: StatusDetails Mapping for Generic Devices (Section 2 of 3)

StatusDetails	JDF/@Status	DeviceStatus	Description
<i>IterationPaused</i> New in JDF 1.4	<i>Suspended</i>	-	" <i>Suspended</i> " specifies that at least one iteration cycle has completed but additional iteration cycles MAY still occur.
<i>JobCanceledByOperator</i> New in JDF 1.3	<i>Aborted</i>	-	The Job was canceled by the Device operator using <i>AbortQueueEntry</i> or means local to the Device.
<i>JobCanceledByUser</i> New in JDF 1.3	<i>Aborted</i>	-	The Job was canceled by the owner of the Job using <i>AbortQueueEntry</i> .
<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 <i>Queue/@Status = "Held"</i> , the Job's <i>QueueSubmissionParams/@Held = "true"</i> , or <i>JDF/@Activation = "Held"</i> .
<i>JobHeld</i> New in JDF 1.3	<i>Waiting</i>	-	The Device held the Job that had been waiting (by performing a <i>HoldQueueEntry</i> request on a <i>Waiting QueueEntry</i>).
<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 (<i>ResumeQueueEntry</i>).
<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 <i>SuspendQueueEntry</i> request on a <i>Running QueueEntry</i>) and other Jobs can be processed by the Device.

Table C-1: StatusDetails Mapping for Generic Devices (Section 3 of 3)

StatusDetails	JDF/@Status	DeviceStatus	Description
<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. "Maintenance" has specialized subcategories: "BlanketChange" and "SleeveChange".
<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 "Pause".
<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 "Failure".
<i>Pause</i>	<i>Stopped</i>	<i>Stopped</i>	Machine paused; restart is possible. "Pause" has specialized subcategories: "MissResources" and "WaitForApproval".
<i>ProcessingToStopPoint</i> New in JDF 1.3	<i>InProgress</i>	<i>Running</i>	The requester has issued an <code>AbortQueueEntry</code> 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 "Pause".
<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.

Table C-2: StatusDetails Mapping for Printing Devices (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 "Maintenance".
<i>BlanketWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of the blanket; subcategory of "WashUp".
<i>CleaningInkFountain</i>	<i>Cleanup</i>	<i>Cleanup</i>	Cleaning of the ink fountain; subcategory of "WashUp".
<i>CylinderWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of impression cylinders; subcategory of "WashUp".
<i>DampeningRollerWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of the dampening roller; subcategory of "WashUp".
<i>FormChange</i>	<i>Setup</i>	<i>Setup</i>	In conventional printing, changing of plates; in direct imaging printing, imaging or re-imaging of plates.
<i>InkRollerWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of the inking roller; subcategory of "WashUp".
<i>PlateWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of the plate; subcategory of "WashUp".
<i>Processing</i> New in JDF 1.4	<i>InProgress</i>	-	Other productive processing (RIP, etc.) is taking place but no final output is being produced. All input data has arrived (not "InProgress"/"JobStreaming" nor "Waiting"/"JobIncoming").
<i>SleeveChange</i>	<i>Stopped</i>	<i>Stopped</i>	Changing of sleeves; subcategory for "Maintenance".
<i>WashUp</i>	<i>Cleanup</i>	<i>Cleanup</i>	Machine is washed before, during or after production. "WashUp" has specialized subcategories: "BlanketWash", "CleaningInkFountain", "CylinderWash", "DampeningRollerWash", "InkRollerWash", or "PlateWash". "WashUp" is the default which is assumed if <i>StatusDetails</i> is not specified.
<i>WaitingForMarker</i> New in JDF 1.4	<i>Suspended</i> or <i>Ready</i>	-	The <i>Status</i> is "Suspended" if the Printing Device models any module prior to the Marker module, otherwise, the <i>Status</i> is "Ready". The Node is automatically Suspended by the Worker because it is waiting behind other Jobs for the Marker module and the Worker will resume the Node when a Marker module becomes available.

Table C-2: StatusDetails Mapping for Printing Devices (Section 2 of 2)

StatusDetails	JDF/@Status	DeviceStatus	Description
<i>WaitingForReference</i> <i>dDataCollector</i> New in JDF 1.4	<i>Suspended</i> or <i>Ready</i>	-	The <i>Status</i> is " <i>Suspended</i> " if the Printing Device models any module prior to the Referenced-Data-Collector module, otherwise, the <i>Status</i> is " <i>Ready</i> ". The Node is automatically Suspended by the Worker because it is waiting behind other Jobs for the Referenced-Data-Collector module and the Worker will resume the Node when a Referenced-Data-Collector module becomes available.
<i>WaitingForRIP</i> New in JDF 1.4	<i>Suspended</i> or <i>Ready</i>	-	The <i>Status</i> is " <i>Suspended</i> " if the Printing Device models any module prior to the RIP module, otherwise, the <i>Status</i> is " <i>Ready</i> ". The Node is automatically Suspended by the Worker because it is waiting behind other Jobs for a RIP module (process slot) and the Worker will resume the Node when a RIP module becomes available.

Table C-3: StatusDetails Mapping for Postpress Devices

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

The *ModuleStatus* Element (see Table 5-72, "ModuleStatus Element," on page 234), the *ModulePhase* Element (see Table 3-40, "ModulePhase Element," on page 129) and **VarnishingParams** (see Table 7.2.196 on page 772) contain a *ModuleType* Attribute that defines individual modules within a Machine. The following table defines individual *ModuleType* values.

Table C-4: ModuleType Attribute Values for Conventional Printing Devices (Section 1 of 2)

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.

Table C-4: ModuleType Attribute Values for Conventional Printing Devices (Section 2 of 2)

ModuleType	Description
<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 Attribute Values for Postpress

ModuleType	Description
<i>BlockPreparer</i> New in JDF 1.4	The Block Preparer prepares the book block for a hardcover book. See Section 6.6.2, BlockPreparation.
<i>BoxFolder</i> New in JDF 1.4	The Box Folder folds and glues blanks into folded boxes for packaging. See Section 6.6.3, BoxFolding
<i>CaseMaker</i> New in JDF 1.4	The Case Maker produces the hard case for books. See Section 6.6.6, CaseMaking
<i>Caser</i> New in JDF 1.4	The Caser joins the hard cover book case and the book block. (<i>CasingIn</i>). See Section 6.6.7, CasingIn.
<i>Chain</i> New in JDF 1.2	Transport chain or conveyer to transport gathered / collected product.
<i>EndSheetGluer</i> New in JDF 1.4	The End-Sheet Gluer merges the front-end sheet, the book block and the back-end sheet together. See Section 6.6.17, EndSheetGluing.
<i>Feeder</i> New in JDF 1.2	Feeder module, feeds the device with paper. See Section 6.6.18, Feeding.
<i>Gluer</i> New in JDF 1.4	The Gluer applies glue to a component. See Section 6.6.21, Gluing.
<i>HeadBandApplicator</i> New in JDF 1.4	The Head Band Applicator applies a head band to the book block. See Section 6.6.22, HeadBandApplication.
<i>InkjetPrinter</i> New in JDF 1.4	Prints images or texts on a component. (<i>Numbering, DigitalPrinting</i>)
<i>Insertter</i> New in JDF 1.4	The Insertter inserts one or more “child” components to one “mother” component. See Section 6.6.24, Inserting.
<i>Jacketer</i> New in JDF 1.4	The Jacketer wraps a jacket around a book. See Section 6.6.25, Jacketing.
<i>PaperPath</i> New in JDF 1.2	Paper path module, path that paper follows through the Machine.
<i>PressingStation</i> New in JDF 1.4	The Pressing Station presses the cover to the book block.

Table C-5: ModuleType Attribute Values for Postpress

ModuleType	Description
<i>ShapeCutter</i> New in JDF 1.4	The Shape Cutter produces special shapes like an envelope window or a heart-shaped beer mat. Note that the Shape Cutter Module MAY contain Tools that correspond to the actual dies etc. See Section 6.6.36, ShapeCutting.
<i>SpinePreparer</i> New in JDF 1.4	The Spine Preparer prepares the spine of a book for hard and soft cover production. See Section 6.6.40, SpinePreparation.
<i>SpineTaper</i> New in JDF 1.4	The Spine Taper applies a tape strip to the spine of a book block. See Section 6.6.41, SpineTaping.
<i>Strapper</i> New in JDF 1.4	The Strapper straps a bundle of products. See Section 6.6.45, Strapping
<i>ThreadSealer</i> New in JDF 1.4	The Thread sealer sews and seals a signature at the spine. See Section 6.6.47, ThreadSealing
<i>ThreadSewer</i> New in JDF 1.4	The Thread sewer sews all signatures of a book block together. See Section 6.6.48, ThreadSewing.

Table C-6: ModuleType Attribute Values 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.
<i>Marker</i> New in JDF 1.4	Marker module, excluding in-line finishing.
<i>MimeUnpacker</i> New in JDF 1.4	Module that receives and unpacks the MIME package and fetches the JDF if it is referenced from the JMF.
<i>ReferencedDataCollector</i> New in JDF 1.4	Module that fetches data referenced from the JDF and MAY include data referenced from the PDL. Does not include accepting MIME, unpacking MIME, or fetching the JDF itself.
<i>RIP</i> New in JDF 1.4	RIP module. See Section 6.4.32, RIPing.

Table C-7: ModuleType Attribute Values for PrintingUnitWebPath Modules of Web Printing Devices (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 Print Unit consists of multiple Print Module units.
<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 Remoistening Module is located between last printing unit and dryer.

Table C-7: ModuleType Attribute Values for PrintingUnitWebPath Modules of Web Printing Devices (Section 2 of 2)

ModuleType	Description
<i>Rollstand</i> New in JDF 1.3	The Roll stand feeds the Web into the Process-unit chain.
<i>UVCoater</i> New in JDF 1.3	The UV-Coater module applies UV-varnish with subsequent drying in a UV-dryer.

Table C-8: ModuleType Attribute Values for FolderSuperstructureWebPath Modules of Web Printing Devices

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>MoebiusDeinfinitizer</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 Chill Unit and Folder.
<i>PlanoModule</i> New in JDF 1.3	The Plano Module cuts the Web / ribbon into Sheets and stacks the Sheets to a pile
<i>PloughFoldModule</i> New in JDF 1.3	The Plough Fold Module does a quarter fold to ribbons or webs, mostly found in front of a Folder module
<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.
<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: ModuleType Attribute Values for PostPressComponentPath Modules of Web Printing Devices

ModuleType	Description
<i>BundlingModule</i> New in JDF 1.3	The Bundling Module is used for bundling components
<i>LabelingModule</i> New in JDF 1.3	The Labeling Module is used for labelling a bundle.
<i>PalletizingModule</i> New in JDF 1.3	The Palletizing Module collects the Bundles on a pallet. See “Palletizing” on page 326.
<i>PrintRoll</i> New in JDF 1.3	The Print Roll is used for rolling components. See “PrintRolling” on page 328.
<i>Stacker</i> New in JDF 1.3	Stacks the component to a pile. See “Stacking” on page 331.
<i>Trimmer</i> New in JDF 1.3	Trims the component to its final size. See “Trimming” on page 333.

C.3 NotificationDetails

The Notification Element is used for messaging and logging of events. It is defined in Section 3.11.4.5, Notification. Notifications are grouped into five Classes: *Event*, *Information*, *Warning*, *Error* and *Fatal*. For more about Notification Classes, see Notification/@Class in Table 3.11.4.5, “Notification,” on page 125. In addition to the Classes, the *Type* Attribute and Abstract NotificationDetails Element provide a container for detailed information about the notification.

Elements derived from the Abstract NotificationDetails Element represent a structured and extensible data type. The structure of various predefined NotificationDetails types and their descriptions are listed in the following sections.

C.3.1 Abstract NotificationDetails

The Abstract NotificationDetails Element is empty.

Table C-10: Abstract NotificationDetails

Name	Page	Description

C.3.2 NotificationDetails

Table C-11 defines the Elements that are derived from the Abstract NotificationDetails Element. The value of Notification/@Type is the same as the Element name for the corresponding Notification/NotificationDetails.

Table C-11: List of NotificationDetail Elements (Section 1 of 2)

Name	Page	Description
Barcode	page 884	A bar code has been scanned
FCNKey	page 884	A function key has been activated at a console.
SystemTimeSet	page 884	The system time of a device/Controller/Agent has been set
CounterReset	page 884	The production counter of a device has been reset.
Error	page 884	This Element provides additional information for common errors
Event	page 885	This Element provides additional information for common events.

Table C-11: List of NotificationDetail Elements (Section 2 of 2)

Name	Page	Description
Milestone	page 885	Tracks certain overall milestones concerning the entire Job across all Resources and Processes.

C.3.2.1 Barcode

A bar code has been scanned.

Table C-12: Barcode Element

Name	Data Type	Description
<i>Code</i>	string	Contains the scanned bar code.

C.3.2.2 FCNKey

A function key has been activated at a console.

Table C-13: FCNKey Element

Name	Data Type	Description
<i>Key</i>	integer	Contains the number of that function key.

C.3.2.3 SystemTimeSet

The system time of a device/Controller/Agent has been set, (e.g., readjusted, changed to daylight saving time, etc.).

Table C-14: SystemTimeSet Element

Name	Data Type	Description
<i>NewTime</i>	dateTime	Contains the new time.
<i>OldTime?</i>	dateTime	Contains the old time.

C.3.2.4 CounterReset

The production counter of a device has been reset.

Table C-15: 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.3.2.5 Error

This Element provides additional information for common errors.

Table C-16: Error Element (Section 1 of 2)

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 re-sending policy to fix the error. Values are: <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 891.

Table C-16: Error Element (Section 2 of 2)

Name	Data Type	Description
ErrorData * New in JDF 1.3	element	Additional details of the error.

C.3.2.5.1 ErrorData

This Element provides additional information for locating errors.

Table C-17: ErrorData Element

Name	Data Type	Description
<i>ErrorType</i>	enumeration	Details of the error of the Attribute or Element specified in <i>Path</i> . 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.3.2.6 Event

[New in JDF 1.2](#)

This Element provides additional information for common events.

Table C-18: 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.3.2.7 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 228.) and available/consumed Resources (See “Resource” on page 214.), many Actors in the workflow want to track certain overall milestones concerning the entire Job across all Resources and Processes in order to display this to the operator. Sometimes the JMF recipients cannot determine these milestones from the detailed JDF/JMF. Therefore a more abstract representation of Job status is described by Milestone events. Note that Milestone Elements usually refer to events involving multiple objects, although the Milestone/*@MilestoneType* is specified as a singular. The scope of the Milestone is defined by the parent Notification element

Table C-19: Milestone Element

Name	Data Type	Description
<i>MilestoneType</i>	NMTOKEN	Type of Milestone. Values include those from: Table C-20, “MessageEvents and MilestoneType Values,” on page 886
<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 C-1: Milestone in JMF

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="WorkflowController"
  Timestamp="2005-07-25T12:32:48+02:00" Version="1.4">
  <Signal ID="S1" Type="Notification" xsi:type="SignalNotification">
    <Notification Class="Event" JobID="myJobID"
      Timestamp="2005-05-25T12:32:48+02:00"
      Type="Milestone">
      <Comment>All Proofs sent to customer</Comment>
      <Milestone MilestoneType="ProofSent" TypeAmount="24"/>
    </Notification>
  </Signal>
</JMF>
```

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 usually refer to events involving multiple objects, although the *Milestone/@MilestoneType* is specified as a singular. The scope of the Milestone is defined by the parent Notification Element.

Note: the following symbols are used in the table below:

- **DigDel** means *DigitalDelivery* in the JDF Process column.
- **Delivery** means *Delivery* in the JDF Process column.
- **ArtDell** 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-20: 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	ArtDell	—	Yes	The receiver acknowledged that the files are accessible for their destination.

Table C-20: MessageEvents and MilestoneType Values (Section 2 of 3)

MessageEvents and MilestoneType Values	JDF Process	JDF Intent Resource	Milestone	Page-Status	Description
<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	ArtDell	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).
<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>			Yes	Yes	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.

Table C-20: MessageEvents and MilestoneType Values (Section 3 of 3)

MessageEvents and MilestoneType Values	JDF Process	JDF Intent Resource	Milestone	Page-Status	Description
<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.
<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 C-2: MessageEvents in CustomerInfo

This example 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”.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <CustomerInfo Class="Parameter" ID="C1" Status="Available">
```

```

    CustomerJobName="Job title ...">
  <CustomerMessage Language="FR"
    MessageEvents="JobCompletedSuccessfully JobCompletedWithErrors"
    ShowList="StartTime EndTime Error">
    <Comment Name="UserText">This is the Mighty Mouse brochure job
      that should be approved by Walt Disney
    </Comment>
    <ComChannel ChannelType="Email" Locator="admin@mycompany.com"/>
  </CustomerMessage>
</CustomerInfo>
</ResourcePool>
<ResourceLinkPool>
  <CustomerInfoLink Usage="Input" rRef="C1"/>
</ResourceLinkPool>
</JDF>

```

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 113 for more details on the use of Location.)

Table C-21: Input Tray and Output Bin Names (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>Back</i>	IO*	The value " <i>Rear</i> " is analogous; " <i>Rear</i> " SHOULD be used instead when possible.
<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.
<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>Cassette</i>	IO	The value " <i>Tray-N</i> " is analogous; " <i>Tray-N</i> " SHOULD be used instead when possible.
<i>Center</i>	—	The bin that, when facing the device, can best be identified as ‘center.’ Depreciated 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.

Table C-21: Input Tray and Output Bin Names (Section 2 of 2)

Name	I/O	Description
<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>Lower</i>	IO*	The value <i>Bottom</i> is analogous; " <i>Bottom</i> " SHOULD be used instead when possible.
<i>Main</i>	IO	The value " <i>LargeCapacity</i> " is analogous; " <i>LargeCapacity</i> " SHOULD be used instead when possible.
<i>MyMailbox</i>	O	The Job will be output to the bin that is best identified as "My Mailbox"
<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>PostMarkerInserter</i> New in JDF 1.4	I	The input tray that is downstream of the marking engine and allows the user to pass media through a non-marking paper path for covers and/or inserts.
<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 Web-Fed media.
<i>Roll-N</i>	IO	The Nth location to handle the Nth Web-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.
<i>Upper</i>	IO*	The value " <i>Top</i> " is analogous; " <i>Top</i> " SHOULD be used instead when possible.

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 Message/Command Message not implemented
6	Invalid parameters
7	Insufficient parameters
8	Device not available (Controller exists but not the Device or queue)
9	Message incomplete.
10 New in JDF 1.3	Message Service is busy.
11 New in JDF 1.4	Synchronous mode not supported for message. No <i>@AcknowledgeURL</i> is specified and the Message can only be processed asynchronously and was not processed. (Error)
12 New in JDF 1.4	Asynchronous acknowledge not supported for message. No <i>@AcknowledgeURL</i> is specified and the Message was processed. The resulting <i>Acknowledge</i> can only be emitted asynchronously. (Warning)
13 New in JDF 1.4	Reliable Signals not supported. Subscription denied.
100 – 199	Device and Controller errors
100	Device not running
101	Device incapable of fulfilling request, (e.g., a RIP that has been asked to cut a Sheet).
102	No executable Node exists in the JDF
103	<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.

Table D-1: Return codes for JMF (Section 2 of 2)

ReturnCode	Description
112 New in JDF 1.1	Queue request failed because the Queue is <i>Closed</i> or <i>Blocked</i> and does not accept new entries.
113 New in JDF 1.2	Queue entry is already in the resulting status.
114 Modified in JDF 1.4	QueueEntry/@ <i>Status</i> is already " <i>PendingReturn</i> ", " <i>Completed</i> " or " <i>Aborted</i> " and therefore does not accept changes. Modification note: starting with JDF 1.4, " <i>PendingReturn</i> " added.
115 New in JDF 1.2	Queue entry is not running.
116 New in JDF 1.3	Queue entry already exists. Used when a QueueEntry with identical <i>JobID</i> , <i>JobPartID</i> and <i>Part</i> already exists.
120 New in JDF 1.3	Cannot access referenced URL. URI Reference cannot be resolved. Used when a referenced entity, e.g., a JDF in a SubmitQueueEntry cannot be found.
121 New in JDF 1.3	Unknown <i>DeviceID</i> . No Device is known with the <i>DeviceID</i> specified.
130 New in JDF 1.3	Ganging is not supported. A gang Job has been submitted to a queue that does not support ganging.
131 New in JDF 1.3	<i>GangName</i> not known. A Job has been submitted with an unknown <i>GangName</i> .
200 – ...	Job and pipe specific errors
200	Invalid Resource parameters
201	Insufficient Resource parameters
202	<i>PipeID</i> unknown
203	Unlinked ResourceLink
204 New in JDF 1.3	Could not create new JDF Node.
300 New in JDF 1.3	Authentication denied.
301 New in JDF 1.3	Secure channel not supported - I don't support secure channel for this Message.
302 New in JDF 1.3	Secure channel required - I require secure channel for this Message.
303 New in JDF 1.3	Certificate expired (Some implementations might not be able to send this response because the SSL layer will reject the Message before passing it to the JMF implementation for parsing)
304 New in JDF 1.4	Authentication pending.
305 New in JDF 1.4	Authentication already established.
306 New in JDF 1.4	No authentication request in process.
307 New in JDF 1.4	Certificate Invalid

Appendix 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 450.) that are intended to allow simple, late-in-the-workflow, minor adjustments to the overall color appearance of a Job or portions of a Job.

Note: These color adjustments are not available in any Intent Resource, such as `ColorIntent`. In order to request such adjustment in a Product Intent Job ticket supplied to a print provider, attach to a Product Intent Node an incomplete `ColorCorrection` Process with a `ColorCorrectionParams` Resource specifying the requested `ColorCorrectionOp` Element Attributes.

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>	-180 to +180
<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 += *AdjustLightness*]
- *AdjustContrast* scales the L* channel about mid-scale (where L = 50).
[L = 50 + (L - 50) * (*AdjustContrast* / 100 + 1)]
- *AdjustSaturation* scales the a* and b* channels about zero. [a *= (*AdjustSaturation* / 100 + 1)] and [b *= (*AdjustSaturation* / 100 + 1)]

AdjustCyanRed, *AdjustMagentaGreen* and *AdjustYellowBlue* offset the colors in the a*b* plane along the respective color vector. Lightness (L*) is not changed. Positive values offset towards red, green or blue, and negative values offset towards cyan, magenta or yellow. The adjustment vectors are aligned with the standard SWOP inks. When adjusting Device colors, these adjustments can be approximated by offsets along the vectors of the actual ink colors being used. The angles and unit vectors for SWOP inks (from the CGATS TR001 print characterization) are:

	Red-cyan	Green-Magenta	Blue-yellow
Angle	-129.9	-5.3	94.5
a*	0.641	-0.996	0.078

b* 0.767 0.092 -0.997

So

$$\begin{aligned} \mathbf{a}^* & += 0.641 * \text{AdjustCyanRed} \\ & - 0.996 * \text{AdjustMagentaGreen} \\ & + 0.078 * \text{AdjustYellowBlue} \\ \mathbf{b}^* & += 0.767 * \text{AdjustCyanRed} \\ & + 0.092 * \text{AdjustMagentaGreen} \\ & - 0.997 * \text{AdjustYellowBlue} \end{aligned}$$

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** Resource 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** Resource 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. Stock type is specified in **Media/@StockType** or **MediaIntent/StockType**.

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 Equivalent 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	204	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
67	45# Bond	216	80# Cover
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)
上質紙 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)
中質紙 Chuushitsushi	--	45	--	30	54.2
	--	55	--	36.5	66.3
アート紙 Aatoshi	73	70.5	50.5	46.5	84.9
	90	87	62.5	57.5	104.7
	110	106	76.5	70.5	127.9
	135	130.5	93.5	86.5	157.0
マシンコート紙 Mashinkootoshi	63	61	--	--	73.3
	68	65.6	47	43.5	79.1
	73	70.5	50.5	46.5	84.9
	90	87	62.5	57.5	104.7
	110	106	76.5	70.5	127.9
	135	130.5	93.5	86.5	157.0
アートポスト紙 Aatoposutoshi	180	--	125	--	209.3
	200	--	139	--	232.6
	220	--	153	--	255.0

* The following describes the five paper grades in the above table:

- 上質紙 Joushitsu (“top-quality paper”) contains 100% chemical pulp;
- 中質紙 Chuushitsu (“medium-quality paper”) contains a minimum of 70% chemical pulp;
- アート紙 Aatoshi (“art paper”) is machine coated paper, available in top quality and medium quality (Joushitsu and Chuushitsu);
- マシンコート紙 Mashinkootoshi (“machine coated paper”), also called Kootoshi (コート紙), is machine coated paper given only a thin coat of clay;
- アートポスト紙 Aatoposutoshi (“art-post paper”) is cover stock coated on one side.

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 4)

Media Size	Size in Points	Size in Millimeters	Size in Inches	Notes
A0	2384 x 3370	841 x 1189	33.11 x 46.81	I, J
A1	1684 x 2384	594 x 841	23.39 x 33.11	I, J
A2	1191 x 1684	420 x 594	16.54 x 23.39	I, J
A3	842 x 1191	297 x 420	11.69 x 16.54	I, J
A3Extra	913 x 1262	322 x 445	12.67 x 17.52	
A4	595 x 842	210 x 297	8.27 x 11.69	I, J
A4Extra	667 x 914	235.5 x 322.3	9.27 x 12.69	
A4Plus	595 x 936	210 x 330	8.27 x 13	
A4Tab New in JDF 1.4	638 x 842	225 x 297	8.86 x 11.69	
A5	420 x 595	148 x 210	5.83 x 8.27	I, J
A5Extra	492 x 668	174 x 235	6.85 x 9.25	
A6	297 x 420	105 x 148	4.13 x 5.83	I, J
A7	210 x 297	74 x 105	2.91 x 4.13	I, J
A8	148 x 210	52 x 74	2.05 x 2.91	I, J
A9	105 x 148	37 x 52	1.46 x 2.05	I, J
A10	73 x 105	26 x 37	1.02 x 1.46	I, J
AnsiC	1224 x 1584	431.8 x 558.8	17 x 22	
AnsiD	1584 x 2448	558.8 x 863.6	22 x 34	
AnsiE	2448 x 3168	863.6 x 1118	34 x 44	
ARCHA	648 x 864	228.6 x 304.8	9 x 12	
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

Table G-1: Media Sizes (Section 2 of 4)

Media Size	Size in Points	Size in Millimeters	Size in Inches	Notes
B4	729 x 1032	257 x 364	10.12 x 14.33	J
B5	516 x 729	182 x 257	7.17 x 10.12	J
B6	363 x 516	128 x 182	5.04 x 7.17	J
B7	258 x 363	91 x 128	3.58 x 5.04	J
B8	181 x 258	64 x 91	2.52 x 3.58	J
B9	127 x 181	45 x 64	1.77 x 2.52	J
B10	91 x 127	32 x 45	1.26 x 1.77	J
C4	649 x 918	229 x 324	9.02 x 12.75	I, E
C5	459 x 649	162 x 229	6.38 x 9.02	I, E
C6	323 x 459	114 x 162	4.51 x 6.38	I, E
Comm10	297 x 684	104.8 x 241.3	4.125 x 9.5	E
DL	312 x 624	110 x 220	4.33 x 8.66	I, E
DoublePostcard	567 x 419	200 x 148	7.87 x 5.83	
Env9	279 x 639	98.4 x 225.4	3.875 x 8.875	E
Env10	297 x 684	104.8 x 241.3	4.125 x 9.5	E
Env11	324 x 747	113.3 x 263.5	4.5 x 10.375	E
Env12	342 x 792	120.7 x 279.4	4.75 x 11	E
Env14	360 x 828	127 x 292.1	5 x 11.5	E
EnvC0	2599 x 3676	917 x 1297	36.10 x 51.06	I, E
EnvC1	1837 x 2599	648 x 917	25.51 x 36.10	I, E
EnvC2	1298 x 1837	458 x 648	18.03 x 25.51	I, E
EnvC3	918 x 1296	324 x 458	12.75 x 18.03	I, E
EnvC4	649 x 918	229 x 324	9.02 x 12.75	I, E
EnvC5	459 x 649	162 x 229	6.38 x 9.02	I, E
EnvC6	323 x 459	114 x 162	4.51 x 6.38	I, E
EnvC65	324 x 648	114 x 229	4.51 x 9	E
EnvC7	230 x 323	81 x 113	3.19 x 4.49	I, E
EnvChou3	340 x 666	120 x 235	4.72 x 9.25	E
EnvChou4	255 x 581	90 x 205	3.54 x 8	E
EnvDL	312 x 624	110 x 220	4.33 x 8.66	I, E
EnvInvite	624 x 624	220 x 220	8.66 x 8.66	E
EnvISOB4	708 x 1001	250 x 353	9.84 x 13.9	E
EnvISOB5	499 x 709	176 x 250	6.9 x 9.8	E
EnvISOB6	499 x 354	176 x 125	6.9 x 4.9	E
EnvItalian	312 x 652	110 x 230	4.33 x 9	E
EnvKaku2	680 x 941	240 x 332	9.45 x 13	E
EnvKaku3	612 x 785	216 x 277	8.5 x 10.9	E
EnvMonarch	279 x 540	98.43 x 190.5	3.875 x 7.5	E
EnvPersonal	261 x 468	92.08 x 165.1	3.625 x 6.5	E
EnvPRC1	289 x 468	102 x 165	4 x 6.5	E
EnvPRC2	289 x 499	102 x 176	4 x 6.9	E

Table G-1: Media Sizes (Section 3 of 4)

Media Size	Size in Points	Size in Millimeters	Size in Inches	Notes
EnvPRC3	354 x 499	125 x 176	4.9 x 6.9	E
EnvPRC4	312 x 590	110 x 208	4.33 x 8.2	E
EnvPRC5	312 x 624	110 x 220	4.33 x 8.66	E
EnvPRC6	340 x 652	120 x 230	4.7 x 9	E
EnvPRC7	454 x 652	160 x 230	6.3 x 9	E
EnvPRC8	340 x 876	120 x 309	4.7 x 12.2	E
EnvPRC9	649 x 918	229 x 324	9 x 12.75	E
EnvPRC10	918 x 1298	324 x 458	12.75 x 18	E
EnvYou4	298 x 666	105 x 235	4.13 x 9.25	E
Executive	522 x 756	184.2 x 266.7	7.25 x 10.5	
FanFoldGerman	612 x 864	215.9 x 304.8	8.5 x 12	
FanFoldGermanLegal	612 x 936	215.9 x 330	8.5 x 13	
FanFoldUS	1071 x 792	377.8 x 279.4	14.875 x 11	
Folio	595 x 935	210 x 330	8.27 x 13	
ISOB0	2835 x 4008	1000 x 1414	39.37 x 55.67	I
ISOB01	2004 x 2835	707 x 1000	27.83 x 39.37	I
ISOB2	1417 x 2004	500 x 707	19.68 x 27.83	I
ISOB3	1001 x 1417	353 x 500	13.90 x 19.68	I
ISOB4	709 x 1001	250 x 353	9.84 x 13.90	I
ISOB5	499 x 709	176 x 250	6.9 x 9.8	I
ISOB5Extra	569 x 782	201 x 276	7.9 x 10.8	
ISOB6	354 x 499	125 x 176	4.92 x 6.93	I
ISOB7	249 x 354	88 x 125	3.46 x 4.92	I
ISOB8	176 x 249	62 x 88	2.44 x 3.46	I
ISOB9	125 x 176	44 x 62	1.73 x 2.44	I
ISOB10	88 x 125	31 x 44	1.22 x 1.73	I
Ledger	1224 x 792	431.8 x 279.4	17 x 11	
Legal	612 x 1008	215.9 x 355.6	8.5 x 14	
LegalExtra	684 x 1080	241.3 x 381	9.5 x 15	
Letter	612 x 792	215.9 x 279.4	8.5 x 11	
LetterExtra	684 x 864	241.3 x 304.8	9.5 x 12	
LetterPlus	612 x 913	215.9 x 322.3	8.5 x 12.69	
LetterTabThreeEighthsInch New in JDF 1.4	639 x 792	225.4 x 279.4	8.875 x 11	
LetterTabHalfInch New in JDF 1.4	648 x 792	228.6 x 279.4	9 x 11	
LetterTabFiveEighthsInch New in JDF 1.4	657 x 792	231.8 x 279.4	9.125 x 11	
Monarch	279 x 540	98.43 x 190.5	3.875 x 7.5	E
Postcard	284 x 419	100 x 148	3.94 x 5.83	
PRC16K	414 x 610	146 x 215	5.75 x 8.5	

Table G-1: Media Sizes (Section 4 of 4)

Media Size	Size in Points	Size in Millimeters	Size in Inches	Notes
SRA3 New in JDF 1.4	907 x 1276	320 x 450	12.60 x 17.72	

Appendix H 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 832 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.

Modification note: starting with JDF 1.4, the second column "Sample MimeVersion" replaces "MimeType Version" and rows with same value of MimeType, but with different values of MimeVersion are reduced to a single row with just a sample MimeVersion

Table H-1: MimeType Attribute Values (IANA Registered) (Section 1 of 3)

MimeType	Sample MimeVersion	File Extension	Description [iana-mt] indicates IANA registration
application/mac-binhex40	HQX/4.0	.hqx	Macintosh BinHex 4.0 7-bit encoding [RFC1741] Note: BinHex encoding converts an 8-bit file into a 7-bit format [RFC1741], similar to Uuencoding. BinHex format preserves file Attributes, as well as Macintosh resource forks, and includes CRC (Cyclic Redundancy Check) error-checking. This encoding method works on any type of file, including formatted word processing and spreadsheet files, graphics files and even executable files (i.e., programs or applications). Note: BinHex is not to be confused with MacBinary encoding, which is an 8-bit format.
application/msword	MSWORD/XP	.doc	Microsoft Word

Table H-1: MIMEType Attribute Values (IANA Registered) (Section 2 of 3)

MIMEType	Sample MIMEType-Version	File Extension	Description [iana-mt] indicates IANA registration
application/pdf	PDF/1.6, PDF/X-3:2003	.pdf	Adobe Portable Document Format [PDF1.6] and Portable Document Format (PDF) PDF/X-3 [ISO15930-6:2003]
application/postscript	PS/3	.ps	Adobe PostScript™ See [RFC2045] and [RFC2046]
application/vnd.cip4-jdf+xml	JDF 1.4	.jdf	CIP4 Job Definition Format (JDF) version 1.4.
application/vnd.cip4-jmf+xml	JMF 1.4	.jmf	CIP4 Job Definition Format (JDF) version 1.4 (See Job Messaging Format).
application/vnd.cip3-ppf	PPF/3.0	.ppf	CIP3 Print Production Format (PPF) version 3.0, 1998 [PPF]
application/vnd.hp-PCL	PCL/X	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.iccprofile New in JDF 1.4		.icc .icm	International Color Consortium (ICC) File Format for Color Profiles taken from the binary coded decimal Profile Header Profile Version Number field (bytes 8 through 11) [ICC.1] Creation note: starting with JDF 1.4 this MIMEType replaces “ICC Profile”. See Table H-2
application/vnd.podi-ppml+xml	PPML/2.1	.ppml	Personalized Print Markup Language [PPML]
application/vnd.Quark.QuarkXPress	XPress/6.0	.qxd .qxt .qwd .qwt .qxl .qxb	QuarkXPress [Quark]
application/zip		.zip	ZIP packaging — The actual compression used for each file in a ZIP package is stored in the ZIP package as metadata for each file. Therefore, the FileSpec/@Compression Attribute for the contained file MAY use any <i>Compression</i> value, including “None”, “Compress”, “Gzip” and “ZLIB”.

Table H-1: MimeType Attribute Values (IANA Registered) (Section 3 of 3)

MimeType	Sample MimeType-Version	File Extension	Description [iana-mt] indicates IANA registration
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 [RFC3302] Note: The image/tiff MIME <i>MediaType</i> is assumed to be TIFF Revision 6.0 as defined in detail by Adobe in [TIFF6]. TIFF/IT is a different MIME type.
multipart/related		.mjd .mjm	Multipart/Related with JDF as the first part [RFC2387]
x-world/x-vrml New in JDF 1.4			

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 and File Type Combinations (Section 1 of 3)

MimeType	File Extension	Description [iana-mt] indicates IANA registration
Base64	.mme	Base64 — A format for encoding arbitrary binary information for transmission by electronic mail. [RFC3548]
Compress		Compress — UNIX compression [RFC1977].
DCS	.eps	Document Color Separation (DCS), version 2.0. [DCS2.0]
Deflate		Deflate — The file is compressed using ZIP public domain compression format [RFC1951].
GZip	.gz	Gzip — GNU zip compression technology [RFC1952].
ICC Profile Deprecated in JDF 1.4	.icc .icm	International Color Consortium (ICC) File Format for Color Profiles taken from the binary coded decimal Profile Header Profile Version Number field (bytes 8 through 11) [ICC.1] Deprecation note: starting with JDF 1.4 this MimeType becomes " <i>application/vnd.iccprofile</i> ". See Table H-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]

Table H-2: MimeType and File Type Combinations (Section 2 of 3)

MimeType	File Extension	Description [iana-mt] indicates IANA registration
Tar	.tar	UNIX packaging format.
TIFF/IT	.fp	TIFF/IT [ISO12639:2004] — Full Page — baseline Note: the file format TIFF/IT MUST NOT use the “application/tiff” <i>MimeType</i> . The “image/tiff” <i>MimeType</i> 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 <i>FileType</i> Attribute instead of the <i>MimeType</i> Attribute.
TIFF/IT	.ct	TIFF/IT [ISO12639:2004] — Continuous Tone picture data — baseline
TIFF/IT	.lw	TIFF/IT [ISO12639:2004] — Continuous Line art — baseline
TIFF/IT	.hc	TIFF/IT [ISO12639:2004] — High-resolution Continuous tone image data — baseline
TIFF/IT	.mp	TIFF/IT [ISO12639:2004] — monochrome picture image data — baseline
TIFF/IT	.bp	TIFF/IT [ISO12639:2004] — Binary Picture image data — baseline
TIFF/IT	.bl	TIFF/IT [ISO12639:2004] — Binary Line art image data — baseline
TIFF/IT	.fp	TIFF/IT [ISO12639:2004] — Full Page — profile 1
TIFF/IT	.ct	TIFF/IT [ISO12639:2004] — Continuous Tone picture data — profile 1
TIFF/IT	.lw	TIFF/IT [ISO12639:2004] — Color Line art data — profile 1
TIFF/IT	.hc	TIFF/IT [ISO12639:2004] — High-resolution Continuous tone image data — profile 1
TIFF/IT	.mp	TIFF/IT [ISO12639:2004] — monochrome picture image data — profile 1
TIFF/IT	.bp	TIFF/IT [ISO12639:2004] — Binary Picture image data — profile 1
TIFF/IT	.bl	TIFF/IT [ISO12639:2004] — Binary Line art image data — profile 1
TIFF/IT	.fp	TIFF/IT [ISO12639:2004] — Full Page — baseline Note: this entry and following ones were created in the context of [ISO12639:2004], whereas preceding entries were created in the context of the 1998 version of [ISO12639:2004]
TIFF/IT	.ct	TIFF/IT [ISO12639:2004] — Continuous Tone picture data — baseline
TIFF/IT	.lw	TIFF/IT [ISO12639:2004] — Color Line art data — baseline
TIFF/IT	.hc	TIFF/IT [ISO12639:2004] — High-resolution Continuous tone image data — baseline
TIFF/IT	.mp	TIFF/IT [ISO12639:2004] — monochrome picture image data — baseline
TIFF/IT	.bp	TIFF/IT [ISO12639:2004] — Binary Picture image data — baseline
TIFF/IT	.bl	TIFF/IT [ISO12639:2004] — Binary Line art image data — baseline
TIFF/IT	.sd	TIFF/IT [ISO12639:2004]
TIFF/IT	.fp	TIFF/IT [ISO12639:2004] — Full Page — profile 1
TIFF/IT	.ct	TIFF/IT [ISO12639:2004] — Continuous Tone picture data — profile 1
TIFF/IT	.lw	TIFF/IT [ISO12639:2004] — Color Line art data — profile 1
TIFF/IT	.hc	TIFF/IT [ISO12639:2004] — High-resolution Continuous tone image data — profile 1

Table H-2: MimeType and File Type Combinations (Section 3 of 3)

MimeType	File Extension	Description [iana-mt] indicates IANA registration
TIFF/IT	.mp	TIFF/IT [ISO12639:2004] — monochrome picture image data — profile 1
TIFF/IT	.bp	TIFF/IT [ISO12639:2004] — Binary Picture image data — profile 1
TIFF/IT	.bl	TIFF/IT [ISO12639:2004] — Binary Line art image data — profile 1. Note: There is no TIFF/IT P1 conformance level of SD in [ISO12639:2004]
TIFF/IT	.fp	TIFF/IT [ISO12639:2004] — Full Page — profile 2
TIFF/IT	.ct	TIFF/IT [ISO12639:2004] — Continuous Tone picture data — profile 2
TIFF/IT	.lw	TIFF/IT [ISO12639:2004] — Color Line art data — profile 2
TIFF/IT	.hc	TIFF/IT [ISO12639:2004] — High-resolution Continuous tone image data — profile 2
TIFF/IT	.mp	TIFF/IT [ISO12639:2004] — monochrome picture image data — profile 2
TIFF/IT	.bp	TIFF/IT [ISO12639:2004] — Binary Picture image data — profile 2
TIFF/IT	.bl	TIFF/IT [ISO12639:2004] — Binary Line art image data — profile 2
TIFF/IT	.sd	TIFF/IT [ISO12639:2004]
Type 1 Font	.pfa .pfb	Type 1 Font [type1font]
True Type Font	.ttf	True Type Font [truetypefont]
Open Type Font	.otf	Open Type Font [opentypefont]
UUEncoded	.uue	Uuencode — A set of encoding algorithms for converting files into a series of 7-bit ASCII characters that can be transmitted over the Internet. Originally, uuencode stood for Unix-to-Unix encode, but it has since become a universal protocol used to transfer files between different platforms such as Unix, Windows and Macintosh. Uuencoding is especially popular for sending Email attachments. [uuencode]
ZLIB		ZLIB — ZLIB compression [RFC1950]

Appendix I 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*;
- *Subscription/@Format* and *Subscription/@Template*
- *CustomerMessage/@ShowList* with a fixed format
- *FileSpec/@FileFormat* and *FileSpec/@FileTemplate*;
- *IdentificationField/@ValueFormat* and *IdentificationField/@ValueTemplate*
- *JobField/@JobFormat* and *JobField/@JobTemplate*
- *JobField/@ShowList* with a fixed format
- *Layout/@SheetNameFormat* and *Layout/@SheetNameTemplate*
- *Layout/MarkObject/DynamicField@Format* and *Layout/MarkObject/DynamicField/@Template*.
- *RunList/MetadataMap/@ValueFormat* and *RunList/MetadataMap/@ValueFormat*
- *StrippingParams/@SheetNameFormat* and *StrippingParams/@SheetNameTemplate*

The function defined when using the Attributes *XXXFormat* and *XXXTemplate* is based on the standard C printf() function. (See [K&R].) *XXXFormat* is the first argument and *XXXTemplate* is a comma-separated list of the additional arguments. *XXXTemplate* MAY contain the following operators: +, -, *, /, %, (,) which are evaluated using standard C-operator precedence and the variables defined in the following table which include any valid Partition Key of a Partitioned Resource

Modification note: starting with JDF 1.4, values from *ShowList* are added to Table I-1 below and when two values differ only in case, the one starting with an uppercase letter is deprecated.

Table I-1: Predefined variables used in @XXXTemplate and @ShowList (Section 1 of 4)

Name	Description
<i><a Partition Key></i>	Any Partition Key that is a value of <i>@PartIDKeys</i> in Table 3-27, "Partitionable Resource Element," on page 103.
<i>AcknowledgeType</i>	Corresponds to the JMF <i>AcknowledgeType</i> in the Acknowledge Message. See "Acknowledge" on page 181.
<i>all</i>	Selects all matching Elements. Valid only when <i>FileSpec</i> is used as an Input Resource.
<i>Amount</i> New in JDF 1.4	Amount of the product that was produced. Creation note: before JDF 1.4, was only for <i>CustomerMessage/@ ShowList</i>
<i>CustomerID</i>	<i>CustomerID</i> .
<i>Date</i>	Current <i>Date</i> in [ISO8601:2004] format.
<i>DeviceID</i> New in JDF 1.4	ID of the Device. This is a unique name within the workflow. Creation note: before JDF 1.4, was only for <i>JobField/@ ShowList</i> and <i>CustomerMessage/@ ShowList</i>
<i>element</i>	Integer iterator over all elements in a given page. Restarts at 0 for each page.

Table I-1: Predefined variables used in @XXXTemplate and @ShowList (Section 2 of 4)

Name	Description
<i>EndTime</i> New in JDF 1.4	Actual end time of the Job. Creation note: before JDF 1.4, was only for JobField/@ ShowList and CustomerMessage/@ ShowList
<i>Error</i> New in JDF 1.4	Errors that happened during the Job. Creation note: before JDF 1.4, was only for JobField/@ ShowList and CustomerMessage/@ ShowList
<i>ErrorStats</i> New in JDF 1.4	Statistics on errors that happened during execution. Creation note: before JDF 1.4, was only for JobField/@ ShowList
<i>ExposedMediaName</i> New in JDF 1.4	<i>DescriptiveName</i> of the exposed media, e.g. plate or proof that is being imaged. Creation note: before JDF 1.4, was only for JobField/@ ShowList
<i>FriendlyName</i> New in JDF 1.4	<i>FriendlyName</i> of the Device. Creation note: before JDF 1.4, was only for JobField/@ ShowList and CustomerMessage/@ ShowList
<i>GeneralID:XXX</i> New in JDF 1.4	GeneralID/@IDValue of a GeneralID[@IDUsage = "XXX"] For example if <i>Format</i> = "%i" and <i>Template</i> = " <i>GeneralID:foo</i> " then for: <GeneralID IDUsage="foo" IDValue="1" /> the extracted value is 1.
<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>JobID</i> Deprecated in JDF 1.4	<i>JobID</i> of the Node that is executing. Deprecation note: starting with JDF 1.4, use <i>jobID</i> . Before JDF 1.4, <i>JobID</i> was only for JobField/@ ShowList
<i>jobName</i>	<i>DescriptiveName</i> of the Node that is being processed.
<i>JobName</i> Deprecated in JDF 1.4	<i>DescriptiveName</i> of the Node that is executing. Deprecation note: starting with JDF 1.4, use <i>jobName</i> . Before JDF 1.4, <i>JobName</i> was only for JobField/@ ShowList and CustomerMessage/@ ShowList
<i>jobPartID</i>	<i>JobPartID</i> string.
<i>JobRecipientName</i> New in JDF 1.4	Name of the recipient of the Job. Creation note: before JDF 1.4, was only for JobField/@ ShowList and CustomerMessage/@ ShowList
<i>JobSubmitterName</i> New in JDF 1.4	Name of the submitter of the Job. Creation note: before JDF 1.4, was only for JobField/@ ShowList and CustomerMessage/@ ShowList
<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 .

Table I-1: Predefined variables used in @XXXTemplate and @ShowList (Section 3 of 4)

Name	Description
<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>MediaBrand</i> New in JDF 1.4	Brand of the media that is being printed. Creation note: before JDF 1.4, was only for <i>JobField/@ShowList</i> and <i>CustomerMessage/@ShowList</i>
<i>MediaType</i> New in JDF 1.4	<i>DescriptiveName</i> of the media that is being printed. Creation note: before JDF 1.4, was only for <i>JobField/@ShowList</i> and <i>CustomerMessage/@ShowList</i>
<i>MoonPhase</i> New in JDF 1.4	Phase of the moon at the <i>StartTime</i> of the Job. Creation note: before JDF 1.4, was only for <i>JobField/@ShowList</i>
<i>Operator</i> New in JDF 1.4	Name of the operator. Creation note: before JDF 1.4, was only for <i>JobField/@ShowList</i> and <i>CustomerMessage/@ShowList</i>
<i>OperatorText</i> New in JDF 1.4	Text from the operator as defined in <i>OperatorText</i> .
<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>PrintQuality</i> New in JDF 1.4	The quality of the printout. (High, Normal, Draft or Device specific name) Creation note: before JDF 1.4, was only for <i>JobField/@ShowList</i>
<i>ProoferProfileName</i> New in JDF 1.4	Name of the ICC profile for the proofing Device. Creation note: before JDF 1.4, was only for <i>JobField/@ShowList</i>
<i>PressProfileName</i> New in JDF 1.4	Name of the ICC profile for the final printing (used as intermediate space during proofing). Creation note: before JDF 1.4, was only for <i>JobField/@ShowList</i>
<i>r</i>	Integer iterator over all RunList Partitions with a Partition Key of "Run" in an input RunList .
<i>ri</i>	Integer iterator over all indices in an input "Run" of a RunList . This index is equivalent to looping over a <i>RunIndex</i> .
<i>Resolution</i> New in JDF 1.4	Output resolution. Creation note: before JDF 1.4, was only for <i>JobField/@ShowList</i> and <i>CustomerMessage/@ShowList</i>
<i>ResolutionX</i> New in JDF 1.4	Output resolution in X direction. Creation note: before JDF 1.4, was only for <i>JobField/@ShowList</i> and <i>CustomerMessage/@ShowList</i>
<i>ResolutionY</i> New in JDF 1.4	Output resolution in Y direction. Creation note: before JDF 1.4, was only for <i>JobField/@ShowList</i> and <i>CustomerMessage/@ShowList</i>
<i>ScreeningFamily</i> New in JDF 1.4	Name of the screening family of the output. Creation note: before JDF 1.4, was only for <i>JobField/@ShowList</i> and <i>CustomerMessage/@ShowList</i>

Table I-1: Predefined variables used in @XXXTemplate and @ShowList (Section 4 of 4)

Name	Description
<i>sep</i> Deprecated in JDF 1.4	Separation as defined in the separation Partition Keys of a Partitioned Resource. Deprecation note: starting with JDF 1.4, use the Partition Key “ <i>Separation</i> ”.
<i>SheetNum</i> New in JDF 1.3	Integer iterator over the sheet number of a document. Creation note: before JDF 1.4, was only for JobField/@ShowList
<i>StartTime</i> New in JDF 1.4	Actual start time of the Job Creation note: before JDF 1.4, was only for JobField/@ShowList and CustomerMessage/@ShowList
<i>surf</i> Deprecated in JDF 1.3	Surface string, “ <i>Front</i> ” or “ <i>Back</i> ”. Deprecation note: starting with JDF 1.3, use the Partition Key “ <i>Side</i> ” 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> New in JDF 1.3	Total # of pages in a document.
<i>UserText</i> New in JDF 1.4	For JobField , <i>UserText</i> references user-defined text in JobField/@UserText . For CustomerMessage , <i>UserText</i> references user defined text in Comment[@Name = “UserText”] . Creation note: before JDF 1.4, was only for JobField/@ShowList and CustomerMessage/@ShowList
<i>Warning</i> New in JDF 1.4	Warnings that happened during the Job. Warnings don't lose information in the resulting Job, while errors do. Creation note: before JDF 1.4, was only for JobField/@ShowList and CustomerMessage/@ShowList

Example I-1: @FileTemplate and @FileFormat

With **JobID**= “j001” and a **RunList** defining 2024 created files, this example will iterate over all created files and place them into:

```
"file://myserver.mydomain.com/next/j001/0000/m0000.pdf"
```

```
...
```

```
"file://myserver.mydomain.com/next/j001/0020/m0023.pdf"
```

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobID="j001" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <RunList Class="Parameter" ID="R1" Status="Available">
      <LayoutElement>
        <FileSpec FileFormat=
          "file://myserver.mydomain.com/next/%s/%4.i/m%4.i.pdf"
          FileTemplate="JobID,i/100,i%100"/>
        </LayoutElement>
      </RunList>
    </ResourcePool>
    <ResourceLinkPool>
      <RunListLink Usage="Output" rRef="R1"/>
    </ResourceLinkPool>
</JDF>
```

Appendix J 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.

J.1 Semantics of the RunList/@Directory Attribute

The *Directory* Attribute defines a directory where the files that are associated with this **RunList** SHOULD be copied to or from. If *Directory* is specified, it 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** Resource is supplied indicating that the **FileSpec** is contained in another file, the Base URI is the Absolute URI of where the JDF Consumer extracted the container file (whether or not *Directory* is specified). (See "FileSpec" on page 531.)

After determining the Base URI depending on the presence or absence of **RunList/@Directory** and **FileSpec/Container/FileSpec** Resource as described above, each *URL* Attribute in a **RunList** relement (e.g., **LayoutElement/FileSpec/@URL** or **InsertSheet/Layout/Media/QualityControlResult/FileSpec/@URL**) is used in combination with the Base URI to form the Resolved URI as follows according to one of the following mutually exclusive patterns.²

- 1 **RunList URL** starts with a scheme (token ending with ":", (e.g., "*file:*" or "*cid:*"): ³

Resolved URI = the entire **RunList** URL (and the Base URI is ignored).

- 2 **RunList URL** starts with an authority/host (starts with "//", (e.g., "*//www.cip4.org*")):

Resolved URI = the Base URI scheme, followed by the **RunList** URL authority/host followed by its absolute path (which MAY be empty).

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/@URL** 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 K References

Throughout this specification references to other documents are indicated by short symbolic names inside square brackets, (e.g., [ICC.1]). Implementers need to read and conform to such referenced documents when implementing a part of this specification with such a reference. The reader is directed to this Document References section to find the full title, date, source and availability of all such references.

Table K-1: References (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.adsm.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 K-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-ffa.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.1:2004-10</i> <i>File Format for Color Profiles, Version 4.2.0.0</i></p> <p>Date: 2004 Produced by: International Color Consortium (ICC) Available at: http://www.color.org/icc_specs2.xalter</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 K-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 K-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 K-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>
[ISO15076-1:2005]	<p>ISO 15076-1:2005</p> <p><i>Image technology colour management -- Architecture, profile format and data structure -- Part 1: Based on ICC.1:2004-10. See [ICC.1].</i></p> <p>Date: 2005 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>

Table K-1: References (Section 6 of 13)

Term	Definition
[ISO15930-3:2002]	<p><i>ISO 15930-3:2002</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: 2002 Produced by: ISO Available at: http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</p>
[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.jisa.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>

Table K-1: References (Section 7 of 13)

Term	Definition
[PJTF]	<p><i>The Portable Job Ticket Format</i> Version 1.1</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>
[PPD]	<p><i>Adobe PostScript Printer Description File Format Specification</i> Version 4.3</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> Version 3.0</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> Version 2.1</p> <p>Produced by: Print On Demand Initiative (PODi) Available at: http://www.podi.org</p>
[PrintTalk]	<p><i>PrintTalk Implementation</i> Version 1.1</p> <p>Produced by: PrintTalk Consortium Available at: http://www.printtalk.org/.</p>
[PS]	<p><i>PostScript Language Reference (Redbook)</i> Third Edition</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>
[PWGFINMIB]	<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]	See http://www.quark.com .
[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>

Table K-1: References (Section 8 of 13)

Term	Definition
[RFC1741]	<p><i>RFC 1741</i> <i>MIME Content Type for BinHex Encoded Files</i>, by Faltstrom, P., Crocker, D. and Fair, E.</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>
[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.</p> <p>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.</p> <p>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.</p> <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>
[RFC1951]	<p><i>RFC 1951</i> <i>DEFLATE Compressed Data Format Specification version 1.3</i>, by Deutsch, P.</p> <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.</p> <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.</p> <p>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)</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>

Table K-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 K-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, by T. Hastings, R. Herriot, R. deBry, S. Isaacson and P. Powell.</i></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, by Parsons, G., Rafferty, J.</i></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 by T. Hastings, H. Lewis and R. Bergman.</i></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 by R. deBry, R. Herriot, T. Hastings, K. Ocke and P. Zehler.</i></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, by S. Josefsson</i></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 by H. Schulzrinne</i></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 K-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.</p> <p>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 K-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>
[Unicode5.0]	<p><i>The Unicode Standard, Version 5.0,</i></p> <p>Date: November 3, 2006 Produced by: The Unicode Consortium Available at: http://www.unicode.org/book/aboutbook.html</p>
[uencode]	<p><i>Unix Uuencode, 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/uuencode.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>

Table K-1: References (Section 13 of 13)

Term	Definition
[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>
[XPath]	<p><i>XML Path Language (XPath) Version 1.0</i> <i>Version W3C Recommendation 16 November 1999</i></p> <p>Date: 16 November 1999 Produced by: World Wide Web Consortium (W3C) Available at: http://www.w3.org/TR/xpath.html.</p>
[X.509]	<p><i>Public-Key Infrastructure (X.509) (pkix)</i> <i>Version 1.1</i></p> <p>Date: 14 February 2008 Produced by: IETF. Available at: http://www.ietf.org/html.charters/pkix-charter.html.</p>

Appendix L 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 L-1: Naming Scheme for Hole Patterns

Name	Description
General	<m i>: m = metric (millimeter is used), i = imperial (inch, where 1 inch = 25.4 mm)
Ring Binding	R<#holes><m i>-<variant> Example: R2m-DIN = RingBind, 2 hole, metric, DIN
Plastic Comb	P<pitch><m i>-<shape>-<#thumbcuts>t Example: P16:9m-round-0t = Plastic Comb, 9/16" pitch (16:9), round, no thumbcut
Wire Comb	W<pitch><m i>-<shape>-<#thumbcuts>t Example: W2:1i-square-1t = Wire Comb, 1/2" pitch (2:1), square, one thumbcut
Coil/Spiral	C<pitch><m i>-<shape>-<#thumbcuts>t Example: C9.5m-round-0t = Coil, 9.5 mm, round, no thumbcut
Special	S<#holes> Example: S1-generic

Table L-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 L-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 L-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 L-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 diameter range: 6/16" - 1/2"	29.25 (≅ 0.25" + ½ x 5/16" = 13/32")	Left		

Table L-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 L-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 for blocks of 7 mm for blocks of <= 15 mm thick	31.18 (≅ 11 mm)	Left	Only used with A5	DIN 5005:1991

Table L-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" + 1/2 diameter 0.375 - 0.5"	29.25 (≅ 0.25" + 1/2 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" + 1/2 diameter 0.375 - 0.5"	29.25 (≅ 13/32")	Left for letter Top for ledger		
11 Holes (R11...)										

Table L-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/ 2 slots Pattern: H-S-H-H-S-H-S-H-H-H-S-H	4.5 - 13 mm	31.18 (≅ 11 mm)	Left for A4 Top for A3		Printer Finishing MIB

Table L-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 L-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 (I)	Default Process Edge	Usage Notes	Source Standard
W2_li-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_li-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_li-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 L-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 (I)	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 M 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 Attributes of **FileSpec**: *MimeType*, *URL*, *Compression* and the **FileSpec Container** Subelement. These use cases include container packaging files, such as tar, zip and Multipart/Related files and container compression and encoding files, each of which require one or more *Container* Subelements to link one **FileSpec** with its container **FileSpec**.

M.1 Examples of Attribute Values of FileSpec

Table M-1 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 531).

Table M-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 M-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.

M.2 Corresponding XML examples

The above use case examples are represented in XML as follows:

Example M-1: FileSpec #1

Single a.pdf PDF file, No Compression:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" ID="F1" Status="Available"
      MimeType="application/pdf" URL="ftp://www.any.com/share/a.pdf" />
  </ResourcePool>
  <ResourceLinkPool>
    <FileSpecLink Usage="Input" rRef="F1" />
  </ResourceLinkPool>
</JDF>
```

Example M-2: FileSpec #2

Single a.pdf PDF file, with Gzip compression:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" ID="F1" Status="Available"
      Compression="Gzip" MimeType="application/pdf" URL="a.pdf">
      <Container>
        <FileSpec MimeType="Gzip" URL="ftp://www.any.com/a.gz" />
      </Container>
    </FileSpec>
  </ResourcePool>
  <ResourceLinkPool>
    <FileSpecLink Usage="Input" rRef="F1" />
  </ResourceLinkPool>
</JDF>
```

Example M-3: FileSpec #3

Single a.pdf PDF file, no compression, but Base64 encoded:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" ID="F1" Status="Available"
      Compression="Base64" MimeType="application/pdf" URL="a.pdf">
      <Container>
        <FileSpec MimeType="Base64" URL="ftp://www.any.com/a.mme"/>
      </Container>
    </FileSpec>
  </ResourcePool>
  <ResourceLinkPool>
    <FileSpecLink Usage="Input" rRef="F1"/>
  </ResourceLinkPool>
</JDF>
```

Example M-4: FileSpec #4

Single PDF file, no compression, but BinHex encoded:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" ID="F1" Status="Available"
      Compression="BinHex" MimeType="application/pdf" URL="a.pdf">
      <Container>
        <FileSpec MimeType="application/mac-binhex40"
          URL="ftp://www.any.com/a.hqx"/>
      </Container>
    </FileSpec>
  </ResourcePool>
  <ResourceLinkPool>
    <FileSpecLink Usage="Input" rRef="F1"/>
  </ResourceLinkPool>
</JDF>
```

Example M-5: FileSpec #5

Single a.pdf PDF file, in b.zip ZIP file containing one or more files:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" ID="F1" Status="Available"
      Compression="ZLIB" MimeType="application/pdf" URL="a.pdf">
      <Container>
        <FileSpec MimeType="application/zip" URL="ftp://www.any.com/b.zip"/>
      </Container>
    </FileSpec>
  </ResourcePool>
  <ResourceLinkPool>
    <FileSpecLink Usage="Input" rRef="F1"/>
  </ResourceLinkPool>
</JDF>
```

Example M-6: FileSpec #6

Single a.pdf PDF file, in a b.zip with one or more files, and the b.zip packaging file itself is Base64 encoded as b.mme. To read, decode, then decompress. To write, compress, then encode.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
```

```

<ResourcePool>
  <FileSpec Class="Parameter" ID="F1" Status="Available"
    Compression="Deflate" MimeType="application/pdf" URL="a.pdf">
    <Container>
      <FileSpec Compression="Base64" MimeType="application/zip" URL="b.zip">
        <Container>
          <FileSpec MimeType="Base64" URL="ftp://www.any.com/b.mme"/>
        </Container>
      </FileSpec>
    </Container>
  </FileSpec>
</ResourcePool>
<ResourceLinkPool>
  <FileSpecLink Usage="Input" rRef="F1"/>
</ResourceLinkPool>
</JDF>

```

Example M-7: FileSpec #7

Single myFiles/myPicture.jpg file in myNestedZip.zip file with one or many files, but the myNestedZip.zip is itself zipped into c.zip

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" ID="F1" Status="Available"
      Compression="Deflate" MimeType="image/jpeg" URL="myFiles/myPicture.jpg">
    <Container>
      <FileSpec Compression="Deflate" MimeType="application/zip"
        URL="myNestedZip.zip">
      <Container>
        <FileSpec MimeType="application/zip"
          URL="ftp://www.any.com/c.zip"/>
        </Container>
      </FileSpec>
    </Container>
  </FileSpec>
</ResourcePool>
<ResourceLinkPool>
  <FileSpecLink Usage="Input" rRef="F1"/>
</ResourceLinkPool>
</JDF>

```

Example M-8: FileSpec #8

Single a.pdf PDF file, which is ZLIB compressed in a c.zip with one or many files which is contained in a tar file and compressed with ZLIB into a.tar.gz file.:

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" ID="F1" Status="Available"
      Compression="ZLIB" MimeType="application/pdf" URL="a.pdf">
    <Container>
      <FileSpec MimeType="application/zip" URL="c.zip">
      <Container>
        <FileSpec Compression="ZLIB" MimeType="Tar" URL="d.tar">
        <Container>
          <FileSpec MimeType="GZip"
            URL="ftp://www.any.com/d.tar.gz"/>
          </Container>
        </FileSpec>
      </Container>
    </FileSpec>
  </Container>
</FileSpec>

```



```

    </Container>
  </FileSpec>
</ResourcePool>
<ResourceLinkPool>
  <FileSpecLink Usage="Input" rRef="F1"/>
</ResourceLinkPool>
</JDF>

```

M.3 Additional examples showing Partitioning of FileSpec

This section has additional examples of container files and various schemes of Partitioning.

Example M-9: FileSpec #9

Package b.zip contains multiple pdf files a.pdf, b.pdf etc.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" Status="Available" ID="ID_002"
      MimeType="application/zip"
      URL="ftp://www.any.com/b.zip"/>
    <FileSpec Class="Parameter" Status="Available" ID="A_FILE"
      Compression="Deflate" MimeType="application/pdf"
      URL="a.pdf">
      <Container>
        <FileSpecRef rRef="ID_002"/>
      </Container>
    </FileSpec>
    <FileSpec Class="Parameter" Status="Available" ID="B_FILE"
      Compression="Deflate" MimeType="application/pdf"
      URL="b.pdf">
      <Container>
        <FileSpecRef rRef="ID_002"/>
      </Container>
    </FileSpec>
  </ResourcePool>
  <ResourceLinkPool>
    <FileSpecLink Usage="Input" rRef="A_FILE"/>
    <FileSpecLink Usage="Input" rRef="B_FILE"/>
  </ResourceLinkPool>
</JDF>

```

Example M-10: FileSpec #10

Package b.zip contains two pdf files a.pdf, b.pdf and a tiff, c.tiff used by a Partitioned Resource

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" Status="Available" ID="ID_003"
      MimeType="application/zip" URL="ftp://www.any.com/b.zip"/>
    <FileSpec Class="Parameter" Status="Available" ID="ALL_FILES"
      Compression="Deflate" MimeType="application/pdf"
      PartIDKeys="PartVersion">
      <Container>
        <FileSpecRef rRef="ID_003"/>
      </Container>
      <FileSpec PartVersion="English" URL="a.pdf"/>
      <FileSpec PartVersion="French" URL="b.pdf"/>
      <FileSpec MimeType="application/tif" PartVersion="German" URL="c.tif"/>
    </FileSpec>
  </ResourcePool>
  <ResourceLinkPool>

```

```

    <FileSpecLink Usage="Input" rRef="ALL_FILES"/>
  </ResourceLinkPool>
</JDF>

```

Example M-11: FileSpec #11

Single a.pdf PDF file, in b.zip which is contained in c.tar file:

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" Status="Available" ID="ID_004_TAR" MimeType="Tar"
      URL="ftp://www.any.com/c.tar"/>
    <FileSpec Class="Parameter" Status="Available" ID="ID_004_ZIP"
      MimeType="application/zip" URL="b.zip">
      <Container>
        <FileSpecRef rRef="ID_004_TAR"/>
      </Container>
    </FileSpec>
    <FileSpec Class="Parameter" Status="Available" ID="C_FILE"
      Compression="Deflate" MimeType="application/pdf" URL="a.pdf">
      <Container>
        <FileSpecRef rRef="ID_004_ZIP"/>
      </Container>
    </FileSpec>
  </ResourcePool>
  <ResourceLinkPool>
    <FileSpecLink Usage="Input" rRef="C_FILE"/>
  </ResourceLinkPool>
</JDF>

```

Example M-12: FileSpec #11.1 — No Partitioning

Multiple files in several zip's contained in a tar file, various examples with and without Partitioning,

So the file layout looks like:

```

e.tar
  c.zip
    a.pdf
    b.pdf
  d.zip
    a.pdf
    b.pdf

```

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" Status="Available" ID="ID_005_TAR"
      MimeType="Tar" URL="ftp://www.any.com/e.tar"/>
    <FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_C"
      MimeType="application/zip" URL="c.zip">
      <Container>
        <FileSpecRef rRef="ID_005_TAR"/>
      </Container>
    </FileSpec>
    <FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_D"
      MimeType="application/zip" URL="d.zip">
      <Container>
        <FileSpecRef rRef="ID_005_TAR"/>
      </Container>
    </FileSpec>
    <FileSpec Class="Parameter" Status="Available" ID="A_ENGLISH_FILE"
      Compression="Deflate" MimeType="application/pdf"
      URL="a.pdf">
      <Container>

```

```

        <FileSpecRef rRef="ID_005_ZIP_C"/>
    </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="B_ENGLISH_FILE"
    Compression="Deflate" MimeType="application/pdf"
    URL="b.pdf">
    <Container>
        <FileSpecRef rRef="ID_005_ZIP_C"/>
    </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="A_GERMAN_FILE"
    Compression="Deflate" MimeType="application/pdf"
    URL="a.pdf">
    <Container>
        <FileSpecRef rRef="ID_005_ZIP_D"/>
    </Container>
</FileSpec>
<FileSpec Class="Parameter" Status="Available" ID="B_GERMAN_FILE"
    Compression="Deflate" MimeType="application/pdf"
    URL="b.pdf">
    <Container>
        <FileSpecRef rRef="ID_005_ZIP_D"/>
    </Container>
</FileSpec>
</ResourcePool>
<ResourceLinkPool>
    <FileSpecLink Usage="Input" rRef="A_ENGLISH_FILE"/>
    <FileSpecLink Usage="Input" rRef="B_ENGLISH_FILE"/>
    <FileSpecLink Usage="Input" rRef="A_GERMAN_FILE"/>
    <FileSpecLink Usage="Input" rRef="B_GERMAN_FILE"/>
</ResourceLinkPool>
</JDF>

```

Example M-13: FileSpec #11.2 — Intermediate container Partitioned

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
    Type="ProcessGroup" JobPartID="ID300" Version="1.4">
    <ResourcePool>
        <FileSpec Class="Parameter" Status="Available" ID="ID_005_TAR" MimeType="Tar"
            URL="ftp://www.any.com/e.tar"/>
        <FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIPS"
            MimeType="application/zip" PartIDKeys="PartVersion">
            <Container>
                <FileSpecRef rRef="ID_005_TAR"/>
            </Container>
            <FileSpec PartVersion="English" URL="c.zip"/>
            <FileSpec PartVersion="German" URL="d.zip"/>
        </FileSpec>
        <FileSpec Class="Parameter" Status="Available" ID="A_ENGLISH_FILE"
            Compression="Deflate" MimeType="application/pdf"
            URL="a.pdf">
            <Container>
                <FileSpecRef rRef="ID_005_ZIPS">
                    <Part PartVersion="English"/>
                </FileSpecRef>
            </Container>
        </FileSpec>
        <FileSpec Class="Parameter" Status="Available" ID="B_ENGLISH_FILE"
            Compression="Deflate" MimeType="application/pdf"
            URL="b.pdf">
            <Container>
                <FileSpecRef rRef="ID_005_ZIPS">
                    <Part PartVersion="English"/>
                </FileSpecRef>
            </Container>
        </FileSpec>
    </ResourcePool>
</JDF>

```

```

    </Container>
  </FileSpec>
  <FileSpec Class="Parameter" Status="Available" ID="A_GERMAN_FILE"
    Compression="Deflate" MimeType="application/pdf"
    URL="a.pdf">
    <Container>
      <FileSpecRef rRef="ID_005_ZIPS">
        <Part PartVersion="German"/>
      </FileSpecRef>
    </Container>
  </FileSpec>
  <FileSpec Class="Parameter" Status="Available" ID="B_GERMAN_FILE"
    Compression="Deflate" MimeType="application/pdf"
    URL="b.pdf">
    <Container>
      <FileSpecRef rRef="ID_005_ZIPS">
        <Part PartVersion="German"/>
      </FileSpecRef>
    </Container>
  </FileSpec>
</ResourcePool>
<ResourceLinkPool>
  <FileSpecLink Usage="Input" rRef="A_ENGLISH_FILE"/>
  <FileSpecLink Usage="Input" rRef="B_ENGLISH_FILE"/>
  <FileSpecLink Usage="Input" rRef="A_GERMAN_FILE"/>
  <FileSpecLink Usage="Input" rRef="B_GERMAN_FILE"/>
</ResourceLinkPool>
</JDF>

```

Example M-14: FileSpec #11.3 — the pdf is Partitioned

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" Status="Available" ID="ID_005_TAR"
      MimeType="Tar" URL="ftp://www.any.com/e.tar"/>
    <FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_C"
      MimeType="application/zip" URL="c.zip">
      <Container>
        <FileSpecRef rRef="ID_005_TAR"/>
      </Container>
    </FileSpec>
    <FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_D"
      MimeType="application/zip" URL="d.zip">
      <Container>
        <FileSpecRef rRef="ID_005_TAR"/>
      </Container>
    </FileSpec>
    <FileSpec Class="Parameter" Status="Available" ID="ALL_FILES"
      Compression="Deflate" PartIDKeys="PartVersion DocIndex">
      <!-- English Files -->
      <FileSpec PartVersion="English">
        <Container>
          <FileSpecRef rRef="ID_005_ZIP_C"/>
        </Container>
        <!-- English File A -->
        <FileSpec DocIndex="1" MimeType="application/pdf" URL="a.pdf"/>
        <!-- English File B -->
        <FileSpec DocIndex="2" MimeType="application/pdf" URL="b.pdf"/>
      </FileSpec>
      <!-- German Files -->
      <FileSpec PartVersion="German">
        <Container>
          <FileSpecRef rRef="ID_005_ZIP_D"/>
        </Container>
      </FileSpec>
    </FileSpec>
  </ResourcePool>
</JDF>

```

```

    </Container>
    <!-- German File A -->
    <FileSpec DocIndex="1" MimeType="application/pdf" URL="a.pdf"/>
    <!-- German File B -->
    <FileSpec DocIndex="2" MimeType="application/pdf" URL="b.pdf"/>
  </FileSpec>
</FileSpec>
</ResourcePool>
<ResourceLinkPool>
  <FileSpecLink Usage="Input" rRef="ALL_FILES"/>
</ResourceLinkPool>
</JDF>

```

Example M-15: FileSpec #11.3a — the pdf is Partitioned, Different File Layout

As above but the file layout is not reflected in the container structure, the files are intermingled

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" Status="Available" ID="ID_005_TAR"
      MimeType="Tar" URL="ftp://www.any.com/e.tar"/>
    <FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_C"
      MimeType="application/zip" URL="c.zip">
      <Container>
        <FileSpecRef rRef="ID_005_TAR"/>
      </Container>
    </FileSpec>
    <FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_D"
      MimeType="application/zip" URL="d.zip">
      <Container>
        <FileSpecRef rRef="ID_005_TAR"/>
      </Container>
    </FileSpec>
    <FileSpec Class="Parameter" Status="Available" ID="ALL_FILES"
      Compression="Deflate" MimeType="application/pdf"
      PartIDKeys="PartVersion DocIndex">
      <!-- English Files -->
      <FileSpec PartVersion="English">
        <!-- English File A -->
        <FileSpec DocIndex="1" URL="a.pdf">
          <Container>
            <FileSpecRef rRef="ID_005_ZIP_C"/>
          </Container>
        </FileSpec>
        <!-- English File B -->
        <FileSpec DocIndex="2" URL="a.pdf">
          <Container>
            <FileSpecRef rRef="ID_005_ZIP_D"/>
          </Container>
        </FileSpec>
      </FileSpec>
      <!-- German Files -->
      <FileSpec PartVersion="German">
        <!-- German File A -->
        <FileSpec DocIndex="1" URL="b.pdf">
          <Container>
            <FileSpecRef rRef="ID_005_ZIP_C"/>
          </Container>
        </FileSpec>
        <!-- German File B -->
        <FileSpec DocIndex="2" URL="b.pdf">
          <Container>
            <FileSpecRef rRef="ID_005_ZIP_D"/>
          </Container>
        </FileSpec>
      </FileSpec>
    </FileSpec>
  </ResourcePool>
</JDF>

```

```

        </Container>
      </FileSpec>
    </FileSpec>
  </ResourcePool>
  <ResourceLinkPool>
    <FileSpecLink Usage="Input" rRef="ALL_FILES"/>
  </ResourceLinkPool>
</JDF>

```

Example M-16: FileSpec #11.4 — Both Partitioned

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" Status="Available" ID="ID_005_TAR" MimeType="Tar"
      URL="ftp://www.any.com/e.tar"/>
    <FileSpec ID="ID_005_ZIPS" MimeType="application/zip"
      PartIDKeys="PartVersion">
      <Container>
        <FileSpecRef rRef="ID_005_TAR"/>
      </Container>
      <FileSpec PartVersion="English" URL="c.zip"/>
      <FileSpec PartVersion="German" URL="d.zip"/>
    </FileSpec>
    <FileSpec Compression="Deflate" ID="ALL_FILES"
      PartIDKeys="PartVersion DocIndex">
      <!-- English Files -->
      <FileSpec PartVersion="English">
        <Container>
          <FileSpecRef rRef="ID_005_ZIPS">
            <Part PartVersion="English"/>
          </FileSpecRef>
        </Container>
        <!-- English File A -->
        <FileSpec DocIndex="1" MimeType="application/pdf" URL="a.pdf"/>
        <!-- English File B -->
        <FileSpec DocIndex="2" MimeType="application/pdf" URL="b.pdf"/>
      </FileSpec>
      <!-- German Files -->
      <FileSpec PartVersion="German">
        <Container>
          <FileSpecRef rRef="ID_005_ZIPS">
            <Part PartVersion="German"/>
          </FileSpecRef>
        </Container>
        <!-- German File A -->
        <FileSpec DocIndex="1" MimeType="application/pdf" URL="a.pdf"/>
        <!-- German File B -->
        <FileSpec DocIndex="2" MimeType="application/pdf" URL="b.pdf"/>
      </FileSpec>
    </FileSpec>
  </ResourcePool>
  <ResourceLinkPool>
    <FileSpecLink Usage="Input" rRef="ALL_FILES"/>
  </ResourceLinkPool>
</JDF>

```

Example M-17: FileSpec #12

Multiple PDF and TIFF files in several zip's contained in a tar file. Use all PDF files in c.zip, using the `FileSpec/@FileFormat` mechanism and just Pictures/TIFS/a.pdf in d.zip. File layout looks like:

```
e.tar
```

```

c.zip
  a.pdf
  a.tif
  b.pdf
  b.tif
d.zip
  PDFS/a.pdf
  PDFS/b.pdf
  Pictures/TIFS/a.pdf
  Pictures/TIFS/b.pdf

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <FileSpec Class="Parameter" Status="Available" ID="ID_005_TAR" MimeType="Tar"
      URL="ftp://www.any.com/e.tar"/>
    <FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_C"
      MimeType="application/zip" URL="c.zip">
      <Container>
        <FileSpecRef rRef="ID_005_TAR"/>
      </Container>
    </FileSpec>
    <FileSpec Class="Parameter" Status="Available" ID="ID_005_ZIP_D"
      MimeType="application/zip" URL="d.zip">
      <Container>
        <FileSpecRef rRef="ID_005_TAR"/>
      </Container>
    </FileSpec>
    <FileSpec Class="Parameter" Status="Available" ID="PDF_FILES"
      Compression="Deflate" FileFormat="%s.pdf" FileTemplate="all"
      MimeType="application/pdf">
      <Container>
        <FileSpecRef rRef="ID_005_ZIP_C"/>
      </Container>
    </FileSpec>
    <FileSpec Class="Parameter" Status="Available" ID="Pictures"
      Compression="Deflate" URL="Pictures/TIFS/a.pdf">
      <Container>
        <FileSpecRef rRef="ID_005_ZIP_D"/>
      </Container>
    </FileSpec>
  </ResourcePool>
  <ResourceLinkPool>
    <FileSpecLink Usage="Input" rRef="PDF_FILES"/>
    <FileSpecLink Usage="Input" rRef="Pictures"/>
  </ResourceLinkPool>
</JDF>

```

M.4 Example of an Intent Job Ticket with a doubly nested ZIP packaging file

Here is a complete example of an intent Job ticket using `ArtDeliveryIntent` with a doubly nested packaging file. The example shows a `myPictures.jpg` file that is contained in `myNestedZip.zip` file which is contained in `myZip.zip` file:

Example M-18: Intent Job Ticket

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="FileSpecProposal01" JobID="bookJob"
  JobPartID="bookJob-1" Status="Waiting" Type="Product" Version="1.4">
  <ResourcePool>
    <ArtDeliveryIntent Class="Intent" ID="FileSpecProposal02" Status="Draft">

```

```

    <ArtDelivery ArtDeliveryType="DigitalMedia">
      <RunListRef rRef="FileSpecProposal05"/>
    </ArtDelivery>
  </ArtDeliveryIntent>
  <RunList ID="FileSpecProposal05" Class="Parameter" Status="Available">
    <LayoutElement>
      <FileSpec Compression="Deflate" MimeType="image/jpeg"
        URL="myFiles/myPicture.jpg">
        <Container>
          <FileSpecRef rRef="ID_002"/>
        </Container>
      </FileSpec>
    </LayoutElement>
  </RunList>
  <Component Amount="100" Class="Quantity" ComponentType="FinalProduct"
    DescriptiveName="FileSpec Test" ID="FileSpecProposal03"
    Status="Unavailable"/>
  <FileSpec Class="Parameter" Status="Available" ID="ID_001"
    MimeType="application/zip" URL="http://www.CIP4.org/myZip.zip"/>
  <FileSpec Class="Parameter" Status="Available" Compression="Deflate"
    ID="ID_002" MimeType="application/zip" URL="myNestedZip.zip">
    <Container>
      <FileSpecRef rRef="ID_001"/>
    </Container>
  </FileSpec>
</ResourcePool>
<ResourceLinkPool>
  <ComponentLink Amount="100" Usage="Output" rRef="FileSpecProposal03"/>
  <ArtDeliveryIntentLink Usage="Input" rRef="FileSpecProposal02"/>
</ResourceLinkPool>
</JDF>

```

M.5 AppOS and OSVersion Attributes

[New in JDF 1.2](#)

This section lists examples values for the following Attributes of the **FileSpec** Resource: *AppOS* and *OSVersion*. The listing is intended to be exhaustive for the most likely operating systems that are routinely used in JDF applications. However, other operating systems and combinations MAY be used as well. When operating systems have new versions, they can be used and SHOULD follow the patterns established in this the following table.

Table M-2: AppOS and OSVersion Examples (Section 1 of 2)

AppOS	OSVersion	Description
Linux	2.2	Linux operation system
Mac	10.2.4	Macintosh operation system
Solaris	4.0	Sun Solaris operation system
UNIX	BSD	Berkeley UNIX
UNIX	V	System V UNIX
UNIX	V.1	System V UNIX
UNIX	V.2	System V UNIX
UNIX	V.3	System V UNIX
UNIX	PC	UNIX for the PC
Windows	95	Windows 95
Windows	98	Windows 98
Windows	NT	Windows NT

Table M-2: AppOS and OSVersion Examples (Section 2 of 2)

AppOS	OSVersion	Description
Windows	NT-5	Windows 2000
Windows	NT-5.1	XP [not yet registered by Microsoft with IANA]

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.

Note: pay special attention to JDF tags that are **orange** and/or JDF attribute names that are **magenta**.

N.1 Brief Example

N.1.1 Before and After Processing

Example N-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"
  JobPartID="ID34" Status="Waiting" Type="ColorSpaceConversion"
  Version="1.4">
  <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"/>
  </ResourceLinkPool>
</JDF>
```

```

    <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>

```

Example N-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"
  JobPartID="ID234" Version="1.4">
  <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:///SMPProcessCMYK.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"/>
    <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" />
<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

Example N-3: 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" JobPartID="ID455" Type="Product" Version="1.4">
  <ResourcePool>
    <Component Amount="100" Class="Quantity" DescriptiveName="Teacher's Book"
      ID="Link0003" Status="Unavailable" ComponentType="Sheet" />
    <Component Amount="2000" Class="Quantity" DescriptiveName="Cover" ID="Link0005"
      Status="Unavailable" ComponentType="Sheet">
      <!--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" ComponentType="Sheet">
      <!--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>
</ResourceLinkPool>

```

```

    <ComponentLink Amount="1000" Usage="Output" rRef="Link0011"/>
  </ResourceLinkPool>
  <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" ComponentType="Sheet"/>
    </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" ComponentType="Sheet"/>
    </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.

Example N-4: 2-Component JDF before Spawning

The following JDF file describes a two-component brochure. The Resources are not fleshed out.

Note: pay special attention to JDF tags that are orange and/or JDF attribute names that are magenta.

```

<JDF ID="SpawnTest" Type="Product" xmlns="http://www.CIP4.org/JDFSchema_1_1"
  Status="Waiting" Version="1.4" 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" ComponentType="Sheet" />
    <BindingIntent ID="r0044" Class="Intent" Status="Available">
      <BindingType Range="SaddleStitch" DataType="EnumerationSpan" />
    </BindingIntent>
    <ProductionIntent ID="r0045" Class="Intent" Status="Available">
      <PrintProcess Range="Gravure" DataType="EnumerationSpan" />
    </ProductionIntent>
    <Component ID="r0047" Class="Quantity" Status="Unavailable"
      ComponentType="Sheet" />
    <Component ID="r0051" Class="Quantity" Status="Unavailable"
      ComponentType="Sheet" />
  </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>

```

Example N-5: 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="1.4" 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" ComponentType="Sheet" />
  <BindingIntent ID="r0044" Class="Intent" Status="Available">
    <BindingType Range="SaddleStitch" DataType="EnumerationSpan" />
  </BindingIntent>
  <ProductionIntent ID="r0045" Class="Intent" Status="Available">
    <PrintProcess Range="Gravure" DataType="EnumerationSpan" />
  </ProductionIntent>
  <Component ID="r0047" Class="Quantity" Status="Unavailable"
    ComponentType="Sheet" SpawnIDs="Sp0057" SpawnStatus="SpawnedRW" />
  <Component ID="r0051" Class="Quantity" Status="Unavailable"
    ComponentType="Sheet" />
</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>
  <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>

```


Example N-6: 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="1.4" 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"
      ComponentType="Sheet" SpawnIDs="Sp0057" />
  </ResourcePool>
  <AncestorPool>
    <Ancestor NodeID="SpawnTest" FileName="testjdf4.jdf" />
  </AncestorPool>
</JDF>
```

Example N-7: 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="1.4" 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" />
    <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"
      ComponentType="Sheet" Status="Unavailable" />
    <BindingIntent ID="r0044" Class="Intent" Status="Available">
      <BindingType Actual="SoftCover" DataType="EnumerationSpan" />
    </BindingIntent>
    <ProductionIntent ID="r0045" Class="Intent" Status="Available">
      <PrintProcess Range="Gravure" DataType="EnumerationSpan" />
    </ProductionIntent>
    <Component ID="r0047" Class="Quantity" ComponentType="Sheet"
      Status="Available" />
    <Component ID="r0051" Class="Quantity" ComponentType="Sheet"
      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="1.4" 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>

```

Example N-8: 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" JobPartID="ID777" Version="1.4">
  <ResourcePool>
    <ImageSetterParams ID="r0052" Class="Parameter" Locked="false" Status="Available" />
    <Media ID="r0053" Class="Consumable" Locked="false" Status="Available" />
    <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>

```

Example N-9: 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" JobPartID="ID467" Version="1.4">
  <AuditPool/>
  <ResourcePool>
    <ImageSetterParams ID="r0052" Class="Parameter" Locked="true"
      Status="Available" />
    <Media ID="r0053" Class="Consumable" Locked="true" Status="Available"
      PartIDKeys="Separation">
      <Media Separation="Cyan" />
    </Media>
    <ExposedMedia ID="r0054" Class="Handling" Locked="true"
      Status="Unavailable"
      PartIDKeys="Separation">
      <ExposedMedia Separation="Cyan">
        <MediaRef rRef="r0053">
          <Part Separation="Cyan" />
        </MediaRef>
      </ExposedMedia>
    </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="1.4" FileName="testjdf5.jdf" />
  </AncestorPool>
</JDF>

```

Example N-10: Root Partitioned ImageSetting Node after Spawning

```

<JDF ID="n20020701190951" Type="ImageSetting"
  xmlns="http://www.CIP4.org/JDFSchema_1_1"
  Status="Pool" JobPartID="ID778" Version="1.4">
  <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 Usage="Input" rRef="r0052" />
    <MediaLink Usage="Input" rRef="r0053" />
    <ExposedMediaLink Usage="Output" rRef="r0054" />
    <RunListLink Usage="Input" rRef="r0055" />
    <NodeInfoLink Usage="Input" rRef="r050722154232_0045" />
  </ResourceLinkPool>
</JDF>

```

Example N-11: 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" JobPartID="ID234"
  Version="1.4">
  <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="r050722154232_0045" NodeStatus="Waiting"
    PartIDKeys="Separation" Status="Available">
    <NodeInfo NodeStatus="Completed" Separation="Cyan"/>
  </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

Example N-12: 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.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <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>
            <FileSpec URL="File:///Cyan.pdf"/>
          </LayoutElement>
        </RunList>
        <RunList Separation="Magenta">
          <LayoutElement >
            <FileSpec URL="File:///Magenta.pdf"/>
          </LayoutElement>
        </RunList>
        <RunList Separation="Yellow">
          <LayoutElement>
            <FileSpec URL="File:///Yellow.pdf"/>
          </LayoutElement>
        </RunList>
        <RunList Separation="Black">
          <LayoutElement>
            <FileSpec URL="File:///Black.pdf"/>
          </LayoutElement>
        </RunList>
        <RunList Separation="SpotGreen">
          <LayoutElement>
            <FileSpec URL="File:///Green.pdf"/>
          </LayoutElement>
        </RunList>
      </RunList>
    </ResourcePool>
  </JDF>

```

```

    </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 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 CCMYYKKG
    </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:///PreSepCCMYYKKG.pdf"/>
</LayoutElement>
</ResourcePool>
<ResourceLinkPool>
    <RunListLink Usage="Input" rRef="Link0003"/>
</ResourceLinkPool>
</JDF>

```

N.5 Messages

N.5.1 Simple KnownMessages

The following simple example shows a KnownMessages Query Message and the Response Message sent by a fairly dumb Controller:

Example N-13: KnownMessages Query

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFCClient"
    TimeStamp="2000-11-07T13:15:56+01:00" MaxVersion="1.4" Version="1.4"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
    <Query ID="Q0001" Type="KnownMessages" xsi:type="QueryKnownMessages">
        <KnownMsgQuParams ListCommands="true" ListQueries="true"
            ListSignals="false"/>
    </Query>
</JMF>

```

Example N-14: KnownMessages Response

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFCClient #2"
    TimeStamp="2000-11-07T13:15:56+01:00" MaxVersion="1.4" Version="1.4"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >

```

```

<Response ID="R0001" Type="KnownMessages" xsi:type="ResponseKnownMessages"
  refID="Q0001">
  <MessageService Query="true" Type="KnownMessages"/>
  <MessageService Persistent="true" Query="true" Type="Status"/>
  <MessageService Command="true" Type="StopPersistentChannel"/>
</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. Then the following four examples are a set of typical, simple responses that are emitted whenever *DeviceStatus* changes; three responses are Signal Messages

Example N-15: Status Query

```

<JMF xmlns="http://www.CIP4.org/JDFSchemas_1_1" SenderID="JMFCClient"
  Timestamp="2000-11-07T16:02:09+01:00" MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <Query ID="Q0011" Type="Status" xsi:type="QueryStatus">
    <Subscription URL="http://123.123.123.123/message/recipient">
      <ObservationTarget ObservationPath="/**/*" />
    </Subscription>
    <StatusQuParams JobDetails="Brief" />
  </Query>
</JMF>

```

Example N-16: Status Response

This is the Response Message that is sent immediately within the same HTTP connection as the Query Message.

```

<JMF xmlns="http://www.CIP4.org/JDFSchemas_1_1" SenderID="JMFCClient #2"
  Timestamp="2000-11-07T16:02:19+01:00"
  MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <Response ID="R0013" Type="Status" xsi:type="ResponseStatus" refID="Q0011">
    <DeviceInfo DeviceStatus="Idle" />
  </Response>
</JMF>

```

Example N-17: Status Signal #1

This is an intermediate Signal that was emitted when *DeviceStatus* changed.

```

<JMF xmlns="http://www.CIP4.org/JDFSchemas_1_1" SenderID="JMFCClient #2"
  Timestamp="2000-11-07T17:02:19+01:00" MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <Signal ID="Q0015" Type="Status" xsi:type="SignalStatus" refID="Q0011">
    <DeviceInfo DeviceStatus="Setup" />
  </Signal>
</JMF>

```

Example N-18: Status Signal #2

This is an intermediate Signal that was emitted when *DeviceStatus* changed.

```

<JMF xmlns="http://www.CIP4.org/JDFSchemas_1_1" SenderID="JMFCClient #2"
  Timestamp="2000-11-07T17:08:19+01:00" MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <Signal ID="Q0017" Type="Status" xsi:type="SignalStatus" refID="Q0011">
    <DeviceInfo DeviceStatus="Running" />
  </Signal>
</JMF>

```


Example N-19: Status Signal #3

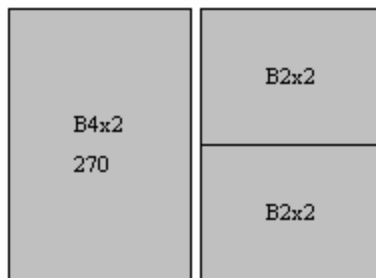
This is the last Signal of the persistent channel.

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFClient #2"
  TimeStamp="2000-11-07T19:02:19+01:00" MaxVersion="1.4" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <Signal ID="Q0017" Type="Status" xsi:type="SignalStatus" refID="Q0011"
    LastRepeat="true">
    <DeviceInfo DeviceStatus="Idle"/>
  </Signal>
</JMF>
```

N.6 Stripping

Example N-20: Using Position

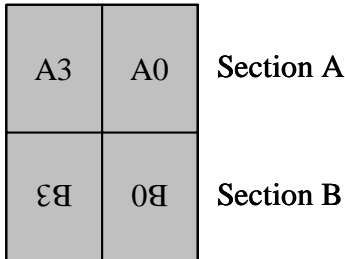
The following example illustrates the more advanced use of the *Position* object. Note that the two B2x2 Signatures are filled independently.



```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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>
  </ResourcePool>
  <ResourceLinkPool>
    <StrippingParamsLink Usage="Input" rRef="L1"/>
  </ResourceLinkPool>
</JDF>
```

Example N-21: 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.



```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool >
    <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">
      <BinderySignatureRef rRef="B4x2"/>
      <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"/>
          <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>
  </ResourcePool >
  <ResourceLinkPool>
    <StrippingParamsLink Usage="Input" rRef="L1"/>
  </ResourceLinkPool>
</JDF>
```

Example N-22: 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

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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>
  </ResourcePool>
  <ResourceLinkPool>
    <StrippingParamsLink Usage="Input" rRef="L1" />
  </ResourceLinkPool>
</JDF>
```

Example N-23: 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.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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" 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">

```

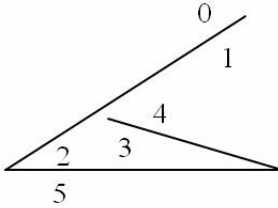
```

        <StrippingParams AssemblyIDs="Book1" SectionIndex="0" SectionList="2"/>
        <StrippingParams AssemblyIDs="Book2" SectionIndex="1" SectionList="7"/>
    </StrippingParams>
    <StrippingParams 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>
</ResourcePool>
<ResourceLinkPool>
    <StrippingParamsLink Usage="Input" rRef="L1"/>
</ResourceLinkPool>
</JDF>

```

Example N-24: 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

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
    Type="ProcessGroup" JobPartID="ID300" Version="1.4">
    <ResourcePool>
        <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 Class="Parameter" ID="Cover" Status="Available"
            WorkStyle="WorkAndBack">
            <BinderySignatureRef rRef="foldout"/>
            <Position RelativeBox="0 0 1 0.5"/>
            <Position RelativeBox="0 0.5 1 1"/>
        </StrippingParams>
    </ResourcePool>
    <ResourceLinkPool>
        <StrippingParamsLink Usage="Input" rRef="Cover"/>
    </ResourceLinkPool>
</JDF>

```

Example N-25: 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	37
31	0	7	24

Web1

81	11	01	17
29	2	5	26

Web2

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <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>
  </ResourcePool>
</JDF>
```

```

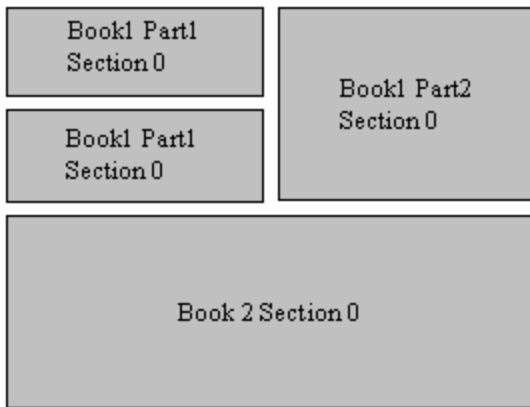
<StrippingParams SignatureName="Signature2">
  <StrippingParams SheetName="Sheet3">
    <BinderySignatureRef rRef="B001">
      <Part WebName="Web1"/>
    </BinderySignatureRef>
  </StrippingParams>
  <StrippingParams SheetName="Sheet4">
    <BinderySignatureRef rRef="B001">
      <Part WebName="Web2"/>
    </BinderySignatureRef>
  </StrippingParams>
</StrippingParams>
</StrippingParams>
</ResourcePool>
<ResourceLinkPool>
  <StrippingParamsLink Usage="Input" rRef="MultiWeb1"/>
</ResourceLinkPool>
</JDF>

```

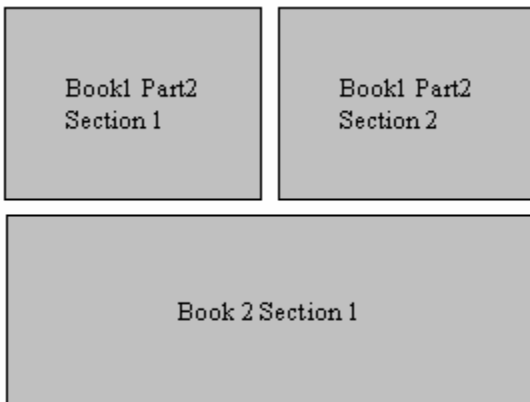
Example N-26: Stripping Process

The next sample illustrates the *Stripping* Process and its *StrippingParams* and *Assembly Resources*.

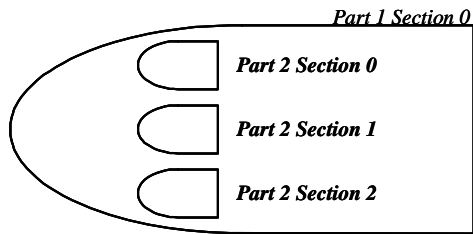
Sheet1:



Sheet2:



Assembly 1



```

<JDF ID="n001" Type="Stripping" JobPartID="ID378" Status="Ready" Version="1.4">
  <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>
    <Assembly Class="Parameter" ID="A1" JobID="Book1" Order="List"
      Status="Available">
      <AssemblySection AssemblyIDs="Part1" Order="Gathering">

```

```

        <AssemblySection AssemblyIDs="Part2" />
        <AssemblySection AssemblyIDs="Part2" />
        <AssemblySection AssemblyIDs="Part2" />
    </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 N-27: DigitalDelivery: Before the Delivery

Instruct the digital delivery Device to compress the files delivered in gzip compression. The part that changes has an orange tag and magenta attributes.

```

<JDF ID="SpawnTest" Type="DigitalDelivery"
    xmlns="http://www.CIP4.org/JDFSchema_1_1"
    Status="Waiting" JobPartID="Part1" Version="1.4" >
  <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>
    <DigitalDeliveryParams ID="DDLLink" Class="Parameter" Status="Available" />
  </ResourcePool>
  <!-- ... -->
  <ResourceLinkPool>
    <DigitalDeliveryParamsLink rRef="DDLLink" Usage="Input" />
    <RunListLink rRef="SourceFilesLink" Usage="Input" />
    <RunListLink rRef="TargetFilesLink" Usage="Output" />
  </ResourceLinkPool>
</JDF>

```

Example N-28: DigitalDelivery: After the Delivery

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.

```

<JDF ID="SpawnTest" Type="DigitalDelivery"
    xmlns="http://www.CIP4.org/JDFSchema_1_1"
    Status="Waiting" JobPartID="Part1" Version="1.4" >
  <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"
      URL="File:///FileServer1/ComingJobs/job702555.gz"/>
    </LayoutElement>
  </RunList>
<DigitalDeliveryParams ID="DDLLink" Class="Parameter" Status="Available"/>
</ResourcePool>
<!-- ... -->
<ResourceLinkPool>
  <DigitalDeliveryParamsLink rRef="DDLLink" Usage="Input"/>
  <RunListLink rRef="SourceFilesLink" Usage="Input"/>
  <RunListLink rRef="TargetFilesLink" Usage="Output"/>
</ResourceLinkPool>
</JDF>

```

Example N-29: Delivery and DigitalDelivery Processes

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/JDFSchema_1_1"
  DescriptiveName="ArtDeliveryIntent translated to Delivery and
  DigitalDelivery processes" ID="ID000"
  Status="InProgress" Type="Product" JobPartID="ID879" Version="1.4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ResourcePool>
    <NodeInfo ID="N01" Class="Parameter" Status="Available"
      JobPriority="100"/>
    <CustomerInfo ID="Cus01" Class="Parameter" Status="Available"
      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="tel:+44-019-1234-4567"/>
        <ComChannel ChannelType="Fax"
          ChannelUsage="Business DayTime NightTime"
          Locator="tel:+44-019-1234-4567"/>
      </Contact>
    </CustomerInfo>
    <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="tel:+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"/>
  <NodeInfoLink Usage="Input" rRef="N01"/>
  <CustomerInfoLink Usage="Input" rRef="Cus01"/>
</ResourceLinkPool>
<JDF ID="J171373" Status="Completed" Type="DigitalDelivery"
  JobPartID="ID877" >
  <ResourcePool>
    <CustomerInfo ID="Cus02" Class="Parameter" Status="Available"
      CustomerJobName="Job title ...">
      <CustomerMessage Language="FR" MessageEvents="Delivered"
        ShowList="StartTime EndTime Error">
        <ComChannel ChannelType="Email"
          Locator="mailto:sender@Email.com"/>
        <ComChannel ChannelType="InstantMessaging"
          ChannelTypeDetails="Yahoo"
          Locator="bill_jones"/>
      </CustomerMessage>
    </CustomerInfo>
    <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="mailto: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>

```

```

        </Container>
        </FileSpec>
    </LayoutElement>
</RunList>
</ResourcePool>
<ResourceLinkPool>
    <DigitalDeliveryParamsLink Usage="Input" rRef="DestinationLink"/>
    <RunListLink Usage="Input" rRef="FileListLink1"/>
    <CustomerInfoLink Usage="Input" rRef="Cus02"/>
    <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" JobPartID="ID878" >
    <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>

```

Example N-30: 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" JobPartID="ID200"
  Version="1.4" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ResourcePool>
    <NodeInfo ID="Node001" Class="Parameter" Status="Available" JobPriority="60"/>
    <Contact Class="Parameter" ContactTypes="Delivery" ID="DestLink"
      PartIDKeys="Location" Status="Available">
      <Contact Location="Dest1">
        <ComChannel ChannelType="Email"
          Locator="mailto:Reciever1@hotmail.com"/>
      </Contact>
      <Contact Location="Dest2">
        <ComChannel ChannelType="PrivateDirectory"
          ChannelTypeDetails="VioAddress"
          Locator="Best Workgroup@Best Company"/>
      </Contact>
    </Contact>
    <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>
  </ResourcePool>
  <JDF DescriptiveName="Upload Job to Server" ID="ID001" JobID="J702555"
    Status="Completed" Type="DigitalDelivery" JobPartID="ID201">
  <ResourcePool>
    <CustomerInfo ID="Cus001" Class="Parameter" Status="Available"
      CustomerJobName="World atlas maps #2">
      <CustomerMessage Language="FR" MessageEvents="Delivered Accepted">
        <ComChannel ChannelType="Email"
          Locator="mailto:sender@email.com"/>
      </CustomerMessage>
    </CustomerInfo>
    <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="mailto: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" />
  <CustomerInfoLink Usage="Input" rRef="Cus001" />
  <NodeInfoLink Usage="Input" rRef="Node001" />
  <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"
  JobPartID="ID202">
  <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>
  <RunList Run="2">

```

```

        <LayoutElement>
            <FileSpec FileSize="33947" URL="./Atlas/America.jpg"/>
        </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"/>
    <NodeInfoLink Usage="Input" rRef="Node001"/>
    <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>

```

N.8 Automated Imposition

Example N-31: Algorithm for Processing an Imposition Template

The pseudocode below describes how a Page Pool or Page Pool List might be processed through an Imposition Template. Note that the algorithm described is a fairly lazy algorithm that relies on detecting that no content can be placed on a sheet to detect end of content. In addition, cut and stack and *BaseOrdReset = "PagePoolList"* are not supported in this example:

Note: numPagesInPagePool and lastPagePoolPositiveIndex will be affected by execution of *Layout/*PageCondition Elements.

```

positiveBaseOrd = negativeBaseOrd = 0 // base index into the PagePool
EvaluateTemplate and determine:
    doingNegativeOrds // If negative ords present in this template, then true
    doingPositiveOrds // If non-negative ords present in this template, then true
    maximumPositiveOrd // Value of the largest non-negative ord in the template

```

```

maximumNegativeOrd // Value of the greatest magnitude negative ord in the template
numPagesInPagePool // includes evaluation for PagePoolList
lastPagePoolPositiveIndex // index of last page in the page pool to be placed using
    non-negative ords (the "midpoint" of the PagePool)
do
numSheetsProcessed = numSheetsExhausted = 0
for each sheet in the Imposition Template
    numSheetsProcessed++
    numSidesExhausted = numSidesInSheet = 0
    for each Side in the Sheet
        numSidesInSheet++
        if ProcessSide() == sideContentExhausted // see below for pseudocode
            numSidesExhausted++
        if numSidesInSheet == numSidesExhausted
            numSheetsExhausted++
    if doingPositiveOrds
        positiveBaseOrd += maximumPositiveOrd + 1
    if doingNegativeOrds
        negativeBaseOrd += maximumNegativeOrd
until numSheetsProcessed == numSheetsExhausted

```

Processing a Side's ContentObjects:

```

numOrdsProcessed = numOrdsExhausted = 0
for each ContentObject in the Side
    numOrdsProcessed++
    if ContentObject/@Ord >= 0
        theOrd = positiveBaseOrd + ContentObject/@Ord
    else
        theOrd = numPagesInPagePool + negativeBaseOrd + ContentObject/@Ord;
        // remember negativeBaseOrd starts at 0
    if ((ContentObject/@Ord >= 0) && (theOrd > lastPagePoolPositiveIndex)) ||
        ((ContentObject/@Ord < 0) && (theOrd <= lastPagePoolPositiveIndex))
        numOrdsExhausted++
        if InsertSheet specifies FillSignature
            place any alternate content or leave blank
        else if the ImpositionTemplate specifies a PageCondition that applies
            place any alternate content or leave blank
        else
            place PagePool[theOrd]
if (numOrdsProcessed == numOrdsExhausted)
    return sideContentExhausted
else
    return sideContentNotExhausted

```

Example N-32: Format of Variable Data Structured Content

The BNF and example below describe a generic variable data structured content format using XML to represent the structure elements. This format is able to represent the most common attributes of existing variable data languages, and will be used to describe the input data sets for this section's examples:

The following is BNF for eVDPML (example **V**ariable **D**ocument **P**rint **M**arkup **L**anguage) XML:

Page ::= [Metadata*] - represents the graphical content of a single page which may contain metadata.

(**Note:** Metadata was added to the Page Element in the meeting discussion)

Metadata - represents arbitrary key/value metadata information

DocPart ::= [Metadata*] [DocPart+ || Page+]

(**Note:** cannot have both DocPart and Page elements in the same DocPart element.)

Record ::= [Metadata*] [DocPart+ || Page+]

(**Note:** Both DocPart and Page elements **MUST** not be specified in the same Record.)

RecordGroup ::= [Metadata] [RecordGroup+ || Record+]

(Note: all Record elements MUST be specified at the same RecordGroup node hierarchical level)

Below is an example structure using eVDPML syntax (which is NOT JDF syntax):

```
<RecordGroup>
  <RecordGroup >
    <Metadata Key="MailRate" Value="Bulk" />
    <Record>
      <Metadata Key="RecID" Value="0" />
      <Metadata Key="LastName" Value="Robinson" />
      <DocPart>
        <DocPart>
          <Page/> <Page/> <Page/> <Page/>
        </DocPart>
        <DocPart>
          <Page/> <Page/> <Page/>
        </DocPart>
      </DocPart>
    </Record>
    <Record>
      <Metadata Key="RecID" Value="1" />
      <Metadata Key="LastName" Value="Smith" />
      <Page/> <Page/> <Page/> <Page/>
    </Record>
    <!-- ... -->
  </RecordGroup>
  <RecordGroup>
    <Metadata Key="MailRate" Value="FirstClass" />
    <Record>
      <Metadata Key="RecID" Value="389" />
      <Metadata Key="LastName" Value="Prosi" />
      <Page/> <Page/> <Page/> <Page/> <Page/>
    </Record>
    <Record>
      <Metadata Key="RecID" Value="390" />
      <Metadata Key="LastName" Value="Doe" />
      <Page/> <Page/> <Page/>
    </Record>
    <!-- ... -->
  </RecordGroup>
</RecordGroup>
```

Example N-33: Page Pools

All recipients receive a one page cover letter printed on substrate A and a personalized document containing two or more customized two sided pages printed on substrate B and all pages to be corner stapled together.

Demonstrates use of Partitioning for mapping Page Pools of two different documents to independent sets of sheets relying on the structure of the PDL.

The eVDPML is:

```
<RecordGroup>
  <Record>
    <Metadata Key="RecID" Value="0" />
    <DocPart>
      <Metadata Key="Part" Value="CoverLetter" />
      <Page/>
    </DocPart>
    <DocPart>
      <Metadata Key="Part" Value="Brochure" />
      <Page/>
      <Page/>
      <Page/>
      <Page/>
    </DocPart>
  </Record>
</RecordGroup>
```



```

</Record>
<Record>
  <Metadata Key="RecID" Value="1" />
  <DocPart>
    <Metadata Key="Part" Value="CoverLetter" />
    <Page />
  </DocPart>
  <DocPart>
    <Metadata Key="Part" Value="Brochure" />
    <Page />
    <Page />
    <Page />
    <Page />
  </DocPart>
</Record>
<!-- ... -->
<Record>
  <Metadata Key="RecID" Value="n" />
  <DocPart>
    <Metadata Key="Part" Value="CoverLetter" />
    <Page />
  </DocPart>
  <DocPart>
    <Metadata Key="Part" Value="Brochure" />
    <Page />
    <Page />
  </DocPart>
</Record>
</RecordGroup>

```

The JDF is:

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <RunList Class="Parameter" ID="Ex3_RunList" Status="Available">
      <LayoutElement>
        <FileSpec URL="MyVDPRecords.vdpml"
          MimeType="application/x-evdpml+xml" />
      </LayoutElement>
      <MetadataMap DataType="PartIDKeys" Name="RunTags"
        ValueFormat="%s" ValueTemplate="doctype" Context="Document">
        <Expr Name="doctype" Value="CoverLetter">
          <and>
            <NameEvaluation
              Path="Metadata/@Key" RegExp="Part" />
            <NameEvaluation
              Path="Metadata/@Value" RegExp="CoverLetter" />
          </and>
        </Expr>
        <Expr Name="doctype" Value="Brochure">
          <and>
            <NameEvaluation
              Path="Metadata/@Key" RegExp="Part" />
            <NameEvaluation
              Path="Metadata/@Value" RegExp="Brochure" />
          </and>
        </Expr>
      </MetadataMap>
    </RunList>
    <Media Class="Consumable" ID="Medial" Status="Available" />
  <!--
    NOTE: MetadataMap(s) are applied to each document
    node input to the Imposition process independent of each other.
  -->

```

```

-->
<Layout Class="Parameter" ID="Ex2_Layout" Status="Available"
  PartIDKeys="RunTags SheetName Side" Automated="true"
  LockOrigins="true" BaseOrdReset="PagePool">
  <MediaRef rRef="Medial"/>
  <!-- Used by Imposition process to obtain media dimensions -->
  <Layout RunTags="CoverLetter">
    <Layout SheetName="LetterSheet">
      <Layout Side="Front">
        <ContentObject CTM="1 0 0 1 0 0" Ord="0"/>
        <!-- First page of CoverLetter DocPart -->
      </Layout>
    </Layout>
  </Layout>
  <Layout RunTags="Brochure">
    <Layout SheetName="BrochureSheets">
      <Layout Side="Front">
        <ContentObject CTM="1 0 0 1 0 0" Ord="0"/>
        <!--Front side of Brochure sheet -->
      </Layout>
      <Layout Side="Back">
        <ContentObject CTM="1 0 0 1 0 0" Ord="1"/>
        <!--Back side of Brochure sheet -->
      </Layout>
    </Layout>
  </Layout>
</ResourcePool>
<ResourceLinkPool>
  <RunListLink Usage="Input" rRef="Ex3_RunList"/>
  <LayoutLink Usage="Input" rRef="Ex2_Layout"/>
</ResourceLinkPool>
</JDF>

```

Example N-34: Booklet Using Automated Imposition

All recipients receive a single customized saddle stitched booklet where the two-page cover (front and back outside only) and the four or more body pages are also printed on substrate A using cut and stack production. The finished sheet size of each component of a booklet is 8.5" x 11" and the finished booklet is 5.5" x 8.5". The production of the component will be performed cut and stack on 17" x 11". Two saddle stitch imposed page pairs will be generated per sheet definition.

The eVDPML is:

```

<RecordGroup>
  <Record>
    <Metadata Key="RecID" Value="0"/>
    <DocPart>
      <Metadata Key="Part" Value="Cover"/>
      <Page/> <Page/>
    </DocPart>
    <DocPart>
      <Metadata Key="Part" Value="Body"/>
      <Page/> <Page/><Page/><Page/> ... <Page/>
    </DocPart>
  </Record>
  <Record>
    <Metadata Key="RecID" Value="1"/>
    <DocPart>
      <Metadata Key="Part" Value="Cover"/>
      <Page/> <Page/>
    </DocPart>
    <DocPart>
      <Metadata Key="Part" Value="Body"/>
      <Page/> <Page/><Page/><Page/> ... <Page/>
    </DocPart>
  </Record>
</RecordGroup>

```

```

    </DocPart>
  </Record>
  <!-- ... -->
  <Record>
    <Metadata Key="RecID" Value="n" />
    <DocPart>
      <Metadata Key="Part" Value="Cover" />
      <Page/> <Page/>
    </DocPart>
    <DocPart>
      <Metadata Key="Part" Value="Body" />
      <Page/> <Page/><Page/><Page/> ... <Page/>
    </DocPart>
  </Record>
</RecordGroup>

```

The JDF is:

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="A1" Status="Waiting"
  Type="ProcessGroup" JobPartID="ID300" Version="1.4">
  <ResourcePool>
    <RunList Class="Parameter" ID="Ex11_RunList" Status="Available">
      <LayoutElement>
        <FileSpec URL="MyVDPRecords.vdpml"
          MimeType="application/x-evdpml+xml" />
      </LayoutElement>
      <MetadataMap DataType="PartIDKeys" Name="RunTags" ValueFormat="%s"
        ValueTemplate="ProductPart">
        <Expr Name="ProductPart" Value="BookletCover">
          <and>
            <NameEvaluation Path="Metadata/@Key" RegExp="Part"/>
            <NameEvaluation Path="Metadata/@Value" RegExp="Cover"/>
          </and>
        </Expr>
        <Expr Name="ProductPart" Value="BookletBody">
          <and>
            <NameEvaluation Path="Metadata/@Key" RegExp="Part"/>
            <NameEvaluation Path="Metadata/@Value" RegExp="Body"/>
          </and>
        </Expr>
      </MetadataMap>
    </RunList>
    <Media Class="Consumable" ID="SubstrateA" Status="Available"/>
    <Layout Class="Parameter" ID="Ex11_Layout" Status="Available"
      PartIDKeys="RunTags SheetName Side" Automated="true"
      LockOrigins="false" BaseOrdReset="PagePool">
      <LogicalStackParams MaxStackDepth="500" Restrictions="None">
        <Stack LogicalStackOrd="0" LogicalSheetSequence="SheetIndex"/>
        <Stack LogicalStackOrd="-1" LogicalSheetSequence="SheetIndex"/>
      </LogicalStackParams>
      <Layout RunTags="BookletCover">
        <MediaRef rRef="SubstrateA"/>
        <!-- Used by Imposition process to obtain media dimensions -->
        <Layout SheetName="BookletCoverSheet">
          <Layout Side="Front">
            <ContentObject CTM="1 0 0 1 378 630" Ord="0"
              LogicalStackOrd="0"/> <!-- front outside cover -->
            <ContentObject CTM="1 0 0 1 18 630" Ord="-1"
              LogicalStackOrd="0"/> <!-- back outside cover -->
            <ContentObject CTM="1 0 0 1 378 18" Ord="0"
              LogicalStackOrd="1"/> <!-- front outside cover -->
            <ContentObject CTM="1 0 0 1 18 18" Ord="-1"
              LogicalStackOrd="1"/> <!-- back outside cover -->
            <MarkObject CTM="1 0 0 1 0 0" LogicalStackOrd="0">
              <MarkActivation Context="CollectSheetIndex"

```

```

        Index="-1"/>
        <DeviceMark Anchor="BottomLeft" Font="Code128A"
        FontSize="48"/>
        <!-- provides formatting for the dynamic barcode -->
        <JobField JobFormat="%d" JobTemplate="TotalSheetsInCollect"/>
    </MarkObject>
    <MarkObject CTM="1 0 0 1 0 0" LogicalStackOrd = "1">
        <MarkActivation Context="CollectSheetIndex" Index="-1"/>
        <DeviceMark Anchor="BottomLeft" Font="Code128A"
        FontSize="48"/>
        <!-- provides formatting for the dynamic barcode -->
        <JobField JobFormat="%d" JobTemplate="TotalSheetsInCollect"/>
    </MarkObject>
</Layout>
</Layout>
</Layout>
<Layout RunTags="BookletBody">
    <MediaRef rRef="SubstrateA"/>
    <!-- Used by Imposition process to obtain media dimensions -->
    <Layout SheetName="BookletBodySheets">
        <Layout Side="Front">
            <ContentObject CTM="1 0 0 1 378 630" Ord="0"
            LogicalStackOrd = "0"/> <!-- Front right -->
            <ContentObject CTM="1 0 0 1 18 630" Ord="-1"
            LogicalStackOrd = "0"/> <!-- Front left -->
            <ContentObject CTM="1 0 0 1 378 18" Ord="0"
            LogicalStackOrd = "1"/> <!-- Front right -->
            <ContentObject CTM="1 0 0 1 18 18" Ord="-1"
            LogicalStackOrd = "1"/> <!-- Front left -->
        </Layout>
        <Layout Side="Back">
            <ContentObject CTM="1 0 0 1 18 630" Ord="1"
            LogicalStackOrd = "0"/> <!-- Back left -->
            <ContentObject CTM="1 0 0 1 378 630" Ord="-2"
            LogicalStackOrd = "0"/> <!-- Back right -->
            <ContentObject CTM="1 0 0 1 18 18" Ord="1"
            LogicalStackOrd = "1"/> <!-- Back left -->
            <ContentObject CTM="1 0 0 1 378 18" Ord="-2"
            LogicalStackOrd = "1"/> <!-- Back right -->
        </Layout>
    </Layout>
</Layout>
</Layout>
</ResourcePool>
<ResourceLinkPool>
    <LayoutLink Usage="Input" rRef="Ex11_Layout"/>
    <RunListLink Usage="Input" rRef="Ex11_RunList"/>
</ResourceLinkPool>
</JDF>

```

Appendix O New, Deprecated & Modified Items

This appendix contains the list of items that have been modified in JDF 1.4. See previous versions of the JDF specification for a complete history of changes.

O.1 Compatibility Warnings

Table O-1: Compatibility Warnings

XPath	Table and Page	Description
QueueEntry/ <i>@QueueEntryID</i>	Table 5-102 on page 251	ReturnQueueEntryParams/ <i>@QueueEntryID</i> was erroneously omitted in JDF 1.2. and JDF 1.3.

O.2 Changed Items

The “C” column contains one of three letters to specify the type of change

- N: [New in JDF 1.4](#)
- M: [Modified in JDF 1.4](#)
- D: [Deprecated in JDF 1.4](#)

Table O-2: Changed Items (Section 1 of 17)

C	XPath	Table and Page	Description
N	Location of Callouts	Section 1.3.3.1 on page 4	
N	hexBinaryList	Table 1-6 on page 13	new Data Type
N	languages	Table 1-6 on page 13	new Data Type
N	WorkStyle	Table 1-6 on page 13	new Data Type
M	Countable Objects	Table 1-6 on page 13	modified Data Type
N	Any Element/GeneralID	Table 3-1 on page 40	moved GeneralID from Abstract Resource to Any Element
N	Any Element/Preview	Table 3-1 on page 40	moved Preview from Process Template for Input Resources to Any Element
M	Comment/ <i>@Name</i>	Table 3-2 on page 42	
N	Comment/ <i>@Name</i> = "DeviceText"	Table 3-2 on page 42	
M	GeneralID	Section 3.1.2 on page 43	Changed from Resource to Element
N	GeneralID/ <i>@DataType</i>	Table 3-3 on page 44	
M	GeneralID/ <i>@IDUsage</i>	Table 3-3 on page 44	
N	GeneralID/ <i>@IDUsage</i> = "DeviceProductID"	Table 3-3 on page 44	
M	JDF/ <i>@Category</i>	Table 3-5 on page 47	
N	JDF/ <i>@Category</i> = "Newsprinting"	Table 3-5 on page 47	
M	JDF/ <i>@Status</i> = "Suspended"	Table 3-6 on page 52	
D	Abstract Resource/ <i>@CatalogID</i>	Table 3-10 on page 63	
D	Abstract Resource/ <i>@CatalogDetails</i>	Table 3-10 on page 63	

Table O-2: Changed Items (Section 2 of 17)

C	XPath	Table and Page	Description
M	Abstract ResourceLink/@ProcessUsage	Table 3-16 on page 77	
N	Abstract ResourceLink/@ProcessUsage = ICS values	Table 3-16 on page 77	Add values from ICSs
M	Abstract ResourceLink/@Duration	Table 3-16 on page 77	Moved to Abstract ResourceLink from Table 3-20, "Abstract ImplementationLink or // AmountPool/PartAmount Element"
M	Abstract ResourceLink/@Start	Table 3-16 on page 77	
M	Abstract ResourceLink/@StartOffset	Table 3-16 on page 77	
M	PartAmount/@Duration	Table 3-19 on page 81	Moved to PartAmount from Table 3-20, "Abstract ImplementationLink or // AmountPool/PartAmount Element"
M	PartAmount/@Start	Table 3-19 on page 81	
M	PartAmount/@StartOffset	Table 3-19 on page 81	
M	Abstract ImplementationLink or //AmountPool/PartAmount	Table 3-20 on page 82	All non-deprecated attributes moved to PartAmount and Abstract ResourceLink.
N	Partitionable Resource/@PartIDKeys = "Metadata0"	Table 3-27 on page 103	Ten values added "Metadata0" through "Metadata9"
M	Part/@DocTags	Table 3-28 on page 104	Data Type was expanded from NMTOKENS to NameRangeList.
N	Part/@Metadata0	Table 3-28 on page 104	Ten Attributes added @Metadata0 through @Metadata9
M	Part/@PageTags	Table 3-28 on page 104	Data Type was expanded from NMTOKENS to NameRangeList.
N	Part/@RunPageRange	Table 3-28 on page 104	
M	Part/@RunTags	Table 3-28 on page 104	Data Type was expanded from NMTOKENS to NameRangeList.
M	Part/@SetTags	Table 3-28 on page 104	Data Type was expanded from NMTOKENS to NameRangeList.
M	Part/@SheetIndex	Table 3-28 on page 104	
D	Part/@Sorting	Table 3-28 on page 104	
D	Part/@SortAmount	Table 3-28 on page 104	
N	Part/@Condition = "AuxiliarySheet"	Table 3-29 on page 113	
N	Part/@Condition = "Reusable"	Table 3-29 on page 113	
D	Abstract Audit/@Author	Table 3-32 on page 123	Replaced by Employee .
N	Abstract Audit/@QueueEntryID	Table 3-32 on page 123	
N	Abstract Audit/ Employee	Table 3-32 on page 123	
N	Notification/@CombinedProcessIndex	Table 3-38 on page 126	

Table O-2: Changed Items (Section 3 of 17)

C	XPath	Table and Page	Description
N	Notification/@ModuleID	Table 3-38 on page 126	
N	Notification/@ModuleIndex	Table 3-38 on page 126	
N	Notification/@ModuleType	Table 3-38 on page 126	
D	ModulePhase/@End	Table 3-40 on page 129	
D	ModulePhase/@Start	Table 3-40 on page 129	
N	ProcessRun/@ReturnTime	Table 3-41 on page 130	
N	ProcessRun/@SubmissionTime	Table 3-41 on page 130	
N	JMF/@AgentName	Table 5-1 on page 168	
N	JMF/@AgentVersion	Table 5-1 on page 168	
N	JMF/Employee	Table 5-1 on page 168	
N	Abstract Message/@AgentName	Table 5-2 on page 169	
N	Abstract Message/@AgentVersion	Table 5-2 on page 169	
N	Abstract Message/@ICSVersions	Table 5-2 on page 169	
N	Abstract Message/@SenderID	Table 5-2 on page 169	
N	Abstract Message/@Version	Table 5-2 on page 169	
N	Abstract Message/Employee	Table 5-2 on page 169	
N	List of JMF Messages/KnownSubscriptions	Table 5-3 on page 172	
N	List of JMF Messages/ RequestForAuthentication	Table 5-3 on page 172	
N	Signal/@ChannelMode	Table 5-6 on page 178	
M	Signal/QueryTypeObj	Table 5-6 on page 178	
N	Command/@RelatedCommands	Table 5-9 on page 180	
N	Command/@TransactionID	Table 5-9 on page 180	
N	Subscription/@ChannelMode	Table 5-12 on page 185	
N	Subscription/@RetryPolicy	Table 5-12 on page 185	
N	Messages for events and capabilities/ KnownSubscriptions	Table 5-15 on page 189	
N	Messages for events and capabilities/ RequestForAuthentication	Table 5-15 on page 189	
N	NotificationFilter/@MilestoneTypes	Table 5-17 on page 191	
N	KnownControllers/ControllerFilter	Table 5-19 on page 192	
N	ControllerFilter	Table 5-20 on page 193	
N	JDFController/@URLType	Table 5-21 on page 193	
N	KnownMsgQuParams/@ChannelMode	Table 5-26 on page 196	
N	MessageService/@ChannelMode	Table 5-27 on page 196	
N	MessageService/State	Table 5-27 on page 196	
N	KnownSubscriptions	Section 5.8.6 on page 198	
N	KnownSubscriptions/SubscriptionFilter	Table 5-28 on page 198	
N	SubscriptionFilter	Section 5.8.6.1 on page 198	
N	SubscriptionInfo	Section 5.8.6.2 on page 199	
N	Notification/NotificationFilter	Table 5-31 on page 200	

Table O-2: Changed Items (Section 4 of 17)

C	XPath	Table and Page	Description
N	RequestForAuthentication	Section 5.8.9 on page 202	
N	RequestForAuthentication Command	Section 5.8.9.1 on page 202	
N	RequestForAuthentication Query	Section 5.8.9.2 on page 204	
N	NewComment/Part	Table 5-50 on page 210	
M	ResourceQuParams/@JobID	Table 5-60 on page 214	
N	ResourceQuParams/@LotDetails	Table 5-60 on page 214	
N	ResourceQuParams/@LotID	Table 5-60 on page 214	
N	ResourceQuParams/@ResourceDetails	Table 5-60 on page 214	
M	ResourceQuParams/@ResourceName	Table 5-60 on page 214	
N	ResourceQuParams/@Scope	Table 5-60 on page 214	
M	ResourceCmdParams/@UpdateMethod	Table 5-62 on page 219	
N	ResourceCmdParams/@UpdateMethod = "Remove"	Table 5-62 on page 219	
N	ResourceCmdParams/@Usage	Table 5-62 on page 219	
N	ResourceInfo/@CommandResult	Table 5-63 on page 221	
M	ResourceInfo/@Level	Table 5-63 on page 221	
N	ResourceInfo/@LotControlled	Table 5-63 on page 221	
N	ResourceInfo/Lot	Table 5-63 on page 221	
M	ResourceInfo/Resource	Table 5-63 on page 221	
D	ResourcePullParams/@ReturnURL	Table 5-65 on page 226	
D	ResourcePullParams/@WatchURL	Table 5-65 on page 226	
N	Status Signal	Example 5-33 on page 228	New example
N	DeviceInfo/@IdleStartTime	Table 5-70 on page 230	
N	JobPhase/@URL	Table 5-71 on page 232	
D	JobPhase/@JDF	Table 5-71 on page 232	
N	UpdateJDF Signal	Section 5.9.11.2 on page 238	
N	PendingReturn Column	Table 5-92 on page 246	New column
D	SubmissionMethods/@HotFolder	Table 5-122 on page 260	
M	SubmissionMethods/@Packaging	Table 5-122 on page 260	
N	SubmissionMethods/@Packaging = "None"	Table 5-122 on page 260	
N	QueueFilter/@JobID	Table 5-126 on page 264	
N	QueueFilter/@JobPartID	Table 5-126 on page 264	
N	QueueFilter/@PreviewUsages	Table 5-126 on page 264	
M	QueueFilter/@QueueEntryDetails	Table 5-126 on page 264	
D	QueueFilter/@QueueEntryDetails = "JDF"	Table 5-126 on page 264	
N	QueueFilter/@UpdateGranularity	Table 5-126 on page 264	
N	QueueFilter/Part	Table 5-126 on page 264	
D	Template for Input Resources/Preview	Table 6-1 on page 269	moved Preview from Process Template for Input Resources to Table 3-1, "Any Element (generic content)," on page 40

Table O-2: Changed Items (Section 5 of 17)

C	XPath	Table and Page	Description
N	Template for Input Resources/ Tool	Table 6-1 on page 269	
M	ManualLabor (Output)/ Resource	Table 6-12 on page 272	multiple Resources are now allowed
N	DieDesign	Section 6.4.10 on page 279	
N	DieLayoutProduction	Section 6.4.11 on page 280	
M	Imposition	Section 6.4.17 on page 284	
N	LayoutShifting	Section 6.4.22 on page 298	
N	PageAssigning	Section 6.4.23 on page 299	
M	ConventionalPrinting (Input)/ Component	Table 6-96 on page 310	
N	ConventionalPrinting (Input)/ ExposedMedia (Sleeve)	Table 6-96 on page 310	
N	ConventionalPrinting (Input)/ Media (MountingTape)	Table 6-96 on page 310	
M	DigitalPrinting (Input)/ Component	Table 6-98 on page 312	
N	Varnishing	Section 6.5.3 on page 313	
D	BoxFolding (Input)/ Component (Application)	Table 6-104 on page 315	
M	CoverApplication (Input)/ Component (Cover)	Table 6-120 on page 319	
N	DieMaking	Section 6.6.14 on page 320	
M	Embossing (Input)/ Media	Table 6-128 on page 320	
M	Embossing (Input)/ Tool	Table 6-128 on page 320	
M	Inserting (Input)/ Component	Table 6-144 on page 325	
M	Palletizing (Input)/ Component	Table 6-154 on page 327	
M	RingBinding (Input)/ Component	Table 6-162 on page 328	
N	ShapeDefProduction	Section 6.6.37 on page 329	
N	StaticBlocking	Section 6.6.43 on page 331	
N	BindingIntent /@BackCoverColorDetails	Table 7-16 on page 352	
N	BindingIntent /@BindingColorDetails	Table 7-16 on page 352	
M	BindingIntent /@BindingOrder	Table 7-16 on page 352	
N	BindingIntent /@BindingOrder = "None"	Table 7-16 on page 352	
N	BindingIntent /@CoverColorDetails	Table 7-16 on page 352	
D	@BindingType = "Sewn"	Table 7-17 on page 354	
D	@BindingType = "SideSewn"	Table 7-17 on page 354	
D	@BindingType = "ThreadSealing"	Table 7-17 on page 354	
N	HardCoverBinding/HeadBandColorDetails	Table 7-23 on page 357	
D	Tape/TapeColor	Table 7-32 on page 363	
D	Tabs/@TabBanks	Table 7-31 on page 363	
N	Tabs/@TabCount	Table 7-31 on page 363	
N	Tabs/TabMylarColorDetails	Table 7-31 on page 363	
M	EmbossingItem/EmbossingType	Table 7-43 on page 375	
N	EmbossingItem/EmbossingType = "Braille"	Table 7-43 on page 375	

Table O-2: Changed Items (Section 6 of 17)

C	XPath	Table and Page	Description
N	EmbossingItem/FoilColorDetails	Table 7-43 on page 375	
N	MediaIntent/Flute	Table 7-51 on page 383	
N	MediaIntent/FluteDirection	Table 7-51 on page 383	
M	MediaIntent/GrainDirection	Table 7-51 on page 383	
N	MediaIntent/GrainDirection = "XDirection"	Table 7-51 on page 383	
N	MediaIntent/GrainDirection = "YDirection"	Table 7-51 on page 383	
N	MediaIntent/MediaQuality	Table 7-51 on page 383	
N	MediaIntent/MediaLayers	Table 7-51 on page 383	
N	NumberingIntent/ColorNameDetails	Table 7-52 on page 389	
N	NumberItem/ColorNameDetails	Table 7-53 on page 389	
N	PublishingIntent/ContentDataRefs	Table 7-58 on page 394	
M	PublishingIntent/IssueType	Table 7-58 on page 394	
N	PublishingIntent/IssueType = "Supplement"	Table 7-58 on page 394	
N	PublishingIntent/ContentList	Table 7-58 on page 394	
D	AssemblySection/@Order	Table 7-69 on page 402	
N	Perfect Bound (Gathering)	Example 7-2 on page 403	New example
N	Saddle-Stitched Brochure (Collecting)	Example 7-3 on page 404	New example
N	BinderySignature/ @AlignmentReferenceWeb	Table 7-76 on page 408	
N	BinderySignature/@FoldLay	Table 7-76 on page 408	
N	BinderySignature/@WebCellAlignment	Table 7-76 on page 408	
N	Pseudo Code to Generate Page Count from SignatureCell Elements	Example 7-4 on page 413	New example
D	SignatureCell/@BackFacePages	Table 7-77 on page 414	
N	SignatureCell/@FaceCells	Table 7-77 on page 414	
D	SignatureCell/@FrontFacePages	Table 7-77 on page 414	
N	StrippingParams: Foldout Using FaceCells	Example 7-5 on page 415	New example
D	BoxFoldingParams/BoxApplication	Table 7-79 on page 417	
D	BoxApplication	Section 7.2.14.1 on page 418	
N	BoxPackingParams/@Columns	Table 7-83 on page 422	
N	BoxPackingParams/ @ComponentOrientation	Table 7-83 on page 422	
N	BoxPackingParams/@Copies	Table 7-83 on page 422	
N	BoxPackingParams/@MaxWeight	Table 7-83 on page 422	
N	ByteMap/@ElementType	Table 7-88 on page 428	
M	ByteMap/@FrameHeight	Table 7-88 on page 428	
M	ByteMap/@FrameWidth	Table 7-88 on page 428	
M	ByteMap/@HalfToned	Table 7-88 on page 428	
M	ByteMap/@Interleaved	Table 7-88 on page 428	

Table O-2: Changed Items (Section 7 of 17)

C	XPath	Table and Page	Description
M	ByteMap / <i>@Resolution</i>	Table 7-88 on page 428	
M	ByteMap /Band	Table 7-88 on page 428	
M	ByteMap /PixelColorant	Table 7-88 on page 428	
M	Band/ <i>@Data</i>	Table 7-89 on page 429	
M	Band/ <i>@Height</i>	Table 7-89 on page 429	
M	Band/ <i>@WasMarked</i>	Table 7-89 on page 429	
M	Band/ <i>@Width</i>	Table 7-89 on page 429	
N	CasingInParams / GlueApplication	Table 7-92 on page 431	
D	CasingInParams / GlueLine	Table 7-92 on page 431	
N	ChannelBindingParams / <i>@ClampColorDetails</i>	Table 7-93 on page 432	
N	CoilBindingParams / <i>@ColorDetails</i>	Table 7-95 on page 434	
N	Color / <i>@ColorDetails</i>	Table 7-97 on page 436	
N	Color / <i>@Gray</i>	Table 7-97 on page 436	
N	ColorantControl: Content-Ignorant MIS	Example 7-9 on page 441	
N	ColorantControl: Synchronized with Input	Example 7-10 on page 442	
N	ColorantControl: Synchronized with Input with Alias	Example 7-11 on page 443	
N	ColorantControl: with ColorantAlias/Replacement-ColorantName	Example 7-12 on page 443	
N	ColorantControl: with Invalid ColorantAlias/ReplacementColorantName	Example 7-13 on page 444	
N	ColorantAlias / <i>@RawNames</i>	Table 7-100 on page 444	
N	ColorantAlias/ <i>@RawNames</i>	Example 7-14 on page 444	
M	ColorantControl / <i>@ProcessColorModel</i>	Table 7-101 on page 446	
N	ColorantControl / <i>@ProcessColorModel = "None"</i>	Table 7-101 on page 446	
N	ColorantControl /ColorantConvertProcess	Table 7-101 on page 446	
N	ColorantConvertProcess	Section 7.2.28.1 on page 448	
M	ColorControlStrip / <i>@Center</i>	Table 7-108 on page 450	
M	ColorControlStrip / <i>@Size</i>	Table 7-108 on page 450	
N	ColorControlStrip /SeparationSpec	Table 7-108 on page 450	
N	ColorCorrectionOp / <i>@ObjectTags</i>	Table 7-110 on page 452	
N	ColorSpaceConversionOp / <i>@ObjectTags</i>	Table 7-113 on page 455	
M	ColorSpaceConversionOp /FileSpec	Table 7-113 on page 455	Changed from '?' to '*'.
N	ColorSpaceConversionOp / <i>@SourceCS="All"</i>	Table 7-113 on page 455	
D	Component / <i>@IsWaste</i>	Table 7-121 on page 468	
N	Component / <i>@SpineThickness</i>	Table 7-121 on page 468	
N	Component /Media	Table 7-121 on page 468	
N	Component / <i>@ProductType = "BlankSheet"</i>	Table 7-122 on page 470	

Table O-2: Changed Items (Section 8 of 17)

C	XPath	Table and Page	Description
N	Component /@ProductType = "BlankWeb"	Table 7-122 on page 470	
N	Component /@ProductType = "Stack"	Table 7-122 on page 470	
M	Contact @ContactTypes	Table 7-123 on page 473	
N	Contact /@ContactTypes= "Agency"	Table 7-123 on page 473	
N	Contact /@ContactTypes= "Author"	Table 7-123 on page 473	
N	Contact /@ContactTypes= "Designer"	Table 7-123 on page 473	
N	Contact /@ContactTypes= "Editor"	Table 7-123 on page 473	
N	Contact /@ContactTypes= "Illustrator"	Table 7-123 on page 473	
N	Contact /@ContactTypes= "Photographer"	Table 7-123 on page 473	
N	Contact /@ContactTypes= "TelephoneSanitizer"	Table 7-123 on page 473	
M	ContentList /ContentData	Table 7-125 on page 474	
N	ContentData /@ContentRefs	Table 7-126 on page 475	
N	ContentData /@ID	Table 7-126 on page 475	
N	ContentData /ContentMetadata	Table 7-126 on page 475	
N	ContentMetadata	Section 7.2.40.2 on page 476	
N	ContentList	Example 7-18 on page 476	
N	ContentList: Extended with ISBN, Author, etc.	Example 7-19 on page 477	
D	ConventionalPrintingParams / @ModuleAvailableIndex	Table 7-129 on page 479	
D	ConventionalPrintingParams / @ModuleIndex	Table 7-129 on page 479	
N	ConventionalPrintingParams / @PrintingTechnology	Table 7-129 on page 479	
D	ConventionalPrintingParams /Ink	Table 7-129 on page 479	
M	CustomerMessage /@ShowList	Table 7-136 on page 485	
N	CustomerMessage /@ShowList description	Table 7-136 on page 485	New paragraph
N	CutBlock /@CutWidth	Table 7-137 on page 486	
N	CuttingParams /@NUpSeparation	Table 7-140 on page 488	
N	Cut /@CutWidth	Table 7-141 on page 489	
N	PageCondition	Example 7-23 on page 598	
D	Device /@FriendlyName	Table 7-154 on page 500	
N	Device /Location	Table 7-154 on page 500	
N	DeviceMark /@Anchor	Table 7-158 on page 503	
M	DeviceMark /@FontSize	Table 7-158 on page 503	
N	DeviceMark /@HorizontalFitPolicy	Table 7-158 on page 503	
D	DeviceMark /@MarkJustification	Table 7-158 on page 503	
D	DeviceMark /@MarkOffset	Table 7-158 on page 503	
D	DeviceMark /@MarkOrientation	Table 7-158 on page 503	
D	DeviceMark /@MarkPosition	Table 7-158 on page 503	
N	DeviceMark /@VerticalFitPolicy	Table 7-158 on page 503	

Table O-2: Changed Items (Section 9 of 17)

C	XPath	Table and Page	Description
N	DeviceMark/BarcodeReproParams	Table 7-158 on page 503	
N	DieLayout/@DieSide	Table 7-160 on page 505	
N	DieLayout/@MediaSide	Table 7-160 on page 505	
N	DieLayout/@Rotated	Table 7-160 on page 505	
N	DieLayout/@Waste	Table 7-160 on page 505	
N	DieLayout/Device	Table 7-160 on page 505	
N	DieLayout/Media	Table 7-160 on page 505	
N	DieLayout/RuleLength	Table 7-160 on page 505	
N	RuleLength	Section 7.2.61.1 on page 506	
N	Station/ShapeDef	Table 7-162 on page 506	
N	DieLayoutProductionParams	Section 7.2.62 on page 506	
N	ConvertingConfig	Section 7.2.62.1 on page 507	
N	RepeatDesc	Section 7.2.62.2 on page 507	
D	DigitalPrintingParams/Ink	Table 7-168 on page 515	
N	Emboss/IdentificationField	Table 7-173 on page 522	
N	Emboss/Media	Table 7-173 on page 522	
N	Emboss/Tool	Table 7-173 on page 522	
M	Employee/@Roles	Table 7-174 on page 523	
N	Employee/@Roles= "StandBy"	Table 7-174 on page 523	
M	Feeder/FeederType	Table 7-180 on page 527	
N	Feeder/FeederType = "BookBlock"	Table 7-180 on page 527	
N	FileSpec/@Encoding	Table 7-183 on page 531	
M	FoldingParams/@FoldCatalog	Table 7-189 on page 541	
M	GeneralID	Section 7.2.84 on page 549	Changed to Element . See "GeneralID" on page 43.
N	HeadBandApplicationParams/@BottomColorDetails	Table 7-201 on page 552	
N	HeadBandApplicationParams/@TopColorDetails	Table 7-201 on page 552	
M	IdentificationField/@Encoding	Table 7-206 on page 558	
N	IdentificationField/@Encoding = "Braille"	Table 7-206 on page 558	
N	IdentificationField/@EncodingDetails = "BrailleASCII"	Table 7-207 on page 560	
N	IdentificationField/@EncodingDetails = "BrailleUnicode"	Table 7-207 on page 560	
D	Ink/@ColorName	Table 7-225 on page 576	
M	Ink/@SpecialInk	Table 7-225 on page 576	Data type changed
N	JacketingParams/@FoldingDistance	Table 7-235 on page 589	
N	JobField/@JobFormat	Table 7-236 on page 590	
N	JobField/@JobTemplate	Table 7-236 on page 590	
M	JobField/@ShowList	Table 7-236 on page 590	

Table O-2: Changed Items (Section 10 of 17)

C	XPath	Table and Page	Description
N	JobField / <i>@ShowList</i> description	Table 7-236 on page 590	New values
M	JobField / <i>@UserText</i>	Table 7-236 on page 590	
N	JobField / <i>@UserText</i> description	Table 7-236 on page 590	New paragraph
D	JobField / <i>@DeviceMark</i>	Table 7-236 on page 590	
N	Layout / <i>@MaxCollect</i>	Table 7-239 on page 592	
D	Layout / <i>@MaxDocOrd</i>	Table 7-239 on page 592	
D	Layout / <i>@MaxOrd</i>	Table 7-239 on page 592	
D	Layout / <i>@MaxSetOrd</i>	Table 7-239 on page 592	
N	Layout / <i>@MinCollect</i>	Table 7-239 on page 592	
D	Layout / <i>@Name</i>	Table 7-239 on page 592	
N	Layout / <i>@NameFormat</i>	Table 7-239 on page 592	
N	Layout / <i>@NameTemplate</i>	Table 7-239 on page 592	
N	Layout / <i>@OrdsConsumed</i>	Table 7-239 on page 592	
N	Layout / <i>@BaseOrdReset</i>	Table 7-239 on page 592	
N	Layout /LogicalStackParams	Table 7-239 on page 592	
N	Layout /PageCondition	Table 7-239 on page 592	
N	Layout /SheetCondition	Table 7-239 on page 592	
N	LogicalStackParams	Section 7.2.109.3 on page 596	
N	Stack	Section 7.2.109.4 on page 596	
N	PageCondition	Section 7.2.109.5 on page 597	
N	SheetCondition	Section 7.2.109.6 on page 599	
N	Abstract PlacedObject / <i>@Anchor</i>	Table 7-246 on page 600	
N	Abstract PlacedObject / <i>@ClipBoxFormat</i>	Table 7-246 on page 600	
N	Abstract PlacedObject / <i>@ClipBoxTemplate</i>	Table 7-246 on page 600	
N	Abstract PlacedObject / <i>@CompensationCTMFormat</i>	Table 7-246 on page 600	
N	Abstract PlacedObject / <i>@CompensationCTMTemplate</i>	Table 7-246 on page 600	
N	Abstract PlacedObject / <i>@LogicalStackOrd</i>	Table 7-246 on page 600	
N	Abstract PlacedObject / <i>@TrimClipPath</i>	Table 7-246 on page 600	
N	Abstract PlacedObject /FitPolicy	Table 7-246 on page 600	
M	ContentObject / <i>@Ord</i>	Table 7-247 on page 603	
N	MarkObject / <i>@ContentRef</i>	Table 7-248 on page 603	
D	MarkObject / <i>@LayoutElementPageNum</i>	Table 7-248 on page 603	
M	MarkObject / <i>@Ord</i>	Table 7-248 on page 603	
N	MarkObject /DeviceMark	Table 7-248 on page 603	
M	MarkObject /JobField	Table 7-248 on page 603	
D	MarkObject /LayoutElement	Table 7-248 on page 603	

Table O-2: Changed Items (Section 11 of 17)

C	XPath	Table and Page	Description
N	MarkObject/MarkActivation	Table 7-248 on page 603	
N	MarkObject/RefAnchor	Table 7-248 on page 603	
N	MarkActivation	Section 7.2.109.10 on page 605	
D	DynamicField/@Ord	Table 7-250 on page 606	
D	DynamicField/@OrdExpression	Table 7-250 on page 606	
D	DynamicField/DeviceMark	Table 7-250 on page 606	
N	LayoutElement/@ContentDataRefs	Table 7-253 on page 612	
N	LayoutElement/@SourceMediaBox	Table 7-253 on page 612	
N	LayoutElement/ContentList	Table 7-253 on page 612	
N	LayoutElementProductionParams/ActionPool	Table 7-256 on page 616	
N	LayoutElementProductionParams/ShapeDef	Table 7-256 on page 616	
N	LayoutElementProductionParams/TestPool	Table 7-256 on page 616	
N	LayoutElementPart/@ID	Table 7-257 on page 617	
N	LayoutElementPart/LayoutElement	Table 7-257 on page 617	
N	LayoutElementPart/PositionObj	Table 7-257 on page 617	
N	PositionObj	Section 7.2.111.3 on page 618	
N	LayoutShift	Section 7.2.113 on page 636	
N	LayoutShift	Example 7-38 on page 637	
M	ManualLaborParams/@LaborType	Table 7-266 on page 638	
N	ManualLaborParams/@LaborType = "SeparateBlanks"	Table 7-266 on page 638	
N	Media/@BackCoatingDetail	Table 7-267 on page 639	
M	Media/@Dimension	Table 7-267 on page 639	
N	Media/@Dimension description	Table 7-267 on page 639	New paragraph
N	Media/@FrontCoatingDetail	Table 7-267 on page 639	
M	Media/@FrontCoatings	Table 7-267 on page 639	
D	Media/@FrontCoatings= "InkJet"	Table 7-267 on page 639	
N	Media/@InnerCoreDiameter	Table 7-267 on page 639	
N	Media/@MediaQuality	Table 7-267 on page 639	
M	Media/@MediaType	Table 7-267 on page 639	
N	Media/@MediaType = "MountingTape"	Table 7-267 on page 639	
N	Media/@MediaType = "Screen"	Table 7-267 on page 639	
N	Media/@MediaType = "Sleeve"	Table 7-267 on page 639	
M	Media/@PlateTechnology	Table 7-267 on page 639	
N	Media/@PlateTechnology = "FlexoAnalogSolvent"	Table 7-267 on page 639	

Table O-2: Changed Items (Section 12 of 17)

C	XPath	Table and Page	Description
N	Media/@PlateTechnology = "FlexoAnalogThermal"	Table 7-267 on page 639	
N	Media/@PlateTechnology = "FlexoDigitalSolvent"	Table 7-267 on page 639	
N	Media/@PlateTechnology = "FlexoDigitalThermal"	Table 7-267 on page 639	
N	Media/@PlateTechnology = "FlexoDirectEngraving"	Table 7-267 on page 639	
N	Media/@PrintingTechnology	Table 7-267 on page 639	
N	Media/@ReliefThickness	Table 7-267 on page 639	
N	Media/@SleeveInterlock	Table 7-267 on page 639	
M	Media/@StockType	Table 7-267 on page 639	
N	Media/@StockType description	Table 7-267 on page 639	New paragraph
N	Media/@StockType = "Aatoposutoshi"	Table 7-267 on page 639	
N	Media/@StockType = "Aatoshi"	Table 7-267 on page 639	
N	Media/@StockType = "Chuushitsushi"	Table 7-267 on page 639	
N	Media/@StockType = "Joushitsushi"	Table 7-267 on page 639	
N	Media/@StockType = "Mashinkootoshi"	Table 7-267 on page 639	
N	Media/TabDimensions	Table 7-267 on page 639	
N	Media/@MediaTypeDetails = "FlexoBase"	Table 7-267 on page 639	
N	Media/@MediaTypeDetails = "FlexoPhotoPolymer"	Table 7-267 on page 639	
N	TabDimensions	Section 7.2.116.2 on page 649	
M	MiscConsumable/@ConsumableType	Table 7-271 on page 656	
N	MiscConsumable/@ConsumableType = "Headband"	Table 7-271 on page 656	
N	MSDetails/@Complexity	Table 7-272 on page 656	
N	NodeInfo/@WorkStepID	Table 7-273 on page 658	
N	ObjectResolution/@ObjectTags	Table 7-276 on page 661	
N	PageAssignParams	Section 7.2.125 on page 662	
N	PageData/@PageIndex	Table 7-280 on page 664	
N	PageElement/@ContentDataRefs	Table 7-281 on page 667	
D	PageElement/@ContentListIndex	Table 7-281 on page 667	
M	Pallet/@PalletType	Table 7-282 on page 667	
D	Pallet/@PalletType = "Euro"	Table 7-282 on page 667	
N	Pallet/@PalletType = "Euro800x600"	Table 7-282 on page 667	
N	Pallet/@PalletType = "Euro800x1200"	Table 7-282 on page 667	
N	Pallet/@PalletType = "Euro1000x1200"	Table 7-282 on page 667	
N	Pallet/@PalletType = "Euro1200x1200"	Table 7-282 on page 667	
N	PalletizingParams/@LayerAmount	Table 7-283 on page 668	
N	PalletizingParams/@Overhang	Table 7-283 on page 668	

Table O-2: Changed Items (Section 13 of 17)

C	XPath	Table and Page	Description
N	PalletizingParams/@OverhangOffset	Table 7-283 on page 668	
N	PalletizingParams/Bundle	Table 7-283 on page 668	
N	Person/@Languages	Table 7-289 on page 674	
N	PlasticCombBindingParams/ @ColorDetails	Table 7-290 on page 675	
N	BasicPreflightTest/@Classes	Table 7-293 on page 677	
N	BasicPreflightTest/@ClassName	Table 7-293 on page 677	
M	BasicPreflightTest/@ListType	Table 7-293 on page 677	
M	BasicPreflightTest/@Name	Table 7-293 on page 677	
N	Preview/@MimeTypeDetails	Table 7-313 on page 687	
M	Preview/@PreviewFileType	Table 7-313 on page 687	
N	Preview/@PreviewFileType description	Table 7-313 on page 687	New paragraph
M	Preview/@PreviewUsage	Table 7-313 on page 687	
N	Preview/@PreviewUsage = "3D"	Table 7-313 on page 687	
N	Preview/@PreviewUsage = "Animation"	Table 7-313 on page 687	
N	RefAnchor	Section 7.2.153 on page 704	
M	RegisterMark/@MarkType	Table 7-333 on page 705	
M	RegisterMark/@MarkUsage	Table 7-333 on page 705	
N	RegisterMark/@MarkUsage = "Tile"	Table 7-333 on page 705	
M	RegisterRibbon/@LengthOverall	Table 7-334 on page 705	
N	RegisterRibbon/@RibbonColorDetails	Table 7-334 on page 705	
M	RegisterRibbon/@VisibleLength	Table 7-334 on page 705	
N	RingBindingParams/@BinderColorDetails	Table 7-338 on page 708	
N	RingBindingParams/@SpineColorDetails	Table 7-338 on page 708	
D	RunList/@ComponentGranularity	Table 7-340 on page 711	
N	RunList/@IgnoreContext	Table 7-340 on page 711	
N	RunList/@SheetSides	Table 7-340 on page 711	
D	RunList/DynamicInput	Table 7-340 on page 711	
N	RunList/MetadataMap	Table 7-340 on page 711	
N	Marks and Reordering of Content using RunList/ @IgnoreContext	Example 7-48 on page 716	
D	DynamicInput	Section 7.2.160.1 on page 717	
N	MetadataMap	Section 7.2.160.2 on page 717	
N	Expr	Section 7.2.160.3 on page 720	
N	RunList/MetadataMap	Example 7-50 on page 721	
N	ScreenSelector/@ObjectTags	Table 7-347 on page 727	
D	Shape/@CutType	Table 7-350 on page 731	
N	Shape/@DDESCutType	Table 7-350 on page 731	
D	Shape/@StationName	Table 7-350 on page 731	

Table O-2: Changed Items (Section 14 of 17)

C	XPath	Table and Page	Description
N	ShapeDef	Section 7.2.169 on page 732	
N	ShapeDefProductionParams	Section 7.2.170 on page 733	
N	ObjectModel	Section 7.2.170.1 on page 734	
N	ShapeTemplate	Section 7.2.170.2 on page 734	
N	SpineTapingParams/ <i>@HorizontalExcessBack</i>	Table 7-359 on page 738	
N	SpineTapingParams/ <i>@StripColorDetails</i>	Table 7-359 on page 738	
N	StackingParams/ <i>@BundleDepth</i>	Table 7-361 on page 741	
N	StackingParams/ <i>@LayerLift</i>	Table 7-361 on page 741	
N	StackingParams/ <i>@LayerCompression</i>	Table 7-361 on page 741	
N	StackingParams/ <i>@MaxHeight</i>	Table 7-361 on page 741	
N	StackingParams/ <i>@PreStackAmount</i>	Table 7-361 on page 741	
N	StackingParams/ <i>@PreStackMethod</i>	Table 7-361 on page 741	
N	StackingParams/ <i>@StackCompression</i>	Table 7-361 on page 741	
N	StaticBlockingParams	Section 7.2.177 on page 743	
N	StitchingParams/ <i>@StitchOrigin</i>	Table 7-363 on page 745	
N	Strap/ <i>@StrapColorDetails</i>	Table 7-364 on page 747	
N	StripBindingParams/ <i>@StripColorDetails</i>	Table 7-366 on page 748	
N	StrippingParams/ <i>@InnermostShingling</i>	Table 7-367 on page 749	
N	StrippingParams/ <i>@NameFormat</i>	Table 7-367 on page 749	
N	StrippingParams/ <i>@NameTemplate</i>	Table 7-367 on page 749	
N	StrippingParams/ <i>@OutermostShingling</i>	Table 7-367 on page 749	
N	StrippingParams/ <i>@StackDepth</i>	Table 7-367 on page 749	
N	StrippingParams/ StripMark	Table 7-367 on page 749	
N	StripMark/ <i>@AbsoluteHeight</i>	Table 7-370 on page 755	
N	StripMark/ <i>@AbsoluteWidth</i>	Table 7-370 on page 755	
N	StripMark/ <i>@Anchor</i>	Table 7-370 on page 755	
N	StripMark/ <i>@HorizontalFitPolicy</i>	Table 7-370 on page 755	
N	StripMark/ <i>@ID</i>	Table 7-370 on page 755	
N	StripMark/ <i>@MarkContext</i>	Table 7-370 on page 755	
N	StripMark/ <i>@Offset</i>	Table 7-370 on page 755	
N	StripMark/ <i>@Ord</i>	Table 7-370 on page 755	
N	StripMark/ <i>@Orientation</i>	Table 7-370 on page 755	
N	StripMark/ <i>@RelativeHeight</i>	Table 7-370 on page 755	
N	StripMark/ <i>@RelativeWidth</i>	Table 7-370 on page 755	
M	StripMark/ <i>@StripMarkDetails</i>	Table 7-370 on page 755	
N	StripMark/ <i>@VerticalFitPolicy</i>	Table 7-370 on page 755	
D	StripMark/Position	Table 7-370 on page 755	
N	StripMark/RefAnchor	Table 7-370 on page 755	

Table O-2: Changed Items (Section 15 of 17)

C	XPath	Table and Page	Description
N	StripMark/@MarkName = "BleedMark"	Table 7-371 on page 757	
N	StripMark/@MarkName = "CenterMark"	Table 7-371 on page 757	
N	StripMark/@MarkName = "CollationMark"	Table 7-371 on page 757	
N	StripMark/@MarkName = "FoldMark"	Table 7-371 on page 757	
N	StripMark/@MarkName = "Set"	Table 7-371 on page 757	
N	StripMark/@MarkName = "TrimMark"	Table 7-371 on page 757	
N	Tile/MarkObject	Table 7-375 on page 761	
M	Tool/@ToolType	Table 7-376 on page 762	
N	Tool/@ToolType = "Braille"	Table 7-376 on page 762	
N	Tool/@ToolType = "ChangingCuttingBlock"	Table 7-376 on page 762	
D	TrappingDetails/@IgnoreFileParams	Table 7-381 on page 765	
N	VarnishingParams	Section 7.2.196 on page 772	
N	WireCombBindingParams/@ColorDetails	Table 7-392 on page 775	
N	Abstract Evaluation/@Path	Table 7-443 on page 812	
M	Abstract Evaluation/@rRef	Table 7-443 on page 812	
N	Font Properties/@IsDoubleByteFont	Table 7-477 on page 837	
N	Pages Properties/ @PageHasOptionalContent	Table 7-483 on page 842	
N	Pages Properties/@PageScalingFactor	Table 7-483 on page 842	
N	languages	Section A.2.25 on page 863	
N	URL/Examples	Section A.2.44 on page 867	New examples
N	Anchor	Section A.3.3.1 on page 870	Enumeration Values
M	@StatusDetails = "ControlDeferred"	Table C-1 on page 875	
N	@StatusDetails = "IterationPaused"	Table C-1 on page 875	
N	@StatusDetails = "Processing"	Table C-2 on page 878	
N	@StatusDetails = "WaitingForMarker"	Table C-2 on page 878	
N	@StatusDetails = "WaitingForReferencedDataCollector "	Table C-2 on page 878	
N	@StatusDetails = "WaitingForReferencedDataCollector "	Table C-2 on page 878	
N	@StatusDetails = "WaitingForRIP"	Table C-2 on page 878	
N	@ModuleType = "BlockPreparer"	Table C-5 on page 880	
N	@ModuleType = "BoxFolder"	Table C-5 on page 880	
N	@ModuleType = "CaseMaker"	Table C-5 on page 880	
N	@ModuleType = "Caser"	Table C-5 on page 880	
N	@ModuleType = "EndSheetGluer"	Table C-5 on page 880	
N	@ModuleType = "Gluer"	Table C-5 on page 880	
N	@ModuleType = "HeadBandApplicator"	Table C-5 on page 880	

Table O-2: Changed Items (Section 16 of 17)

C	XPath	Table and Page	Description
N	@ModuleType = "InkjetPrinter"	Table C-5 on page 880	
N	@ModuleType = "Inserter"	Table C-5 on page 880	
N	@ModuleType = "Jacketer"	Table C-5 on page 880	
N	@ModuleType = "PressingStation"	Table C-5 on page 880	
N	@ModuleType = "ShapeCutter"	Table C-5 on page 880	
N	@ModuleType = "SpinePreparer"	Table C-5 on page 880	
N	@ModuleType = "SpineTaper"	Table C-5 on page 880	
N	@ModuleType = "Strapper"	Table C-5 on page 880	
N	@ModuleType = "ThreadSealer"	Table C-5 on page 880	
N	@ModuleType = "ThreadSewer"	Table C-5 on page 880	
N	@ModuleType = "Marker"	Table C-6 on page 881	
N	@ModuleType = "MimeUnpacker"	Table C-6 on page 881	
N	@ModuleType = "ReferencedDataCollector"	Table C-6 on page 881	
N	@ModuleType = "RIP"	Table C-6 on page 881	
N	@ModuleType = "PostMarkerInserter"	Table C-21 on page 889	
N	@ReturnCode = "11"	Table D-1 on page 891	
N	@ReturnCode = "12"	Table D-1 on page 891	
N	@ReturnCode = "13"	Table D-1 on page 891	
M	@ReturnCode = "114"	Table D-1 on page 891	
N	@ReturnCode = "304"	Table D-1 on page 891	
N	@ReturnCode = "305"	Table D-1 on page 891	
N	@ReturnCode = "306"	Table D-1 on page 891	
N	@ReturnCode = "307"	Table D-1 on page 891	
N	Media Sizes = "A4Tab"	Table G-1 on page 899	
N	Media Sizes = "LetterTabThreeEighthsInch"	Table G-1 on page 899	
N	Media Sizes = "LetterTabHalfInch"	Table G-1 on page 899	
N	Media Sizes = "LetterTabFiveEighthsInch"	Table G-1 on page 899	
N	Media Sizes = "SRA3"	Table G-1 on page 899	
N	@MimeType	Table H-1 on page 903	Modified column and rows.
N	@MimeType = "application/vnd.iccprofile"	Table H-1 on page 903	
N	@MimeType = "x-world/x-vrml"	Table H-1 on page 903	
D	@MimeType = "ICC Profile"	Table H-2 on page 905	
N	@XXXTemplate = "Amount"	Table I-1 on page 909	
N	@XXXTemplate = "DeviceID"	Table I-1 on page 909	
N	@XXXTemplate = "EndTime"	Table I-1 on page 909	
N	@XXXTemplate = "Error"	Table I-1 on page 909	
N	@XXXTemplate = "ErrorStats"	Table I-1 on page 909	
N	@XXXTemplate = "ExposedMediaName"	Table I-1 on page 909	
N	@XXXTemplate = "FriendlyName"	Table I-1 on page 909	
N	@XXXTemplate = "GeneralID:XXX"	Table I-1 on page 909	

Table O-2: Changed Items (Section 17 of 17)

C	XPath	Table and Page	Description
D	@XXXTemplate="JobID"	Table I-1 on page 909	
D	@XXXTemplate="JobName"	Table I-1 on page 909	
N	@XXXTemplate="JobRecipientName"	Table I-1 on page 909	
N	@XXXTemplate="JobSubmitterName"	Table I-1 on page 909	
N	@XXXTemplate="MediaBrand"	Table I-1 on page 909	
N	@XXXTemplate="MediaType"	Table I-1 on page 909	
N	@XXXTemplate="MoonPhase"	Table I-1 on page 909	
N	@XXXTemplate="Operator"	Table I-1 on page 909	
N	@XXXTemplate="OperatorText"	Table I-1 on page 909	
N	@XXXTemplate="PrintQuality"	Table I-1 on page 909	
N	@XXXTemplate="ProoferProfileName"	Table I-1 on page 909	
N	@XXXTemplate="PressProfileName"	Table I-1 on page 909	
N	@XXXTemplate="Resolution"	Table I-1 on page 909	
N	@XXXTemplate="ResolutionX"	Table I-1 on page 909	
N	@XXXTemplate="ResolutionY"	Table I-1 on page 909	
N	@XXXTemplate="ScreeningFamily"	Table I-1 on page 909	
D	@XXXTemplate="sep"	Table I-1 on page 909	
N	@XXXTemplate="StartTime"	Table I-1 on page 909	
N	@XXXTemplate="UserText"	Table I-1 on page 909	
N	@XXXTemplate="Warning"	Table I-1 on page 909	

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 a 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.

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

[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 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 spe-

cific Partitions. It is an error to define PartStatus Elements that reference identical or overlapping parts within one StatusPool. Partitioned Resources are described in Section 3.10.5, Description of Partitioned Resources.

Table P-2: Contents of the StatusPool Element

Name	Data Type	Description
<i>Status?</i>	enumeration	Identifies the <i>Status</i> of the Node when <code>JDF/@Status = "Pool"</code> . Individual PartStatus 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> . Values are from: <code>JDF/@Status</code> (except <code>Pool</code>)
<i>StatusDetails?</i> New in JDF 1.2	string	Identifies the <i>StatusDetails</i> of the Node when <code>JDF/@Status = "Pool"</code> . Individual PartStatus 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> . Values include those from: "StatusDetails Supported Strings" on page 875
PartStatus *	element	Element that defines the Node's status for a set of parts.

P.1.2.1 Element: PartStatus

The following table describes the PartStatus Element.

Table P-3: Contents of the PartStatus Element

Name	Data Type	Description
<i>Status?</i>	enumeration	Identifies the status of an individual part of the Node. If not specified, defaults to <code>StatusPool/@Status</code> . In JDF 1.3 and beyond, <i>Status</i> has been replaced by <code>NodeInfo/@NodeStatus</code> . Values are from: <code>JDF/@Status</code>
<i>StatusDetails?</i> 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> . In JDF 1.3 and beyond, <i>StatusDetails</i> has been replaced by <code>NodeInfo/@NodeStatus</code> . Values include those from: "StatusDetails Supported Strings" on page 875
Part Modified in JDF 1.2	element	Specifies the selected part that the PartStatus 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 Part may contain either <i>Side</i> only or <i>Side</i> and <i>Separation</i> , but not <i>Separation</i> only. See Table 3-28, "Part Element," on page 104 for details of the Part Element. For details on Partitioned Resources, see "Description of Partitioned Resources" on page 94. Note: the cardinality of Part 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

P.2.1.1 Element: Trigger

The following 3 Elements were deprecated from Signal/Trigger in JDF 1.2.

P.2.1.2 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 <code>ResponseTypeObj</code> like <code>NotificationDef</code>, a <code>JDFController</code>, a <code>Device</code>, a <code>JDFService</code> or a <code>MessageService</code>.</p> <p>For details on these Elements see Section 5.8, Messages for Events and Capabilities.</p>

P.2.1.3 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.

P.2.1.4 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 <code>ResponseTypeObj</code> like <code>NotificationDef</code>, a <code>JDFController</code>, a <code>Device</code>, a <code>JDFService</code> or a <code>MessageService</code>.</p> <p>For details on these Elements see Section 5.8, 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 Message or a Query Message to modify or to query JDF `NodeInfo` Elements. The Query Message simply retrieves information about the `NodeInfo` without modifying it, while the command modifies those settings within the `NodeInfo` that are specified. Settings that are not specified remain unchanged.

P.2.2.1 NodeInfo Query

The NodeInfo Query Message is made selective by specifying a NodeInfoQuParams Element. The query's Response Message returns a list of NodeInfoResp Elements that contains the queried information concerning the NodeInfo Elements. If the list is empty because the selective query parameters of the NodeInfoQuParams lead to a null selection, then the *ReturnCode* is 103 (*JobID* unknown), 104 (*JobPartID* unknown) or 108 (empty list) and SHOULD be flagged as a warning with Notification [*@Class* = "Warning" and *@Type* = "Error"].

Table P-7: NodeInfo Query Message

Object Type	Element Name	Description
QueryTypeObj	NodeInfoQuParams	Specifies the Node queried.
ResponseTypeObj	NodeInfoResp *	Details of the NodeInfo Elements

— Element: NodeInfoQuParams

Table P-8: NodeInfoQuParams 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. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> , or <i>QueueEntryID</i> are specified, ResourceQuParams applies to all Jobs.
Part *	element	Part Elements that describe the Partition of the Job whose NodeInfo is modified. For details on Node Partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 150.

P.2.2.2 NodeInfo Command

The NodeInfo Command Message is used to modify the NodeInfo – generally scheduling information – of a submitted JDF Node. It is made selective by specifying the OPTIONAL Attributes in the NodeInfoCmdParams Element.

The Response Message contains a list of NodeInfoResp Elements with a copy of NodeInfo after the changes have been applied. If the NodeInfo Command Message was successful, the value of the *ReturnCode* Attribute is "0". If it is not successful, the value of *ReturnCode* might be one of those described in the above section about the NodeInfo Query Message; it might also be "200" (invalid parameters) or "201" (insufficient parameters). Partial application of the NodeInfo SHOULD also be flagged as a warning. If the value of *ReturnCode* is larger than "0", the Controller that issued the command can evaluate the returned NodeInfo in order to find the setting that could not be applied.

Table P-9: NodeInfo Command Message

Object Type	Element name	Description
CommandTypeObj	NodeInfoCmdParams	Specifies the NodeInfo Elements to be modified.
ResponseTypeObj	NodeInfoResp *	Contains information about the NodeInfo and the NodeInfo after modification.

— Element: NodeInfoCmdParams

Table P-10: NodeInfoCmdParams 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, NodeInfoCmdParams applies to all Jobs.
<i>UpdateMethod</i> = "Complete"	enumeration	Method how NodeInfo is applied to the JDF. Values are: <i>Complete</i> – The NodeInfo in the JDF is completely overwritten by NodeInfo in this Message. <i>Incremental</i> – The NodeInfo in the JDF is incrementally updated by the values that are explicitly set in NodeInfo in this Message.
Part *	element	Part Elements that describe the Partition of the Job whose NodeInfo is modified. For details on Node Partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 150.
NodeInfo ?	element	NodeInfo to be uploaded to the Device.

— Element: NodeInfoResp

Table P-11: 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, NodeInfoResp applies to all Jobs.
Part *	element	Part Elements that describe the Partition of the Job NodeInfo is modified. For details on Node Partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 150.
NodeInfo ?	element	NodeInfo after uploading to the Controller.

The following is an example for retrieving NodeInfo settings:

```
<Query ID="Q1" Type="NodeInfo">
  <NodeInfoQuParams JobID="J1"/>
</Query>
```

The following is a possible Response Message to the Query Message above:

```
<Response ID="M1" Type="NodeInfo" refID="Q1">
  <NodeInfoResp JobID="J1" JobPartID="P1">
    <NodeInfo/>
  </NodeInfoResp>
  <NodeInfoResp JobID="J1" JobPartID="P2">
    <NodeInfo/>
  </NodeInfoResp>
</Response>
```

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: KnownJDFServices Message

Object Type	Element name	Description
QueryTypeObj	—	—
ResponseTypeObj	JDFService *	Processes that the Controller or Device can execute.

The KnownJDFServices Query Message returns a list of services that are defined in the JDF specification, such as **ConventionalPrinting**, **RIPing**, or **EndSheetGluing**. It allows a Controller to publish the services that the Devices it controls are capable of providing. The response is a list of JDFService Elements, one for each supported Process type.

P.2.3.1 Element: JDFService

JDFService Elements define the Node types that can be processed by the Controller. A JDF processor SHOULD be capable of processing Combined Process Nodes of any of the individual JDFService Elements that are specified. It is therefore not necessary to define every permutation of allowed combinations. It need not be able to process individual Nodes with a type defined in the *Types* Attribute of a *Combined* JDFService Element.

Table P-13: JDFService Element (Section 1 of 2)

Name	Data Type	Description
CombinedMethod ? New in JDF 1.1	enumeration	Specifies how the Processes specified in <i>Types</i> may be specified. Values are: <i>Combined</i> – The list of Processes in <i>Types</i> MUST be specified as a Combined 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 Combined 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. Values include those from: JDF/@ <i>Type</i>
TypeOrder ? New in JDF 1.1	enumeration	Ordering restriction for Combined Process Nodes. Values are: <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.

Table P-13: JDFService Element (Section 2 of 2)

Name	Data Type	Description
<i>Types?</i>	NMTOKENS	If <i>Type = Combined</i> , or <i>Type = 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 3.3.3, Combined Process Nodes. Values include those from: JDF/@ <i>Types</i>

The following is an example of a Response Message to a KnownJDFServices Query Message:

```
<Response ID="M1" refID="Q1" Type="KnownJDFServices">
  <JDFService Type="Rendering" />
  <JDFService Type="Folding" />
  <JDFService Type="Combined" Types="Gathering Stitching"/>
  <JDFService Type="AnyCompaniesNamespace:MyFolding" />
  ...
</Response>
```

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: QueueEntryStatus Message

Object Type	Element name	Description
QueryTypeObj Modified in JDF 1.1A	QueueEntryDefList	Defines the addressed queue entries. Note that this Element was QueueEntryDef * prior to JDF 1.1A.
ResponseTypeObj	QueueEntry *	Describes the status of the queried queue entries.
For the definition of the Elements above see Section 5.14, Elements for Queues.		

The QueueEntryStatus Message returns queue entry descriptions. The QueueEntryDef Elements specify the queue entries to be queried. If no QueueEntryDef Element is specified, the query returns a list of QueueEntry Elements, one for each entry in the queue. If no QueueEntryDef is specified and the query defines a persistent channel, a Signal is emitted for any entry whose status changes. This includes changes as a result of modifications of the queue status, such as hold or resume.

P.2.4.1 Element: QueueEntryDefList

[New in JDF 1.1A](#)

[Deprecated in JDF 1.2](#)

The QueryTypeObj of QueueEntryStatus has been modified from QueueEntryDef * to QueueEntryDefList because of a type collision in the XML Schema. QueueEntryDef had been used both as a QueryTypeObj and as a CommandTypeObj.

Table P-15: 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.

Table P-16: Packing – Input Resources

Name	Description
PackingParams	Necessary information about the Packing Process.
PhysicalResource	All kinds of Physical Resources can be packed.

Table P-17: Packing – 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.

Table P-18: FilmToPlateCopying – Input Resources

Name	Description
DevelopingParams ?	Controls the physical and chemical specifics of the media development Process.
ExposedMedia	The film or films to be copied onto the plate.
Media	The unexposed plate.
PlateCopyParams	The settings of the exposure task.

Table P-19: FilmToPlateCopying – 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*

Table P-20: PreflightAnalysis Resource

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.

P.3.3.1 Element: PreflightDetail

PreflightDetail Subelements are used to describe one property within the **PreflightAnalysis** category in which they occur. This Subelement is also used by **PreflightInventory** Resource.

Table P-21: PreflightDetail Element

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	<p>Values are:</p> <p><i>Error</i> – Value violates the ConstraintValue specified in the associated PreflightConstraint Element. The constraint was flagged as an Error in the profile.</p> <p><i>Warning</i> – Value violates the ConstraintValue specified in the associated PreflightConstraint Element. The constraint was flagged as a Warning in the profile.</p> <p><i>Ignore</i> – The constraint is ignored, and no PreflightDetail or PreflightInstanceDetail Elements are created for this constraint.</p> <p><i>IgnoreValue</i> – No comparison was made against a ConstraintValue. 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.</p>
Value ?	element	Identifies the value of the property. The semantics are PDL-specific.

P.3.3.2 Element: PreflightInstance

PreflightInstance Subelements are used to collect PreflightInstanceDetail Elements for one instance of some object which occurs in the PDL files referenced by a run list. For example, there might be one PreflightInstance

Element for each font that occurs in the pages of a run list. This Subelement is also used by **PreflightInventory** Resources.

Table P-22: PreflightInstance Element

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.
PreflightInstanceDetail * Modified in JDF 1.1	element	A pool of PreflightInstanceDetail Elements that describe the properties for this instance

P.3.3.3 Element: PreflightInstanceDetail

PreflightInstanceDetail Subelements describe one property of one instance of some object type that occurs in a PDL file. For example, several PreflightInstanceDetail Elements might describe the properties of a single font. This Subelement is also used by **PreflightInventory** Resources

Table P-23: PreflightInstanceDetail Element

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. 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*

Table P-24: PreflightInventory Resource

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.
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 for the file. The Elements in this Resource, detailed in the table below, place the results in specific categories. The value for each of these Elements is an array of PreflightConstraint Subelements. Within the PreflightConstraint Resources, the ConstraintValue Element indicates allowable values and the *Status* Attribute indicates the error level (if any) to be flagged when exceptions to the constraints are identified.

Resource Properties

Resource Class: **Parameter**

Resource referenced by: —

Example Partition: —

Input of Processes: *Preflight*

Output of Processes: —

Table P-25: PreflightProfile Resource (Section 1 of 2)

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

Table P-25: PreflightProfile Resource (Section 2 of 2)

Name	Data Type	Description
FileTypeConstraintsPool ?	element	A Preflight constraint. The <i>Type</i> 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 <i>Value</i> 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

P.3.5.1 Element: PreflightConstraint

Table P-26: PreflightConstraint Element

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	Values are: <i>Error</i> – Values that violate the ConstraintValue specified are flagged as Errors in PreflightDetail and PreflightInstanceDetail Elements. <i>Warning</i> – Values that violate the ConstraintValue specified are flagged as Warnings in PreflightDetail and PreflightInstanceDetail Elements. <i>Ignore</i> – The constraint is ignored, and no PreflightDetail or PreflightInstanceDetail Elements are created for this constraint. <i>IgnoreValue</i> – No comparison is made against the ConstraintValue.
ConstraintValue ?	element	Provides a value against which to test occurrences of the characteristic in the file. Note: The semantics of the ConstraintValue Element depend on the PDL characteristic in question.

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.

Table P-27: Proofing – Input Resources

Name	Description
ColorantControl ? Modified in JDF 1.1A	Identifies the color model used by the Job.
ColorSpaceConversionParams ?	This Resource provides information needed to convert colorspaces in the pages for proofing. Generally present if a color proof is desired, unless the pages in the RunList have already been operated on by a previous colorspace conversion process.
Layout ?	REQUIRED if an imposition proof is desired.
Media	This Resource characterizes the output media for the proof.
ProofingParams	This Resource provides the parameters needed to produce the desired proof.
RunList (<i>Document</i>)	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 (<i>Marks</i>) ?	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 .

Table P-28: Proofing – 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.

Table P-29: SoftProofing – Input Resources

Name	Description
ColorantControl ? Modified in JDF 1.1A	Identifies the color model used by the Job.
ColorSpaceConversionParams ?	This Resource provides information needed to convert colorspaces in the pages for proofing. Generally present if a color proof is desired, unless the pages in the RunList have already been operated on by a previous colorspace conversion process.
Layout ?	REQUIRED if an imposition proof is desired.
ProofingParams	Provides the parameters needed to produce the desired proof.
RunList (<i>Document</i>)	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 (<i>Marks</i>) ?	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 .

Table P-30: SoftProofing – Output Resources

Name	Description

P.3.8 IDPrinting

[Deprecated in JDF 1.1](#)

The **IDPrinting** Process was deprecated in JDF/1.1. Instead, implementations SHOULD use a Combined Process with the **DigitalPrinting** Process and other Processes, thus improving interoperability by reducing one of the combinations of Processes. Also the **IDPrinting** Process defined a number of Resources and Subelements which are deprecated since they duplicate other Resources.

IDPrinting, which stands for Integrated Digital Printing, is a specific form of digital printing. It combines functionality that might be represented by the **Interpreting**, **Rendering**, **Screening**, and **DigitalPrinting** Processes in a single Process. In addition, Devices which support **IDPrinting** frequently provide some degree of finishing capabilities, such as collating and stapling, as well as some automated layout capabilities, such as N-up and duplex printing.

Controls for **IDPrinting** are provided in the **IDPrintingParams** Resource. These controls are intended to be somewhat limited in their scope. If greater control over various aspects of the printing Process is needed, **IDPrinting** SHOULD NOT be used. Ultimately, the controls specified for **IDPrinting** can be used to generate an Internet Printing Protocol (IPP) Job. See JDF/1.0 Appendix F for a mapping between JDF **IDPrinting** and IPP. **IDPrinting** may be combined with other Processes, such as **Trapping** or **ColorSpaceConversion**.

Table P-31: IDPrinting – Input Resources (Section 1 of 2)

Name	Description
ColorantControl ?	The ColorantControl Resources that define the ordering and usage of inks in print modules.
Component (<i>Cover</i>) ?	A finished cover may be combined with the pages that will be output by this Process.

Table P-31: IDPrinting – Input Resources (Section 2 of 2)

Name	Description
Component (<i>Input</i>) ?	Various components can be used in IDPrinting instead of Media . Examples include waste, precut Media , or a set of preprinted Sheets or webs.
Component (<i>Proof</i>) ?	A Proof component is used if a proof was produced during an earlier ConventionalPrinting Process.
ExposedMedia ?	A Proof is useful for comparisons (completeness, color accuracy) with the print out of the IDPrinting Process.
FontPolicy ?	Describes the behavior of the font machinery in absence of requested fonts.
Ink ?	Ink or toner and information about it is needed for IDPrinting .
InterpretingParams *	A set of Resources that specify how the Device SHOULD interpret the PDL files which are referenced by the RunList for the Process. Note that InterpretingParams is an Abstract Resource. Instances are PDL-specific.
IDPrintingParams ?	Specific parameters to set up the machinery.
Media ?	The physical Media and information about the Media , such as thickness, type, and size, are used to set up paper travel in the press. This has to be present if no preprinted Component (<i>Input</i>) 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.

Table P-32: IDPrinting – Output Resources

Name	Description
Component (<i>Good</i>)	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 (<i>Waste</i>) ?	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**.

Table P-33: Dividing – Input Resources

Name	Description
Component	The Dividing Process consumes one Component : the Web(s) or ribbon(s) entering the crosscutting machinery. The substrate might have been treated with LongitudinalRibbonOperations and may be folded with a former fold.
DividingParams	Specific parameters to set up the machinery.

Table P-34: Dividing – 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 Process Node with **ConventionalPrinting**).

Table P-35: LongitudinalRibbonOperations – Input Resources

Name	Description
Component	The Component can consist of more than one Web or ribbon that has been combined with the Gathering Process.
LongitudinalRibbonOperationParams	Specific parameters to set up the machinery tools for the LongitudinalRibbonOperations Process.

Table P-36: LongitudinalRibbonOperations – Output Resources

Name	Description
Component +	A ribbon is produced that is used in other postpress Processes. If the LongitudinalRibbonOperations Process was slitting, more than one Component is produced.

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.

Table P-37: SaddleStitching – Input Resources

Name	Description
Component	The only REQUIRED Component is the collected pile.
SaddleStitchingParams	Specific parameters to set up the machinery.

Table P-38: SaddleStitching – 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*.

Table P-39: SideSewing – Input Resources

Name	Description
Component	The only REQUIRED Component is the gathered Sheets.
SideSewingParams	Specific parameters to set up the machinery.

Table P-40: SideSewing – 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.

P.4.1.1 Element: AdhesiveBinding[Deprecated in JDF 1.1](#)**Table P-41: AdhesiveBinding Element**

Name	Data Type	Description
<i>Scoring?</i>	EnumerationSpan	Scoring option for AdhesiveBinding . Values are: <i>TwiceScored</i> <i>QuadScored</i> <i>None</i> Note: Values are based on viewing the cover in its flat pre-binding state.
<i>SpineGlue?</i>	EnumerationSpan	Glue type used to define AdhesiveBinding procedures. 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).

P.4.1.2 Element: BookCase[Deprecated in JDF 1.1](#)

This Subelements contains details of the book case for hard-cover book binding. The actual binding parameters are set in the appropriate AdhesiveBinding, ThreadSewing, or ThreadSealing Elements.

Table P-42: BookCase Element

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. 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].

P4.2.1 Element: Pricing[Deprecated in JDF 1.3](#)**Table P-43: Pricing Element**

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 = "true"</i>	boolean	Specifies whether the line item defined by this quote has a price. If <i>false</i> , the line item is not included in the parent quote, and the price is unknown and MUST be added. If <i>true</i> , 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 = false</i>). The sum of line items need not be identical to the parent quote.

P4.2.2 Element: Payment[New in JDF 1.1](#)[Deprecated in JDF 1.3](#)**Table P-44: Payment Element**

Name	Data Type	Description
<i>PayTerm?</i>	telem	Describes the payment terms & conditions.
<i>CreditCard?</i>	element	Specifies credit card information

P4.2.3 Element: CreditCard[New in JDF 1.1](#)[Deprecated in JDF 1.3](#)**Table P-45: CreditCard Element (Section 1 of 2)**

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.

Table P-45: CreditCard Element (Section 2 of 2)

Name	Data Type	Description
<i>Type</i>	NMTOKEN	Credit card brand. 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 Intent Node
Output of Processes:	—

Table P-46: SizeIntent Resource

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. 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:** —

Table P-47: AdhesiveBindingParams Resource

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.

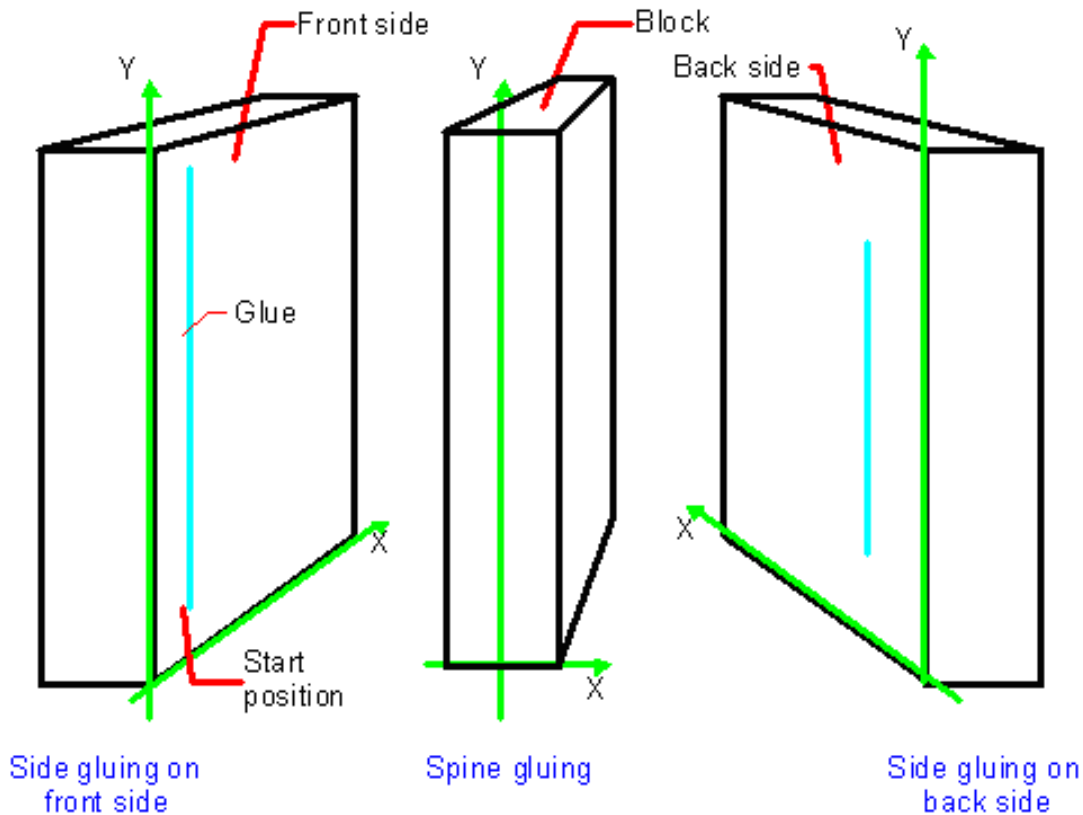


Figure P-1: Parameters and coordinate system for glue application

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:	<i>RibbonName, SheetName, SignatureName, WebName</i>
Input of Processes:	<i>Dividing</i>
Output of Processes:	—

Table P-48: DividingParams Resource

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:	<i>DocIndex, DocRunIndex, DocSheetIndex, PartVersion, Run, RunIndex, RunTags, SheetIndex, SheetName, Side</i>
Input of Processes:	<i>IDPrinting</i>
Output of Processes:	—

Table P-49: IDPrintingParams Resource (Section 1 of 2)

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.
<i>OutputBin</i> ?	NMTOKEN	Specifies the bin to which the finished document is to be output. Values include those from: Table P-50, "OutputBin Attribute Values," on page 1027

Table P-49: IDPrintingParams Resource (Section 2 of 2)

Name	Data Type	Description
<i>PageDelivery</i> ?	enumeration	Indicates how pages are to be delivered to the output bin or finisher. Values are: <i>SameOrderFaceUp</i> – Order as defined by the RunList, with the “front” sides of the media up. <i>SameOrderFaceDown</i> – Order as defined by the RunList, with the “front” sides of the media up. <i>ReverseOrderFaceUp</i> – Order reversed, as defined by the RunList, with the “front” sides of the media up. <i>ReverseOrderFaceDown</i> – Order reversed, as defined by the RunList, with the “front” sides of the media down.
<i>PrintQuality</i> ?	enumeration	Indicates how pages are to be delivered to the output bin or finisher. 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>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.

— Attribute: OutputBin

Table P-50: OutputBin Attribute Values (Section 1 of 2)

Value	Description
<i>Top</i>	The bin that, when facing the Device, can best be identified as “top”.
<i>Middle</i>	The bin that, when facing the Device, can best be identified as “middle”.
<i>Bottom</i>	The bin that, when facing the Device, can best be identified as “bottom”.

Table P-50: OutputBin Attribute Values (Section 2 of 2)

Value	Description
<i>Side</i>	The bin that, when facing the Device, can best be identified as “side”.
<i>Left</i>	The bin that, when facing the Device, can best be identified as “left”.
<i>Right</i>	The bin that, when facing the Device, can best be identified as “right”.
<i>Center</i>	The bin that, when facing the Device, can best be identified as “center”.
<i>Rear</i>	The bin that, when facing the Device, can best be identified as “rear”.
<i>FaceUp</i>	The bin that can best be identified as “face up” with respect to the Device.
<i>FaceDown</i>	The bin that can best be identified as “face down” with respect to the Device.
<i>FitMedia</i>	Requests the Device to select a bin based on the size of the media.
<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.
<i>Mailbox-N</i>	The Job will be output to the bin that is best identified as “Mailbox-1”, “Mailbox-2”...etc.
<i>Stacker-N</i>	The Job will be output to the bin that is best identified as “Stacker-1”, “Stacker-2” ...etc.
<i>Tray-N</i>	The Job will be output to the tray that is best identified as “Tray-1”, “Tray-2” ... etc.

P4.6.1 Element: Cover[Deprecated in JDF 1.1](#)

This Element describes the cover requested for the Job. Covers may be applied to the whole Job, or to each Instance Document in the Job. Note that front and back covers may be specified.

Table P-51: Cover Element

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. Values are: <i>Front</i> – The front cover. <i>Back</i> – The back cover.
<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 <i>IDPrinting</i> 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.

P4.6.2 Element: IDPFinishing

[Deprecated in JDF 1.1](#)

IDPFinishing Elements describe finishing operations that are to be applied to sets of Sheets that are output by the *IDPrinting* Process. The finishings are applied to the entire Job when there are no Instance Documents. Otherwise, each Instance Document is finished separately. Operation-specific Subelements may also be present when a Device provides controls for a finishing operation. Additional Subelements are expected to be defined over time. Also, more detail will be added to the currently defined Elements

Table P-52: IDPFinishing Element

Name	Data Type	Description
<i>Finishings</i> ?	IntegerList	A set of finishing operations to apply to the Job. The operations are encoded as an enumeration. Values include those from: Table P-53, “Finishings Attribute Values”
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.

— Attribute: Finishings

Table P-53: Finishings Attribute Values (Section 1 of 2)

Value	Description
3	(none) Perform no finishing
4	(staple) Bind the document(s) with one or more staples. The exact number and placement of the staples is site-defined.
5	(punch) This value indicates that holes are REQUIRED in the finished document. The exact number and placement of the holes is site-defined. The punch specification may be satisfied (in a site- and implementation-specific manner) either by drilling/punching, or by substituting predrilled media.
6	(cover) This value is specified when it is desired to select a non-printed (or preprinted) cover for the document. This does not supplant the specification of a printed cover (on cover stock medium) by the document itself.
7	(bind) This value indicates that a binding is to be applied to the document; the type and placement of the binding is site-defined.
8	(saddle-stitch) Bind the document(s) with one or more staples (wire stitches) along the middle fold. The exact number and placement of the staples and the middle fold is implementation and/or site-defined.
9	(edge-stitch) Bind the document(s) with one or more staples (wire stitches) along one edge. The exact number and placement of the staples is implementation and/or site-defined.
10	(fold) Fold the document(s) with one or more folds. The exact number and orientations of the folds is implementation and/or site-defined.
11	(trim) Trim the document(s) on one or more edges. The exact number of edges and the amount to be trimmed is implementation and/or site-defined.
12	(bale) Bale the document(s). The type of baling is implementation and/or site-defined.
13	(booklet-maker) Deliver the document(s) to the Signature booklet maker. This value is a short cut for specifying a Job that is to be folded, trimmed and then saddle-stitched.

Table P-53: Finishings Attribute Values (Section 2 of 2)

Value	Description
14	(jog-offset) Shift each copy of an output document from the previous copy by a small amount which is Device dependent. This value has no effect on the "Job-Sheet." This value SHOULD NOT have an effect if each copy of the Job consists of one Sheet.
50	(bind-left) Bind the document(s) along the left edge. The type of the binding is site-defined.
51	(bind-top) Bind the document(s) along the top edge. The type of the binding is site-defined.
52	(bind-right) Bind the document(s) along the right edge. The type of the binding is site-defined.
53	(bind-bottom) Bind the document(s) along the bottom edge. The type of the binding is site-defined.

P.4.6.3 Element: IDPFolding[Deprecated in JDF 1.1](#)

This Element describes the folding requested for a set of pages in the document.

Table P-54: IDPFolding Element

Name	Data Type	Description
FoldingParams ?	Refelement	Describes the details of how to fold the media.

P.4.6.4 Element: IDPHoleMaking[Deprecated in JDF 1.1](#)

This Element describes the hole making requested for a set of pages in the document.

Table P-55: IDPHoleMaking Element

Name	Data Type	Description
HoleMakingParams ?	refelement	Describes the details of the holes to be punched into the Media.

P.4.6.5 Element: IDPLayout[Deprecated in JDF 1.1](#)**Table P-56: IDPLayout Element (Section 1 of 3)**

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. 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.
<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.

Table P-56: IDPLayout Element (Section 2 of 3)

Name	Data Type	Description
ImageShift ?	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 ImageShift, and all contents for each surface are shifted the same amount.
<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	<p>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.</p> <p>Values are:</p> <p><i>ToBottomToRight</i> – Pages are imaged in successive columns, from left to right, starting at the top of each column.</p> <p><i>ToBottomToLeft</i> – Pages are imaged in successive columns, from right to left, starting at the top of each column.</p> <p><i>ToTopToRight</i> – Pages are imaged in successive columns, from left to right, starting at the bottom of each column.</p> <p><i>ToTopToLeft</i> – Pages are imaged in successive columns, from right to left, starting at the bottom of each column.</p> <p><i>ToRightToBottom</i> – Pages are imaged in successive rows, from top to bottom, starting at the left of each row.</p> <p><i>ToRightToTop</i> – Pages are imaged in successive rows, from bottom to top, starting at the left of each row.</p> <p><i>ToLeftToBottom</i> – Pages are imaged in successive rows, from top to bottom, starting at the right of each row.</p> <p><i>ToLeftToTop</i> – Pages are imaged in successive rows, from bottom to top, starting at the right of each row.</p>
<i>Rotate</i> = "0"	number	<p>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.</p> <p>Note: Text will be reflowed in cases where the PDL for the page allows reflow by the Device.</p> <p>Default = "0"</p>

Table P-56: IDPLayout Element (Section 3 of 3)

Name	Data Type	Description
<i>Sides = "OneSided"</i>	enumeration	<p>Indicates how pages are to be imposed onto sides of the medium.</p> <p>Values are:</p> <p><i>OneSided</i> – Page contents will only be imaged on one side of the media. The default.</p> <p><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 “<i>work-and-turn</i>”.</p> <p><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 “<i>work-and-tumble</i>”.</p>

P.4.6.6 Element: IDPStitching[Deprecated in JDF 1.1](#)

This Element describes the stitching requested for a set of pages in the document

Table P-57: IDPStitching Element

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.</p> <p>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>.</p> <p>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>

P.4.6.7 Element: IDPTrimming[Deprecated in JDF 1.1](#)

This Element describes the trimming requested for a set of pages in the document.

Table P-58: IDPTrimming Element

Name	Data Type	Description
TrimmingParams ?	refelement	Describes the details of how to trim the media.

P.4.6.8 Element: ImageShift[Deprecated in JDF 1.1](#)

ImageShift Elements describe how finished page contents will be imaged onto media. All Attributes refer to positioning along the “X” or “Y” axis. The “X” dimension is the first number of the Media *Dimension* Attribute; “Y” is the second number

Table P-59: ImageShift Element

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.

P.4.6.9 Element: JobSheet[Deprecated in JDF 1.1](#)

This Element describes a Job Sheet which may be produced along with the Job. Job Sheets include separators, Sheets, and error Sheets. The information provided on the Sheet depends on the type of Sheet. In addition, any Sheet type

may include an optional Message as a comment Subelement for the Sheet Element. Such a Message comment MUST have a *Name* Attribute with the value 'SheetMessage'.

Table P-60: JobSheet Element

Name	Data Type	Description
<i>SheetFormat</i> = "Standard"	NMTOKEN	Identifies the format of the <i>JobSheet</i> . One one value is defined here, but site-specific values may be defined. Values include: <i>Standard</i>
<i>SheetOccurrence</i>	enumeration	Indicates when the Sheet is to be produced and inserted into the set of output pages. 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. 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 3.10.5, *Description of Partitioned Resources*. Overrides MAY apply to a set of Instance Documents, set of copies of Instance Documents, or to a set of finished pages, output surfaces, Sheets of media in a personalized printing Job, or header or trailer insert Sheets added by a RunList. Note: If more than one override refers to the same content, the lowest level override takes precedence. The following list defines Partitioning precedence, from lowest to highest, (i.e., the lower entries in the list take precedence):

Job level Partitioning (*lowest priority*):

PartVersion, Run, SheetName, Side, RunTags

Page level Partitioning:

RunIndex

SheetIndex

Instance Document level Partitioning (*highest priority*):

DocCopies

DocIndex

DocSheetIndex

DocRunIndex

Note: It is strongly discouraged to mix page-level Partitions and Instance Document-level Partitions. Cover Elements in **IDPrintingParams** are counted when calculating *DocSheetIndex* or *DocRunIndex*.

Example of a Partitioned IDPrinting Node

The following example shows how Partitioning can be used to describe a fairly complex example. Three color models (**ColorantControl** Partitions) are applied to a set of Sheets using the *DocSheetIndex* key;

- 1 DeviceN:*DocSheetIndex* = "0" defines the cover;
- 2 DeviceCMYK *DocSheetIndex* = "1" defines the first Sheet (non cover);
- 3 DeviceGray:*DocSheetIndex* = "2 ~ -1" defines all other Sheets;

The cover is selected from a different input tray using the *Location* key. The same key is used to describe the **Media** in each tray.

```
<?xml version='1.0' encoding='utf-8' ?>
<JDF ID="HDM20010402140111" Type="IDPrinting" JobID="HDM20010402140111"
Status="Waiting" Version="1.2">
  <ResourcePool>
    <Media ID="Link0003" Class="Consumable" Locked="false" Status="Available"
Dimension="700 900" MediaType="Paper" PartIDKeys="Location">
      <Media Weight="90" Location="Tray 1"/>
      <Media Weight="120" Location="Tray 2"/>
    </Media>
    <RunList ID="Link0004" Class="Parameter" Locked="false" Status="Available"
PartIDKeys="Run">
      <RunList Run="Run0005" Pages="0">
        <LayoutElement>
          <FileSpec URL="Cover.pdf"/>
        </LayoutElement>
      </RunList>
      <RunList Run="Run0006" Pages="0 ~ 7">
        <LayoutElement>
          <FileSpec URL="File2.pdf"/>
        </LayoutElement>
      </RunList>
    </RunList>
    <IDPrintingParams ID="Link0008" Class="Parameter" Locked="false"
Status="Available">
      <IDPLayout NumberUp="2 2"/>
    </IDPrintingParams>
  </ResourcePool>
</JDF>
```

```

    <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.

P.4.7.1 Element: Signature

[Deprecated in JDF 1.3](#)

This Element groups individual **Sheet** Resources into one Signature Subelement. In JDF 1.3 and beyond, Signature is represented as a Partition of **Layout** with **Layout/@PartIDKeysSignatureName** set.

Table P-61: Signature Element

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: *RibbonName, SheetName, SignatureName, WebName*
Input of Processes: *LongitudinalRibbonOperations*
Output of Processes: —

Table P-62: LongitudinalRibbonOperationParams Resource

Name	Data Type	Description
<i>LROperation</i> +	element	Abstract Element which is a placeholder for a longitudinal ribbon operation.

P.4.8.1 Element: 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

Table P-63: LROperation Element

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.

P.4.8.2 Element: 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.

P.4.8.3 Element: LongGlue[Deprecated in JDF 1.1.](#)

LongGlue is derived from the Abstract Element LROperation and describes a longitudinal gluing operation and has the following contents in addition to those of LROperation.

Table P-64: LongGlue Element

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 listed values can be used: Values are: <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> .

P.4.8.4 Element: LongPerforate

[Deprecated in JDF 1.1.](#)

LongPerforate is derived from the Abstract Element LROperation and describes a longitudinal gluing operation and has the following contents in addition to those of LROperation.

Table P-65: LongPerforate Element

Name	Data Type	Description
<i>TeethPerDimension ?</i>	integer	If <i>Operation = 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).

P.4.8.5 Element: 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:	<i>DigitalPrinting, IDPrinting</i>
Output of Processes:	—

Table P-66: MediaSource Resource

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 3.10.6.4, Locations of Physical Resources) there SHOULD be a match between one <i>Location</i> Partition Key and this <i>MediaLocation</i> value.
<i>ManualFeed = "false"</i>	boolean	Indicates whether the media will be fed manually. Default = "false"
Media ?	reference	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:	<i>Packing</i>
Output of Processes:	—

Table P-67: PackingParams Resource

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. Values are:: <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. Values are: <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:	—

Table P-68: PlateCopyParams Resource

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.
<i>Diffusion ?</i>	enumeration	The diffusion foil setting. 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:	—

Table P-69: ProofingParams Resource (Section 1 of 2)

Name	Data Type	Description
<i>ColorType ?</i>	enumeration	Color quality of the proof. 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>

Table P-69: ProofingParams Resource (Section 2 of 2)

Name	Data Type	Description
<i>ImageViewingStrategy</i> = "NoImages"	string	Identifies which images will be displayed during the SoftProofing Process. 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</i> = "false" New in JDF 1.1	boolean	Indicates whether the media will be fed manually. Default = "false"
<i>ProofRenderingIntent</i> = "Perceptual" New in JDF 1.1	enumeration	Identifies the rendering intents associated with the proof. Values are ICC-defined rendering intent values: Values are: <i>Saturation</i> <i>Perceptual</i> – The default. <i>RelativeColorimetric</i> <i>AbsoluteColorimetric</i>
<i>ProofType</i> = "None"	enumeration	Describes the type of the proof. 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 FileSpec MUST be " <i>ProoferProfile</i> ".
<i>Media</i> ?	reference	Describes the media to be used.

P.4.13 SaddleStitchingParams

This Resource provides the parameters of the **SaddleStitching** Process.

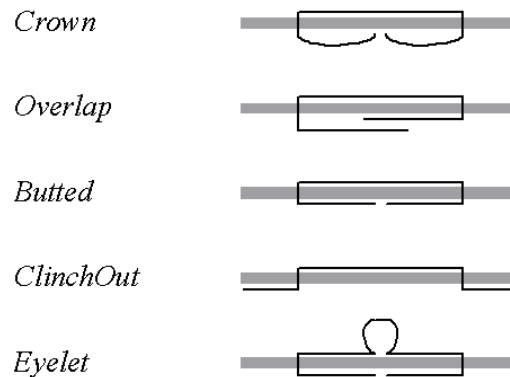
[Deprecated in JDF 1.1](#)

Resource Properties

Resource Class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of Processes:	<i>SaddleStitching</i>
Output of Processes:	—

Table P-70: SaddleStitchingParams Resource

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. Values are: <i>Crown</i> <i>Overlap</i> <i>Butted</i> <i>ClinchOut</i> <i>Eyelet</i> Note: 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: *SheetName.*

Input of Processes: —

Output of Processes: —

Table P-71: Sheet Resource

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.
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 (<i>Front</i>) ?	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 (<i>Back</i>) ?	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).

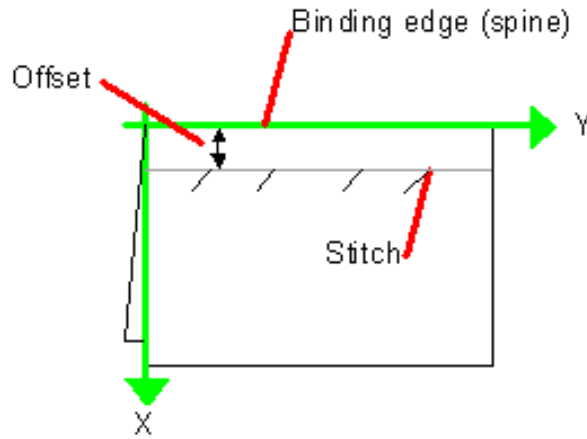


Figure P-3: Parameters and coordinate system used for side sewing

Resource Properties

- Resource Class: Parameter
- Resource referenced by: —
- Example Partition: —
- Input of Processes: *SideSewing*
- Output of Processes: —

Table P-72: SideSewingParams Resource

Name	Data Type	Description
<i>NumberOfNeedles</i>	integer	Specifies the number of needles to be used.
<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. Values are: <i>Normal</i> <i>Staggered</i> <i>CombinedStaggered</i>
<i>ThreadMaterial?</i>	enumeration	Specifies the thread material to be used. 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: —

Table P-73: Surface Resource

Name	Data Type	Description
<i>Side</i>	enumeration	The side of the Sheet that the Surface describes. 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.	12
Figure 2-1 Example of JDF and JMF workflow interactions	22
Figure 2-2 JDF tree structure	23
Figure 2-3 Example of a hierarchical tree structure of JDF Nodes	25
Figure 2-4 Example of a Process chain linked by Input Resources and Output Resources . . .	26
Figure 2-5 Standard coordinate system	27
Figure 2-6 Relation between Resource and process coordinate systems	28
Figure 2-7 Layout of simple saddle stitched brochure (product example)	31
Figure 2-8 Equation for Surface Coordinate System Transformations	31
Figure 2-9 Surface coordinate system	32
Figure 2-10 Press coordinate system used for Sheet-Fed Printing	32
Figure 2-11 Press coordinate system used for Web Printing	33
Figure 2-12 Coordinate systems after Folding (product example)	33
Figure 2-13 Coordinate systems after Collecting (product example)	34
Figure 2-14 Examples of Transformations and Coordinate Systems in JDF.	35
Figure 2-15 Transforming a point (example)	37
Figure 3-1 Any-Element (generic content) – a diagram of its structure	45
Figure 3-2 JDF Node – a Diagram of its Structure	53
Figure 3-3 Job hierarchy with Process, Process Group and Product Intent Nodes	54
Figure 3-4 Combined Process Node dependencies	59
Figure 3-5 Abstract Resource Element – a diagram of its structure	68
Figure 3-6 Nodes linked by a Resource	73
Figure 3-7 ResourceLink Elements and ResourceRef Elements	74
Figure 3-8 Abstract ResourceLink Element – a diagram of its structure	76
Figure 3-9 Amount handling	91
Figure 3-10 Workflow for splitting shared Input Resources	119
Figure 3-11 Workflow for combining shared Output Resources	120
Figure 3-12 Workflow for splitting independent Input Resources	120
Figure 3-13 Workflow for combining independent Output Resources	120
Figure 3-14 Abstract Audit Element – a diagram of its structure	122
Figure 4-1 Simplified PrintTalk workflow (negotiation phase)	144
Figure 4-2 Life Cycle of a JDF Node	147
Figure 4-3 Example of a simple Process chain linked by Resources	149
Figure 4-4 Example of a Pipe Resource linking two Processes	152
Figure 4-5 Example of status transitions in case of overlapping Processing	153
Figure 4-6 The spawning and merging mechanism and its phases	157
Figure 4-7 JDF Node structure that requires Resource copying during spawning and merging . .	159
Figure 4-8 Example for a JDF Node structure with nested spawning	161
Figure 4-9 Example of the spawning and merging of independent Jobs	162
Figure 4-10 Parameter space in Device capabilities	165
Figure 5-1 JMF Root Element – a diagram of its structure	171
Figure 5-2 Interaction of Messages with a Subscription	175
Figure 5-3 Interaction of Command and Acknowledge Messages	182
Figure 5-4 Example of Reliable Signaling	184
Figure 5-5 Without UpdateJDF Message	237
Figure 5-6 With UpdateJDF Message	238

Figure 5-7 Mechanism of a PipePull Message	242
Figure 5-8 Mechanism of a PipePush Message	244
Figure 5-9 JMF QueueEntry Status Transition Diagram	247
Figure 5-10 Effects of the global queue Messages on the queue Status	257
Figure 6-1 Imposition for Cut and Stack	294
Figure 6-2 Worst case scenario for area coverage calculation	301
Figure 6-3 Overview of Web Printing	310
Figure 6-4 Bundle Creation	316
Figure 6-5 Bundle Transport	316
Figure 6-6 Combined Process with Feeding Process	322
Figure 6-7 Input Components	322
Figure 6-8 Output Component	322
Figure 6-9 Packaging Process Coordinate System	337
Figure 7-1 Structure of a normal hardcover book	359
Figure 7-2 Structure of a padded hardcover book	359
Figure 7-3 Structure of a book with GlueProcedure = "SideOnly" (Layflat)	362
Figure 7-4 WebCellAlignment, Example 1	411
Figure 7-5 WebCellAlignment Example 2	412
Figure 7-6 WebCellAlignment Example 3	413
Figure 7-7 Folding examples for some values of BoxFoldAction/@Action	420
Figure 7-8 BoxFoldingType Attribute for values of Type00, Type01 and Type02	420
Figure 7-9 BoxFoldingType Attribute for values of Type03, Type04 and Type10	421
Figure 7-10 BoxFoldingType Attribute for values of Type 11, Type12 and Type13	421
Figure 7-11 BoxFoldingType Attribute for values of Type15 and Type20	422
Figure 7-12 Box packing	423
Figure 7-13 BundlingParams Coordinate System	427
Figure 7-14 CaseMakingParams	430
Figure 7-15 Parameters and coordinate system for CasingIn	431
Figure 7-16 Parameters used for channel binding	432
Figure 7-17 Coordinate systems used for collecting	435
Figure 7-18 Component – terms and definitions	467
Figure 7-19 Parameters and coordinate system for cover application	483
Figure 7-20 Definition of the PlatePosition Attribute on a newspaper-Web Press	491
Figure 7-21 Example of a single physical section of eight pages	491
Figure 7-22 Basic Shape for RepeatDesc/@LayoutStyle Examples	509
Figure 7-23 RepeatDesc/@LayoutStyle = "StraightNest"	509
Figure 7-24 RepeatDesc/@LayoutStyle = "Reverse2ndRow"	510
Figure 7-25 RepeatDesc/@LayoutStyle = "Reverse2ndRowAligned"	510
Figure 7-26 RepeatDesc/@LayoutStyle = "Reverse2ndColumn"	511
Figure 7-27 RepeatDesc/@LayoutStyle = "Reverse2ndColumnAligned"	511
Figure 7-28 RepeatDesc/@GutterX2 and @GutterY2: Secondary Gutters	512
Figure 7-29 Parameters and coordinate system used for end-Sheet gluing	525
Figure 7-30 Names of the reference edges of a Sheet in the FoldingParams Resource	540
Figure 7-31 Fold catalog part 1	542
Figure 7-32 Fold catalog part 2	543
Figure 7-33 Coordinate system used for Gathering	548
Figure 7-34 Parameters and coordinate system for glue application	550
Figure 7-35 Hole line parameters	554
Figure 7-36 Line hole punching for multiple webs	554
Figure 7-37 Parameters and coordinate system used for Inserting	578

Figure 7-38 Parameters and coordinate system for jacketing	589
Figure 7-39 Diagram of a 4-up cross-folded saddle-stitch imposition with vertical gutter creep ..	634
Figure 7-40 Diagram of a step-and-repeat 2-up saddle-stitch imposition with vertical spine gutter creep	634
Figure 7-41 a paper Roll with some roll-specific information	648
Figure 7-42 Relief and Floor Thickness for a Flexo Plate or Flexo Sleeve	648
Figure 7-43 Types of Interlocks for Flexo Sleeve	649
Figure 7-44 TabSetCollationOrder Attribute Values	651
Figure 7-45 Diagram of a Single Bank of Tabs	652
Figure 7-46 Inside Loss, Outside Gain	653
Figure 7-47 PRGroup – a diagram of its structure	681
Figure 7-48 PrintRollingParams Coordinate System	693
Figure 7-49 RegisterRibbon lengths and coordinate system for BlockPreparation	706
Figure 7-50 ShapeTemplate Example 1	735
Figure 7-51 ShapeTemplate Example 2	735
Figure 7-52 ShapeTemplate Example 3	736
Figure 7-53 Parameters and coordinate systems for the SpinePreparation Process	738
Figure 7-54 Parameters and coordinate system for the SpineTaping Process	739
Figure 7-55 Stacking Layers	740
Figure 7-56 Pile Patterns	740
Figure 7-57 Odd count handling for a Bundle	741
Figure 7-58 Odd count handling for a Layer	741
Figure 7-59 Staple shapes	743
Figure 7-60 Parameters and coordinate system used for saddle stitching	744
Figure 7-61 Parameters and coordinate system used for Stitching	744
Figure 7-62 Stitching Coordinate System for StitchOrigin Values	745
Figure 7-63 Strapped Bundle	748
Figure 7-64 Strapped Bundle with Sub-bundles	748
Figure 7-65 Shingling for Stripping	751
Figure 7-66 Shingling for Stripping – Details	751
Figure 7-67 RelativeBox including margins	752
Figure 7-68 Definition of margins in StripCellParams	755
Figure 7-69 Parameters and coordinate system used for thread sewing	759
Figure 7-70 Parameters and coordinate system used for side sewing	760
Figure 7-71 Parameters and coordinate system used for trimming	770
Figure 7-72 DeviceCap – a diagram of its structure	777
Figure 7-73 Abstract State Element – a diagram of its structure	788
Figure 7-74 macro Element – a diagram of its structure	804
Figure 7-75 Abstract Term Element – a diagram of its structure	808
Figure 7-76 Abstract Evaluation Element – a diagram of its structure	811
Figure 8-1 Example of Exchange of Certificates	850
Figure P-1 Parameters and coordinate system for glue application	1025
Figure P-2 Staple shapes	1043
Figure P-3 Parameters and coordinate system used for side sewing	1045

Appendix R List of Tables

Table 1-1 Basic references	1
Table 1-2 Callouts	4
Table 1-3 Cardinality symbols	4
Table 1-4 Glossary	5
Table 1-5 Conformance terminology	10
Table 1-6 JDF data types	13
Table 1-7 Units used in JDF	16
Table 2-1 Information contained in JDF Nodes, arranged numerically	23
Table 2-2 Information contained in JDF Nodes, arranged by group	24
Table 2-3 Data types for specifying coordinates and transformation	28
Table 2-4 Matrices and Orientation values for describing the orientation of a Component	30
Table 2-5 JDF Processes used for the production of the simple brochure	31
Table 3-1 Any Element (generic content)	40
Table 3-2 Comment Element	42
Table 3-3 GeneralID Element.....	44
Table 3-4 Definition of “Scope” Terms used in Table 3-5 on page 47	46
Table 3-5 JDF Node	47
Table 3-6 Status Attribute Values	52
Table 3-7 AncestorPool Element	60
Table 3-8 Ancestor Element	61
Table 3-9 ResourcePool Element	62
Table 3-10 Abstract Resource Element	63
Table 3-11 SourceResource Element	67
Table 3-12 Abstract Parameter Resource Element	69
Table 3-13 Abstract Physical Resource Element	70
Table 3-14 Location Element	71
Table 3-15 ResourceLinkPool Element	73
Table 3-16 Abstract ResourceLink Element	77
Table 3-17 ProcessUsage Attribute Values	79
Table 3-18 AmountPool Element	80
Table 3-19 PartAmount Element	81
Table 3-20 Abstract ImplementationLink or //AmountPool/PartAmount Element	82
Table 3-21 Abstract PhysicalLink or //AmountPool/PartAmount Element	83
Table 3-22 Lot Element	85
Table 3-23 Abstract ResourceElement	87
Table 3-24 Abstract ResourceRef Element	87
Table 3-25 Example of actual amount and amount handling	92
Table 3-26 Identical Element	99
Table 3-27 Partitionable Resource Element	103
Table 3-28 Part Element	104
Table 3-29 Condition Attribute Values	113
Table 3-30 PartUsage Attribute examples	117
Table 3-31 AuditPool Element	121
Table 3-32 Abstract Audit Element	123
Table 3-33 List of Audit Elements	123
Table 3-34 Created Audit Element	124
Table 3-35 Deleted Audit Element	124

Table 3-36 Merged Audit Element	124
Table 3-37 Modified Audit Element	125
Table 3-38 Notification Audit Element	126
Table 3-39 PhaseTime Audit Element	128
Table 3-40 ModulePhase Element	129
Table 3-41 ProcessRun Audit Element	130
Table 3-42 ResourceAudit Audit Element	131
Table 3-43 Spawned Audit Element	134
Table 3-44 Excerpt from TrappingParams	137
Table 4-1 Business objects as defined by PrintTalk	142
Table 4-2 Examples of Resource and Process states in the case of simple Process routing .	149
Table 4-3 Examples of Partitioning across multiple Resources	150
Table 4-4 Actions generated when a dynamic-pipe buffer passes various levels	154
Table 5-1 JMF Element	168
Table 5-2 Abstract Message Element	169
Table 5-3 List of JMF Messages	172
Table 5-4 Query Message Element	175
Table 5-5 Response Message Element	176
Table 5-6 Signal Message Element	178
Table 5-7 Trigger Element	179
Table 5-8 ChangedPath Element	179
Table 5-9 Command Message Element	180
Table 5-10 Acknowledge Message Element	182
Table 5-11 Registration Message Element	183
Table 5-12 Subscription Element	185
Table 5-13 ObservationTarget Element	186
Table 5-14 Template for Message tables	188
Table 5-15 Messages for events and capabilities	189
Table 5-16 Events Message	190
Table 5-17 NotificationFilter Element	191
Table 5-18 NotificationDef Element	192
Table 5-19 KnownControllers Message	192
Table 5-20 ControllerFilter Element	193
Table 5-21 JDFController Element	193
Table 5-22 KnownDevices Message	194
Table 5-23 DeviceFilter Element	195
Table 5-24 DeviceList Element	195
Table 5-25 KnownMessages Message	196
Table 5-26 KnownMsgQuParams Element	196
Table 5-27 MessageService Element	196
Table 5-28 KnownSubscriptions Message	198
Table 5-29 SubscriptionFilter Element	198
Table 5-30 SubscriptionInfo Element	199
Table 5-31 Notification Signal	200
Table 5-32 RepeatMessages Message	200
Table 5-33 MsgFilter Element	200
Table 5-34 RequestForAuthentication Command Message	202
Table 5-35 AuthenticationCmdParams Element	203
Table 5-36 Certificate Element.....	204
Table 5-37 AuthenticationResp Element.....	204

Table 5-38 RequestForAuthentication Query Message	204
Table 5-39 AuthenticationCmdParams Element	205
Table 5-40 StopPersistentChannel Message	206
Table 5-41 StopPersChParams Element	206
Table 5-42 Messages to query/affect a Job, Device or Controller	207
Table 5-43 FlushResources Command.....	208
Table 5-44 FlushResources Command.....	208
Table 5-45 FlushResourceParams Element	209
Table 5-46 FlushedResources Element	209
Table 5-47 ModifyNode Command	209
Table 5-48 ModifyNode Signal.....	209
Table 5-49 ModifyNodeCmdParams Element	210
Table 5-50 NewComment Element	210
Table 5-51 NewJDF Query Message	211
Table 5-52 NewJDFQuParams Element	211
Table 5-53 NewJDF Command Message	211
Table 5-54 NewJDFCmdParams Element	211
Table 5-55 IDInfo Element	212
Table 5-56 Occupation Message	212
Table 5-57 EmployeeDef Element	212
Table 5-58 Occupation Element	213
Table 5-59 Resource Query Message	214
Table 5-60 ResourceQuParams Element	214
Table 5-61 Resource Command Message	218
Table 5-62 ResourceCmdParams Element	219
Table 5-63 ResourceInfo Element	221
Table 5-64 ResourcePull Message	225
Table 5-65 ResourcePullParams Element	226
Table 5-66 ShutDown Message	227
Table 5-67 ShutDownCmdParams Element	228
Table 5-68 Status Message	228
Table 5-69 StatusQuParams Element	229
Table 5-70 DeviceInfo Element	230
Table 5-71 JobPhase Element	232
Table 5-72 ModuleStatus Element	234
Table 5-73 Track Message	236
Table 5-74 TrackFilter Element	236
Table 5-75 TrackResult Element	237
Table 5-76 UpdateJDF Command	238
Table 5-77 UpdateJDF Signal.....	238
Table 5-78 UpdateJDFCmdParams Element	238
Table 5-79 CreateLink Element	239
Table 5-80 CreateResource Element	239
Table 5-81 MoveResource Element	239
Table 5-82 RemoveLink Element	240
Table 5-83 WakeUp Message	241
Table 5-84 WakeUpCmdParams Element	241
Table 5-85 Messages for Control of Dynamic Pipes	241
Table 5-86 PipeClose Message	241
Table 5-87 PipePull Message	242

Table 5-88 PipeParams Element	242
Table 5-89 PipePush Message	243
Table 5-90 PipePause Message	244
Table 5-91 Messages for queue entry handling	245
Table 5-92 Status Transitions for QueueEntry Handling Messages	246
Table 5-93 AbortQueueEntry Message	248
Table 5-94 HoldQueueEntry Message	249
Table 5-95 RemoveQueueEntry Message	249
Table 5-96 RequestQueueEntry Message	249
Table 5-97 RequestQueueEntryParams Element	250
Table 5-98 ResubmitQueueEntry Message	250
Table 5-99 ResubmissionParams Element	251
Table 5-100 ResumeQueueEntry Message	251
Table 5-101 ReturnQueueEntry Message	251
Table 5-102 ReturnQueueEntryParams Element	251
Table 5-103 SetQueueEntry Message	252
Table 5-104 QueueEntryPosParams Element	252
Table 5-105 SetQueueEntryPriority Message	253
Table 5-106 QueueEntryPriParams Element	253
Table 5-107 SubmitQueueEntry Message	253
Table 5-108 QueueSubmissionParams Element	254
Table 5-109 SuspendQueueEntry Message	256
Table 5-110 Messages for global handling of queues	256
Table 5-111 Definition of the Queue Status Attribute Values	256
Table 5-112 CloseQueue Message	257
Table 5-113 FlushQueue Command Message	258
Table 5-114 FlushQueueParams Element	258
Table 5-115 FlushQueue Query Message	258
Table 5-116 FlushQueueInfo Element	258
Table 5-117 HoldQueue Message	259
Table 5-118 OpenQueue Message	259
Table 5-119 QueueStatus Message	259
Table 5-120 ResumeQueue Message	259
Table 5-121 SubmissionMethods Message	260
Table 5-122 SubmissionMethods Element	260
Table 5-123 Queue Element	261
Table 5-124 QueueEntry Element	262
Table 5-125 QueueEntryDef Element	264
Table 5-126 QueueFilter Element	264
Table 5-127 Messages for Gang Jobs	265
Table 5-128 Contents of the ForceGang Command Message	266
Table 5-129 GangCmdFilter Element	266
Table 5-130 GangStatus Message	266
Table 5-131 GangQuFilter Element	266
Table 5-132 GangInfo Element	266
Table 6-1 Template for Input Resources	269
Table 6-3 Approval – Input Resources	270
Table 6-2 Template for Output Resources	270
Table 6-5 Buffer – Input Resources	271
Table 6-6 Buffer – Output Resources	271

Table 6-7 Combine – Input Resources	271
Table 6-8 Combine – Output Resources	271
Table 6-4 Approval – Output Resources	271
Table 6-9 Delivery – Input Resources	272
Table 6-10 Delivery – Output Resources	272
Table 6-11 ManualLabor – Input Resources	272
Table 6-12 ManualLabor – Output Resources	272
Table 6-13 Ordering – Input Resources	272
Table 6-14 Ordering – Output Resources	272
Table 6-15 QualityControl – Input Resources	273
Table 6-16 QualityControl – Output Resources	273
Table 6-17 ResourceDefinition – Input Resources	273
Table 6-18 ResourceDefinition – Output Resources	273
Table 6-19 Split – Input Resources	273
Table 6-21 Verification – Input Resources	274
Table 6-22 Verification – Output Resources	274
Table 6-23 Product Intent – Input Resources	274
Table 6-20 Split – Output Resources	274
Table 6-25 AssetListCreation – Input Resources	275
Table 6-24 Product Intent – Output Resources	275
Table 6-27 Bending – Input Resources	276
Table 6-28 Bending – Output Resources	276
Table 6-29 ColorCorrection – Input Resources	276
Table 6-26 AssetListCreation – Output Resources	276
Table 6-31 ColorSpaceConversion – Input Resources	277
Table 6-32 ColorSpaceConversion – Output Resources	277
Table 6-30 ColorCorrection – Output Resources	277
Table 6-33 ContactCopying – Input Resources	278
Table 6-34 ContactCopying – Output Resources	278
Table 6-35 ContoneCalibration – Input Resources	278
Table 6-36 ContoneCalibration – Output Resources	278
Table 6-37 CylinderLayoutPreparation – Input Resources	278
Table 6-39 DBDocTemplateLayout – Input Resources	279
Table 6-40 DBDocTemplateLayout – Output Resources	279
Table 6-41 DBTemplateMerging – Input Resources	279
Table 6-42 DBTemplateMerging – Output Resources	279
Table 6-38 CylinderLayoutPreparation – Output Resources	279
Table 6-43 DieDesign – Input Resources	280
Table 6-44 DieDesign – Output Resources	280
Table 6-45 DieLayoutProduction – Input Resources	280
Table 6-46 DieLayoutProduction – Output Resources	280
Table 6-47 DigitalDelivery – Input Resources	282
Table 6-49 FormatConversion – Input Resources	283
Table 6-50 FormatConversion – Output Resources	283
Table 6-51 ImageReplacement – Input Resources	283
Table 6-48 DigitalDelivery – Output Resources	283
Table 6-53 ImageSetting – Input Resources	284
Table 6-54 ImageSetting – Output Resources	284
Table 6-52 ImageReplacement – Output Resources	284
Table 6-55 Imposition – Input Resources	285

Table 6-56 Imposition – Output Resources	285
Table 6-57 Glossary for Automated Imposition	286
Table 6-58 InkZoneCalculation – Input Resources	296
Table 6-59 InkZoneCalculation – Output Resources	296
Table 6-60 Interpreting – Input Resources	297
Table 6-61 Interpreting – Output Resources	297
Table 6-62 LayoutElementProduction – Input Resources	297
Table 6-63 LayoutElementProduction – Output Resources	297
Table 6-64 LayoutPreparation – Input Resources	298
Table 6-65 LayoutPreparation – Output Resources	298
Table 6-66 LayoutShifting – Input Resources	298
Table 6-67 LayoutPreparation – Output Resources	298
Table 6-68 PageAssigning – Input Resources	299
Table 6-69 PageAssigning – Output Resources	299
Table 6-70 PDFToPSConversion – Input Resources	299
Table 6-71 PDFToPSConversion – Output Resources	299
Table 6-72 PDLCreation – Input Resources	299
Table 6-74 Preflight – Input Resources	300
Table 6-75 Preflight – Output Resources	300
Table 6-73 PDLCreation – Output Resources	300
Table 6-76 PreviewGeneration – Input Resources	302
Table 6-77 PreviewGeneration – Output Resources	302
Table 6-78 PSToPDFConversion – Input Resources	303
Table 6-79 PSToPDFConversion – Output Resources	303
Table 6-80 RasterReading – Input Resources	303
Table 6-81 RasterReading – Output Resources	303
Table 6-82 Rendering – Input Resources	303
Table 6-83 Rendering – Output Resources	304
Table 6-84 Scanning – Input Resources	305
Table 6-85 Scanning – Output Resources	305
Table 6-86 Screening – Input Resources	305
Table 6-87 Screening – Output Resources	305
Table 6-88 Separation – Input Resources	306
Table 6-89 Separation – Output Resources	306
Table 6-90 Stripping – Input Resources	306
Table 6-91 Stripping – Output Resources	307
Table 6-92 Tiling – Input Resources	308
Table 6-93 Tiling – Output Resources	308
Table 6-94 Trapping – Input Resources	309
Table 6-95 Trapping – Output Resources	309
Table 6-96 ConventionalPrinting – Input Resources	310
Table 6-98 DigitalPrinting – Input Resources	312
Table 6-97 ConventionalPrinting – Output Resources	312
Table 6-99 DigitalPrinting – Output Resources	313
Table 6-100 Varnishing – Input Resources	314
Table 6-101 Varnishing – Output Resources	314
Table 6-102 BlockPreparation – Input Resources	315
Table 6-103 BlockPreparation – Output Resources	315
Table 6-104 BoxFolding – Input Resources	315
Table 6-105 BoxFolding – Output Resources	315

Table 6-106 BoxPacking – Input Resources	315
Table 6-108 Bundling – Input Resources	316
Table 6-109 Bundling – Output Resources	316
Table 6-107 BoxPacking – Output Resources	316
Table 6-110 CaseMaking – Input Resources	317
Table 6-111 CaseMaking – Output Resources	317
Table 6-112 CasingIn – Input Resources	317
Table 6-113 CasingIn – Output Resources	317
Table 6-114 ChannelBinding – Input Resources	318
Table 6-115 ChannelBinding – Output Resources	318
Table 6-116 CoilBinding – Input Resources	318
Table 6-117 CoilBinding – Output Resources	318
Table 6-118 Collecting – Input Resources	319
Table 6-119 Collecting – Output Resources	319
Table 6-120 CoverApplication – Input Resources	319
Table 6-121 CoverApplication – Output Resources	319
Table 6-122 Creasing – Input Resources	320
Table 6-123 Creasing – Output Resources	320
Table 6-124 Cutting – Input Resources	320
Table 6-125 Cutting – Output Resources	320
Table 6-126 DieMaking – Input Resources	321
Table 6-127 DieMaking – Output Resources	321
Table 6-128 Embossing – Input Resources	321
Table 6-129 Embossing – Output Resources	321
Table 6-130 EndSheetGluing – Input Resources	321
Table 6-131 EndSheetGluing – Output Resources	322
Table 6-132 Feeding – Input Resources	323
Table 6-133 Feeding – Output Resources	323
Table 6-134 Folding – Input Resources	323
Table 6-135 Folding – Output Resources	323
Table 6-136 Gathering – Input Resources	324
Table 6-137 Gathering – Output Resources	324
Table 6-138 Gluing – Input Resources	324
Table 6-139 Gluing – Output Resources	324
Table 6-140 HeadBandApplication – Input Resources	324
Table 6-141 HeadBandApplication – Output Resources	324
Table 6-142 HoleMaking – Input Resources	325
Table 6-143 HoleMaking – Output Resources	325
Table 6-144 Inserting – Input Resources	325
Table 6-145 Inserting – Output Resources	325
Table 6-146 Jacketing – Input Resources	326
Table 6-147 Jacketing – Output Resources	326
Table 6-148 Labeling – Input Resources	326
Table 6-149 Labeling – Output Resources	326
Table 6-150 Laminating – Input Resources	326
Table 6-151 Laminating – Output Resources	326
Table 6-152 Numbering – Input Resources	327
Table 6-153 Numbering – Output Resources	327
Table 6-154 Palletizing – Input Resources	327
Table 6-155 Palletizing – Output Resources	327

Table 6-156 Perforating – Input Resources	327
Table 6-158 PlasticCombBinding – Input Resources	328
Table 6-159 PlasticCombBinding – Output Resources	328
Table 6-160 PrintRolling – Input Resources	328
Table 6-161 PrintRolling – Output Resources	328
Table 6-157 Perforating – Output Resources	328
Table 6-162 RingBinding – Input Resources	329
Table 6-163 RingBinding – Output Resources	329
Table 6-164 ShapeCutting – Input Resources	329
Table 6-165 ShapeCutting – Output Resources	329
Table 6-166 ShapeDefProduction – Input Resources	330
Table 6-167 ShapeDefProduction – Output Resources	330
Table 6-168 Shrinking – Input Resources	330
Table 6-169 Shrinking – Output Resources	330
Table 6-170 SpinePreparation – Input Resources	330
Table 6-171 SpinePreparation – Output Resources	330
Table 6-172 SpineTaping – Input Resources	331
Table 6-173 SpineTaping – Output Resources	331
Table 6-174 Stacking – Input Resources	331
Table 6-175 Stacking – Output Resources	331
Table 6-176 StaticBlocking – Input Resources	331
Table 6-177 StaticBlocking – Output Resources	331
Table 6-178 Stitching – Input Resources	332
Table 6-179 Stitching – Output Resources	332
Table 6-180 Strapping – Input Resources	332
Table 6-181 Strapping – Output Resources	332
Table 6-182 StripBinding – Input Resources	333
Table 6-183 StripBinding – Output Resources	333
Table 6-184 ThreadSealing – Input Resources	333
Table 6-185 ThreadSealing – Output Resources	333
Table 6-186 ThreadSewing – Input Resources	333
Table 6-187 ThreadSewing – Output Resources	333
Table 6-188 Trimming – Input Resources	334
Table 6-189 Trimming – Output Resources	334
Table 6-190 WebInlineFinishing – Input Resources	334
Table 6-191 WebInlineFinishing – Output Resources	334
Table 6-192 WireCombBinding – Input Resources	335
Table 6-193 WireCombBinding – Output Resources	335
Table 6-194 Wrapping – Input Resources	335
Table 6-195 Wrapping – Output Resources	335
Table 7-1 Template for Intent Resources	340
Table 7-2 Abstract Span Element	340
Table 7-3 List of Span Elements	341
Table 7-4 DurationSpan Element	341
Table 7-5 EnumerationSpan Element	342
Table 7-6 IntegerSpan Element	343
Table 7-7 NameSpan Element	343
Table 7-8 NumberSpan Element	343
Table 7-9 OptionSpan Element	344
Table 7-10 ShapeSpan Element	344

Table 7-11 StringSpan Element	345
Table 7-12 TimeSpan Element	345
Table 7-13 XYPairSpan Element	345
Table 7-14 ArtDeliveryIntent Resource	346
Table 7-15 ArtDelivery Element	349
Table 7-16 BindingIntent Resource	352
Table 7-17 BindingType Attribute Values	354
Table 7-18 BindList Element	355
Table 7-19 BindItem Element	355
Table 7-20 ChannelBinding Element	356
Table 7-21 CoilBinding Element	356
Table 7-22 EdgeGluing Element	357
Table 7-23 HardcoverBinding Element	357
Table 7-24 PlasticCombBinding Element	359
Table 7-25 RingBinding Element	360
Table 7-26 SaddleStitching Element	361
Table 7-27 SideSewing Element	361
Table 7-28 SideStitching Element	361
Table 7-29 SoftCoverBinding Element	361
Table 7-30 StripBinding Element	362
Table 7-31 Tabs Element	363
Table 7-32 Tape Element	363
Table 7-33 ThreadSealing Element	363
Table 7-34 ThreadSewing Element	364
Table 7-35 WireCombBinding Element	364
Table 7-36 ColorIntent Resource	365
Table 7-37 ColorStandard Attribute Values	367
Table 7-38 ColorsUsed Element	367
Table 7-39 DeliveryIntent Resource	368
Table 7-40 DropIntent Element	371
Table 7-41 DropItemIntent Element	373
Table 7-42 EmbossingIntent Resource	374
Table 7-43 EmbossingItem Element	375
Table 7-44 FoldingIntent Resource	376
Table 7-45 HoleMakingIntent Resource	377
Table 7-46 InsertingIntent Resource	378
Table 7-47 InsertList Element	378
Table 7-48 Insert Element	378
Table 7-49 LaminatingIntent Resource	379
Table 7-50 LayoutIntent Resource	380
Table 7-51 MediaIntent Resource	383
Table 7-52 NumberingIntent Resource	389
Table 7-53 NumberItem Element	389
Table 7-54 PackingIntent Resource	390
Table 7-55 ProductionIntent Resource	391
Table 7-56 ProofingIntent Resource	392
Table 7-57 ProofItem Element	392
Table 7-58 PublishingIntent Resource	394
Table 7-59 ScreeningIntent Resource	395
Table 7-60 ShapeCuttingIntent Resource	395

Table 7-61 ShapeCut Element	396
Table 7-62 Template for Process Resource	397
Table 7-63 Address Resource	398
Table 7-64 ApprovalParams Resource	399
Table 7-65 ApprovalPerson Element	399
Table 7-66 ApprovalSuccess Resource	400
Table 7-67 ApprovalDetails Element	400
Table 7-68 Assembly Resource	401
Table 7-69 AssemblySection Element	402
Table 7-70 PageAssignedList Element	403
Table 7-71 AssetListCreationParams Resource	404
Table 7-72 AutomatedOverPrintParams Resource	405
Table 7-73 BarcodeCompParams Resource	406
Table 7-74 BarcodeReproParams Resource	406
Table 7-75 BendingParams Resource	407
Table 7-76 BinderySignature Resource	408
Table 7-77 SignatureCell Element	414
Table 7-78 BlockPreparationParams Resource	416
Table 7-79 BoxFoldingParams Resource	417
Table 7-80 BoxApplication Element	418
Table 7-81 BoxFoldAction Element	418
Table 7-82 Action Attribute Values	419
Table 7-83 BoxPackingParams Resource	422
Table 7-84 BufferParams Resource	424
Table 7-85 Bundle Resource	424
Table 7-86 BundleItem Element	425
Table 7-87 BundlingParams Resource	427
Table 7-88 ByteMap Resource	428
Table 7-89 Band Element	429
Table 7-90 PixelColorant Element	429
Table 7-91 CaseMakingParams Resource	430
Table 7-92 CasingInParams Resource	431
Table 7-93 ChannelBindingParams Resource	432
Table 7-94 CIELABMeasuringField Resource	433
Table 7-95 CoilBindingParams Resource	434
Table 7-96 CollectingParams Resource	434
Table 7-97 Color Resource	436
Table 7-98 DeviceNColor Element	439
Table 7-99 PrintConditionColor Element	440
Table 7-100 ColorantAlias Resource	444
Table 7-101 ColorantControl Resource	446
Table 7-102 ColorantConvertProcess Element	448
Table 7-103 ColorantOrder Element	448
Table 7-104 ColorantParams Element	448
Table 7-105 DeviceColorantOrder Element	448
Table 7-106 ColorSpaceSubstitute Element	448
Table 7-107 Sample output for different values of ProcessColorModel, ColorantParams, ColorantOrder, ColorantControlLink and DeviceColorantOrder Elements.	449
Table 7-108 ColorControlStrip Resource	450
Table 7-109 ColorCorrectionParams Resource	451

Table 7-110 ColorCorrectionOp Element	452
Table 7-111 ColorMeasurementConditions Resource	453
Table 7-112 ColorPool Resource	455
Table 7-113 ColorSpaceConversionOp Resource	455
Table 7-114 SourceCS Attribute Values	459
Table 7-115 Mapping of SourceCS enumeration values to color spaces in the most common input file formats	460
Table 7-116 ColorSpaceConversionParams Resource	462
Table 7-117 ComChannel Resource	464
Table 7-118 ChannelTypeDetails Attribute – predefined values for certain ChannelType values ..	465
Table 7-119 Company Resource	466
Table 7-120 Glossary – Component	467
Table 7-121 Component Resource	468
Table 7-122 ProductType Attribute Values	470
Table 7-123 Contact Resource	473
Table 7-124 ContactCopyParams Resource	474
Table 7-125 ContentList Resource	474
Table 7-126 ContentData Element	475
Table 7-127 ContentType Attribute Values	475
Table 7-128 ContentMetadata Element	476
Table 7-129 ConventionalPrintingParams Resource	479
Table 7-130 CostCenter Resource	482
Table 7-131 CoverApplicationParams Resource	482
Table 7-132 Score Element	482
Table 7-133 CreasingParams Resource	483
Table 7-134 Crease Element	483
Table 7-135 CustomerInfo Resource	485
Table 7-136 CustomerMessage Element	485
Table 7-137 CutBlock Resource	486
Table 7-138 CutMark Resource	487
Table 7-139 Cut mark types as specified by CutMark/@MarkType	488
Table 7-140 CuttingParams Resource	488
Table 7-141 Cut Element	489
Table 7-142 CylinderLayout Resource	490
Table 7-143 CylinderPosition Element	490
Table 7-144 CylinderLayoutPreparationParams Resource	493
Table 7-145 DBMergeParams Resource	494
Table 7-146 DBRules Resource	494
Table 7-147 DBSchema Resource	495
Table 7-148 DBSelection Resource	495
Table 7-149 DeliveryParams Resource	495
Table 7-150 Drop Element	496
Table 7-151 Dropltem Element	497
Table 7-152 DensityMeasuringField Resource	498
Table 7-153 DevelopingParams Resource	499
Table 7-154 Device Resource	500
Table 7-155 IconList Element	502
Table 7-156 Icon Element	502
Table 7-157 Module Element	502

Table 7-158 DeviceMark Resource	503
Table 7-159 DeviceNSpace Resource	505
Table 7-160 DieLayout Resource	505
Table 7-161 RuleLength Element	506
Table 7-162 Station Element	506
Table 7-163 DieLayoutProductionParams Resource	507
Table 7-164 ConvertingConfig Element	507
Table 7-165 RepeatDesc Element	507
Table 7-166 DigitalDeliveryParams Resource	512
Table 7-167 DigitalMedia Resource	513
Table 7-168 DigitalPrintingParams Resource	515
Table 7-169 Disjointing Resource	518
Table 7-170 Disposition Resource	519
Table 7-171 ElementColorParams Resource	520
Table 7-172 EmbossingParams Resource	521
Table 7-173 Emboss Element	522
Table 7-174 Employee Resource	523
Table 7-175 EndSheetGluingParams Resource	524
Table 7-176 EndSheet Element	524
Table 7-177 ExposedMedia Resource	526
Table 7-178 ExternalImpositionTemplate Resource	527
Table 7-179 FeedingParams Resource	527
Table 7-180 Feeder Element	527
Table 7-181 FeederQualityParams Element	529
Table 7-182 CollatingItem Element	529
Table 7-183 FileSpec Resource	531
Table 7-184 ResourceUsage Attribute Values	536
Table 7-185 Container Element	537
Table 7-186 FileAlias Element	537
Table 7-187 FitPolicy Resource	538
Table 7-188 Fold Resource	539
Table 7-189 FoldingParams Resource	541
Table 7-190 FontParams Resource	544
Table 7-191 FontPolicy Resource	545
Table 7-192 FormatConversionParams Resource	545
Table 7-193 TIFFFormatParams Element	546
Table 7-194 TIFFtag Element	547
Table 7-195 TIFFEmbeddedFile Element	548
Table 7-196 GatheringParams Resource	549
Table 7-197 GlueApplication Resource	549
Table 7-198 GlueLine Resource	550
Table 7-199 GluingParams Resource	551
Table 7-200 Glue Element	552
Table 7-201 HeadBandApplicationParams Resource	552
Table 7-202 Hole Resource	553
Table 7-203 HoleLine Resource	554
Table 7-204 HoleList Resource	555
Table 7-205 HoleMakingParams Resource	555
Table 7-206 IdentificationField Resource	558
Table 7-207 EncodingDetails Attribute Values	560

Table 7-208 BarcodeDetails Element	561
Table 7-209 ExtraValues Element	562
Table 7-210 Usage of barcode Attributes for certain barcode types	562
Table 7-211 BarcodeVersion Values – for HIBC_DATAMATRIX	563
Table 7-212 BarcodeVersion Values – for QR barcodes	563
Table 7-213 ImageCompressionParams Resource	564
Table 7-214 ImageCompression Element	565
Table 7-215 CCITTFaxParams Element	567
Table 7-216 DCTParams Element	568
Table 7-217 SourceCSs Attribute Values	568
Table 7-218 FlateParams Element	569
Table 7-219 JBIG2Params Element	570
Table 7-220 JPEG2000Params Element	570
Table 7-221 LZWParams Element	571
Table 7-222 ImageReplacementParams Resource	572
Table 7-223 ImageSetterParams Resource	573
Table 7-224 Sides Attribute Values	575
Table 7-225 Ink Resource	576
Table 7-226 InkZoneCalculationParams Resource	577
Table 7-227 InkZoneProfile Resource	577
Table 7-228 InsertingParams Resource	578
Table 7-229 Location of Inserts	579
Table 7-230 InsertSheet Resource	581
Table 7-231 SheetUsage Attribute Values	583
Table 7-232 InterpretingParams Resource	584
Table 7-233 PDFInterpretingParams Element	586
Table 7-234 OCGControl Element	588
Table 7-235 JacketingParams Resource	589
Table 7-236 JobField Resource	590
Table 7-237 LabelingParams Resource	591
Table 7-238 LaminatingParams Resource	591
Table 7-239 Layout Resource	592
Table 7-240 LayerList Element	596
Table 7-241 LayerDetails Element	596
Table 7-242 LogicalStackParams Element	596
Table 7-243 Stack Element	596
Table 7-244 PageCondition Element	597
Table 7-245 SheetCondition Element	599
Table 7-246 Abstract PlacedObject Element	600
Table 7-247 ContentObject Element	603
Table 7-248 MarkObject Element	603
Table 7-249 MarkActivation Element	605
Table 7-250 DynamicField Element	606
Table 7-251 Example (1) of Ord Attribute in PlacedObject Elements	610
Table 7-252 Example (2) of Ord Attribute in PlacedObject Elements	611
Table 7-253 LayoutElement Resource	612
Table 7-254 ElementType Attribute Values	614
Table 7-255 Dependencies Element	615
Table 7-256 LayoutElementProductionParams Resource	616
Table 7-257 LayoutElementPart Element	617

Table 7-258 BarcodeProductionParams Element	618
Table 7-259 PositionObj Element	618
Table 7-260 LayoutPreparationParams Resource	624
Table 7-261 FrontMarkList Attribute Values	631
Table 7-262 PageCell Element	632
Table 7-263 ImageShift Element	633
Table 7-264 LayoutShift Resource	636
Table 7-265 ShiftPoint Element	636
Table 7-266 ManualLaborParams Resource	638
Table 7-267 Media Resource	639
Table 7-268 MediaTypeDetails Attribute Values	646
Table 7-269 MediaLayers Element	649
Table 7-270 TabDimensions Element	649
Table 7-271 MiscConsumable Resource	656
Table 7-272 MISDetails Element	656
Table 7-273 NodeInfo Resource	658
Table 7-274 NumberingParams Resource	660
Table 7-275 NumberingParam Element	660
Table 7-276 ObjectResolution Resource	661
Table 7-277 OrderingParams Resource	662
Table 7-278 PageAssignParams Resource	662
Table 7-279 PageList Resource	663
Table 7-280 PageData Element	664
Table 7-281 PageElement Element	667
Table 7-282 Pallet Resource	667
Table 7-283 PalletizingParams Resource	668
Table 7-284 PDFToPSConversionParams Resource	669
Table 7-285 PDLCreationParams Resource	672
Table 7-286 PDLResourceAlias Resource	672
Table 7-287 PerforatingParams Resource	673
Table 7-288 Perforate Element	673
Table 7-289 Person Resource	674
Table 7-290 PlasticCombBindingParams Resource	675
Table 7-291 PreflightParams Resource	676
Table 7-292 PreflightAction Element	676
Table 7-293 BasicPreflightTest Element	677
Table 7-294 PreflightArgument Element	678
Table 7-295 BoxArgument Element	678
Table 7-296 Box Attribute Values	678
Table 7-297 BoxToBoxDifference Element	679
Table 7-298 PreflightReport Resource	680
Table 7-299 PRItem Element	680
Table 7-300 PRError Element	681
Table 7-301 PRGroup Element	681
Table 7-302 Abstract PRGroupOccurrenceBase Element	683
Table 7-303 List of PRGroupOccurrenceBase Elements	683
Table 7-304 ArgumentValue Element	683
Table 7-305 PRGroupOccurrence Element	684
Table 7-306 StringListValue Element	684
Table 7-307 PROccurrence Element	684

Table 7-308 PreflightReportRulePool Resource	685
Table 7-309 PRRule Element	685
Table 7-310 PRRuleAttr Element	685
Table 7-311 ReportAttr Attribute Values	686
Table 7-312 Contingent Report Behavior	686
Table 7-313 Preview Resource	687
Table 7-314 PreviewGenerationParams Resource	689
Table 7-315 PrintCondition Resource	691
Table 7-316 PrintRollingParams Resource	692
Table 7-317 ProductionPath Resource	693
Table 7-318 FolderSuperstructureWebPath Element	694
Table 7-319 PostPressComponentPath Element	694
Table 7-320 PrintingUnitWebPath Element	694
Table 7-321 PStoPDFConversionParams Resource	695
Table 7-322 AdvancedParams Element	697
Table 7-323 PDFXParams Element	699
Table 7-324 PDFXOutputIntentProfile Attribute Values	700
Table 7-325 ThinPDFParams Element	701
Table 7-326 QualityControlParams Resource	701
Table 7-327 BindingQualityParams Element	701
Table 7-328 QualityControlResult Resource	702
Table 7-329 QualityMeasurement Element	702
Table 7-330 BindingQualityMeasurement Element	702
Table 7-331 RasterReadingParams Resource	703
Table 7-332 RefAnchor Element	704
Table 7-333 RegisterMark Resource	705
Table 7-334 RegisterRibbon Resource	705
Table 7-335 RenderingParams Resource	707
Table 7-336 ResourceDefinitionParams Resource	707
Table 7-337 ResourceParam Element	708
Table 7-338 RingBindingParams Resource	708
Table 7-339 RollStand Resource	710
Table 7-340 RunList Resource	711
Table 7-341 DynamicInput Element	717
Table 7-342 MetadataMap Element	718
Table 7-343 Expr Element	720
Table 7-344 ScanParams Resource	725
Table 7-345 ScavengerArea Resource	726
Table 7-346 ScreeningParams Resource	727
Table 7-347 ScreenSelector Element	727
Table 7-348 SeparationControlParams Resource	730
Table 7-349 SeparationSpec Resource	730
Table 7-350 Shape Resource	731
Table 7-351 ShapeCuttingParams Resource	732
Table 7-352 ShapeDef Resource	732
Table 7-353 ShapeDefProductionParams Resource	734
Table 7-354 ObjectModel Element	734
Table 7-355 ShapeTemplate Element	734
Table 7-356 ShrinkingParams Resource	736
Table 7-357 SpinePreparationParams Resource	737

Table 7-358 Operations Attribute Values	737
Table 7-359 SpineTapingParams Resource	738
Table 7-360 Parameters in Stacking	740
Table 7-361 StackingParams Resource	741
Table 7-362 StrappingParams Resource	743
Table 7-363 StitchingParams Resource	745
Table 7-364 Strap Resource	747
Table 7-365 StrappingParams Resource	747
Table 7-366 StripBindingParams Resource	748
Table 7-367 StrippingParams Resource	749
Table 7-368 Position Element	752
Table 7-369 StripCellParams Element	753
Table 7-370 StripMark Element	755
Table 7-371 MarkName Attribute Values	757
Table 7-372 MarkSide Attribute Values	758
Table 7-373 ThreadSealingParams Resource	759
Table 7-374 ThreadSewingParams Resource	760
Table 7-375 Tile Resource	761
Table 7-376 Tool Resource	762
Table 7-377 TransferCurve Resource	763
Table 7-378 TransferCurvePool Resource	763
Table 7-379 TransferCurveSet Element	764
Table 7-380 TransferFunctionControl Resource	764
Table 7-381 TrappingDetails Resource	765
Table 7-382 TrappingOrder Element	765
Table 7-383 TrappingParams Resource	766
Table 7-384 ColorantZoneDetails Element	769
Table 7-385 TrapRegion Resource	769
Table 7-386 TrimmingParams Resource	771
Table 7-387 UsageCounter Resource	771
Table 7-388 VarnishingParams Resource	772
Table 7-389 VerificationParams Resource	773
Table 7-390 WebInlineFinishingParams Resource	774
Table 7-391 FolderProduction Element	774
Table 7-392 WireCombBindingParams Resource	775
Table 7-393 WrappingParams Resource	776
Table 7-394 DeviceCap Element	778
Table 7-395 ActionPool Element	780
Table 7-396 Action Element	781
Table 7-397 DevCapPool Element	781
Table 7-398 ModulePool Element	781
Table 7-399 ModuleCap Element	782
Table 7-400 DevCaps Element	782
Table 7-401 Loc Element	785
Table 7-402 DevCap Element	786
Table 7-403 Abstract State Element	789
Table 7-404 ListType Attribute Values	791
Table 7-405 List of State Elements	791
Table 7-406 BooleanState Element	792
Table 7-407 ValueLoc Element	792

Table 7-408 DateTimeState Element	793
Table 7-409 DurationState Element	793
Table 7-410 EnumerationState Element	794
Table 7-411 IntegerState Element	794
Table 7-412 MatrixState Element	796
Table 7-413 MatrixState/Value Element	796
Table 7-414 NameState Element	797
Table 7-415 NumberState Element	797
Table 7-416 PDFPathState Element	798
Table 7-417 PDFPathState/Value Element	799
Table 7-418 RectangleState Element	799
Table 7-419 ShapeState Element	800
Table 7-420 StringState Element	801
Table 7-421 StringState/Value Element	801
Table 7-422 XYPairState Element	801
Table 7-423 DisplayGroupPool Element	802
Table 7-424 DisplayGroup Element	803
Table 7-425 FeaturePool Element	803
Table 7-426 MacroPool Element	804
Table 7-427 macro Element	805
Table 7-428 choice Element	805
Table 7-429 otherwise Element	805
Table 7-430 when Element	805
Table 7-431 set Element	806
Table 7-432 FeatureAttribute Element	806
Table 7-433 call Element	806
Table 7-434 Performance Element	806
Table 7-435 TestPool Element	807
Table 7-436 Test Element	807
Table 7-437 List of Term Elements	808
Table 7-438 and Element	809
Table 7-439 or Element	810
Table 7-440 xor Element	810
Table 7-441 not Element	810
Table 7-442 TestRef Element	810
Table 7-443 Abstract Evaluation Element	812
Table 7-444 List of Abstract Evaluation Elements	812
Table 7-445 Mapping of Evaluation Element to State Element	813
Table 7-446 BooleanEvaluation Element	813
Table 7-447 DateTimeEvaluation Element	813
Table 7-448 DurationEvaluation Element	814
Table 7-449 EnumerationEvaluation Element	814
Table 7-450 IntegerEvaluation Element	814
Table 7-451 IsPresentEvaluation Element	814
Table 7-452 MatrixEvaluation Element	815
Table 7-453 MatrixEvaluation/Value Element	815
Table 7-454 NameEvaluation Element	815
Table 7-455 NumberEvaluation Element	815
Table 7-456 PDFPathEvaluation Element	816
Table 7-457 PDFPathEvaluation/Value Element	816

Table 7-458 RectangleEvaluation Element	816
Table 7-459 ShapeEvaluation Element	816
Table 7-460 StringEvaluation Element	817
Table 7-461 StringEvaluation/Value Element	817
Table 7-462 XYPairEvaluation Element	817
Table 7-463 Object Classes for a Document	826
Table 7-464 Properties Implemented by each Class of Object	827
Table 7-465 Mapping between property types (in the preflight spec) and evaluations	828
Table 7-466 List of Properties Categories	829
Table 7-467 Annotation Properties	830
Table 7-468 AnnotationType Attribute Values	830
Table 7-469 Box Properties	830
Table 7-470 Class Properties	831
Table 7-471 ClassName Attribute Values	831
Table 7-472 PropertyList Attribute Values	831
Table 7-473 Colorant Properties	832
Table 7-474 Document Properties	832
Table 7-475 Fill Properties	836
Table 7-476 FillColorType Attribute Values	836
Table 7-477 Font Properties	837
Table 7-478 FontType Attribute Values	837
Table 7-479 Graphic Properties	838
Table 7-480 Image Properties	839
Table 7-481 Logical Properties	841
Table 7-482 PageBox Properties	842
Table 7-483 Pages Properties	842
Table 7-484 PDLObject Properties	843
Table 7-485 Reference Properties	843
Table 7-486 Shading Properties	844
Table 7-487 Stroke Properties	844
Table 7-488 Text Properties	845
Table 7-489 Vector Properties	846
Table 8-1 MIME Content-Types	852
Table A-1 Anchor Enumeration Values	870
Table A-2 JDFJMFVersion Enumeration Values	870
Table A-3 NamedColor Enumeration Values	870
Table A-4 Orientation Enumeration Values	871
Table A-5 WorkStyle Enumeration Values	871
Table A-6 XYRelation Enumeration Values	872
Table C-1 StatusDetails Mapping for Generic Devices	875
Table C-2 StatusDetails Mapping for Printing Devices	878
Table C-4 ModuleType Attribute Values for Conventional Printing Devices	879
Table C-3 StatusDetails Mapping for Postpress Devices	879
Table C-5 ModuleType Attribute Values for Postpress	880
Table C-6 ModuleType Attribute Values for DigitalPrinting	881
Table C-7 ModuleType Attribute Values for PrintingUnitWebPath Modules of Web Printing Devices	881
Table C-8 ModuleType Attribute Values for FolderSuperstructureWebPath Modules of Web Printing Devices	882
Table C-9 ModuleType Attribute Values for PostPressComponentPath Modules of Web Printing	

Devices	883
Table C-10 Abstract NotificationDetails.....	883
Table C-11 List of NotificationDetail Elements	883
Table C-12 Barcode Element	884
Table C-13 FCNKey Element	884
Table C-14 SystemTimeSet Element	884
Table C-15 CounterReset Element	884
Table C-16 Error Element	884
Table C-17 ErrorData Element	885
Table C-18 Event Element	885
Table C-19 Milestone Element	886
Table C-20 MessageEvents and MilestoneType Values	886
Table C-21 Input Tray and Output Bin Names	889
Table D-1 Return codes for JMF	891
Table E-1 Attributes for Color Space Adjustment	893
Table F-1 Conversion Factor from USWeight (lbs) to Weight (g/m2)	895
Table F-2 Grammage Equivalents for Common (US) Basis Weights	895
Table F-3 Japanese Media Weight	896
Table G-1 Media Sizes	899
Table H-1 MimeType Attribute Values (IANA Registered)	903
Table H-2 MimeType and File Type Combinations	905
Table I-1 Predefined variables used in @XXXTemplate and @ShowList	909
Table K-1 References	915
Table L-1 Naming Scheme for Hole Patterns	929
Table L-2 Hole Details for R2 Series	930
Table L-3 Hole Details for R3 and R4 Series	931
Table L-4 Hole Details for R5 and R6 Series	932
Table L-5 Hole Details for R7 and R11 Series	934
Table L-6 Hole Details for P, W, C and S Series	935
Table M-1 Use Cases showing MimeType, URL and Compression Attribute Values	939
Table M-2 AppOS and OSVersion Examples	950
Table O-1 Compatibility Warnings	987
Table O-2 Changed Items	987
Table P-1 Contents of the Abstract ResourceUpdate Element	1005
Table P-2 Contents of the StatusPool Element	1006
Table P-3 Contents of the PartStatus Element	1006
Table P-4 Added Element	1007
Table P-5 ChangedAttribute Element	1007
Table P-6 Removed Element	1007
Table P-7 NodeInfo Query Message.....	1008
Table P-8 NodeInfoQuParams Element.....	1008
Table P-9 NodeInfo Command Message	1008
Table P-10 NodeInfoCmdParams Element.....	1009
Table P-11 NodeInfoResp Element.....	1009
Table P-12 KnownJDFServices Message	1010
Table P-13 JDFService Element	1010
Table P-14 QueueEntryStatus Message	1011
Table P-15 QueueEntryDefList Element	1011
Table P-16 Packing – Input Resources.....	1012
Table P-17 Packing – Output Resources	1012

Table P-18 FilmToPlateCopying – Input Resources.....	1012
Table P-19 FilmToPlateCopying – Output Resources.....	1012
Table P-20 PreflightAnalysis Resource.....	1013
Table P-21 PreflightDetail Element.....	1013
Table P-22 PreflightInstance Element.....	1014
Table P-23 PreflightInstanceDetail Element.....	1014
Table P-24 PreflightInventory Resource.....	1015
Table P-25 PreflightProfile Resource.....	1015
Table P-26 PreflightConstraint Element.....	1016
Table P-27 Proofing – Input Resources.....	1017
Table P-28 Proofing – Output Resources.....	1017
Table P-29 SoftProofing – Input Resources.....	1018
Table P-30 SoftProofing – Output Resources.....	1018
Table P-31 IDPrinting – Input Resources.....	1018
Table P-32 IDPrinting – Output Resources.....	1019
Table P-33 Dividing – Input Resources.....	1020
Table P-34 Dividing – Output Resources.....	1020
Table P-35 LongitudinalRibbonOperations – Input Resources.....	1020
Table P-36 LongitudinalRibbonOperations – Output Resources.....	1020
Table P-37 SaddleStitching – Input Resources.....	1020
Table P-39 SideSewing – Input Resources.....	1021
Table P-40 SideSewing – Output Resources.....	1021
Table P-38 SaddleStitching – Output Resources.....	1021
Table P-41 AdhesiveBinding Element.....	1022
Table P-42 BookCase Element.....	1022
Table P-43 Pricing Element.....	1023
Table P-44 Payment Element.....	1023
Table P-45 CreditCard Element.....	1023
Table P-46 SizeIntent Resource.....	1024
Table P-47 AdhesiveBindingParams Resource.....	1025
Table P-48 DividingParams Resource.....	1026
Table P-49 IDPrintingParams Resource.....	1026
Table P-50 OutputBin Attribute Values.....	1027
Table P-51 Cover Element.....	1028
Table P-52 IDPFinishing Element.....	1029
Table P-53 Finishings Attribute Values.....	1029
Table P-54 IDPFolding Element.....	1030
Table P-55 IDPHoleMaking Element.....	1030
Table P-56 IDPLayout Element.....	1030
Table P-57 IDPStitching Element.....	1033
Table P-58 IDPTrimming Element.....	1034
Table P-59 ImageShift Element.....	1034
Table P-60 JobSheet Element.....	1035
Table P-61 Signature Element.....	1037
Table P-62 LongitudinalRibbonOperationParams Resource.....	1038
Table P-63 LROperation Element.....	1038
Table P-64 LongGlue Element.....	1038
Table P-65 LongPerforate Element.....	1039
Table P-66 MediaSource Resource.....	1039
Table P-67 PackingParams Resource.....	1040

Table P-68 PlateCopyParams Resource 1041
Table P-69 ProofingParams Resource..... 1041
Table P-70 SaddleStitchingParams Resource 1043
Table P-71 Sheet Resource 1044
Table P-72 SideSewingParams Resource 1045
Table P-73 Surface Resource 1046

Appendix S List of Examples

Example 1-1 XPath Expression	3
Example 3-1 ResourceLink Structure for a ProcessGroup	56
Example 3-2 Combined Process Node	58
Example 3-3 ResourceLinkPool for Combined Process Node	58
Example 3-4 Complex Combined Process Node	59
Example 3-5 EmployeeLink As ImplementationLink	82
Example 3-6 PartAmount	84
Example 3-7 MediaLink with Lot	86
Example 3-8 MediaRef to Partitioned Media	87
Example 3-9 Equivalent Inline Media	88
Example 3-10 Invalid Inline Partitioned Media	88
Example 3-11 MediaLink and MediaRef	89
Example 3-12 Inheritance for Subelements of a Partitioned Resource	94
Example 3-13 Partitioned ExposedMedia	94
Example 3-14 Legal Incomplete Partition	95
Example 3-15 Illegal Incomplete Partition	96
Example 3-16 Legal Complete Partition	96
Example 3-17 Illegal Partition	96
Example 3-18 Degenerate Partition	96
Example 3-19 Invalid Degenerate Partition	97
Example 3-20 Partitioned ExposedMedia with Media Subelements	97
Example 3-21 Partitioned ExposedMedia with Incomplete Media Subelements	98
Example 3-22 Partitioned ExposedMedia with Invalid Partitioning of Media Subelements	98
Example 3-23 Partitioned ExposedMedia with MediaRef Subelements	98
Example 3-24 Partitioned ExposedMedia with Invalid MediaRef Subelements	99
Example 3-25 Partitioning with the Identical Element	99
Example 3-26 ResourceLink with Part Element	100
Example 3-27 Partitioning with an Invalid Identical Element	101
Example 3-28 ExposedMedia with Location Elements	114
Example 3-29 Media with Location Elements	114
Example 3-30 Linking to Subsets of Resources	115
Example 3-31 @Amount in an ExposedMediaLink to a Partitioned ExposedMedia	115
Example 3-32 PartUsage in a Partitioned Resource	117
Example 3-33 Explicit Reference of Ordered Partitioned Resources	118
Example 3-34 Implicit Reference of Ordered Partitioned Resources	119
Example 3-35 ResourceAudit: Before Logging	132
Example 3-36 ResourceAudit: Logging of Consumption	133
Example 3-37 ResourceAudit: Logging Changes	133
Example 3-38 Namespaces in XML	135
Example 3-39 Extending Process Types	136
Example 3-40 Extending NMTOKEN Lists	137
Example 4-1 PrintTalk	144
Example 4-2 Equivalent Pure JDF	144
Example 4-3 Product Intent Node	145
Example 4-4 Family Tree of Spawned Nodes	158
Example 5-1 Query Message	176
Example 5-2 Response Message for Query	177

Example 5-3 Signal Message	179
Example 5-4 ResumeQueueEntry Command Message	181
Example 5-5 ResumeQueueEntry Response Message	181
Example 5-6 Acknowledge Message	182
Example 5-7 Response with Notification Element	187
Example 5-8 Query with Subscription to All Events	190
Example 5-9 Response for Subscription to All Events	190
Example 5-10 KnownControllers Query	193
Example 5-11 KnownControllers Response	193
Example 5-12 KnownDevices Response	194
Example 5-13 KnownMessages Response	198
Example 5-14 Notification Signal	200
Example 5-15 RepeatMessages Response	201
Example 5-16 RequestForAuthentication Command	205
Example 5-17 RequestForAuthentication Response	205
Example 5-18 RequestForAuthentication Query Subsequently	206
Example 5-19 RequestForAuthentication Response from Subsequent Query	206
Example 5-20 Occupation Response	213
Example 5-21 Resource Query about Paper	216
Example 5-22 Resource Response about Paper	216
Example 5-23 Resource Query about Employees	217
Example 5-24 Resource Response about Employees	217
Example 5-25 Resource Signal about Consumed Resources	217
Example 5-26 Resource Command: Single Resource is Available	221
Example 5-27 Resource Command: Multiple Resources are Available	221
Example 5-28 Resource Query for Consumables	224
Example 5-29 Resource Response about Consumables	224
Example 5-30 Resource Command for Changing Amount	225
Example 5-31 Resource Response for Changing Amount	225
Example 5-32 ResourcePull Command	227
Example 5-33 Status Signal	228
Example 5-34 Status Response to Query	235
Example 5-35 Track Response	237
Example 5-36 UpdateJDF Command	240
Example 5-37 AbortQueueEntry Command	248
Example 5-38 AbortQueueEntry Response	248
Example 5-39 SubmitQueueEntry Command with "file" Scheme	255
Example 5-40 SubmitQueueEntry Command with "http" Scheme	255
Example 5-41 SubmissionMethods Response	261
Example 5-42 Queue Element	262
Example 5-43 Custom Query	267
Example 5-44 Custom Response	267
Example 5-45 Custom Query for IfraTrack	267
Example 5-46 Custom Response for IfraTrack	268
Example 6-1 DieLayoutProduction: Single Shape and Two Sheet Sizes	280
Example 6-2 DieLayoutProduction: Single Shape and Range of Sheet Sizes	281
Example 6-3 DieLayoutProduction: Two Shapes and Range of Sheet Sizes	281
Example 6-4 Automated Imposition: MarkObject	289
Example 6-5 Imposition Template: Layout	290
Example 6-6 Output RunList (Surfaces)	291

Example 6-7 Automated Imposition: Ord Values	295
Example 6-8 RIPing	304
Example 6-9 Stripping: Simple Example	307
Example 6-10 Stripping: Complex Example	307
Example 6-11 Stitching: Combined Process	332
Example 7-1 EnumerationSpan	342
Example 7-2 Perfect Bound (Gathering)	403
Example 7-3 Saddle-Stitched Brochure (Collecting)	404
Example 7-4 Pseudo Code to Generate Page Count from SignatureCell Elements	413
Example 7-5 StrippingParams: Foldout Using FaceCells	415
Example 7-6 BoxFoldingParams/BoxFoldAction	419
Example 7-7 Bundle: Boxing and Palletizing	425
Example 7-8 Color	441
Example 7-9 ColorantControl: Content-Ignorant MIS	441
Example 7-10 ColorantControl: Synchronized with Input	442
Example 7-11 ColorantControl: Synchronized with Input with Alias	443
Example 7-12 ColorantControl: with ColorantAlias/ReplacementColorantName	443
Example 7-13 ColorantControl: with Invalid ColorantAlias/ReplacementColorantName	444
Example 7-14 ColorantAlias/@RawNames	444
Example 7-15 ComChannel for Telephone	465
Example 7-16 ComChannel for Instant Messaging	465
Example 7-17 Component: Partitioned by the Condition Partition Key	471
Example 7-18 ContentList	476
Example 7-19 ContentList: Extended with ISBN, Author, etc.	477
Example 7-20 CylinderLayout	491
Example 7-21 CylinderLayout: Double-Spread-Page Plate	493
Example 7-22 Barcode	563
Example 7-23 PageCondition	598
Example 7-24 Layout: DynamicField Element	607
Example 7-25 Invalid MarkObject	608
Example 7-26 MarkObject	608
Example 7-27 RunList: Simple Multi-File Unseparated RunList	610
Example 7-28 RunList: Simple Multi-File Separated RunList	610
Example 7-29 OrdExpression	611
Example 7-30 DocOrd Usage	612
Example 7-31 LayoutElementProductionParams: Page Shape	616
Example 7-32 LayoutElementProductionParams: Label Shape	616
Example 7-33 LayoutElementProductionParams: Box Shape	617
Example 7-34 LayoutElementProductionParams: PositionObj	619
Example 7-35 LayoutElementProductionParams: Preflight	621
Example 7-36 LayoutPreparationParams: JDF for Figure 7-39	635
Example 7-37 LayoutPreparationParams: JDF for Figure 7-40	635
Example 7-38 LayoutShift	637
Example 7-39 Media: Corrugated	653
Example 7-40 Media: Self Adhesive	654
Example 7-41 Media: Flat Plate	654
Example 7-42 Media: Flexo Sleeve	655
Example 7-43 Test with InsideBox and a BoxArgument Subelement	679
Example 7-44 PRItem	682
Example 7-45 PrintCondition	691

Example 7-46 ProductionPath: on Path Level:	694
Example 7-47 ProductionPath: on Part Path Level:.....	694
Example 7-48 Marks and Reordering of Content using RunList/@IgnoreContext.....	716
Example 7-49 MetadataMap: Setting Attributes	721
Example 7-50 RunList/MetadataMap.....	721
Example 7-51 RunList: Unstructured Single-File RunList.....	722
Example 7-52 RunList: Multi-File Unseparated RunList	723
Example 7-53 RunList: Multi-File Unseparated RunList with Spawning	723
Example 7-54 RunList: Spawned RunList.....	723
Example 7-55 RunList: Multi-File Separated RunList	724
Example 7-56 ShapeTemplate for Figure 7-50.....	734
Example 7-57 DisplayGroupPool.....	802
Example 7-58 FeaturePool	803
Example 7-59 ActionPool and TestPool.....	809
Example 7-60 KnownDevices Query for a Scanner.....	818
Example 7-61 KnownDevices Response for a Scanner	818
Example 7-62 KnownDevices Query for a Scanner #2.....	821
Example 7-63 KnownDevices Response for a Scanner #2	821
Example 7-64 JDF Accepted by Previous Scanner	824
Example 7-65 JDF Rejected by Previous Scanner.....	824
Example 7-66 Test for Existence of TrappedKey	827
Example 7-67 Test for TrappedKey Equal to “Unknown”	828
Example 7-68 Test with BoxToBoxDifference Element	843
Example 8-1 Packaging of Individual JDF/JMF files in MIME.....	853
Example 8-2 CID URL Scheme	853
Example 8-3 MIME Multipart/Related	854
Example A-1 boolean.....	857
Example A-2 CMYKColor	858
Example A-3 date	858
Example A-4 dateTime	858
Example A-5 DateTimeRange	858
Example A-6 DateTimeRangeList.....	859
Example A-7 double.....	859
Example A-8 DoubleList	859
Example A-9 DoubleRange	859
Example A-10 DoubleRangeList.....	859
Example A-11 duration.....	860
Example A-12 DurationRange	860
Example A-13 DurationRangeList.....	860
Example A-14 gYearMonth	860
Example A-15 hexBinary	861
Example A-16 ID.....	861
Example A-17 IDREF.....	861
Example A-18 IDREFS	861
Example A-19 integer	861
Example A-20 IntegerList.....	862
Example A-21 IntegerRange.....	862
Example A-22 IntegerRangeList.....	862
Example A-23 LabColor.....	862
Example A-24 language.....	863

Example A-25 languages.....	863
Example A-26 matrix.....	863
Example A-27 NameRange	863
Example A-28 NameRangeList.....	864
Example A-29 NMTOKEN	864
Example A-30 NMTOKENS	864
Example A-31 PDFPath.....	864
Example A-32 rectangle.....	865
Example A-33 RectangleRange.....	865
Example A-34 RectangleRangeList	865
Example A-35 regExp	865
Example A-36 shape.....	865
Example A-37 ShapeRange	866
Example A-38 ShapeRangeList.....	866
Example A-39 sRGBColor	866
Example A-40 string.....	866
Example A-41 TransferFunction	867
Example A-42 URI	867
Example A-43 URL	868
Example A-44 URL: UTF-8.....	868
Example A-45 URL: Windows Locale 1252	868
Example A-46 URL: Escaped Characters.....	868
Example A-47 XPath.....	868
Example A-48 XYPair	868
Example A-49 XYPairRange.....	869
Example A-50 XYPairRangeList.....	869
Example A-51 enumeration	869
Example A-52 enumerations.....	869
Example B-1 JDF Nodes: xsi:type	873
Example B-2 JDF Nodes: xsi:type (not in Default Namespace)	874
Example B-3 JMF: xsi:type	874
Example C-1 Milestone in JMF	886
Example C-2 MessageEvents in CustomerInfo	888
Example I-1 @FileTemplate and @FileFormat.....	912
Example M-1 FileSpec #1	940
Example M-2 FileSpec #2	940
Example M-3 FileSpec #3	941
Example M-4 FileSpec #4	941
Example M-5 FileSpec #5	941
Example M-6 FileSpec #6	941
Example M-7 FileSpec #7	942
Example M-8 FileSpec #8	942
Example M-9 FileSpec #9	943
Example M-10 FileSpec #10	943
Example M-11 FileSpec #11	944
Example M-12 FileSpec #11.1 — No Partitioning	944
Example M-13 FileSpec #11.2 — Intermediate container Partitioned.....	945
Example M-14 FileSpec #11.3 — the pdf is Partitioned.....	946
Example M-15 FileSpec #11.3a — the pdf is Partitioned, Different File Layout.....	947
Example M-16 FileSpec #11.4 — Both Partitioned	948

Example M-17 FileSpec #12	948
Example M-18 Intent Job Ticket	949
Example N-1 Before Processing	953
Example N-2 After Processing	954
Example N-3 Product JDF	955
Example N-4 2-Component JDF before Spawning	956
Example N-5 2-Component JDF Parent after Spawning the Cover Node	957
Example N-6 2-Component JDF Spawmed Node	959
Example N-7 2-Component JDF after Merging	959
Example N-8 Partitioned ImageSetting Node before Spawning	960
Example N-9 Spawmed Cyan Partition of the ImageSetting Node	961
Example N-10 Root Partitioned ImageSetting Node after Spawning	961
Example N-11 Merged ImageSetting Node	962
Example N-12 RunList	963
Example N-13 KnownMessages Query	965
Example N-14 KnownMessages Response	965
Example N-15 Status Query	966
Example N-16 Status Response	966
Example N-17 Status Signal #1	966
Example N-18 Status Signal #2	966
Example N-19 Status Signal #3	967
Example N-20 Using Position	967
Example N-21 Multiple Bindery Signatures	968
Example N-22 Multisection Bindery Signatures	968
Example N-23 Multiple Job Parts in One Imposition	969
Example N-24 FoldOuts	970
Example N-25 Multiple Web Layout	971
Example N-26 Stripping Process	972
Example N-27 DigitalDelivery: Before the Delivery	974
Example N-28 DigitalDelivery: After the Delivery	974
Example N-29 Delivery and DigitalDelivery Processes	975
Example N-30 Full Example of Digital Delivery through Central Server	978
Example N-31 Algorithm for Processing an Imposition Template	980
Example N-32 Format of Variable Data Structured Content	981
Example N-33 Page Pools	982
Example N-34 Booklet Using Automated Imposition	984