

# JDF Specification

Release 1.2





## 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 © 2000-2004, 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 © 2000-2004, International Cooperation for the Integration of Processes in Prepress, Press and Postpress (CIP4) with registered office in Zurich, Switzerland.

### Trademarks and Tradenames

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

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

### Waiver of Liability

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



# Table of Contents

## **Front Matter**

<b>Legal Notice</b> .....	<b>i</b>
<b>Table of Contents</b> .....	<b>iii</b>
<b>List of Figures</b> .....	<b>xxvii</b>
<b>Chapter 1 Introduction</b> .....	<b>1</b>
<b>1.1 Background on JDF</b> .....	<b>1</b>
<b>1.2 Document References</b> .....	<b>1</b>
<b>1.3 Conventions Used in This Specification</b> .....	<b>2</b>
1.3.1 Text Styles .....	2
1.3.2 XPath Notation Used in this Specification.....	3
1.3.3 Call-Outs .....	3
1.3.4 Specification of Cardinality.....	4
<b>1.4 Glossary of Terminology</b> .....	<b>4</b>
1.4.1 Conformance Terminology.....	7
1.4.2 Conformance Requirements for JDF Entities .....	7
1.4.2.1 Conformance Requirements for Support of Attributes and Attribute Values.....	7
1.4.2.2 Conformance Requirements for Support of Elements.....	8
1.4.2.3 Conformance Requirements for Support of Processes .....	8
1.4.2.4 Conformance Requirements for Support of Combined Processes.....	9
1.4.3 Conformance to SettingsPolicy.....	9
<b>1.5 Data Structures</b> .....	<b>9</b>
<b>1.6 Units</b> .....	<b>11</b>
<b>Chapter 2 Overview of JDF</b> .....	<b>13</b>
<b>2.1 System Components</b> .....	<b>13</b>
2.1.1 Job Components.....	13
2.1.1.1 Jobs and Nodes .....	13
2.1.1.2 Elements.....	13
2.1.1.3 Attributes .....	13
2.1.1.4 Relationships.....	13
2.1.1.5 Links .....	14
2.1.2 Workflow Component Roles .....	14
2.1.2.1 Machines.....	14
2.1.2.2 Devices.....	14
2.1.2.3 Agents .....	14
2.1.2.4 Controllers .....	15

2.1.2.5 Management Information Systems—MIS .....	15
2.1.2.6 System Interaction .....	16
<b>2.2 JDF Workflow .....</b>	<b>16</b>
2.2.1 Job Structure.....	17
<b>2.3 Hierarchical Tree Structure and Networks in JDF .....</b>	<b>19</b>
<b>2.4 Role of Messaging in JDF .....</b>	<b>20</b>
<b>2.5 Coordinate Systems in JDF .....</b>	<b>21</b>
2.5.1 Introduction .....	21
2.5.2 How and Where Coordinates and Transformations Are Used/Defined in JDF ....	22
2.5.3 Coordinate Systems of Resources and Processes.....	23
2.5.3.1 Coordinate Systems of Combined Processes.....	23
2.5.3.2 Coordinate System Transformations .....	23
2.5.4 Product Example: Simple Brochure .....	25
2.5.5 General Rules.....	30
2.5.6 Homogeneous Coordinates .....	30
<b>Chapter 3 Structure of JDF Nodes and Jobs .....</b>	<b>33</b>
<b>3.1 JDF Nodes .....</b>	<b>35</b>
3.1.1 Generic Contents of JDF Elements .....	35
3.1.2 JDF Node Attributes and Elements.....	37
3.1.2.1 Common Node Types .....	42
3.1.3 Product Intent Nodes .....	43
3.1.4 Process Group Nodes.....	43
3.1.4.1 Use of the Types attribute in ProcessGroup nodes .....	44
3.1.4.2 Use of the NamedFeatures attribute in Product and ProcessGroup nodes.....	44
3.1.4.3 ResourceLink Structure in ProcessGroup nodes .....	45
3.1.5 Combined Process Nodes .....	46
3.1.5.1 Combined Process Nodes with Multiple Processes of the Same Type.....	46
3.1.5.2 Examples of Combined Process Nodes .....	47
3.1.6 Process Nodes.....	47
<b>3.2 AncestorPool .....</b>	<b>48</b>
<b>3.3 Customer Information in CustomerInfo .....</b>	<b>49</b>
<b>3.4 Node Information in NodeInfo .....</b>	<b>51</b>
<b>3.5 StatusPool .....</b>	<b>52</b>
<b>3.6 Resources .....</b>	<b>53</b>
3.6.1 Resource Classes.....	56
3.6.1.1 Parameter Resources.....	56
3.6.1.2 Intent Resources.....	57
3.6.1.3 Implementation Resources.....	57
3.6.1.4 Physical Resources (Consumable, Quantity, Handling).....	57

3.6.1.5 Placeholder Resources .....	59
3.6.2 Position of Resources within JDF Nodes.....	59
3.6.3 Pipe Resources.....	59
3.6.4 ResourceUpdate Elements.....	59
<b>3.7 Resource Links .....</b>	<b>61</b>
3.7.1 Links to Parameter Resources.....	66
3.7.2 Links to Implementation Resources.....	66
3.7.3 Links to Physical Resources.....	67
3.7.4 Links to Placeholder Resources.....	68
3.7.5 Links to Intent Resources .....	68
3.7.6 Inter-Resource Linking Using ResourceRef.....	68
3.7.6.1 Status of Resources That Contain rRef References .....	70
3.7.6.2 Alignment of ResourceLink and ResourceRef.....	70
<b>3.8 Subsets of Resources .....</b>	<b>71</b>
3.8.1 Resource Amount .....	71
3.8.1.1 Evaluating and Updating Amount related attributes in a Device .....	71
3.8.1.2 Specifying Amount for a partially completed process.....	72
3.8.2 Description of Partitionable Resources.....	74
3.8.2.1 Amount in Partitionable Resources .....	75
3.8.2.2 Relating PartIDKeys and Partitions.....	75
3.8.2.2.1 Incomplete Partitions .....	76
3.8.2.2.2 Multiple Keys per Partitioned Leaf or Node .....	76
3.8.2.2.3 Degenerate Partitions .....	76
3.8.2.3 Partitioning of Resource sub-Elements.....	77
3.8.2.4 Additional Attributes for use with partitioned Resources .....	78
3.8.2.5 Options in Intent Resources.....	84
3.8.2.6 Locations of Physical Resources .....	84
3.8.3 Linking to Subsets of Resources .....	85
3.8.3.1 Handling Amount in a ResourceLink to a Partitioned Resource.....	85
3.8.3.2 Implicit and Explicit PartUsage in Partitioned Resources.....	85
3.8.3.3 Referencing Partitioned Resources from Nodes That Allow Multiple ResourceLinks....	86
3.8.4 Splitting and Combining Resources.....	87
<b>3.9 AuditPool .....</b>	<b>88</b>
3.9.1 Audit Elements.....	91
3.9.1.1 ProcessRun.....	91
3.9.1.2 Notification .....	92
3.9.1.2.1 NotificationDetails .....	93
3.9.1.3 PhaseTime.....	93
3.9.1.4 ResourceAudit .....	94
3.9.1.4.1 Logging Machine Data by Using the ResourceAudit .....	95
3.9.1.4.2 Logging Changes in Product Descriptions by Using the ResourceAudit .....	96
3.9.1.5 Created.....	96
3.9.1.6 Deleted.....	97

3.9.1.7 Modified.....	97
3.9.1.8 Spawned.....	97
3.9.1.9 Merged.....	98
<b>3.10 JDF Extensibility . . . . .</b>	<b>98</b>
3.10.1 Namespaces in XML.....	98
3.10.1.1 JDF Namespace .....	99
3.10.1.2 JDF Extension Namespace .....	99
3.10.2 Extending Process Types .....	99
3.10.3 Extending Existing Resources .....	100
3.10.4 Extending NMTOKEN Lists.....	100
3.10.5 Creating New Resources .....	100
3.10.6 Future JDF Extensions .....	100
3.10.7 Maintaining Extensions .....	101
3.10.8 Processing Unknown Extensions.....	101
3.10.9 Derivation of Types in XMLSchema.....	101
<b>3.11 JDF Versioning . . . . .</b>	<b>101</b>
3.11.1 JDF Versioning Requirements.....	101
3.11.2 JDF Version Definition .....	102
3.11.3 JDF Version Policies.....	102
3.11.3.1 JDF Specification Version Policies .....	102
3.11.3.2 JDF Schema Version Policies.....	102
3.11.3.3 JDF Application Version Policies.....	103
3.11.3.3.1 JDF Agent Version Policies .....	103
3.11.3.3.2 JDF Device/Controller Version Policies .....	103
<b>Chapter 4 Life Cycle of JDF . . . . .</b>	<b>105</b>
<b>4.1 Creation and Modification . . . . .</b>	<b>105</b>
4.1.1 Product Intent Constructs .....	105
4.1.1.1 Representation of Product Intent .....	106
4.1.1.2 Representation of Product Binding.....	106
4.1.2 Defining Business Objects Using Intent Resources.....	106
4.1.3 Specification of Delivery of End Products .....	108
4.1.4 Specification of Process Specifics for Product Intent Nodes .....	108
<b>4.2 Process Routing . . . . .</b>	<b>109</b>
4.2.1 Determining Executable Nodes .....	110
4.2.2 Distributing Processing to Work Centers or Devices .....	110
4.2.3 Device / Controller Selection.....	111
<b>4.3 Execution Model . . . . .</b>	<b>111</b>
4.3.1 Serial Processing .....	111
4.3.2 Partial Processing of Nodes with Partitioned Resources.....	112
4.3.3 Overlapping Processing Using Pipes.....	114



4.3.3.1 Pipes of Partitionable Resources.....	116
4.3.3.2 Dynamic Pipes .....	116
4.3.3.3 Comparison of Non-Dynamic and Dynamic Pipes.....	117
4.3.4 Parallel Processing .....	117
4.3.5 Iterative Processing .....	117
4.3.5.1 Informal Iterative Processing.....	118
4.3.5.2 Formal Iterative Processing .....	118
4.3.6 Approval, Quality Control, and Verification .....	118
<b>4.4 Spawning and Merging .....</b>	<b>118</b>
4.4.1 Case 1: Standard Spawning and Merging .....	120
4.4.2 Case 2: Spawning and Merging with Resource Copying.....	121
4.4.2.1 Spawning of Resources with Inter-Resource Links .....	121
4.4.3 Case 3: Parallel Spawning and Merging of Partitioned Resources .....	122
4.4.4 Case 4: Nested Spawning and Merging in Reverse Sequence .....	122
4.4.5 Case 5: Spawning and Merging of Independent Jobs .....	123
4.4.6 Case 6: Simultaneous Spawning and Merging of Multiple Nodes .....	125
<b>4.5 Node and Resource IDs .....</b>	<b>125</b>
<b>4.6 Error Handling .....</b>	<b>126</b>
4.6.1 Classification of Notifications .....	126
4.6.2 Event Description.....	126
4.6.3 Error Logging in the JDF File .....	126
4.6.4 Error Handling via Messaging (JMF) .....	126
<b>4.7 Test Running .....</b>	<b>126</b>
4.7.1 Resource Status During Testrun.....	127
<b>4.8 Capability and Constraint Definitions .....</b>	<b>127</b>
<b>Chapter 5 JDF Messaging with the Job Messaging Format .....</b>	<b>129</b>
<b>5.1 JMF Root .....</b>	<b>129</b>
<b>5.2 JMF Semantics .....</b>	<b>131</b>
5.2.1 Message Families .....	131
5.2.1.1 Query .....	132
5.2.1.2 Response .....	133
5.2.1.3 Signal .....	134
5.2.1.4 Command.....	136
5.2.1.5 Acknowledge .....	137
5.2.2 JMF Handshaking .....	138
5.2.2.1 Single Query/Command Response Communication .....	138
5.2.2.2 Signal .....	138
5.2.2.3 Persistent Channels .....	138
<b>5.3 JMF Messaging Levels .....</b>	<b>140</b>

<b>5.4 Error and Event Messages</b> .....	<b>140</b>
5.4.1 Pure Event Messages.....	141
<b>5.5 Standard Messages</b> .....	<b>141</b>
5.5.1 Controller Registration and Communication Messages.....	142
5.5.1.1 Events.....	142
5.5.1.2 KnownControllers.....	144
5.5.1.3 KnownJDFServices .....	146
5.5.1.4 KnownMessages .....	146
5.5.1.5 RepeatMessages.....	147
5.5.1.6 StopPersistentChannel .....	148
5.5.2 Device/Operator Status and Job Progress Messages.....	149
5.5.2.1 FlushResources .....	150
5.5.2.2 NewJDF .....	150
5.5.2.3 NodeInfo .....	152
5.5.2.4 Occupation .....	154
5.5.2.5 Resource.....	155
5.5.2.6 ResourcePull .....	160
5.5.2.7 Shutdown .....	162
5.5.2.8 Status.....	162
5.5.2.9 Track .....	168
5.5.2.10 WakeUp .....	169
5.5.3 Pipe Control .....	170
5.5.3.1 PipeClose .....	170
5.5.3.2 PipePull.....	170
5.5.3.3 PipePush.....	172
5.5.3.4 PipePause .....	173
<b>5.6 Queue Support</b> .....	<b>173</b>
5.6.1 Queue Entry ID Generation .....	173
5.6.2 Use of QueueFilter in Queue Entry Handling commands.....	173
5.6.3 Queue Entry Handling Commands .....	174
5.6.3.1 AbortQueueEntry .....	176
5.6.3.2 HoldQueueEntry .....	176
5.6.3.3 RemoveQueueEntry .....	177
5.6.3.4 RequestQueueEntry .....	177
5.6.3.5 ResubmitQueueEntry.....	178
5.6.3.6 ResumeQueueEntry .....	178
5.6.3.7 ReturnQueueEntry .....	178
5.6.3.8 SetQueueEntryPosition.....	179
5.6.3.9 SetQueueEntryPriority .....	180
5.6.3.10 SubmitQueueEntry.....	180
5.6.3.11 SuspendQueueEntry.....	182
5.6.4 Global Queue Handling.....	182
5.6.4.1 CloseQueue.....	183
5.6.4.2 FlushQueue .....	184

5.6.4.3 HoldQueue .....	185
5.6.4.4 OpenQueue .....	185
5.6.4.5 QueueEntryStatus .....	185
5.6.4.6 QueueStatus .....	185
5.6.4.7 ResumeQueue .....	186
5.6.4.8 SubmissionMethods .....	186
5.6.5 Queue-Handling Elements .....	187
<b>5.7 Extending Messages .....</b>	<b>190</b>
5.7.1 IfraTrack Support .....	190
<b>Chapter 6 Processes .....</b>	<b>191</b>
<b>6.1 Process Template .....</b>	<b>191</b>
<b>6.2 General Processes .....</b>	<b>191</b>
6.2.1 Approval .....	191
6.2.2 Buffer .....	192
6.2.3 Combine .....	192
6.2.4 Delivery .....	193
6.2.5 ManualLabor .....	193
6.2.6 Ordering .....	193
6.2.7 Packing .....	194
6.2.8 QualityControl .....	194
6.2.9 ResourceDefinition .....	194
6.2.10 Split .....	194
6.2.11 Verification .....	195
<b>6.3 Product Intent Descriptions .....</b>	<b>195</b>
<b>6.4 Prepress Processes .....</b>	<b>196</b>
6.4.1 AssetListCreation .....	196
6.4.2 ColorCorrection .....	197
6.4.3 ColorSpaceConversion .....	197
6.4.4 ContactCopying .....	198
6.4.5 ContoneCalibration .....	198
6.4.6 DBDocTemplateLayout .....	199
6.4.7 DBTemplateMerging .....	199
6.4.8 DigitalDelivery .....	199
6.4.9 FilmToPlateCopying .....	200
6.4.10 FormatConversion .....	200
6.4.11 ImageReplacement .....	200
6.4.12 ImageSetting .....	201
6.4.13 Imposition .....	201
6.4.14 InkZoneCalculation .....	202

6.4.15 Interpreting.....	203
6.4.16 LayoutElementProduction.....	203
6.4.17 LayoutPreparation.....	204
6.4.18 PDFToPSConversion.....	204
6.4.19 Preflight.....	204
6.4.20 PreviewGeneration .....	205
6.4.21 Proofing.....	207
6.4.22 PSToPDFConversion.....	207
6.4.23 Rendering .....	208
6.4.24 RIPing .....	208
6.4.25 Scanning .....	209
6.4.26 Screening.....	209
6.4.27 Separation.....	210
6.4.28 SoftProofing .....	210
6.4.29 Stripping.....	210
6.4.30 Tiling .....	212
6.4.31 Trapping.....	212
<b>6.5 Press Processes . . . . .</b>	<b>213</b>
6.5.1 ConventionalPrinting.....	213
6.5.2 DigitalPrinting.....	214
6.5.3 IDPrinting .....	216
<b>6.6 Postpress Processes . . . . .</b>	<b>216</b>
6.6.1 AdhesiveBinding .....	216
6.6.2 BlockPreparation.....	216
6.6.3 BoxPacking .....	217
6.6.4 Bundling.....	217
6.6.5 CaseMaking .....	218
6.6.6 CasingIn.....	218
6.6.7 ChannelBinding.....	219
6.6.8 CoilBinding.....	219
6.6.9 Collecting .....	219
6.6.10 CoverApplication.....	220
6.6.11 Creasing.....	220
6.6.12 Cutting.....	221
6.6.13 Dividing .....	221
6.6.14 Embossing .....	221
6.6.15 EndSheetGluing.....	222
6.6.16 Feeding .....	222
6.6.17 Folding .....	223
6.6.18 Gathering .....	224

---

6.6.19 Gluing.....	224
6.6.20 HeadBandApplication .....	224
6.6.21 HoleMaking.....	225
6.6.22 Inserting.....	225
6.6.23 Jacketing.....	226
6.6.24 Labeling.....	226
6.6.25 Laminating.....	226
6.6.26 LongitudinalRibbonOperations.....	227
6.6.27 Numbering.....	227
6.6.28 Palletizing.....	227
6.6.29 Perforating.....	227
6.6.30 PlasticCombBinding.....	228
6.6.31 PrintRolling.....	228
6.6.32 RingBinding.....	228
6.6.33 SaddleStitching.....	229
6.6.34 ShapeCutting.....	229
6.6.35 Shrinking.....	229
6.6.36 SideSewing.....	230
6.6.37 SpinePreparation.....	230
6.6.38 SpineTaping.....	230
6.6.39 Stacking.....	230
6.6.40 Stitching.....	231
6.6.41 Strapping.....	231
6.6.42 StripBinding.....	231
6.6.43 ThreadSealing.....	232
6.6.44 ThreadSewing.....	232
6.6.45 Trimming.....	232
6.6.46 WireCombBinding.....	233
6.6.47 Wrapping.....	233
6.6.48 Postpress Processes Structure.....	233
6.6.48.1 Block Production.....	233
6.6.48.1.1 Block Compiling.....	234
6.6.48.1.2 Block Joining.....	234
6.6.48.1.2.1 Single-Leaf Binding Methods.....	234
6.6.48.1.2.2 Loose-Leaf Binding Method.....	234
6.6.48.1.2.2.1 Mechanical Binding Methods.....	234
6.6.48.2 HoleMaking.....	234
6.6.48.3 Laminating.....	235
6.6.48.4 Numbering.....	235
6.6.48.5 Packaging Processes.....	235
6.6.48.6 Processes in Hardcover Book Production.....	235
6.6.48.7 Sheet Processes.....	236

6.6.48.8 Tip-on/in .....	236
6.6.48.9 Trimming .....	236
6.6.48.10 Web Processes .....	236
<b>Chapter 7 Resources .....</b>	<b>237</b>
<b>7.1 Intent Resources .....</b>	<b>237</b>
7.1.1 Intent Resource Span Subelements .....	238
7.1.1.1 Structure of Abstract Span Subelement .....	238
7.1.1.2 Structure of the DurationSpan Subelement .....	239
7.1.1.3 Structure of the EnumerationSpan Subelement .....	240
7.1.1.4 Structure of the IntegerSpan Subelement .....	240
7.1.1.5 Structure of the NameSpan Subelement .....	240
7.1.1.5.1 Specifying New Values in a NameSpan Subelement .....	240
7.1.1.6 Structure of the NumberSpan Subelement .....	241
7.1.1.7 Structure of the OptionSpan Subelement .....	241
7.1.1.8 Structure of the ShapeSpan Subelement .....	241
7.1.1.9 Structure of the StringSpan Subelement .....	241
7.1.1.10 Structure of the TimeSpan Subelement .....	242
7.1.1.11 Structure of the XYPairSpan Subelement .....	242
7.1.2 ArtDeliveryIntent .....	242
7.1.3 BindingIntent .....	247
7.1.4 ColorIntent .....	255
7.1.5 DeliveryIntent .....	258
7.1.6 EmbossingIntent .....	263
7.1.7 FoldingIntent .....	264
7.1.8 HoleMakingIntent .....	265
7.1.9 InsertingIntent .....	265
7.1.10 LaminatingIntent .....	267
7.1.11 LayoutIntent .....	267
7.1.12 MediaIntent .....	270
7.1.13 NumberingIntent .....	275
7.1.14 PackingIntent .....	275
7.1.15 ProductionIntent .....	276
7.1.16 ProofingIntent .....	277
7.1.17 ScreeningIntent .....	279
7.1.18 ShapeCuttingIntent .....	279
7.1.19 SizeIntent .....	280
<b>7.2 Process Resources .....</b>	<b>280</b>
7.2.1 Process Resource Template .....	280
7.2.2 Address .....	281
7.2.3 AdhesiveBindingParams .....	282
7.2.4 ApprovalParams .....	282

---

7.2.5 ApprovalSuccess .....	283
7.2.6 Assembly .....	283
7.2.7 AssetListCreationParams .....	284
7.2.8 AutomatedOverPrintParams .....	285
7.2.9 BinderySignature .....	285
7.2.10 BlockPreparationParams .....	287
7.2.11 BoxPackingParams.....	288
7.2.12 BufferParams .....	288
7.2.13 Bundle.....	289
7.2.14 BundlingParams.....	290
7.2.15 ByteMap.....	291
7.2.16 CaseMakingParams.....	292
7.2.17 CasingInParams .....	293
7.2.18 ChannelBindingParams .....	294
7.2.19 CIELABMeasuringField.....	295
7.2.20 CoilBindingParams .....	296
7.2.21 CollectingParams.....	296
7.2.22 Color .....	297
7.2.23 ColorantAlias.....	303
7.2.24 ColorantControl.....	303
7.2.25 ColorControlStrip .....	306
7.2.26 ColorCorrectionParams .....	307
7.2.27 ColorMeasurementConditions .....	309
7.2.28 ColorPool .....	310
7.2.29 ColorSpaceConversionParams.....	311
7.2.30 ColorSpaceConversionOp .....	313
7.2.31 ComChannel.....	321
7.2.32 Company.....	323
7.2.33 Component .....	323
7.2.34 Contact.....	327
7.2.35 ContactCopyParams.....	328
7.2.36 ConventionalPrintingParams .....	328
7.2.37 CostCenter.....	331
7.2.38 CoverApplicationParams .....	331
7.2.39 CreasingParams .....	332
7.2.40 CutBlock.....	333
7.2.41 CutMark .....	334
7.2.42 CuttingParams .....	335
7.2.43 DBMergeParams .....	336
7.2.44 DBRules.....	337

7.2.45 DBSchema.....	337
7.2.46 DBSelection .....	338
7.2.47 DeliveryParams.....	338
7.2.48 DensityMeasuringField .....	340
7.2.49 DevelopingParams.....	341
7.2.50 Device .....	342
7.2.51 DeviceMark .....	344
7.2.52 DeviceNSpace .....	344
7.2.53 DigitalDeliveryParams.....	345
7.2.54 DigitalMedia .....	346
7.2.55 DigitalPrintingParams .....	347
7.2.55.1 Coordinate systems in DigitalPrinting.....	347
7.2.56 Disjointing .....	349
7.2.57 Disposition .....	350
7.2.58 DividingParams.....	351
7.2.59 ElementColorParams.....	351
7.2.60 EmbossingParams.....	352
7.2.61 Employee .....	353
7.2.62 EndSheetGluingParams .....	353
7.2.63 ExposedMedia .....	355
7.2.64 FeedingParams.....	356
7.2.65 FileSpec.....	359
7.2.66 FitPolicy .....	365
7.2.67 Fold .....	366
7.2.68 FoldingParams.....	366
7.2.69 FontParams .....	370
7.2.70 FontPolicy .....	370
7.2.71 FormatConversionParams .....	371
7.2.72 GatheringParams.....	374
7.2.73 GlueApplication.....	375
7.2.74 GluingParams .....	376
7.2.75 GlueLine.....	376
7.2.76 HeadBandApplicationParams .....	377
7.2.77 Hole.....	378
7.2.78 HoleLine.....	378
7.2.79 HoleList.....	380
7.2.80 HoleMakingParams.....	380
7.2.81 IdentificationField .....	382
7.2.82 IDPrintingParams.....	383
7.2.83 ImageCompressionParams .....	383



---

7.2.84 ImageReplacementParams .....	388
7.2.85 ImageSetterParams .....	390
7.2.86 Ink .....	391
7.2.87 InkZoneCalculationParams .....	392
7.2.88 InkZoneProfile .....	393
7.2.89 InsertingParams .....	394
7.2.90 InsertSheet .....	396
7.2.91 InterpretedPDLData .....	399
7.2.92 InterpretingParams .....	400
7.2.93 JacketingParams .....	402
7.2.94 JobField .....	403
7.2.95 LabelingParams .....	404
7.2.96 LaminatingParams .....	404
7.2.97 Layout .....	405
7.2.98 LayoutElement .....	406
7.2.99 LayoutPreparationParams .....	408
7.2.100 LongitudinalRibbonOperationParams .....	417
7.2.101 ManualLaborParams .....	417
7.2.102 Media .....	417
7.2.103 MediaSource .....	422
7.2.104 MISDetails .....	422
7.2.105 NumberingParams .....	423
7.2.106 ObjectResolution .....	424
7.2.107 OrderingParams .....	424
7.2.108 PackingParams .....	425
7.2.109 PageList .....	425
7.2.110 PalletizingParams .....	427
7.2.111 Pallet .....	428
7.2.112 PDFToPSConversionParams .....	428
7.2.113 PDLResourceAlias .....	431
7.2.114 PerforatingParams .....	432
7.2.115 Person .....	433
7.2.116 PlaceholderResource .....	433
7.2.117 PlasticCombBindingParams .....	433
7.2.118 PlateCopyParams .....	434
7.2.119 PreflightAnalysis .....	434
7.2.120 PreflightInventory .....	434
7.2.121 PreflightParams .....	434
7.2.122 PreflightProfile .....	436
7.2.123 PreflightReport .....	437

7.2.124 PreflightReportRulePool .....	440
7.2.125 Preview .....	442
7.2.126 PreviewGenerationParams .....	444
7.2.127 PrintCondition .....	445
7.2.128 PrintRollingParams .....	446
7.2.129 ProofingParams .....	447
7.2.130 PSToPDFConversionParams .....	447
7.2.131 QualityControlParams .....	452
7.2.132 QualityControlResult .....	452
7.2.133 RegisterMark .....	453
7.2.134 RegisterRibbon .....	454
7.2.135 RenderingParams .....	455
7.2.136 ResourceDefinitionParams .....	455
7.2.137 RingBindingParams .....	456
7.2.138 RollStand .....	457
7.2.139 RunList .....	458
7.2.140 SaddleStitchingParams .....	463
7.2.141 ScanParams .....	463
7.2.142 ScavengerArea .....	465
7.2.143 ScreeningParams .....	465
7.2.144 SeparationControlParams .....	467
7.2.145 SeparationSpec .....	468
7.2.146 ShapeCuttingParams .....	468
7.2.147 Sheet .....	469
7.2.148 ShrinkingParams .....	470
7.2.149 SideSewingParams .....	470
7.2.150 SpinePreparationParams .....	470
7.2.151 SpineTapingParams .....	472
7.2.152 StackingParams .....	473
7.2.153 StitchingParams .....	475
7.2.154 Strap .....	478
7.2.155 StrappingParams .....	478
7.2.156 StripBindingParams .....	479
7.2.157 StrippingParams .....	479
7.2.158 Surface .....	483
7.2.159 ThreadSealingParams .....	489
7.2.160 ThreadSewingParams .....	490
7.2.161 Tile .....	491
7.2.162 Tool .....	492
7.2.163 TransferCurve .....	493

7.2.164 TransferCurvePool.....	493
7.2.165 TransferFunctionControl .....	494
7.2.166 TrappingDetails.....	494
7.2.167 TrappingParams .....	495
7.2.168 TrapRegion .....	499
7.2.169 TrimmingParams.....	499
7.2.170 VerificationParams.....	500
7.2.171 WireCombBindingParams.....	501
7.2.172 WrappingParams .....	502
<b>7.3 Device Capability Definitions . . . . .</b>	<b>502</b>
7.3.1 Structure of the DeviceCap Subelement.....	502
7.3.1.1 Structure of the ActionPool Subelement .....	504
7.3.1.1.1 Structure of the Action Subelement .....	504
7.3.1.2 Structure of the DevCaps Subelement .....	505
7.3.1.2.1 Structure of the Loc Subelement .....	507
7.3.1.2.2 Structure of the DevCap Subelement .....	507
7.3.1.2.2.1 Structure of the Abstract State Subelement .....	508
7.3.1.2.2.1.1 Structure of the BooleanState Subelement.....	511
7.3.1.2.2.1.2 Structure of the DateTimeState Subelement.....	511
7.3.1.2.2.1.3 Structure of the DurationState Subelement.....	512
7.3.1.2.2.1.4 Structure of the EnumerationState Subelement .....	512
7.3.1.2.2.1.5 Structure of the IntegerState Subelement.....	513
7.3.1.2.2.1.6 Structure of the MatrixState Subelement .....	514
7.3.1.2.2.1.7 Structure of the NameState Subelement .....	515
7.3.1.2.2.1.8 Structure of the NumberState Subelement.....	516
7.3.1.2.2.1.9 Structure of the PDFPathState Subelement .....	517
7.3.1.2.2.1.10 Structure of the RectangleState Subelement.....	517
7.3.1.2.2.1.11 Structure of the ShapeState Subelement .....	518
7.3.1.2.2.1.12 Structure of the StringState Subelement .....	519
7.3.1.2.2.1.13 Structure of the XYPairState Subelement.....	520
7.3.1.3 Structure of the DisplayGroupPool Subelement.....	520
7.3.1.3.1 Structure of the DisplayGroup Subelement .....	521
7.3.1.4 Structure of the FeaturePool Subelement .....	521
7.3.1.5 Structure of the MacroPool Subelement.....	522
7.3.1.5.1 Structure of the macro Subelement .....	522
7.3.1.5.1.1 Structure of the choice Subelement .....	522
7.3.1.5.1.1.1 Structure of the otherwise Subelement .....	523
7.3.1.5.1.1.2 Structure of the when Subelement .....	523
7.3.1.5.1.2 Structure of the set Subelement .....	523
7.3.1.5.1.2.1 Structure of the FeatureAttribute Subelement .....	523
7.3.1.5.1.3 Structure of the call Subelement .....	523
7.3.1.6 Structure of the Performance Subelement .....	524
7.3.1.7 Structure of the TestPool Subelement .....	524
7.3.1.7.1 Structure of the Test Subelement .....	524
7.3.1.7.1.1 Structure of the abstract Term Subelement.....	524
7.3.1.7.1.2.1 Boolean Operators.....	526
7.3.1.7.1.2.2 Evaluation Subelements.....	527
7.3.2 Examples of Device Capabilities.....	532

<b>7.4 Concept of the Preflight Process</b> .....	<b>538</b>
7.4.1 Object Classes.....	539
7.4.1.1 Checking for the Presence of a Property.....	540
7.4.1.2 Basic tests on set of objects .....	541
7.4.2 Properties.....	541
7.4.2.1 Annotation Properties .....	542
7.4.2.2 Box Properties.....	542
7.4.2.3 Class Properties.....	544
7.4.2.4 Colorant Properties .....	545
7.4.2.5 Document Properties.....	545
7.4.2.6 Fill Properties.....	548
7.4.2.7 Font Properties.....	549
7.4.2.8 Graphic Properties .....	550
7.4.2.9 Image Properties .....	551
7.4.2.10 Logical Properties .....	553
7.4.2.11 PageBox Properties.....	553
7.4.2.12 Pages Properties .....	554
7.4.2.13 PDLObject Properties .....	555
7.4.2.14 Reference Properties .....	555
7.4.2.15 Shading Properties .....	555
7.4.2.16 Stroke Properties.....	556
7.4.2.17 Text Properties.....	556
7.4.2.18 Vector Properties .....	557
<b>Chapter 8 Building a System Around JDF</b> .....	<b>559</b>
<b>8.1 Implementation Considerations and Guidelines</b> .....	<b>559</b>
<b>8.2 JDF and JMF Interchange Protocol</b> .....	<b>559</b>
8.2.1 File-Based Protocol (JDF + JMF) .....	559
8.2.1.1 JMF Transport Using The File Protocol.....	559
8.2.2 HTTP-Based Protocol (JDF + JMF) .....	560
8.2.2.1 Protocol Implementation Details .....	560
<b>8.3 JDF Packaging</b> .....	<b>560</b>
8.3.1 MIME Basics .....	560
8.3.2 MIME Types and File Extensions .....	561
8.3.2.1 MIME Fields.....	561
8.3.2.1.1 Content Type .....	561
8.3.2.1.2 Content ID .....	561
8.3.2.1.3 Content Length .....	561
8.3.2.1.4 Content Transfer Encoding .....	562
8.3.2.1.5 Content Disposition .....	562
8.3.2.2 Example Packaging of Individual JDF/JMF files in MIME.....	562
8.3.2.3 CID URL Scheme.....	562
8.3.2.4 Ordering of JDF/JMF in MIME Multipart/Related .....	563
<b>8.4 MIS Requirements</b> .....	<b>564</b>

<b>8.5 Interoperability Conformance Specifications</b> .....	<b>564</b>
<b>Appendix A Encoding</b> .....	<b>565</b>
<b>A.1 Notes About Encoding</b> .....	<b>565</b>
A.1.1 Ranges and RangeLists .....	565
A.1.2 Whitespace .....	565
A.1.3 Infinity Limits .....	565
<b>A.2 Simple Types — Attribute Values</b> .....	<b>565</b>
A.2.1 boolean .....	565
A.2.2 CMYKColor .....	565
A.2.3 date .....	566
A.2.4 dateTime .....	566
A.2.5 DateTimeRange .....	566
A.2.6 DateTimeRangeList .....	566
A.2.7 double .....	566
A.2.8 DoubleList .....	567
A.2.9 DoubleRange .....	567
A.2.10 DoubleRangeList .....	567
A.2.11 duration .....	567
A.2.12 DurationRange .....	567
A.2.13 DurationRangeList .....	568
A.2.14 gYearMonth .....	568
A.2.15 hexBinary .....	568
A.2.16 ID .....	568
A.2.17 IDREF .....	568
A.2.18 IDREFS .....	568
A.2.19 integer .....	569
A.2.20 IntegerList .....	569
A.2.21 IntegerRange .....	569
A.2.22 IntegerRangeList .....	569
A.2.23 LabColor .....	569
A.2.24 language .....	570
A.2.25 matrix .....	570
A.2.26 NameRange .....	570
A.2.27 NameRangeList .....	570
A.2.28 NMTOKEN .....	571
A.2.29 NMTOKENS .....	571
A.2.30 PDFPath .....	571
A.2.31 rectangle .....	571
A.2.32 RectangleRange .....	572

A.2.33 RectangleRange List .....	572
A.2.34 regExp .....	572
A.2.35 shape .....	572
A.2.36 ShapeRange .....	572
A.2.37 ShapeRangeList .....	573
A.2.38 sRGBColor .....	573
A.2.39 string .....	573
A.2.40 TimeRange .....	573
A.2.41 TransferFunction .....	573
A.2.42 URI .....	574
A.2.43 URL .....	574
A.2.44 XYPair .....	574
A.2.45 XYPairRange .....	574
A.2.46 XYPairRangeList .....	574
A.2.47 XPath .....	575
<b>A.3 Enumerations and Lists .....</b>	<b>575</b>
A.3.1 enumeration .....	575
A.3.2 enumerations .....	575
A.3.3 Defined JDF enumeration Data Types .....	575
A.3.3.1 JDFJMFVersion .....	575
A.3.3.2 NamedColor .....	576
A.3.3.3 Orientation .....	576
A.3.3.4 Side .....	577
A.3.3.5 WorkStyle .....	577
A.3.4 XYRelation .....	577
<b>A.4 JDF File Formats .....</b>	<b>578</b>
A.4.1 PNG Image Format .....	578
<b>Appendix B Schema .....</b>	<b>579</b>
<b>B.1 Using xsi:type .....</b>	<b>579</b>
B.1.1 Using xsi:type with JDF Nodes .....	579
B.1.2 Using xsi:type with JMF Messages .....	580
<b>Appendix C Converting PJTF to JDF .....</b>	<b>581</b>
<b>C.1 PJTF Object Conversion .....</b>	<b>581</b>
C.1.1 Accounting .....	581
C.1.2 Address .....	581
C.1.3 Analysis .....	581
C.1.4 AuditObject .....	581
C.1.5 ColorantAlias .....	582
C.1.6 ColorantControl .....	582

C.1.7 ColorantDetails.....	582
C.1.8 ColorantZoneDetails.....	582
C.1.9 ColorSpaceSubstitute.....	582
C.1.10 Delivery .....	582
C.1.11 DeviceColorant.....	582
C.1.12 Document.....	582
C.1.13 Finishing.....	584
C.1.14 FontPolicy.....	584
C.1.15 InsertPage.....	584
C.1.16 InsertSheet.....	584
C.1.17 Inventory.....	584
C.1.18 Ticket.....	584
C.1.19 JobTicketContents.....	584
C.1.20 JTFile.....	587
C.1.21 Layout.....	587
C.1.22 Media.....	587
C.1.23 MediaSource .....	587
C.1.24 MediaUsage .....	587
C.1.25 PageRange .....	587
C.1.26 PlacedObject.....	589
C.1.27 PlaneOrder.....	589
C.1.28 Preflight .....	589
C.1.29 PreflightConstraint.....	589
C.1.30 PreflightDetail.....	589
C.1.31 PreflightInstance.....	590
C.1.32 PreflightInstanceDetail .....	590
C.1.33 PreflightResults .....	590
C.1.34 PrintLayout .....	590
C.1.35 Profile .....	590
C.1.36 Rendering.....	591
C.1.37 ResourceAlias .....	591
C.1.38 Scheduling.....	591
C.1.39 Signature .....	592
<b>C.2 Sheet .....</b>	<b>592</b>
C.2.1 SlipSheet.....	592
C.2.2 Surface .....	592
C.2.3 Tile.....	592
C.2.4 Trapping .....	592
C.2.5 TrappingDetails .....	592
C.2.6 TrappingParameters.....	592

C.2.7 TrapRegion.....	592
<b>C.3 Translating Values</b> .....	<b>592</b>
<b>C.4 Translating the Contents Hierarchy</b> .....	<b>593</b>
<b>C.5 Representing Pages</b> .....	<b>593</b>
<b>C.6 Representing Preseparated Documents</b> .....	<b>594</b>
<b>C.7 Representing Inherited Characteristics</b> .....	<b>594</b>
<b>C.8 Translating Layout</b> .....	<b>594</b>
<b>C.9 Translating PrintLayout</b> .....	<b>594</b>
<b>C.10 Translating Trapping</b> .....	<b>595</b>
<b>Appendix D Converting PPF to JDF</b> .....	<b>597</b>
<b>D.1 Converting PPF Data Types</b> .....	<b>598</b>
<b>D.2 PPF Product Definitions</b> .....	<b>598</b>
D.2.1 Comparison of the PPF Component to the JDF Component .....	599
D.2.2 Collecting.....	599
D.2.3 Gathering.....	599
D.2.4 ThreadSewing .....	600
D.2.5 SaddleStitching .....	600
D.2.6 Stitching.....	600
D.2.7 SideSewing .....	600
D.2.8 EndSheetGluing .....	600
D.2.9 AdhesiveBinding.....	600
D.2.10 Trimming .....	601
D.2.11 GluingIn.....	601
D.2.12 Folding.....	602
<b>D.3 PPF Sheet Structure</b> .....	<b>603</b>
D.3.1 Administration Data .....	605
D.3.2 Preview Images.....	606
D.3.3 Transfer Curves.....	607
D.3.4 Register Marks .....	607
D.3.5 Color and Ink Control .....	607
D.3.6 Cutting Data .....	608
D.3.7 Folding Data .....	609
D.3.8 Comments and Annotations.....	609
D.3.9 Private Data and Content.....	609
<b>Appendix E Modeling IfraTrack in JDF</b> .....	<b>611</b>
<b>E.1 IFRA Objects and JDF Nodes</b> .....	<b>611</b>



E.1.1 Object Identification .....	611
E.1.2 IFRA Object Hierarchy.....	611
E.1.3 Object States .....	611
E.1.4 Deadlines and Scheduling.....	612
<b>E.2 JMF Messages that Translate IfraTrack Messages .....</b>	<b>612</b>
<b>Appendix F Mapping between JDF and IPP .....</b>	<b>613</b>
<b>F.1 IPP References .....</b>	<b>613</b>
<b>Appendix G StatusDetails Supported Strings .....</b>	<b>615</b>
<b>Appendix H ModuleType Supported Strings .....</b>	<b>617</b>
<b>Appendix I Supported Error Codes in JMF and Notification elements .</b>	<b>619</b>
<b>Appendix J NotificationDetails .....</b>	<b>621</b>
<b>J.1 Predefined NotificationDetails .....</b>	<b>621</b>
J.1.1 Barcode.....	621
J.1.2 FCNKey.....	621
J.1.3 SystemTimeSet.....	621
J.1.4 CounterReset.....	621
J.1.5 Error .....	622
J.1.6 Event.....	622
<b>Appendix K MessageEvents Values .....</b>	<b>623</b>
<b>Appendix L Color Adjustment Attribute Description and Usage . . .</b>	<b>625</b>
<b>L.1 Adjustment Using Direct Attributes .....</b>	<b>625</b>
<b>L.2 Adjustment using ICC Profile Attributes .....</b>	<b>626</b>
<b>L.3 Adjustment using an ICC Abstract Profile Attribute .....</b>	<b>626</b>
<b>L.4 Adjustment using an ICC DeviceLink Profile Attribute .....</b>	<b>626</b>
<b>Appendix M North American Media Weight Explained .....</b>	<b>627</b>
<b>Appendix N Media Sizes. ....</b>	<b>629</b>
<b>Appendix O Input Tray and Output Bin Names .....</b>	<b>633</b>
<b>Appendix P FileSpec Attribute Examples for MimeType and MimeType-</b>	

<b>Version Attributes</b> .....	<b>635</b>
<b>Appendix Q FileSpec mimeType, URL, and Compression attributes, and Container subelement</b> .....	<b>641</b>
Q.1 FileSpec attribute value examples .....	641
Q.2 Corresponding XML examples .....	642
Q.3 Additional examples showing partitioning of FileSpec .....	643
Q.4 Example of an Intent Job Ticket with a doubly nested ZIP packaging file ..	647
<b>Appendix R Resolving RunList/@Directory and FileSpec/@URL URI references</b> .....	<b>649</b>
R.1 Semantics of the RunList/@Directory attribute .....	649
<b>Appendix S AppOS and OSVersion Attributes</b> .....	<b>651</b>
<b>Appendix T References</b> .....	<b>653</b>
<b>Appendix U JDF/CIP4 Hole Pattern Catalog</b> .....	<b>663</b>
<b>Appendix V Examples</b> .....	<b>669</b>
<b>V.1 Brief Example</b> .....	<b>669</b>
V.1.1 Before Processing .....	669
V.1.2 After Processing .....	670
<b>V.2 Product JDF</b> .....	<b>671</b>
<b>V.3 Spawning and Merging</b> .....	<b>672</b>
V.3.1 Example 2 Component JDF before Spawning .....	672
V.3.2 Example 2 Component JDF Parent after spawning the cover node.....	673
V.3.3 Example 2 Component JDF spawned node .....	674
V.3.4 Example 2 Component JDF after merging .....	674
V.3.5 Example of a Partitioned ImageSetting Node before Spawning.....	676
V.3.6 The Spawned Cyan Partition of the ImageSetting Node .....	676
V.3.7 The Root Partitioned ImageSetting Node after Spawning .....	677
V.3.8 The Merged ImageSetting Node .....	677
<b>V.4 Conversion of PJTF to JDF</b> .....	<b>678</b>
V.4.1 PJTF input .....	678
V.4.2 JDF output .....	681
<b>V.5 Conversion of PPF to JDF</b> .....	<b>682</b>
<b>V.6 RunList</b> .....	<b>689</b>
<b>V.7 Messages</b> .....	<b>691</b>

V.7.1 Simple KnownMessages .....	691
V.7.2 Simple persistent channel .....	692
<b>V.8 Stripping .....</b>	<b>693</b>
V.8.1 Using Position.....	693
V.8.2 Multiple BinderySignatures.....	693
V.8.3 Multisection BinderySignatures .....	694
V.8.4 Multiple job parts in one imposition .....	694
V.8.5 FoldOuts .....	695
V.8.6 Multiple Web Layout.....	695
V.8.7 Stripping Process .....	697
<b>V.9 DigitalDelivery Examples .....</b>	<b>699</b>
<b>Appendix W New, Deprecated, Modified, Illegal, &amp; Removed Items</b>	<b>705</b>
<b>W.1 Compatibility Warnings .....</b>	<b>706</b>
<b>W.2 New Items .....</b>	<b>706</b>
<b>W.3 Deprecated Items .....</b>	<b>711</b>
<b>W.4 Modified Items .....</b>	<b>716</b>
<b>W.5 Clarified Items .....</b>	<b>718</b>
<b>W.6 New/Modified Attributes and Elements .....</b>	<b>721</b>
W.6.1 Structure of JDF Nodes and Jobs .....	721
W.6.2 JDF Messaging with the Job Messaging Format.....	723
W.6.3 Processes.....	727
W.6.4 Resources .....	727
<b>Appendix X Deprecated Processes, Resources, and JMF Messaging Elements .....</b>	<b>743</b>
<b>X.1 Deprecated Processes .....</b>	<b>743</b>
X.1.1 Packing.....	743
X.1.2 FilmToPlateCopying .....	743
X.1.3 PreflightAnalysis .....	744
X.1.4 PreflightInventory.....	746
X.1.5 PreflightProfile .....	746
X.1.6 Proofing .....	747
X.1.7 SoftProofing.....	748
X.1.8 IDPrinting.....	749
X.1.9 AdhesiveBinding.....	750
X.1.10 Dividing.....	750
X.1.11 LongitudinalRibbonOperations .....	751
X.1.12 SaddleStitching.....	751

X.1.13 SideSewing.....	751
<b>X.2 Deprecated Resources . . . . .</b>	<b>752</b>
X.2.1 BindingIntent Deprecated Subelements.....	752
X.2.2 SizeIntent.....	753
X.2.3 AdhesiveBindingParams .....	753
X.2.4 DividingParams .....	755
X.2.5 IDPrintingParams .....	755
X.2.6 LongitudinalRibbonOperationParams.....	765
X.2.7 MediaSource .....	766
X.2.8 PackingParams .....	767
X.2.9 PlateCopyParams.....	768
X.2.10 ProofingParams.....	768
X.2.11 SaddleStitchingParams .....	770
X.2.12 SideSewingParams .....	771
<b>X.3 JMF Messaging Elements . . . . .</b>	<b>772</b>
X.3.1 KnownJDFServices .....	772
X.3.2 QueueEntryStatus .....	773
<b>Appendix Y Table of Tables . . . . .</b>	<b>775</b>
<b>Appendix Z Terminology Usage . . . . .</b>	<b>781</b>

## List of Figures

Figure 1.1 Handling of Default Values of JDF Attributes. . . . .	8
Figure 2.1 Example of JDF and JMF workflow interactions . . . . .	16
Figure 2.2 JDF tree structure . . . . .	17
Figure 2.3 Example of a hierarchical tree structure of JDF nodes . . . . .	19
Figure 2.4 Example of a process chain linked by input and output resources . . . . .	20
Figure 2.5 Standard coordinate system . . . . .	21
Figure 2.6 Relation between resource and process coordinate systems . . . . .	22
Figure 2.7 Layout of simple saddle stitched brochure (product example) . . . . .	25
Figure 2.8 Surface coordinate system . . . . .	26
Figure 2.9 Press coordinate system used for sheet-fed printing . . . . .	26
Figure 2.10 Press coordinate system used for web printing . . . . .	27
Figure 2.11 Coordinate systems after Folding (product example) . . . . .	27
Figure 2.12 Coordinate systems after Collecting (product example) . . . . .	28
Figure 2.13 Examples of Transformations and Coordinate Systems in JDF. . . . .	29
Figure 2.14 Transforming a point (example) . . . . .	31
Figure 3.1 Structure of the JDF Node . . . . .	34
Figure 3.2 Structure of JDF Generic Contents . . . . .	37
Figure 3.3 Job hierarchy with process, process group, and product intent nodes . . . . .	43
Figure 3.4 Structure of the abstract resource types . . . . .	56
Figure 3.5 Resource Links and ResourceRefs . . . . .	61
Figure 3.6 Nodes linked by a resource . . . . .	61
Figure 3.7 Structure of the abstract ResourceLink types . . . . .	63
Figure 3.8 Amount Handling . . . . .	72
Figure 3.9 Splitting and combining physical resources . . . . .	88
Figure 3.10 Structure of Audit element types derived from the abstract Audit type . . . . .	90
Figure 4.1 Simplified PrintTalk workflow (negotiation phase) . . . . .	107
Figure 4.2 Life Cycle of a JDF node . . . . .	110
Figure 4.3 Example of a simple process chain linked by resources . . . . .	111
Figure 4.4 Example of a Pipe resource linking two processes . . . . .	114
Figure 4.5 Example of status transitions in case of overlapping processing . . . . .	115
Figure 4.6 The spawning and merging mechanism and its phases . . . . .	119
Figure 4.7 JDF node structure that requires resource copying during spawning and merging	121
Figure 4.8 Example for a JDF node structure with nested spawning . . . . .	123
Figure 4.9 Example of the spawning and merging of independent jobs . . . . .	124
Figure 4.10 Parameter Space in device Capabilities . . . . .	127
Figure 5.1 Contents of a JMF root element and the message families . . . . .	131
Figure 5.2 Interaction of Messages with a subscription . . . . .	132
Figure 5.3 Interaction of Command and Acknowledge Messages . . . . .	137
Figure 5.4 Mechanism of a PipePull message . . . . .	171
Figure 5.5 Mechanism of a PipePush message . . . . .	172
Figure 5.6 JMF QueueEntry Status Transition Diagram . . . . .	175
Figure 5.7 Effects of the global queue messages on the queue Status . . . . .	183
Figure 6.1 Worst case scenario for area coverage calculation . . . . .	206
Figure 6.2 Packaging Process Coordinate System . . . . .	235
Figure 7.1 CaseMakingParams . . . . .	292
Figure 7.2 Parameters and Coordinate System for CasingIn . . . . .	293

Figure 7.3 Parameters used for channel binding .....	294
Figure 7.4 Coordinate systems used for collecting .....	297
Figure 7.5 Terms and definitions for components .....	324
Figure 7.6 Parameters and coordinate system for cover application .....	332
Figure 7.7 Parameters and coordinate system used for end-sheet gluing .....	354
Figure 7.8 Names of the reference edges of a sheet in the FoldingParams resource .....	366
Figure 7.9 Fold Catalog part 1 .....	368
Figure 7.10 Fold Catalog part 2 .....	369
Figure 7.11 Coordinate system used for gathering .....	374
Figure 7.12 Parameters and coordinate system for glue application .....	375
Figure 7.13 Parameters and Coordinate system used for Inserting .....	394
Figure 7.14 Parameters and Coordinate System for Jacketing .....	402
Figure 7.15 PRGroup Structure .....	438
Figure 7.16 Parameters and Coordinate System for BlockPreparation .....	454
Figure 7.17 Parameters and coordinate systems for the SpinePreparation process .....	471
Figure 7.18 Parameters and coordinate system for the SpineTaping process .....	472
Figure 7.19 Odd Count Handling for Bundling .....	474
Figure 7.20 Staple shapes .....	475
Figure 7.21 Parameters and coordinate system used for saddle stitching .....	476
Figure 7.22 Parameters and coordinate system used for stitching .....	476
Figure 7.23 Definition of margins in StripCellParams .....	482
Figure 7.24 Parameters and coordinate system used for thread sewing .....	490
Figure 7.25 Parameters and coordinate system used for side sewing .....	490
Figure 7.26 Parameters and coordinate system used for trimming .....	499
Figure D.1 JDF node of a CIP3 product structure .....	597
Figure D.2 JDF representation of sheets .....	604
Figure X.1 Parameters and coordinate system for glue application .....	754
Figure X.2 Staple shapes .....	770
Figure X.3 Parameters and coordinate system used for side sewing .....	771

---




## JDF Preface and User Overview

This specification is immense ... there little doubt about that ... but it is also a keystone standard for the future of graphic communications. The members of CIP4 believe that users and developers alike should 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 E-mail 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 call-outs 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 you may want to consider as you formulate your JDF implementation plan. Look for these call-out icons:

### Call-Out Icon Usage


Icon	Call-Out 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 second version of JDF (v.1.1a) 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 may provide include:

- A common language for describing a print job across enterprises, departments, and software and systems;
- A tool for verifying the accuracy and completeness of job tools;
- A systems interface language that can be used to benchmark the performance of new equipment (hardware and software) and that can reduce the cost of expensive custom integration for printers, prepress services, and others;

- A basis for total workflow automation that incorporates all aspects of production: human, machine, and computer;
- A standard that can be applied to eliminate wasteful re-keying and redundancy of information; and
- A common computer language for printing and related industries as well as a platform for more effective communication.

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.



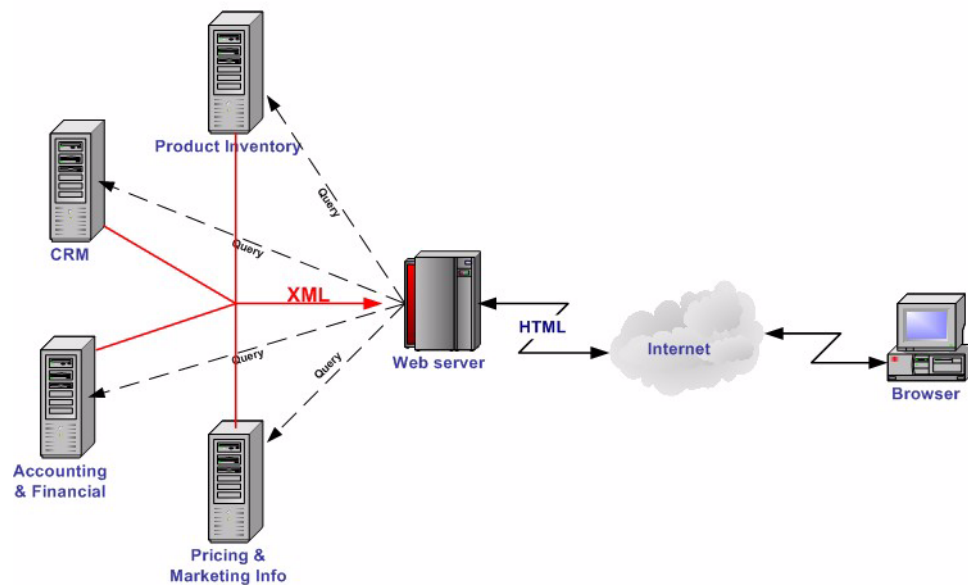
### Implementation Strategy

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

**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 "middleware" element to create the "glue" that ties integrated systems together.

For instance, complex systems such as enterprise resource planning (ERP), data warehousing, or E-commerce systems often tap into numerous legacy databases and application environments. A manager may wish to have a "view" of corporate information that is actually an aggregate of information that may 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 may play in your JDF-enabled workflow include:

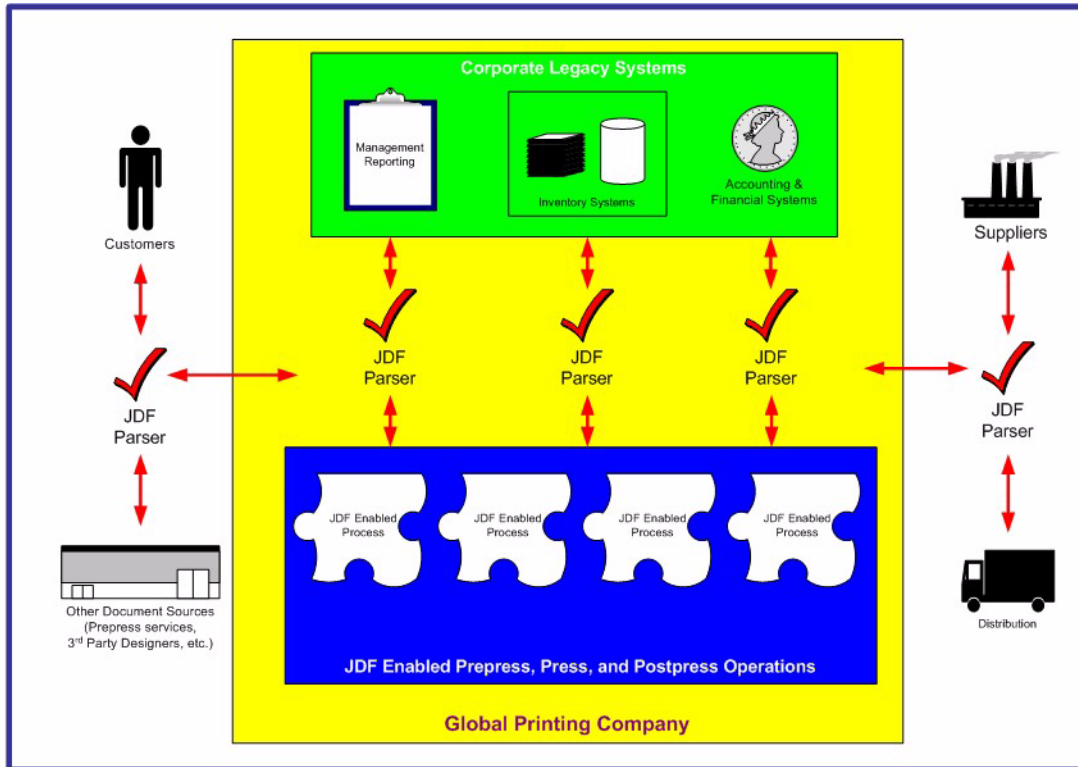
- 1 Acceptance checking of client job tickets;
- 2 Validation of JDF prior to or following transformation of data into and out of databases;
- 3 Ensuring that source job information is collected as a document is created (embedded in document layout software);
- 4 Determining if equipment reads and writes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



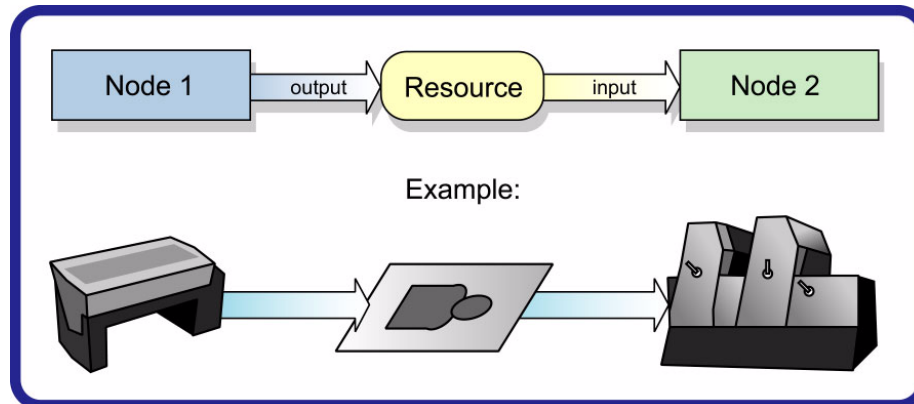
**The JDF Concept.** The JDF schema is quite complex and detailed—something best left to programmers, MIS personnel, and XML experts. But the language and concepts behind JDF are quite simple and straightforward. The schema itself can be downloaded from the CIP4 Website, but is not part of this specification. 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 required 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 may 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 should 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. There are a variety of XML support tools available on the market to address the databases underlying all MIS systems.



### 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.



### XML and Databases

To learn more about how XML and database work together, check out the white papers and tutorials available from XML.org at [http://www.xml.org/xml/resources\\_focus\\_rdbms.shtml](http://www.xml.org/xml/resources_focus_rdbms.shtml).

**Changes to JDF 1.2, ICS Documents, and Certification.** JDF 1.2 includes both some wholly new material, as well as many improvements and refinements to JDF 1.1 and JDF 1.1a. A complete catalog of changes can be found in “New, Deprecated, Modified, Illegal, & Removed Items” on page 705. You will also find [Modified in JDF 1.2](#) and [New in JDF 1.2](#) flags throughout this document. A few of the more “administrative” changes that you may find important include:

- All number, NumberList, NumberRange, and NumberRangeList data types were changed to double, DoubleList, DoubleRange, and DoubleRangeList data types throughout the document. The old “number” allowed for interpretation as either an integer or a double, which could cause compatibility issues. Note that this change has no effect on the encoding. Integer is still used, but with the elimination of all types of “number” data types, this source of potential confusion is eliminated.

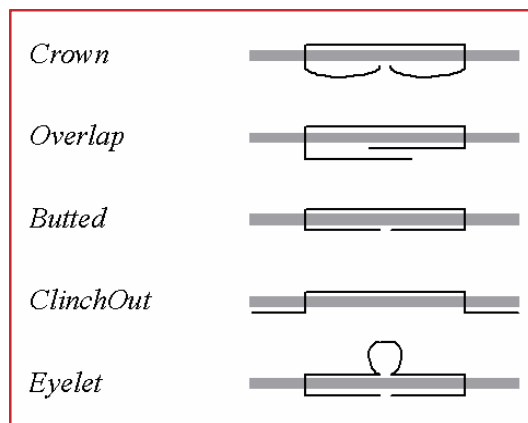
- A “page count” could be interpreted differently, depending on the context of its usage. A designer may only count number of folio pages, someone estimating paper usage may count the recto and verso sides of a pages as one page (or leave), and some one in estimating may count all finished pages, which include both blank pages and folio pages. This is cleared up in the glossary and edits have been made throughout to ensure consistency.
- In JDF 1.1 and 1.1a many JDF attributes had default values and enumerations of “system specified.” As a default, the JDF sender would have to add these values of “system specified,” which the receiving system would then replace with the actual value used by the system. This creates unnecessary work and may result in errors; hence, in JDF 1.2 “system specified” has been removed in almost all cases and it is expected that the receiving system will use its own default where a processing parameter is not defined with an attribute value, and that system will add the JDF value it uses to the JDF instance.
- All deprecated Resources, Processes, and other major deprecated sections have been removed to an appendix to make the JDF standard easier to read.

Several more substantial changes were made to JDF 1.2 as well. The Job Messaging Format and the FileSpec resource have been greatly improved. Several new finishing processes have been added to JDF 1.2, as well as processes and resources for handling quality control measurement and data collection. As indicated in JDF 1.1a, this new edition includes the much anticipated definitions of preflighting processes and resources — replacing the page holder preflighting processes and resources in JDF 1.1a.

Perhaps the most significant change in JDF 1.2 is the completion of device capabilities (see “Device Capability Definitions” on page 502.) Device capabilities provides a language internal to JDF that can be used to construct tests and queries of systems. These capabilities are used heavily by the new preflighting functions and also provide for a new type of preflighting: process preflighting. Given a set of customer files, intent, and processing instructions in JDF, device capabilities features could be used to query a JDF system (e.g., production line, plant, company, etc.) to determine if all of the capabilities are there to complete the job. Conceivably this could be used in selecting a location to produce a job or to balance work across an organization.

Device capabilities in JDF 1.2 have a third important function and that is to facilitate the automation of the “handshake.” For instance, in JDF there are five staple folds that a stitcher may use. If a new stitcher is added to your JDF workflow, the governing workflow or MIS system must know which of those five folds the new stitcher supports. Communicating the set of JDF elements and attributes supported by a device to the MIS system or workflow system is creating the “handshake.”

Prior to JDF 1.2, printers had to make this reconciliation or “handshake” for themselves or with the help of their vendors and/or consultants. Some groups, such as NGP or Print City, are constructing handshakes between devices of partnering companies so that come drupa, if you buy JDF-enabled products from the companies in one of these groups, you'll have some assurance that the handshakes have been established and the devices among the partners have been pre-integrated.



Device capabilities in JDF 1.2 allows for JDF 1.2-capable devices to be automatically queried for the details of what aspects of JDF they can and cannot manage. This is an important step towards total “plug-n-play” interoperability, but the reader is cautioned that it may be a year or more before there are enough JDF 1.2-capable products on the market for buyers to specify and rely on this automated handshake functionality.

Finally, in JDF 1.2 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 may not care to facilitate data used for case binding. A RIP need not be required to facilitate 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. The ICS documents are all currently in draft form, and only in circulation among members of CIP4, but once published, like the standard, they will be freely available to the public and we expect that they will become part of your buying practices.

---

# Chapter 1 Introduction

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

## 1.1 Background on JDF

JDF is an extensible, XML-based format built upon the existing technologies of CIP3's Print Production Format (PPF) and Adobe's Portable Job Ticket Format (PJTF). It provides three primary benefits to the printing industry: 1.) the ability to unify the prepress, press, and postpress aspects of any printing job, unlike any previous format; 2.) the means to bridge the communication gap between production services and Management Information Systems (MIS); and 3.) the ability to carry out both of these functions no matter what system architecture is already in place and no matter what tools are being used to complete the job. In short, JDF is extremely versatile and comprehensive.

JDF is an interchange data format to be used by a system of administrative and implementation-oriented components, which together produce printed products. It provides the means to describe print jobs in terms of the products eventually to be created, as well as in terms of the processes needed to create those products. The format provides a mechanism to explicitly specify the controls needed by each process, which may 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 may 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 must 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 653 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

Term	Definition
[JDF11a]	<i>Job Definition Format 1.1a</i> Date: 2002 Produced by: International Cooperation for Integration of Processes in Prepress, Press and Postpress (CIP4) Available at: <a href="http://www.cip4.org">http://www.cip4.org</a>
[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: <a href="http://www.w3.org/TR/REC-xml">http://www.w3.org/TR/REC-xml</a> .
[XMLNS]	<i>Namespaces in XML</i> <i>Version (W3C Recommendation of 14 January 1999)</i> Date: 14 January 1999 Produced by: World Wide Web Consortium (W3C) Available at: <a href="http://www.w3.org/TR/REC-xml-names/">http://www.w3.org/TR/REC-xml-names/</a>
[XPath]	<i>XML Path Language (XPath) Version 1.0</i> <i>Version W3C Recommendation 16 November 1999</i> Date: 16 November 1999 Produced by: World Wide Web Consortium (W3C) Available at: <a href="http://www.w3.org/TR/xpath.html">http://www.w3.org/TR/xpath.html</a> .
[XMLSchema]	<i>XML Schema Part 0+1+2: Primer, Structures and Datatypes</i> <i>Version (W3C Recommendation of 02 May 2001)</i> Date: 02 May 2001 Produced by: World Wide Web Consortium (W3C) XML Schema working group Available at: <a href="http://www.w3.org/TR/xmlschema-0/">http://www.w3.org/TR/xmlschema-0/</a> , <a href="http://www.w3.org/TR/xmlschema-1/">http://www.w3.org/TR/xmlschema-1/</a> and <a href="http://www.w3.org/TR/xmlschema-2/">http://www.w3.org/TR/xmlschema-2/</a> .

## 1.3 Conventions Used in This Specification

This section contains conventions and notations used within this document.

### 1.3.1 Text Styles

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

- Elements are written in sans serif. Examples are Comment, CustomerInfo, and ResourceLinks.
- Attributes are written in italic sans serif. Examples are Status, ResourceID, and ID.
- Resources are written in bold sans serif. Examples are **ImpositionProof**, **Toner**, and **ExposedMedia**.
- Processes are written in bold-italic sans serif. Examples are **ColorSpaceConversion**, **Rendering**, and **Scanning**.
- Enumerative and Boolean values of attributes are written in italics. Examples are *true*, *Waiting*, *Completed*, and *Stopped*.
- Standard bold text is used for the following purposes



#### 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.

- 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 1.4 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 job** in Section 4.4, Spawning and Merging.
  - to designate PPF objects in Appendix D, Converting PPF to JDF. Examples are **CIP3ProductName** and **CIP3ProductComponent**.
- For the benefit of those who are reading this document in PDF or online, cross-reference links are denoted by gray text. Examples are Chapter 6 *Processes*, and Section 1.2 *Conventions Used in This Specification*. To follow a link, click the highlighted text. The examples provided are not actual links.
  - Also for the benefit of online readers, external hyperlinks are graphically designated. An example is <http://URL.com>. To follow a link, click the highlighted text. The example provided is not an actual link.

### 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 (ex. “element/element”)
- Element attribute hierarchy is indicated by a slash and an at (@) symbol) (ex., “element/@attribute”), and
- Attribute value hierarch is indicated by an equal sign (ex., @Attribute = “Value”)
- 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 (e.g., enumeration, string, etc.).

#### Example:

The Xpath expression:

**Surface/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 **Surface** resource element.

Locally, (and within context), just the basic attribute dependency may be 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 Call-Outs

#### [New in JDF 1.2](#)

To help the reader familiar with JDF/1.0 or JDF/1.1, this specification indicates additions, deprecations, and clarifications using the following call-outs. Please note that not all changes are identified with modified or clarified call-out flags. A few changes have been made globally and are explained in the body of the document and only significant changes have been flagged with call-outs, as determined by CIP4 Working Groups.

Example	Call-Out Meaning
<a href="#">New in JDF 1.2</a>	New sections, attributes/elements, and attribute values
<a href="#">Deprecated in JDF 1.2</a>	Deprecated sections, attributes/elements, and attribute values
<a href="#">Modified in JDF 1.2</a>	Changed syntax or semantics of sections and attributes/elements, may include clarification as well.
<a href="#">Clarified in JDF 1.2</a>	Clarified sections or attributes.

### 1.3.4 Specification of Cardinality

The cardinality of JDF Data Types is expressed using a simple Extended Backus-Naur Form (EBNF) notation. The symbols in this notation may be combined to indicate both simple and complex patterns, as demonstrated in the following table. A and B represent simple expressions.

Notation	Description
(expression)	Expression is treated as a unit and may be combined as described in this list.
A	Matches A. A must occur exactly one time.
A ?	Matches A or nothing. A is optional, or is required only in the circumstances explained in the description field. If A 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.
A +	Matches one or more occurrences of A.
A *	Matches zero or more occurrences of A.
A = "value"	Matches on occurrence of A set to the default value shown. A may be set to other values other than the default. A default that is specified as A = "value" 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 7. This notation is only valid for XML attributes, not XML elements.

## 1.4 Glossary of Terminology

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. To locate the sections that explain these terms in more detail see Terminology Usage.

Term	Definition
<b>Agent</b>	The component of a JDF-based workflow that writes JDF.
<b>Attribute</b>	An XML-based syntactic construct describing an unstructured characteristic of a JDF <b>node</b> or <b>element</b> .
<b>Big job</b>	The combined <b>job</b> that independent jobs are merged into in the case of independent spawning and merging.
<b>Class</b>	A set of complex data types with common content in an object-oriented sense. A complex data type may consist of <b>elements</b> and <b>attributes</b> .
<b>Controller</b>	The component of a JDF-based workflow that initiates <b>devices</b> , routes JDF, and communicates status information.
<b>Default</b>	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.
<b>Deprecated</b>	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 may 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 <a href="#">Deprecated in JDF 1.X</a> in this specification.
<b>Device</b>	The component of a JDF workflow part that interprets JDF and executes the instructions. If a Device controls a <b>machine</b> , it does so in a proprietary manner.
<b>Document set</b>	A set of instance documents presumed to be related.
<b>Element</b>	An XML-based syntactic construct describing structured data in JDF.



Term	Definition
<b>Finished page</b>	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-folds 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.
<b>Folio</b>	A numbered finished page of a printed book or publication. (Pages are not all necessarily numbered. A 72-page book may have 68 pages that are numbered, which are referred to as either “folio pages” or “folios.”)
<b>Form</b>	A collection of imposed (ordered) finished pages set for printing or imaging to plate or film.
<b>Instance document</b>	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.
<b>JDF</b>	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 <b>node</b> (see below.)
<b>JDF Consumer</b>	A Device, Controller, Process, Queue, or Agent that consumes JDF instances.
<b>JMF</b>	Job Messaging Format. A communication format with multi-level capabilities. Structures information between MIS and <b>controllers</b> . There is also the JMF element, which is a top-level element within JDF.
<b>Job</b>	A hierarchical tree structure comprised of <b>nodes</b> . Describes the output that is desired by a customer.
<b>Job part</b>	One or more <b>nodes</b> which comprise the smallest level of control of interest to MIS.
<b>Leaf</b>	Both the recto and verso finished pages on one piece of paper with “leaves” being the plural usage.
<b>Link</b>	A pointer to information that is located elsewhere in a JDF document or that is located in another document.
<b>Machine</b>	The part of a device that does not know JDF and is controlled by a JDF device in a proprietary manner.
<b>MIS</b>	Management Information Systems. The functional part of a JDF workflow that oversees all processes and communication between system components and system control.
<b>Node</b>	The JDF <b>element</b> type detailing the resources and process specification required to produce a final or intermediate product or resource.
<b>Partition</b>	Enumerations of the <i>PartIDKeys</i> attribute of the <b>Resource</b> element used to identify individual physical and logical parts of a job. (See Table 3-27, “Contents of the Partitionable Resource Element,” on page 78.)
<b>Partitioned resource</b>	Structured <b>resource</b> that represents multiple physical or logical entities, such as separated plates.
<b>PDL</b>	Page Description Language. A generic term for any language that describes pages which may be printed. Examples are PDF®, PostScript® or PCL®.
<b>Process</b>	An individual step in the workflow.
<b>Queue</b>	Entity that accepts job entries via a <b>JMF</b> messaging system.
<b>Reader page</b>	A logical page as perceived by a reader, for example one <b>RunList</b> entry. One reader page may span more than one <b>finished page</b> , (e.g., a centerfold). One <b>finished page</b> may contain contents defined by multiple reader pages, (e.g., NUp imposition. Reader pages are defined independently of <b>finished pages</b> ).
<b>Resource</b>	A physical or conceptual entity that is modified or used by a <b>node</b> . Examples include paper, images, or process parameters.

Term	Definition
<b>Sheet</b>	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 vs. facing down at the feeder or off the roll.)
<b>Signature</b>	A <b>signature</b> is a set of printed <b>sheets</b> that may be folded or unfolded. Note that there are multiple usages of the word <b>Signature</b> in the industry. A sheet may contain multiple <b>BinderySignature</b> 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 <b>Layout</b> resource, on the other hand, describes a <b>Signature</b> as a set of sheets. This is appropriate for digital printing, where typically only one or two pages are printed per <b>Surface</b> and multiple sheets are gathered prior to folding.
<b>Slave Controller</b>	The component of a JDF workflow that accepts JDF as a <b>device</b> from other <b>controllers</b> and/or Slave Controllers and sends JDF to other Slave Controllers and/or Devices.
<b>Small job</b>	An independent job that is merged into a <b>big job</b> .
<b>Support</b>	A JDF Consumer <b>supports</b> 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 <i>should</i> support that JDF syntactic construct.
<b>Surface</b>	A single side of either a Sheet or a Signature
<b>Tag</b>	A syntactic construct that marks the start or end of an <b>element</b> .
<b>Work center</b>	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 may also have "recto" and "verso" forms for identification of orientation through the press (facing up vs. 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 may 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 may be described as a "72-page book, plus four cover pages" or just "plus cover." Cover pages may 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 should be interpreted as "finished page."

### 1.4.1 Conformance Terminology

The words “**must**”, “**must not**”, “**required**”, “*should*”, “**should not**”, “**recommended**”, “**may**”, and “**optional**” are used in this specification to define a requirement for the indicated **Agent** or the indicated **JDF Consumer** as follows.

Table 1-2: Conformance Terminology

Term	Meaning
<b>Must, Required</b>	Mean that the definition is an absolute requirement of the specification.
<b>Must not</b>	Means that the definition is an absolute prohibition of the specification.
<b>Should, Recommended</b>	Mean that there may 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.
<b>Should not, Not recommended</b>	Mean that there may 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.
<b>May, Optional</b>	<p>Mean that an item is truly optional. Unless specified otherwise, the word “optional” refers to JDF syntax, (i.e., what an Agent <b>may</b> include in a JDF instance), and does not refer to a JDF Consumer option, (i.e., not to what a JDF Consumer <b>may</b> support). 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 = “DefaultValue”</i> 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 may 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 may omit support of that item. Similarly, one vendor of an Agent may choose to supply such an item in a JDF instance, while another vendor may 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 may use a system-specified value.

See Figure 1.1 below. Such a system-specified attribute value may be configurable by a system administrator for the JDF Consumer or may depend on the values of other supplied attributes and/or the current setting of the JDF Consumer Device or the actual machine for which the Device is providing a JDF interface.

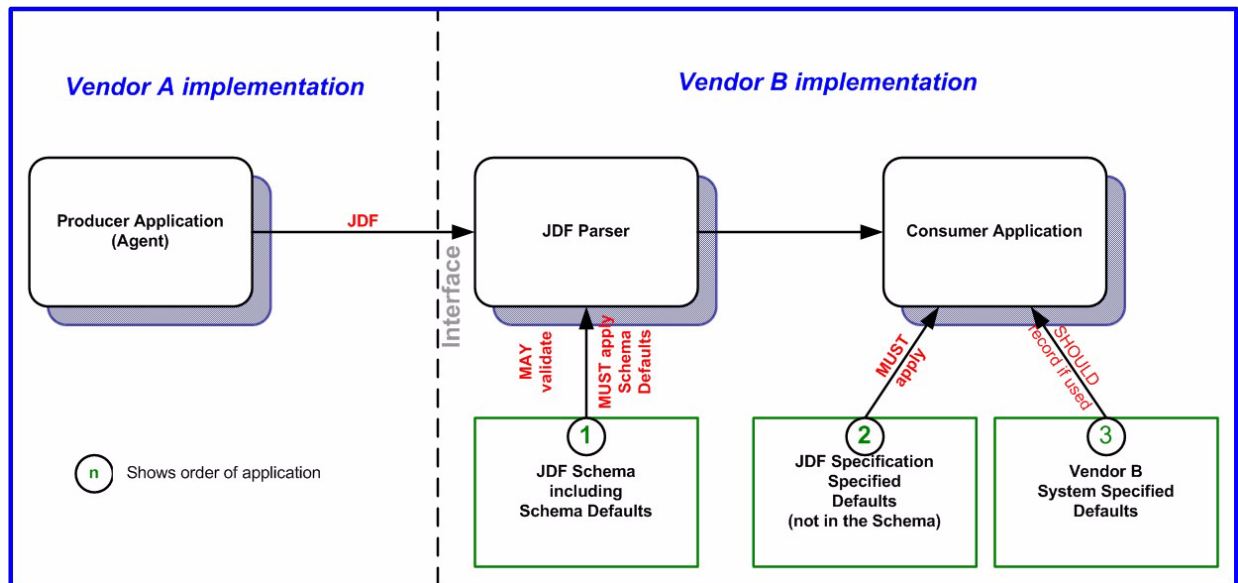


Figure 1.1: Handling of Default Values of JDF Attributes.

### 1.4.2.2 Conformance Requirements for Support of Elements

If a JDF Consumer supports an element, it

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

### 1.4.2.3 Conformance Requirements for Support of Processes

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

- 1 must support all of the input and output **ResourceLinks** and referenced **Resources** as described in Section 1.4.2.2 that this specification defines for that process,
- 2 may make its own assumptions regarding attributes and subelements of an optional input resource (resources with either a "?" or an "\*" – see Section 1.3.4) that an Agent has omitted from the process in the JDF instance; therefore, default attribute values defined in this specification are not guaranteed when the Agent omits the resource from the process in the JDF instance (see Section 6.1, Process Template), and
- 3 should find the processes that it supports in a JDF instance and must ignore all other processes, independent of the *SettingsPolicy* attribute for those other processes.

#### 1.4.2.4 Conformance Requirements for Support of Combined Processes

All combined processes are optional for a JDF Consumer to support. The rules for processes specified in Section 1.4.2.3 apply. If a JDF Consumer supports a combined process, it

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

#### 1.4.3 Conformance to SettingsPolicy

The *SettingsPolicy*, *BestEffortExceptions*, *MustHonorExceptions* and *OperatorIntervention-Exceptions* attributes defined in “Generic Contents of elements” on page 35 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

[Modified in JDF 1.2](#)

The following table describes the data structures as they are used in this specification. For more details on JDF Schema and Datatypes, see “Encoding” on page 565.

In JDF 1.2, some datatypes 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

A important reason for using a W3C Schema is to make use of user-defined datatypes. Even datatypes 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 565 for all datatype definitions.

Table 1-3: JDF data types

Data Type	Description
boolean	Binary-valued logic: (true   false).
CMYKColor	Represents a CMYK color specification.
date	Represents a time period that starts at midnight of a specified day and lasts for 24 hours.
dateTime	Represents a specific instant of time. It must be a UTC-time or a local time that includes the time zone.
DateTimeRange	Two dateTimes separated by a “~” (tilde) character that defines the closed interval of the two. TimeRange corresponds semantically to the time interval (two time instants separated by a slash) defined in ISO 8601.
DateTimeRange-List	Whitespace-separated list of DateTimeRanges.
double	Corresponds to IEEE754 double-precision, 64-bit floating point type, (see [IEEE754]), including special tokens INF and -INF. This corresponds to the standard XML double with NaN removed. For details, see [XMLSchema]. <b>Note:</b> Prior to JDF 1.2 the data type “number” was used. The double and number datatypes are syntactically equivalent.

Table 1-3: JDF data types

Data Type	Description
<a href="#">DoubleList</a> <a href="#">New in JDF 1.2</a>	Whitespace-separated list of doubles. Note that this datatype was named NumberList prior to JDF 1.2.
<a href="#">DoubleRange</a> <a href="#">New in JDF 1.2</a>	Two doubles separated by a “~” (tilde) character that defines the closed interval of the two. Note that this datatype was named NumberRange prior to JDF 1.2.
<a href="#">DoubleRangeList</a> <a href="#">New in JDF 1.2</a>	Whitespace-separated list of double and DoubleRanges. Note that this datatype was named NumberRangeList prior to JDF 1.2.
duration	Represents a duration of time.
DurationRange	<i>DurationRange</i> is used to describe a range of time durations. More specifically, it describes a time span that has a relative start and end.
DurationRange-List	Whitespace-separated list of DurationRanges.
element	Structured data. The specific data type is defined by the element name.
enumeration	Limited set of <b>NMTOKEN</b> (see below).
enumerations	Whitespace-separated list of enumeration data types.
gYearMonth	Represents a specific Gregorian month in a specific Gregorian year.
hexBinary	Represents arbitrary hex encoded binary data.
ID	Unique identifier as defined by [XML Specification 1.0] (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 (IDREFs) separated by white spaces as defined by [XML Specification 1.0].
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 datatype. For details, see [XMLSchema].
IntegerList	Whitespace-separated list of <b>integers</b> .
IntegerRange	Two <b>integers</b> separated by a “~” character that define a closed interval.
IntegerRangeList	Whitespace-separated list of <b>integers</b> and <b>IntegerRanges</b> .
LabColor	Represents a Lab color specification.
language	Represents a language and country code (for example, en-US) for a natural language.
LongInteger	Represents numerical integer values, including the special tokens INF and -INF. This corresponds to the standard XML integer with INF and -INF added. Values greater than +/-2**31 are expected to occur for this datatype. For details, see [XMLSchema].
matrix	Whitespace-separated list of six numbers representing a coordinate transformation matrix.
NamedColor	Represents a color definition by name. A list of valid NamedColor values is provided in Section A.3.3.2, NamedColor.
NameRange	Two <b>NMTOKEN</b> separated by a “~” (tilde) character that define an interval of NMTOKEN.
NameRangeList	Whitespace-separated list of <b>NMTOKEN</b> and <b>NameRanges</b> .
NMTOKEN	A continuous sequence of special characters as defined by the [XML Specification 1.0].
NMTOKENS	Whitespace-separated list of <b>NMTOKEN</b> .
Orientation <a href="#">New in JDF 1.2</a>	Enumeration that specifies named orthogonal two-dimensional orientations.
Orientations <a href="#">New in JDF 1.2</a>	Whitespace separated list of Orientation enumerations that specify named orthogonal two-dimensional orientations.
PDFpath	Whitespace-separated list of path operators as defined in PDF.

Table 1-3: JDF data types

Data Type	Description
rectangle	Whitespace-separated list of four numbers representing a rectangle.
refelement	<b>element</b> or a reference to an element. Used to define candidates for inter-resource linking in resources.
regExp <a href="#">New in JDF 1.2</a>	Regular expression as defined by [XMLSchema]
shape	Whitespace-separated list of three numbers representing a three-dimensional shape consisting of a width, height, and length. Unless specified otherwise in the attribute description, these three numbers are an X-dimension, a Y-dimension, and a Z-dimension, respectively.
ShapeRange	Two <b>Shapes</b> separated by a “~” (tilde) character that defines a 3-dimensional box bounded by x1 y1 z1~x2 y2 z2.
ShapeRangeList	Whitespace-separated list of <b>shapes</b> or <b>ShapeRanges</b> .
sRGBColor	Represents an sRGB color specification.
string <a href="#">Modified in JDF 1.2</a>	Character strings without tabs or line feeds. Corresponds to the standard XML normalizedString datatype [XMLSchema].
telem	Text elements that contain larger chunks of character data and may include line feeds.
text	Text data contained in a telem (text <b>element</b> ).
TimeRange	Two dateTimes separated by a “~” (tilde) character that defines the closed interval of the two. Time-Range corresponds semantically to the time interval (two time instants separated by a slash) defined in ISO 8601.
TransferFunction	Whitespace separated list of an even number of numbers representing a set of XY coordinates of a transfer function.
URI	URI-reference. Represents a Uniform Resource Identifier (URI) Reference as defined in Section 4 of [RFC 2396]. For the “file:” URL scheme, see [RFC1738].
URL	URL-reference. Represents a Uniform Resource Locator (URL) Reference as defined in Section 4 of [RFC 2396]. For the “file:” URL scheme, see [RFC1738].
JDFJMFVersion	Version label of a JDF or JMF instance. See Section 3.11, JDF Versioning for a discussion of versioning in JDF.
JDFJMFVersions	Whitespace separated list of JDFJMFVersion.
XPath	Represents an XPath expression of an XML node set (attributes or elements), boolean, number, or string.[XPath]
XYPair	Whitespace-separated list of two numbers. Unless specified otherwise in the attribute Description, these two numbers are an X-dimension and a Y-dimension, respectively.
XYPairRange	Two <b>XYPairs</b> separated by a “~” (tilde) character that defines a rectangle bounded by x1 y1 ~ x2 y2
XYPairRangeList	Whitespace-separated list of <b>XYPairRanges</b> .
XYRelation <a href="#">New in JDF 1.2</a>	Defines the relationship between two ordered numbers. One of a set of NMTOKENs, a list of valid values is provided in “XYRelation” on page 577.

## 1.6 Units

JDF specifies most values in default units. That means you can't use alternate units instead of the defined default units. All measurable quantities are stated in double precision. Processors should only specify a unit if no default exists, such as when new resources are defined. Then the units must be based on metric units. Overriding the default units that are defined in this table is non-standard and may lead to undefined behavior. Any exceptions are specified in the appropriate descriptive tables.

The following table lists the units used in JDF. The representation column specifies the XML representation in the unit attribute of resources.

Table 1-4: Units used in JDF

Measurement	Unit	Representation	Remarks
Length	point (1/72 inch)	pt	Used for all except microscopic lengths (see below)
	micron	mu	Used for microscopic lengths — where used (instead of points) it will be explicitly stated in the definition of the item. See <b>Media/@Thickness</b> .
Volume	liter	l	—
Weight	gram	g	—
Area	m <sup>2</sup>	m2	—
Resolution	dpi	dpi	The dots per inch (dpi) for print output and bitmap image (TIFF, BMP, etc.) file resolution.
Line Screen	lpi	lpi	The lines per inch (lpi) for conventionally screened halftone, screened grayscale, and screened monotone bitmap images.
Screen Resolution	ppi	ppi	The pixels per inch (ppi) for screen display (e.g., soft-proof display and user interface display), scanner capture settings, and digital camera settings.
Spot Resolution	spi	spi	For imaging devices such as filmsetters, platesetters, and proofers, the fundamental imaging unit, (e.g., one “on” laser or imaging head imaged unit). Note: Many imaging devices construct dots from multiple imaging spots, so dpi and spots per inch (spi) are not equivalent.
Paper weight	g/m <sup>2</sup>	g/m2	—
Speed	units/hour	*/h	Replace the “*” in the representation with the appropriate unit
Temperature	C° (Celsius)	C	degree centigrade
Angle	degrees°	degree	—
Countable Objects	1	—	Countable objects, such as sheets, have no unit specification.



# 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 required to complete the intended project. The information is collected logically into what is called a **node**. Each node in the tree structure represents an aspect of the job to be executed.

The nodes in a job are organized in a hierarchical structure that resembles a pyramid. The node at the top of the pyramid describes the overall intention of the job. The intermediate nodes describe increasingly process-oriented aspects of the job, until the nodes at the bottom of the pyramid each describe a single, simple process. Depending on where in the job structure a node resides, it can represent a portion of the product to be created, one or many processing steps, or other job parts. For more information about jobs and nodes, see Section 3, Structure of JDF Nodes and Jobs.

#### 2.1.1.2 Elements

An element is a standard XML syntactic construct [XML]. (See also: Section 2.1.1.3, Attributes.) Elements that are subparts of other elements are often referred to as subelements. JDF elements are represented by two kinds of data types: element and text element. The latter is abbreviated as telem. For more information about elements, see Section 3.1.2, JDF Node Attributes and Elements.



**XML Crash Course**

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

#### 2.1.1.3 Attributes

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

For more information about attributes, see Section 3.1.2, JDF Node Attributes and Elements. Note that an attribute with an empty (zero length) value string is illegal except when the attribute value is defined as an arbitrary string or as a list, (e.g., when not used nor required, attributes should 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.
- **Ancessor:** 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 may be referenced by multiple links. However, no link references more than one target.

## 2.1.2 Workflow Component Roles

The four components required 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 a JDF/JMF system should be designed, built, or implemented. The intention is to name the component mechanisms required 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 may be software processes, automated tools, or even text editors. Anything that can be used in composing JDF can be considered an agent.

Actual implementations of **devices** or **controllers** will most often be able to modify JDF. These system components have agent properties in the terms of this specification.



### Agents, Controllers & Devices

“Agents”, “Controllers”, and “Devices” are special, logical descriptions. You probably won’t ever buy one. An agent (writes and reads JDF) may be any software tool that can parse JDF. Controllers communicate instructions that devices act upon. They are functions that may be embedded into your software, production equipment, or MIS systems.

### 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. 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 may be derived from the messages that the controller receives or from the audit records in the job. (For more information on audits, see Section 3.9.1, *Audit Elements*.) Alternatively, the completed job may be passed to the job accounting system, which examines the audit records to determine the costs of all the processes in the job.



### Automating Data Flows

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

### 2.1.2.6 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 uni-directional communication channels and double arrows indicate bi-directional communication.

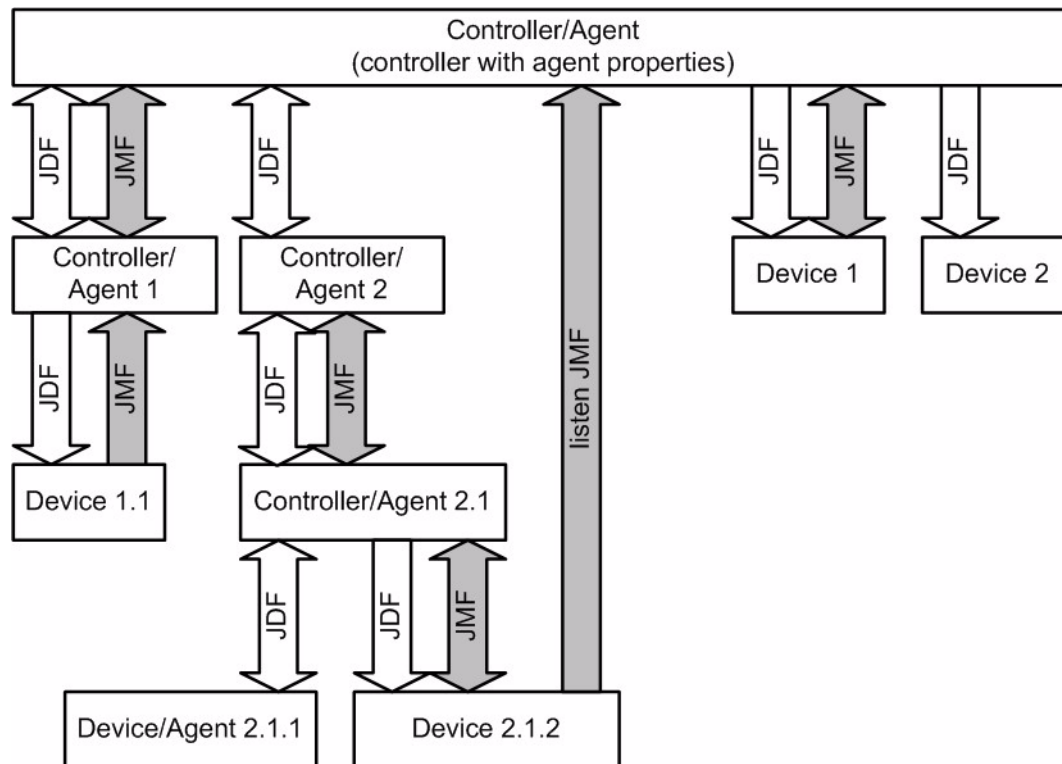


Figure 2.1: Example of JDF and JMF workflow interactions

## 2.2 JDF Workflow

JDF does not dictate that a workflow be constructed in any pre-specified way 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 should be produced. 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 may 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 may 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 may 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 required 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.

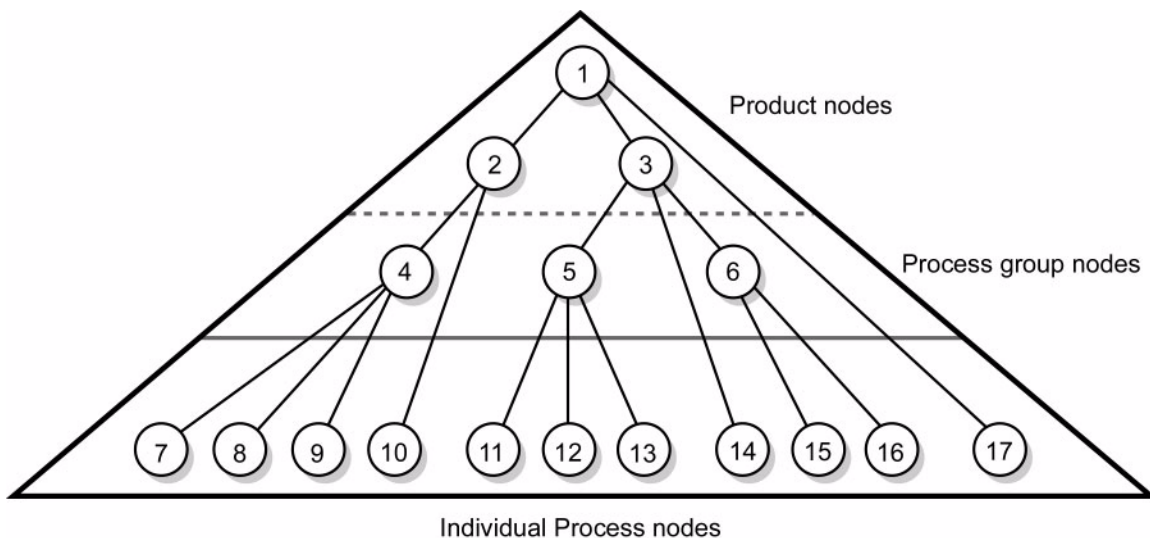


Figure 2.2: JDF tree structure

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*

Node #	Meaning
1	Entire book
2	Cover
3	Contents
4	Production of cover
5	Production of all color pages
6	Production of all black-and-white pages
7	Cover production process 1
8	Cover production process 2
9	Cover production process 3
10	Cover Finishing process
11	RIPing for color pages
12	Plate making for color pages
13	Printing for color pages
14	Color page finishing process
15	RIPing for black-and-white pages
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 required to describe the elements of the job in broad terms. Within the content pages there are some black-and-white pages and some color pages. Since fabricating each requires a different set of processes, further branching is necessary. The following table arranges the nodes in groups according to the processes they will be executing.

*Table 2-2: Information contained in JDF nodes, arranged by group*

Process Group	Node #	Meaning
<b>Entire book</b>	1	Entire book
	17	Assemble book
<b>Cover</b>	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
	3	Contents
<b>Contents</b>		

Table 2-2: Information contained in JDF nodes, arranged by group


Process Group	Node #	Meaning
Color Pages	5	Production of all color pages
	11	RIPing for color pages
	12	Plate making for color pages
	13	Printing for color pages
	14	Color page finishing
Black-and-white pages	6	Production of all black-and-white pages
	15	RIPing for black-and-white pages
	16	Printing for black-and-white pages on a digital press

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

## 2.3 Hierarchical Tree Structure and Networks in JDF

Output resources of JDF nodes are often the input resources for other JDF nodes. Nodes 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 required by a **Conventional Printing** 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 may 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?.

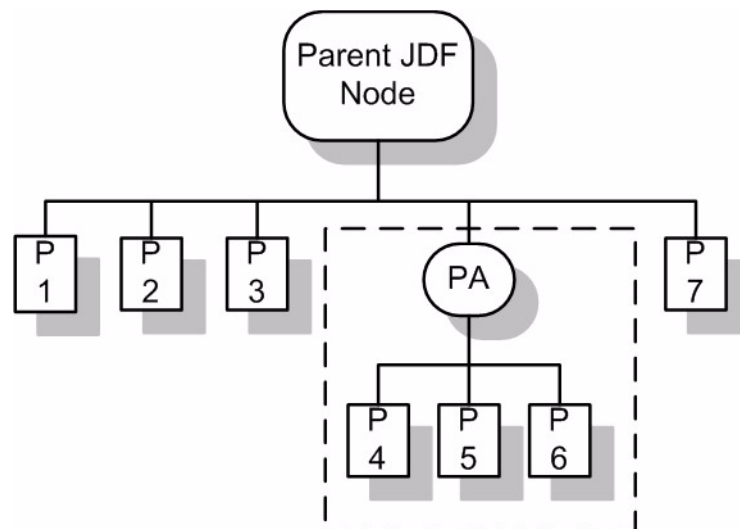


Figure 2.3: Example of a hierarchical tree structure of JDF nodes

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. Figure 2.4 displays an alternate representation of the process described in Figure 2.3. Whereas Figure 2.3 represents a hierarchical structure, Figure 2.4 shows an example of the linking mechanism of the same job. Note that there are many possible process networks that map to the same node hierarchy.

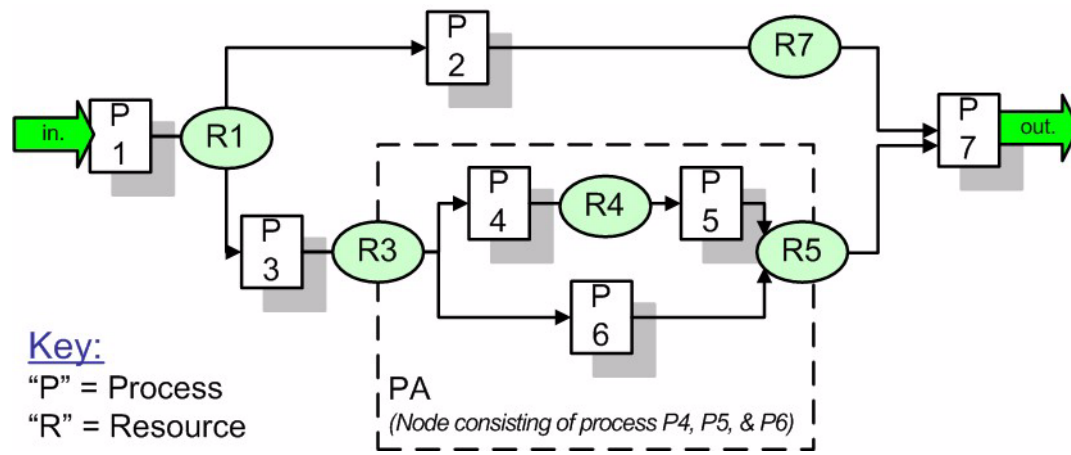


Figure 2.4: Example of a process chain linked by input and output resources

In the JDF specification, the linking of processes is not explicitly specified. In other words, nodes are not arranged in an abstract chronology, dictating, for example, that the trapping node must 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 may be combined in any way.

- 1 **Serial processing** that is subsequent production and consumption of resources as a whole, represented by a simple process chain
  - 2 **Overlapping processing** that is simultaneous production and consumption of resources by pipes
  - 3 **Parallel processing** that involves the splitting and sharing of resources
  - 4 **Iterative processing** that is a circular or back and forward processing for developing resources by repeated activity
- These mechanisms are discussed in greater detail in Section 4.3, Execution Model.

## 2.4 Role of Messaging in JDF

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

JMF is specifically designed for communication between the production system controller and the work centers or devices with which it interacts. It provides a series of queries and commands to check the status of processes and, in some cases, to dictate the next course of action. For example, the `KnownDevices` query allows the controller to determine what processes can be executed by a particular device or work center. These processes are likely to be determined at system initialization time. The `SubmitQueueEntry` message provides a means for the controller to



submit a job ticket to individual work centers or devices. And the **Status**, **Resource** and **Occupation** messages allow the device or work center to communicate quasi real-time<sup>1</sup> processing status to a controller. Depending on the system configuration, the message handler may 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.)

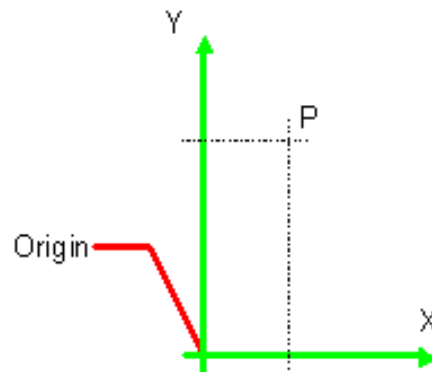


Figure 2.5: Standard coordinate system

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

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.

system to the coordinate systems of the output resources is defined. The process coordinate system is also used to define the meaning of terms like *Top* or *Left*, which are used as values for parameters in some processes.

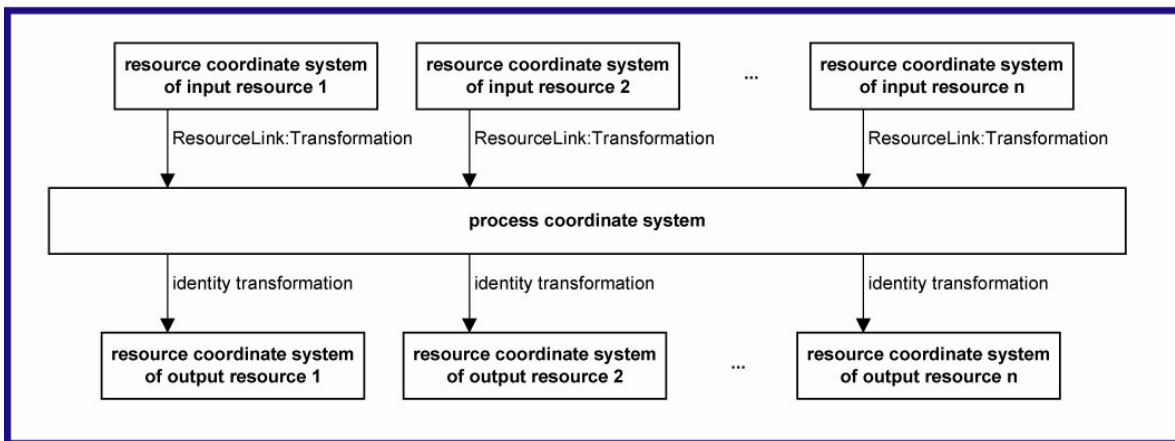


Figure 2.6: Relation between resource and process coordinate systems

It is important that no implicit transformations (such as rotations) are assumed if the dimensions of the input resources of a process do not match each other. Instead every transformation (e.g., a rotation) 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.

## 2.5.2 How and Where Coordinates and Transformations Are Used/Defined in JDF

The following data types are used for the specification of 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

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

[Modified in JDF 1.2](#)

Each physical input **Resource**, (i.e., **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 **ResourceLinks**, see section “Resource Links” on page 61.

### 2.5.3.1 Coordinate Systems of Combined Processes







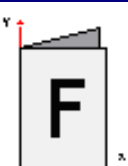









[New in JDF 1.2](#)

Combined Processes (See “Combined Process Nodes” on page 46.) 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 may be modified by defining it as a pipe by specifying **Resource/@PipeID** and **Resource/@PipeProtocol=“Internal”**. (See “Overlapping Processing Using Pipes” on page 114.) and linking it to the combined process with both an input and output **ResourceLink**. The input **ResourceLink** defines the coordinate transformation using the standard *Transformation* or *Orientation* attributes. **Resource/@Status** of the exchange resource must be “*Complete*”.

### 2.5.3.2 Coordinate System Transformations

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

Table 2-3: Matrices and Orientation values used to describe the orientation of a Component

Orientation Value	Source Coordinate System	Transformation Matrix According Action	Target Coordinate System
<i>Rotate0</i>		$1\ 0\ 0\ 1\ 0\ 0$ No Action	
<i>Rotate90</i>		$0\ 1\ -1\ 0\ h\ 0$ 90° Counterclockwise Rotation	
<i>Rotate180</i>		$-1\ 0\ 0\ -1\ w\ h$ 180° Rotation	
<i>Rotate270</i>		$0\ -1\ 1\ 0\ 0\ w$ 270° Counterclockwise Rotation	
<i>Flip0</i>		$1\ 0\ 0\ -1\ 0\ h$ Flip around X	
<i>Flip90</i>		$0\ -1\ -1\ 0\ h\ w$ 90° Counterclockwise Rotation + Flip around X	
<i>Flip180</i>		$-1\ 0\ 0\ 1\ w\ 0$ 180° Rotation + Flip around X	
<i>Flip270</i>		$0\ 1\ 1\ 0\ 0\ 0$ 270° Counterclockwise Rotation + Flip around X	

## 2.5.4 Product Example: Simple Brochure

To illustrate the use of coordinate systems in JDF, a simple saddle stitched brochure with eight pages is used as an example. 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. The following table lists the JDF processes used for the production of the simple brochure.

Input Resources	Process	Output Resources
<b>Layout</b> <b>RunList</b> (Document) <b>RunList</b> (Marks)	<b>Imposition</b>	<b>RunList</b>
<b>RunList</b>	<b>Interpreting</b>	<b>RunList</b> (InterpretedPDLData)
<b>RunList</b> (InterpretedPDLData) <b>Media</b> <b>RenderingParams</b>	<b>Rendering</b>	<b>RunList</b> (rasterized ByteMaps)
<b>RunList</b> (rasterized ByteMaps)	<b>Screening</b>	<b>RunList</b> (Bitmaps)
<b>ImageSetterParams</b> <b>Media</b> (Film) <b>RunList</b> (Bitmaps)	<b>ImageSetting</b> (to Film)	<b>ExposedMedia</b> (Film)
<b>ExposedMedia</b> (Film)	<b>ContactCopying</b>	<b>ExposedMedia</b> (Plate)
<b>ExposedMedia</b> (Plate) <b>ConventionalPrintingParams</b>	<b>ConventionalPrinting</b>	<b>Component</b>
<b>FoldingParams</b> <b>Component</b>	<b>Folding</b>	<b>Component</b>
<b>CollectingParams</b> <b>Component</b>	<b>Collecting</b>	<b>Component</b>
<b>SaddleStitchingParams</b> <b>Component</b>	<b>SaddleStitching</b>	<b>Component</b>
<b>TrimmingParams</b> <b>Component</b>	<b>Trimming</b>	<b>Component</b>

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.)

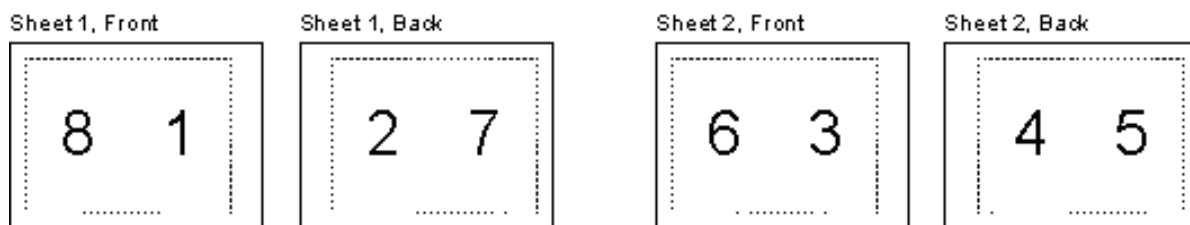


Figure 2.7: Layout of simple saddle stitched brochure (product example)

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 **Surface**, **Sheet**, and **Signature** elements are defined within the hierarchy of the **Layout** resource. 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:

$$P_{\text{Surface}} = P_{\text{Page}} \times \text{CTM}_{\text{Page}} + [\text{SurfaceContentsBox}_{\text{Xlowerleft}} \text{SurfaceContentsBox}_{\text{Ylowerleft}} 0]$$

Please note, that the width and height of the surface are not known at this point.

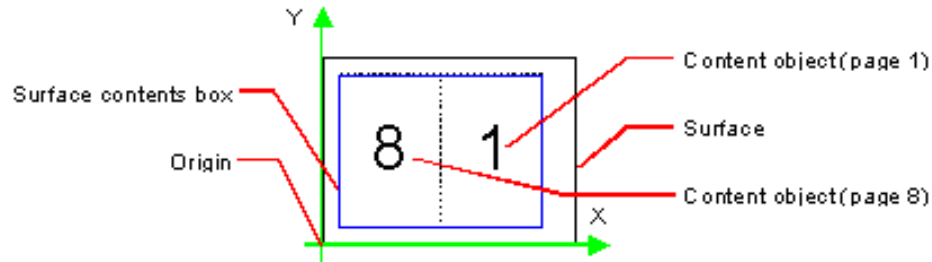


Figure 2.8: Surface coordinate system

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 *Sheet/@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*.

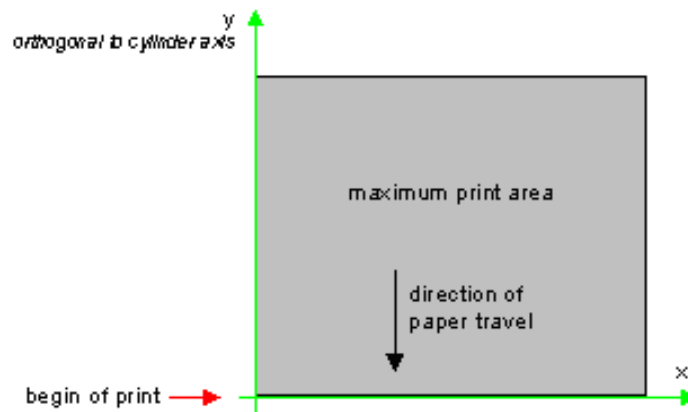


Figure 2.9: Press coordinate system used for sheet-fed printing

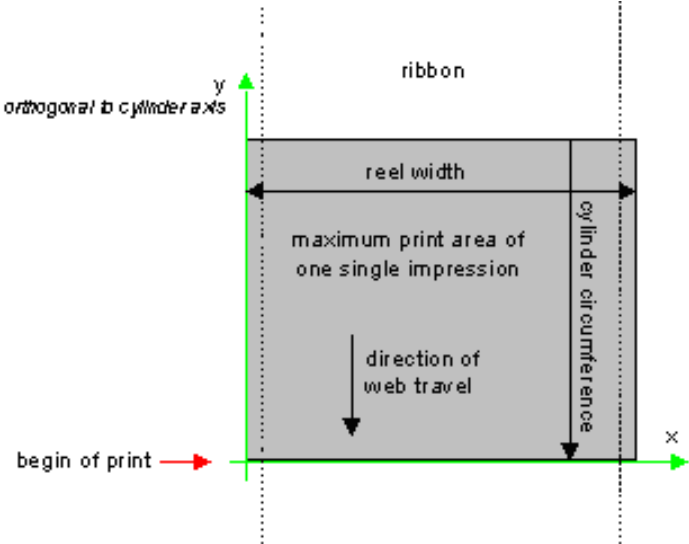


Figure 2.10: Press coordinate system used for web printing

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.)

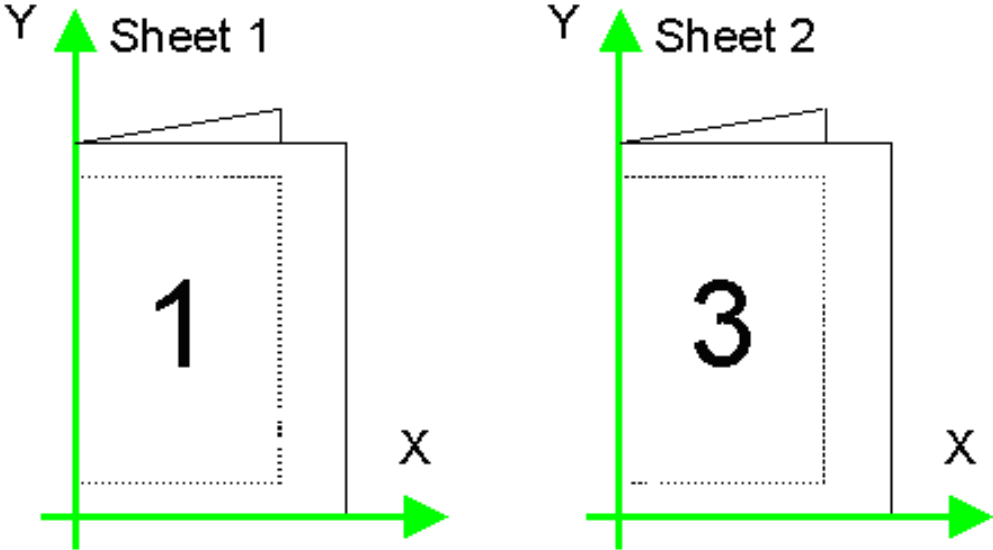


Figure 2.11: Coordinate systems after Folding (product example)

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.

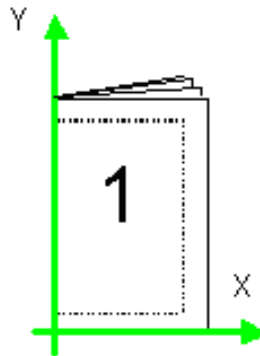


Figure 2.12: Coordinate systems after Collecting (product example)

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 2.12. 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 resource link of each input and output component contains a *Transformation* attribute or an *Orientation* attribute. The *Transformation* attribute may be used if the width and the height of the component are known or a non-orthogonal rotation is required. Otherwise the *Orientation* attribute may 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 resource links 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 resource links 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 resource links. 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.



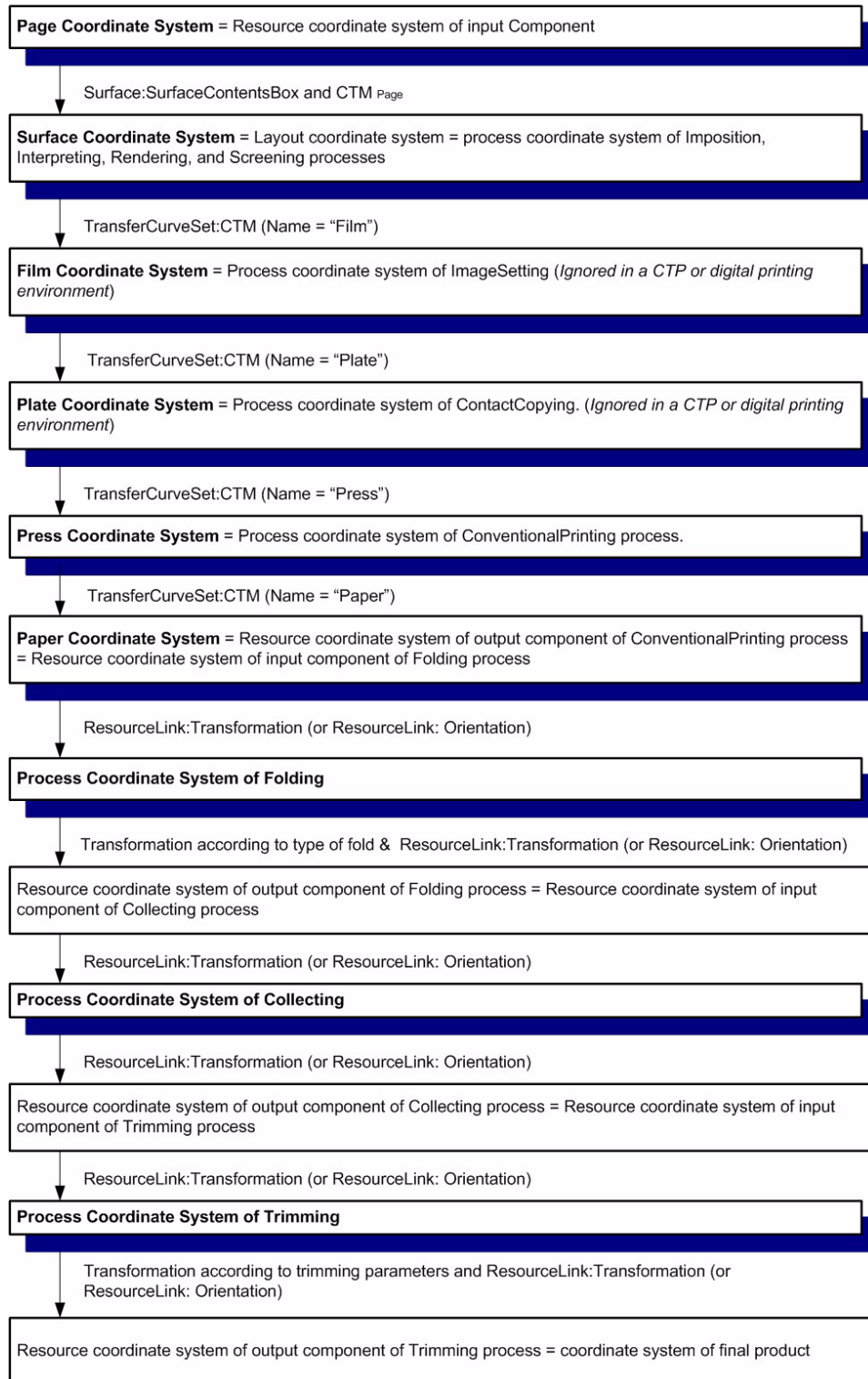


Figure 2.13: Examples of Transformations and Coordinate Systems in JDF.

### 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 may 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*

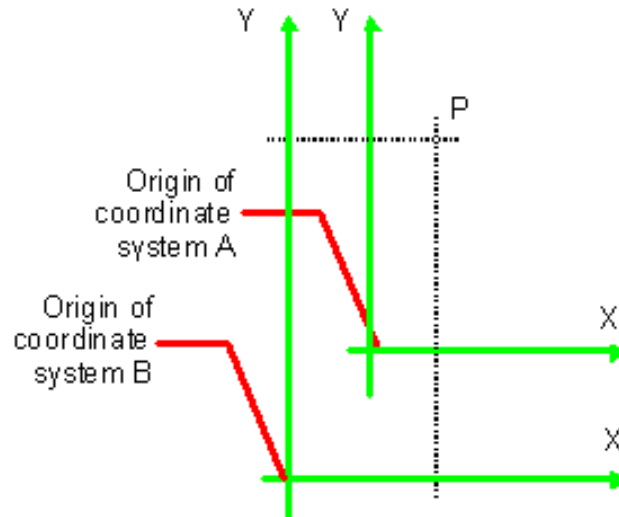


Figure 2.14: Transforming a point (example)

$$P_A = (30, 100)$$

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

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

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: 1.) in terms of a process necessary to produce the end result, such as imposing, printing, or binding; 2.) in terms of a product that contributes to the end result, such as a brochure; or 3.) in terms of some combination of the previous two. In short, a node describes a product or a process.

In addition to describing the structure of an individual JDF node, this chapter examines in what way those nodes interact to form a coherent job structure. The visual correlative of this structure resembles a family tree with a single node describing the entire job at the top, and a number of nodes at the bottom that each describes only one specific process. JDF-supported, leaf-level processes are described in “Processes” on page 191.

Resource linking specifies the transformation of input Resources into output Resources, which in turn may 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 may 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 ResourceLinks, which are defined in Section 3.7, *Resource Links*.

Figure 3.1 is a schematic structure of the JDF node type. In this figure, generic attributes and elements (see Section 3.1.1, *Generic Contents of JDF Elements*) are inserted only in the JDF root node. The element types that are displayed in this figure are described in the subsequent sections. Abstract data types are surrounded by a dashed line. Types derived from the abstract data type Resource are shown schematically in Figure 3.4.

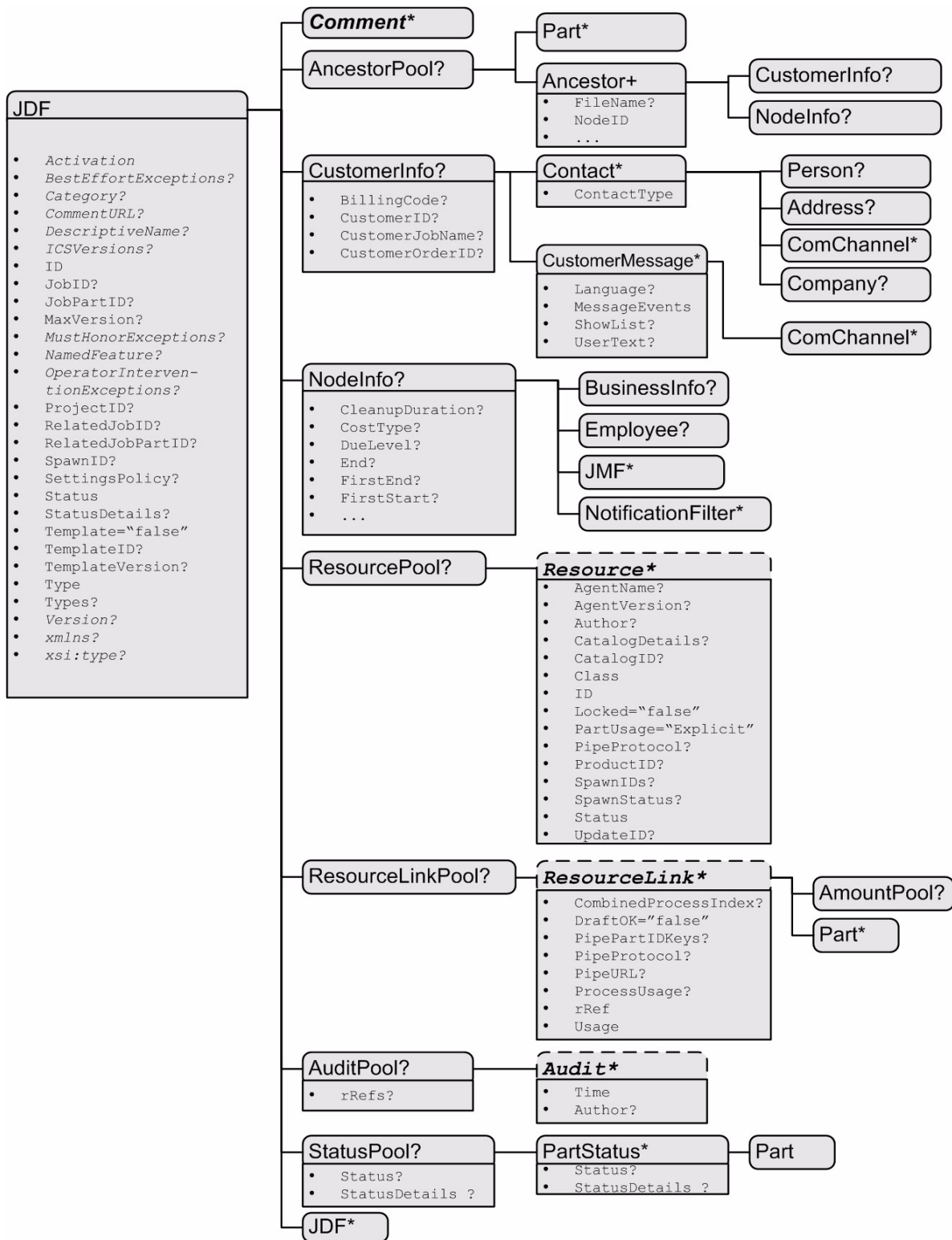


Figure 3.1: Structure of the JDF Node

## 3.1 JDF Nodes

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

### 3.1.1 Generic Contents of JDF Elements

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

Table 3-1: Generic Contents of elements

Name	Data Type	Description
<i>BestEffortExceptions</i> ? <a href="#">New in JDF 1.1</a>	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.
<i>DescriptiveName</i> ?	string	Human-readable descriptive name of the JDF element, (e.g., a descriptive name of a resource, process, or product). It is strongly recommended to supply <i>DescriptiveName</i> in all JDF nodes, <i>Quantity</i> resources (for example: <b>Component</b> resources), and <i>Handling</i> resources (for example, <b>ExposedMedia</b> ) for communication from applications to humans in order to reference the process or resource.
<i>MustHonorExceptions</i> ? <a href="#">New in JDF 1.1</a>	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>OperatorIntervention Exceptions</i> ? <a href="#">New in JDF 1.1</a>	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> .
<i>SettingsPolicy</i> ? <a href="#">New in JDF 1.2</a>	enumeration	The policy for this element indicates what happens when unsupported settings, (i.e., subelements, attributes or attribute values), are present in the element. Possible values are: <i>BestEffort</i> – Substitute or ignore unsupported attributes, attribute values, default attribute values, or elements and continue processing the job. <i>MustHonor</i> – Reject the job when any unsupported attributes, attribute values, or elements are present. <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> . If not specified, <i>SettingsPolicy</i> is inherited from the parent element, and if not specified in the parent element or further superior element, the default value defaults to “ <i>BestEffort</i> ”. In JDF 1.1 <i>SettingsPolicy</i> was specified in “Contents of a JDF node” on page 38 and “Contents of the abstract Resource element” on page 53. It has been removed from JDF node and Resource and been promoted to all JDF elements. For details on <i>SettingsPolicy</i> , see “Conformance to SettingsPolicy” on page 9.
<i>Comment</i> *	telem	Any human-readable text. The <i>Comment</i> element is different from an XML comment <code>&lt;!-- XML Comment --&gt;</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.

Table 3-2: Contents of the Comment element

Name	Data Type	Description
<i>Attribute</i> ? <a href="#">New in JDF 1.1</a>	NMTOKEN	Name of the attribute in this element that the comment refers to. The name should include the prefix if the attribute is in a non-JDF namespace. If omitted, the <b>Comment</b> refers to the entire element
<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>Language</i> ?	language	Human readable language of the <b>Comment</b> .
<i>Name</i> = "Description" <a href="#">Modified in JDF 1.2</a>	NMTOKEN	A name that defines the usage of a comment. For example, it may determine whether two comments should fill two distinct fields of a user interface. Pre-defined values include: <i>Description</i> – Human readable description, which is required if the <b>Comment</b> element is required in a given context, as is the case in the <b>Notification</b> element (see Table 3-33, "Contents of the Notification element," on page 92). <i>Instruction</i> – Message to the operator that contains information regarding the processing of the job. <a href="#">New in JDF 1.2</a> <i>JobDescription</i> – Description of the Job. A <b>Comment</b> element that contains <i>Name</i> = "JobDescription" must only be specified in a JDF node or <b>CustomerInfo</b> element. See also <b>CustomerInfo/@CustomerJobName</b> in Section 3.3, Customer Information in <b>CustomerInfo</b> . <a href="#">New in JDF 1.2</a> <i>OperatorText</i> – Message from the operator that contains information regarding the processing of the job. <a href="#">New in JDF 1.2</a> <i>Orientation</i> – Description of the orientation of a physical resource. <i>TemplateDescription</i> – Description of the job ticket template. A <b>Comment</b> element that contains <i>Name</i> = "TemplateDescription" must only be specified in the root JDF node. <a href="#">New in JDF 1.2</a> <i>UserText</i> – Message to a user that contains information regarding the processing of the job. Used in <b>CustomerInfo/CustomerMessage</b> . See "Customer Information in CustomerInfo" on page 49. <a href="#">New in JDF 1.2</a>
<i>Path</i> ?	PDFPath	Description of the area that the comment is associated with in the coordinate system of the element where the path resides. In the case of physical resources, <b>Layout</b> resources and resources that are related to <b>Layout</b> , <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 <b>ExposedMedia</b> 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 <b>Comment</b> element containing <i>Path</i> is defined in. Note that there are cases where a coordinate system is not available and therefore defining <i>Path</i> is not recommended, (for example: <b>CustomerInfo</b> .)
	text	Body of the comment. Note that whitespace is preserved only as generic whitespace in XML. Thus carriage returns, line feeds or tabs may be lost.



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

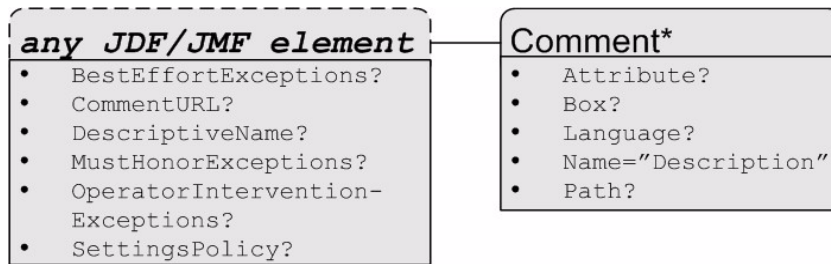


Figure 3.2: Structure of JDF Generic Contents

### 3.1.2 JDF Node Attributes and Elements

The following table presents the attributes and elements likely to be found in any given JDF node. Three of the attributes in Table 3-4, below, are required and must appear in every JDF node. Although the rest are designated as optional, they are optional in the sense that they are required only under certain circumstances, not that they may be left out if desired. The circumstances under which they are required are described in the Description column.

The most important of the attributes is the *Type* attribute, which defines the node type. The value of the *Type* attribute defines the product or process the JDF node represents. As is detailed in Section , 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 "Contents of a JDF node" on page 38 contains a fourth column that provides further details about the valid range of the attribute/element content, how the content is inherited by descendants (children, grandchildren, etc.), and where the attribute/element may reside in the JDF tree. The heading for this column is "Scope," which is short for "Scope and Position." The following abbreviations are defined:

Table 3-3: Definition of "Scope" Terms used in Table 3-4 on page 38

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 may only be specified 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 3-4: Contents of a JDF node

Name	Data Type	Scope	Description																					
<a href="#">Activation ?</a> <a href="#">Modified in JDF 1.1</a>	enumeration	(D)	<p>Describes the activation status of the JDF node. Allows for a range of activity, including deactivation and test running. Possible values, in order of involvement from least to most active, are:</p> <p><i>Inactive</i> – The node and all its descendents must not be executed or tested. This value is set if certain parts of a JDF job must not be executed or tested.</p> <p><i>Informative</i> – The JDF ticket is for information only. If a job is <i>Informative</i>, it must not be processed. Jobs with <i>Activation = "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 should 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 testrun has been completed successfully.</p> <p><i>Active</i> – The default value if not specified in a parent node – The node maybe executed as soon as all inputs are Available or Complete and all outputs are not incomplete.</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>. 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> <table border="1"> <thead> <tr> <th>Activation</th> <th>Test Node</th> <th>Execute Node</th> </tr> </thead> <tbody> <tr> <td><i>Inactive</i></td> <td><i>false</i></td> <td><i>false</i></td> </tr> <tr> <td><i>Informative</i></td> <td><i>false</i></td> <td><i>false</i></td> </tr> <tr> <td><i>Held</i></td> <td><i>false</i></td> <td><i>false</i></td> </tr> <tr> <td><i>Active</i></td> <td><i>false</i></td> <td><i>true</i></td> </tr> <tr> <td><i>TestRun</i></td> <td><i>true</i></td> <td><i>false</i></td> </tr> <tr> <td><i>TestRunAndGo</i></td> <td><i>true</i></td> <td><i>true</i></td> </tr> </tbody> </table>	Activation	Test Node	Execute Node	<i>Inactive</i>	<i>false</i>	<i>false</i>	<i>Informative</i>	<i>false</i>	<i>false</i>	<i>Held</i>	<i>false</i>	<i>false</i>	<i>Active</i>	<i>false</i>	<i>true</i>	<i>TestRun</i>	<i>true</i>	<i>false</i>	<i>TestRunAndGo</i>	<i>true</i>	<i>true</i>
Activation	Test Node	Execute Node																						
<i>Inactive</i>	<i>false</i>	<i>false</i>																						
<i>Informative</i>	<i>false</i>	<i>false</i>																						
<i>Held</i>	<i>false</i>	<i>false</i>																						
<i>Active</i>	<i>false</i>	<i>true</i>																						
<i>TestRun</i>	<i>true</i>	<i>false</i>																						
<i>TestRunAndGo</i>	<i>true</i>	<i>true</i>																						

Table 3-4: Contents of a JDF node

Name	Data Type	Scope	Description
<i>Category</i> ? <a href="#">New in JDF 1.2</a>	NMTOKEN	L	Named category of this node. Used when <i>Type</i> = “ <i>Combined</i> ” or <i>Type</i> = “ <i>ProcessGroup</i> ” 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 may have identical <i>Types</i> attribute values but will have <i>Category</i> = “ <i>ProofRIPing</i> ” or <i>Category</i> = “ <i>RIPing</i> ” respectively. Values include: <ul style="list-style-type: none"> <li>• <i>Binding</i> – Binding of a bound product.</li> <li>• <i>DigitalPrinting</i> – A RIP and print run on a digital printer that produces final output.</li> <li>• <i>FinalImaging</i> – A RIP and image that produces final output that is ready for further processing, (e.g. film or plates).</li> <li>• <i>FinalRIPing</i> – RIP process for generating final output. Includes <b><i>ContoneCalibration, ColorSpaceConversion, ImageReplacement, Interpreting, Rendering, Screening, Separation, and Trapping.</i></b></li> <li>• <i>Folding</i> – Folding of a product.</li> <li>• <i>ImpositionPreparation</i>: Setup for impositioning. Includes <b><i>LayoutPreparation</i></b> or <b><i>Stripping.</i></b></li> <li>• <i>PostPress</i> – General postpress. Includes <i>Folding</i> and <i>Binding</i>.</li> <li>• <i>PrePress</i> – General prepress. Includes <i>PrePressPreparation, ImpositionPreparation, Proof/FinalRIPing, and Proof/FinalImaging.</i></li> <li>• <i>PrePressPreparation</i> – contains all prepress processes needed to prepare the content files ready for RIPing, (e.g. a preflighted, normalized PDF or Postscript file). Includes <b><i>AssetListCreation, DigitalDelivery, Preflight, FormatConversion, LayoutElementProduction, PStoPDFConversion, and Trapping.</i></b></li> <li>• <i>Printing</i> – A press run that produces final output.</li> <li>• <i>ProofImaging</i> – A RIP&amp;Proof that produces proof output.</li> <li>• <i>ProofRIPing</i> – RIP process for generating a proof. The processes are identical to those in specified for <i>FinalRIPing</i>.</li> </ul>
<i>ICSVersions</i> ? <a href="#">New in JDF 1.2</a>	NMTO- KENS	D	CIP4 Interoperability Conformance Specification (ICS) Versions that this JDF node complies with. The format is <ICSName>_L<ICSLevel>-<ICS-Version>. For instance: <i>DP_L1-1.0</i> for ICS for Digital Printing, level 1, version 1.0. See “Interoperability Conformance Specifications” on page 564 for more information on ICS documents.
<i>ID</i>	ID	L	Unique identifier of a JDF node. This ID is used to refer to the JDF node.
<i>JobID</i> ?	string	D	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.
<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 <b><i>Resource/@ProductID</i></b> and not by <i>JobPartID</i> .

Table 3-4: Contents of a JDF node

Name	Data Type	Scope	Description
<a href="#">MaxVersion ?</a> <a href="#">New in JDF 1.2</a>	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 3.11, JDF Versioning for a discussion of versioning in JDF.
<a href="#">NamedFeatures ?</a> <a href="#">New in JDF 1.2</a>	NMTO- KENS	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 “Structure of the DeviceCap Subelement” on page 502. <i>NamedFeatures</i> must only be specified in <i>ProcessGroup</i> nodes, typically with a <i>Types</i> attribute supplied, or <i>Product</i> JDF nodes. See “Use of the NamedFeatures attribute in Product and ProcessGroup nodes” on page 44 for details.
<a href="#">ProjectID ?</a> <a href="#">New in JDF 1.1</a>	string	D	Identification of the project context that this JDF belongs to. Used by the MIS to group a set of JDF jobs.
<a href="#">RelatedJobID ?</a> <a href="#">New in JDF 1.2</a>	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. May be used to retrieve additional job and device specific settings from a data store.
<a href="#">RelatedJobPartID ?</a> <a href="#">New in JDF 1.2</a>	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. May be used to retrieve additional job and device specific settings from a data store.
<a href="#">SpawnID ?</a> <a href="#">New in JDF 1.1</a>	NMTOKEN	D	Identification of a spawned part of a job. Typically this is used to map Audits and JMF messages to a spawned processing step in the workflow. For details on job spawning, see “Spawning and Merging” on page 118.
<a href="#">Status</a> <a href="#">Modified in JDF 1.1</a>	enumeration	L	Identifies the status of the node. Possible values are: <i>Waiting</i> – The node may 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 <i>ResourceLinks</i> 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 <i>Notification</i> element, which is an optional subelement of the <i>AuditPool</i> element described in Section 3.9, <i>AuditPool</i> . <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 <i>Spawned</i> can only be assigned to the original instance of the spawned job. For details, see Section 4.4, <i>Spawning and Merging</i> . <i>Stopped</i> – Execution has been stopped. If a job is <i>Stopped</i> , running may be resumed later. This status may 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.

Table 3-4: Contents of a JDF node

Name	Data Type	Scope	Description
			<p><i>Pool</i> – Indicates that the node processes partitioned resources and that the <i>Status</i> varies depending on the partition keys. Details are provided in the <i>StatusPool</i> element of the node.</p> <p>Derivation of the <i>Status</i> of a parent node from the <i>Status</i> of child nodes is non-trivial and implementation-dependent.</p>
<a href="#">StatusDetails ?</a> <a href="#">New in JDF 1.2</a>	string	L	Description of the status phase that provides details beyond the enumerative values given by the <i>Status</i> attribute. For a list of supported values, see “StatusDetails Supported Strings” on page 615.
<a href="#">Template = “false”</a> <a href="#">New in JDF 1.1</a>	boolean	R	Indicates that this JDF ticket (or instance) is a template that is used to generate JDFs but must not be exchanged as a job definition. A Device must reject a job ticket that contains <i>Template</i> = “true”.
<a href="#">TemplateID ?</a> <a href="#">New in JDF 1.2</a>	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.
<a href="#">TemplateVersion ?</a> <a href="#">New in JDF 1.2</a>	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	<p>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 191, although the flexibility of JDF allows anyone to create processes.</p> <p>In addition to these, there are three values which are described in greater detail in the sections that follow.</p> <p><i>Combined</i></p> <p><i>ProcessGroup</i></p> <p><i>Product</i> – Identifies a product intent node.</p>
<a href="#">Types ?</a> <a href="#">Modified in JDF 1.2</a>	NMTO-KENS	L	<p>List of the <i>Type</i> attributes of the nodes that are combined to create this node. This attribute is required if <i>Type</i> = “<i>Combined</i>”, optional when <i>Type</i> = “<i>ProcessGroup</i>”, and is ignored if <i>Type</i> equals any other value. For details on using <i>Combined</i> nodes, see Section 3.1.5, Combined Process Nodes. If the <i>Types</i> attribute is specified, that JDF node must not contain child JDF nodes. For details on using <i>ProcessGroup</i> nodes, see Section 3.1.4, Process Group Nodes.</p> <p>The following special tokens are defined to allow a JDF-enabled MIS or workflow system to roughly specify finishing, proofing, and RIPing without knowing the details of the respective combined processes. Use these special tokens only for a <i>ProcessGroup</i> and not for a <i>Combined</i> Process. See the <i>Category</i> attribute above for more details on the tokens:</p> <p><i>Finishing</i></p> <p><i>ImpositionPreparation</i>: Setup for impositioning. Includes <b>LayoutPreparation</b> or <b>Stripping</b>.</p> <p><i>PrePressPreparation</i> – contains all prepress processes needed to prepare the content files ready for RIPing.</p> <p><i>PrePress</i> – General prepress</p> <p><i>ProofImaging</i> – A RIP&amp;Proof that produces a proof.</p> <p><i>RIPing</i></p>

Table 3-4: Contents of a JDF node

Name	Data Type	Scope	Description
<a href="#">Version ?</a> <a href="#">Modified in JDF 1.2</a>	JDFJMF-Version	RD	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 101.
<a href="#">xmlns ?</a> <a href="#">New in JDF 1.1</a>	URI	RD	JDF supports use of XML namespaces. The namespace must be declared in the root JDF element. For details on using namespaces in XML, see[XMLNS]. For version 1.1 and 1.2 of JDF, <i>xmlns</i> = “ <a href="http://www.CIP4.org/JDFSchema_1_1">http://www.CIP4.org/JDFSchema_1_1</a> ”.
<a href="#">xsi:type ?</a> <a href="#">New in JDF 1.2</a>	NMTOKEN	L	Informs schema aware validators of the JDF node type definition that the containing node is to be validated against. The schema for this version includes definitions for all the JDF nodes defined in Section 6. If omitted, then a general definition for JDF nodes will be used. See “JDF Nodes” on page 35.
<a href="#">AncestorPool ?</a>	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 3.2, <i>AncestorPool</i> .
<a href="#">AuditPool ?</a>	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 3.9, <i>AuditPool</i> .
<a href="#">CustomerInfo ?</a>	element	D	Container element for customer-specific information. See Section 3.3, <i>Customer Information</i> in <i>CustomerInfo</i> .
<a href="#">JDF *</a>	element	L	Child JDF nodes. The nesting of JDF nodes defines the JDF tree.
<a href="#">NodeInfo ?</a>	element	L	Container element for process-specific information such as scheduling and messaging setup. Scheduling affects the planned times when a node should be executed. Actual times are saved in the <i>AuditPool</i> . See Section 3.4, <i>Node Information</i> in <i>NodeInfo</i> .
<a href="#">ResourceLinkPool ?</a>	element	L	Container element for <i>ResourceLink</i> elements, which describe the input and output resources of the node. See Section 3.7, <i>Resource Links</i> .
<a href="#">ResourcePool ?</a>	element	L <sup>a</sup>	Container element for resources. See Section 3.5, <i>StatusPool</i> .
<a href="#">StatusPool ?</a>	element	L	Container for <i>PartStatus</i> 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 “ <i>Pool</i> ”.

- a. Resources are unique and cannot be overwritten by descendants. Rather, they can only be used by descendants. An exception to this is described in Section 4.4.5, Case 5: Spawning and Merging of Independent Jobs . In this case, resources may also be used by a parent node.

### 3.1.2.1 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 191, known simply as process nodes. The other categories are made up of three enumerative values of the *Type* attribute: *ProcessGroup*, *Combined*, and *Product*, which is also known as product intent. These three node types are described in this section.

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

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

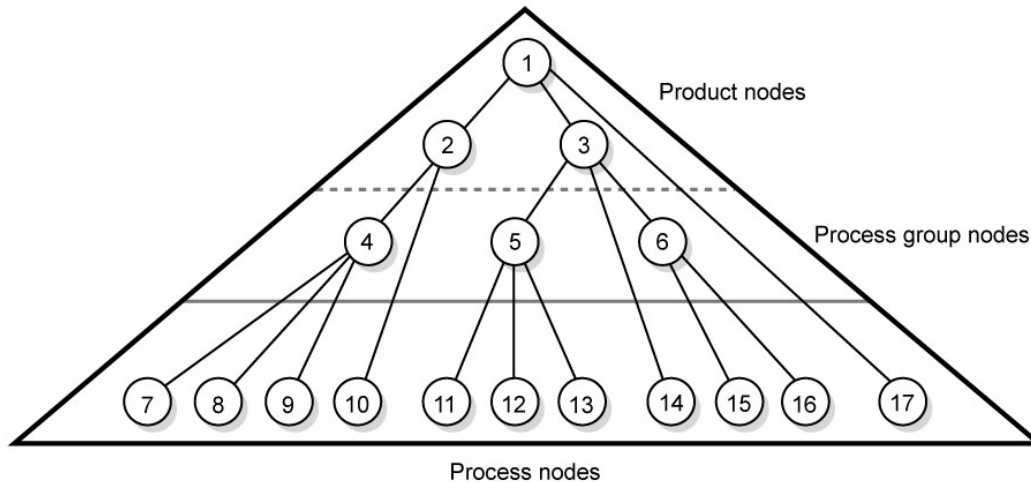


Figure 3.3: Job hierarchy with process, process group, and product intent nodes

### 3.1.3 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 may not know what devices can execute various steps or even which steps will be required.

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, Intent Resource Span Subelements. 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 required, 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 **Components** as Output Resources. For more information about product intent, see Section 4.1.1, Product Intent Constructs.

### 3.1.4 Process Group Nodes

Intermediate nodes in the JDF job hierarchy, (i.e., nodes 4, 5, and 6 in Figure 3.3), describe groups of processes. The *Type* attribute value of these kinds of nodes is *ProcessGroup*, (and they are referred to as “ProcessGroup Nodes”.) These nodes are used to describe multiple steps in a process chain that have common resources or scheduling data.

Since the agent writing the job has the option of grouping processes in any way that seems logical, custom workflows can be modeled flexibly. ProcessGroup nodes may contain further ProcessGroup nodes, individual process nodes, or a mixture of both node types. Sequencing of ProcessGroup nodes should be defined by linking resources of the appropriate leaves or, if the nature of the interchange resources is unknown, by linking **PlaceHolder** resources.

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

### 3.1.4.1 Use of the Types attribute in ProcessGroup nodes

[New in JDF 1.2](#)

ProcessGroup nodes may contain an optional *Types* attribute that allows a controller (e.g. an MIS system) to define a set of processes that must be executed without defining the exact structure or grouping of these processes into individual JDF nodes. A *CombinedProcessIndex* is used to link ResourceLinks to the *Types* in the ProcessGroup. A ResourceLink/@*CombinedProcessIndex* is used to link ResourceLinks to **JDF/@Types** in the ProcessGroup. ProcessGroup nodes with a non-empty *Types* attribute must not be executed. An Agent that receives the ProcessGroup node must define the exact structure of the ProcessGroup node by executing the following steps until the *Types* list referenced by the ProcessGroup node is empty:

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

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

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

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

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

**Step 5** — Finalize the newly created subordinate JDF node by adding any missing Resources and Resource parameters. Note that newly created resources must not be linked to the ProcessGroup node but only to the subordinate JDF node created in this process.

An Agent must instantiate all of the processes in the *Types* attribute before releasing the created JDF nodes for processing and production. The ordering of the processes in the *Types* attribute must be maintained when instantiating the child nodes. JDF ProcessGroup nodes that contain both a non-empty *Types* attribute and child JDF nodes are *not* supported, although a *ProcessGroup* node may contain child *ProcessGroup* nodes with a non-empty *Types* attribute.

### 3.1.4.2 Use of the NamedFeatures attribute in Product and ProcessGroup nodes

[New in JDF 1.2](#)

ProcessGroup and Product nodes may contain an optional *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 sub-nodes. 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/EnumerationSpan/@Name and DeviceCap/FeaturePool/EnumerationSpan/@AllowedValueList (See “Structure of the DeviceCap Subelement” on page 502.), 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 ProcessGroups as described in “Use of the Types attribute in ProcessGroup nodes” on page 44. *Types* abstractly specifies the set of processes to execute, whereas *NamedFeatures* abstractly specifies the set of Resources for the processes specified in *Types*.

### 3.1.4.3 ResourceLink Structure in ProcessGroup nodes

[New in JDF 1.2](#)

The contents of the ResourceLinkPool of a ProcessGroup node define the **Resources** that must be available for the ProcessGroup Node itself to be executed.

The following example shows the ResourceLink structure for a *ProcessGroup* digital printing with near-line finishing node. The input **Media** must be Available and the Output **Component** is of interest to the submitting Controller. The Params resources are assumed to be supplied by the sub-controller that executes the ProcessGroup node. Note the presence of intermediate component links that link the individual processes. The corresponding ResourcePools and **Resources** have been omitted for brevity.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
Type="ProcessGroup" Version="1.2">
  <!--the resource links in the ProcessGroup define the input resources that must be
available for the ProcessGroup to be submitted and the output resources that are
produced by the ProcessGroup -->
  <ResourceLinkPool>
    <!-- print input media -->
    <MediaLink Usage="Input" rRef="L2"/>
    <!-- gathered output components -->
    <ComponentLink Usage="Output" rRef="L7"/>
  </ResourceLinkPool>
  <JDF ID="J2" Status="Waiting" Type="DigitalPrinting">
    <ResourceLinkPool>
      <!-- digital printing parameters -->
      <DigitalPrintingParamsLink Usage="Input" rRef="L1"/>
      <!-- input sheets -->
      <MediaLink Usage="Input" rRef="L2"/>
      <!-- printed output components -->
      <ComponentLink Usage="Output" rRef="L3"/>
    </ResourceLinkPool>
  </JDF>
  <JDF ID="J3" Status="Waiting" Type="Gathering">
    <ResourceLinkPool>
      <!-- gathering parameters -->
      <GatheringParamsLink Usage="Input" rRef="L4"/>
      <!-- printed output components -->
      <ComponentLink Usage="Input" rRef="L3"/>
      <!-- gathered output components -->
      <ComponentLink Usage="Output" rRef="L5"/>
    </ResourceLinkPool>
  </JDF>
  <JDF ID="J4" Status="Waiting" Type="Stitching">
    <ResourceLinkPool>
      <!-- Stitching parameters -->
      <StitchingParamsLink Usage="Input" rRef="L6"/>
```

```

    <!-- gathered output components -->
    <ComponentLink Usage="Input" rRef="L5"/>
    <!-- stitched output components -->
    <ComponentLink Usage="Output" rRef="L7"/>
  </ResourceLinkPool>
</JDF>
</JDF>

```

### 3.1.5 Combined Process Nodes

#### [Clarified in JDF 1.2](#)

The processes described in “Processes” on page 191 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 may 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 are assumed to 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 subprocesses that define the Combined node and that are explicitly defined as *ResourceLinks* must be available before the node can be executed. In addition, all input and output resources that are consumed and produced externally by the process must be specified in the *ResourceLinkPool* element of the node. This includes all required *Parameter* resources as well as the initial input resources and final output resources. Intermediate resources that are internally produced and consumed, on the other hand, need not be specified.

In a Combined Process node, the information defined by the various resources linked as input to the various subprocesses are logically available to all processes of the combined node. In situations where the parameter resource of more than one subprocess specifies the mapping of sheet surface content to media, the subprocess that specifies such a mapping that is defined earliest in the *Types* attribute list must be used, and any other mappings specified by any down-stream subprocess *Resource* must be ignored.

#### 3.1.5.1 Combined Process Nodes with Multiple Processes of the Same Type

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

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting" Type="Combined" Types="Cutting
Folding Cutting" Version="1.2">
  <!--Resources (incomplete...) -->
  <ResourcePool>
    <!-- parameters of the first Cutting Process-->
    <CuttingParams Class="Parameter" ID="L1" Status="Available"/>
    <!-- Folding parameters -->
    <FoldingParams Class="Parameter" ID="L2" Status="Available"/>
    <!-- parameters of the third Cutting Process-->
    <CuttingParams Class="Parameter" ID="L3" Status="Available"/>
    <!-- raw input components -->

```

```

<Component Class="Quantity" ID="L4" Status="Available"/>
<!-- completed output components -->
<Component Class="Quantity" ID="L5" Status="Unavailable"/>
</ResourcePool>
<!-- Links -->
<ResourceLinkPool>
  <!-- parameters of the first Cutting Process-->
  <CuttingParamsLink CombinedProcessIndex="0" Usage="Input" rRef="L1"/>
  <!-- Folding parameters -->
  <FoldingParamsLink CombinedProcessIndex="1" Usage="Input" rRef="L2"/>
  <!-- parameters of the first Cutting Process-->
  <CuttingParamsLink CombinedProcessIndex="2" Usage="Input" rRef="L3"/>
  <!-- raw input components -->
  <ComponentLink Usage="Input" rRef="L4"/>
  <!-- completed output components -->
  <ComponentLink Usage="Output" rRef="L5"/>
</ResourceLinkPool>
</JDF>

```

### 3.1.5.2 Examples of Combined Process Nodes

The following example of the `ResourceLinkPool` of a JDF node describes digital printing with in-line finishing and includes the same processes as the previous `ProcessGroup` example. The node requires the parameter resources and consumable resources of all three processes as inputs, and produces a completed booklet as output. The intermediate printed sheets and gathered piles are not declared, since they exist only internally within the device and cannot be accessed or manipulated by an external controller.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting" Type="Combined"
Types="DigitalPrinting Gathering Stitching" Version="1.2">
  <ResourceLinkPool>
    <!-- digital printing input RunList -->
    <RunListLink CombinedProcessIndex="0" Usage="Input" rRef="L1"/>
    <!-- digital printing parameters -->
    <DigitalPrintingParamsLink CombinedProcessIndex="0" Usage="Input" rRef="L2"/>
    <!-- gathering parameters -->
    <GatheringParamsLink CombinedProcessIndex="1" Usage="Input" rRef="L3"/>
    <!-- Stitching parameters -->
    <StitchingParamsLink CombinedProcessIndex="2" Usage="Input" rRef="L4"/>
    <!-- input sheets -->
    <MediaLink CombinedProcessIndex="0" Usage="Input" rRef="L5"/>
    <!-- stitched output components -->
    <ComponentLink CombinedProcessIndex="2" Usage="Output" rRef="L6"/>
  </ResourceLinkPool>
</JDF>

```

### 3.1.6 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 may 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*.

## 3.2 AncestorPool

When a job is spawned, an **AncestorPool** is created in the spawned job 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 job. 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 job. 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 optionally customer information.

Table 3-5: Contents of the **AncestorPool** element

Name	Data Type	Description
<b>Ancestor +</b>	element	Ordered list of one or more <b>Ancestor</b> elements, which reflect the family tree of a spawned job.
<b>Part *</b> <a href="#">New in JDF 1.1</a>	element	List of parts that this node was spawned with. Used in case of parallel spawning of a node. This defines the aggregated <b>Part(s)</b> in case of nested spawns, (i.e., a logical AND of all spZawn <b>Part(s)</b> ). 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 <b>Part</b> .

An **Ancestor** element may 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 cannot, however, contain further subelements except for read-only copies of **CustomerInfo** and **NodeInfo**. The attributes of **Ancestor** elements are described in Table 3-6 below.

Table 3-6: Attributes of the **Ancestor** element

Name	Data Type	Description
<i>Activation</i> ?	enumeration	Copy of the <i>Activation</i> attribute from the ancestor node. For details, see Table 3-4, "Contents of a JDF node," on page 38.
<i>FileName</i> ?	URL	The URL of the JDF file where the ancestor node resided prior to spawning.
<i>JobID</i> ?	string	Copy of the <i>JobID</i> attribute from the ancestor node. For details, see Table 3-4, "Contents of a JDF node," on page 38.
<i>JobPartID</i> ?	string	Copy of the <i>JobPartID</i> attribute from the original ancestor node. For details, see Table 3-4, "Contents of a JDF node," on page 38.
<i>MaxVersion</i> ? <a href="#">New in JDF 1.2</a>	JDFJMFVersion	Copy of the <i>MaxVersion</i> attribute from the original ancestor node. For details, see Table 3-4, "Contents of a JDF node," on page 38.
<i>NodeID</i>	NMTOKEN <sup>a</sup>	Copy of the <i>ID</i> attribute of the ancestor node.
<i>ProjectID</i> ?	string	Identification of the project context that this JDF belongs to. Used by the application that created the JDF job.
<i>SpawnID</i> ? <a href="#">New in JDF 1.1</a>	NMTOKEN	Copy of the <i>SpawnID</i> attribute of the ancestor node.
<i>Status</i> ?	enumeration	Copy of the <i>Status</i> attribute from the original ancestor node. For details, see Table 3-4, "Contents of a JDF node," on page 38.
<i>StatusDetails</i> ? <a href="#">New in JDF 1.2</a>	string	Copy of the <i>StatusDetails</i> attribute from the original ancestor node. For details, see Table 3-4, "Contents of a JDF node," on page 38.
<i>Type</i> ?	NMTOKEN	Copy of the <i>Type</i> attribute from the original ancestor node. For details, see Table 3-4, "Contents of a JDF node," on page 38.

Table 3-6: Attributes of the Ancestor element

Name	Data Type	Description
<i>Types</i> ?	NMTOKENS	Copy of the <i>Types</i> attribute from the original ancestor node. For details, see Table 3-4, “Contents of a JDF node,” on page 38.
<i>Version</i> ?	JDFJMFVersion	Copy of the <i>Version</i> attribute from the original ancestor node. For details, see Table 3-4, “Contents of a JDF node,” on page 38.
<i>CustomerInfo</i> ? <a href="#">New in JDF 1.1</a>	element	Reference copy of the <i>CustomerInfo</i> element from the original node. For details, see Table 3-4, “Contents of a JDF node,” on page 38.
<i>NodeInfo</i> ? <a href="#">New in JDF 1.1</a>	element	Reference copy of the <i>NodeInfo</i> element from the original node. For details, see Table 3-4, “Contents of a JDF node,” on page 38.

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

### 3.3 Customer Information in CustomerInfo

The *CustomerInfo* element contains information about the customer who orders the job. Usually this element is specified in the uppermost node of a job, (i.e., the root node), although it is also valid in lower nodes in situations such as model subcontracting. Table 3-7, “Contents of the *CustomerInfo* element,” on page 49 describes the contents of this element.



#### Creating Better Job Tracking & Reporting

Customer information within JDF can provide a bridge between your CRM systems and production. How could JDF be used to automate the process of reporting to customers on the status of their jobs?

Table 3-7: Contents of the CustomerInfo element

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> ? <a href="#">New in JDF 1.2</a>	string	The internal project id in the system of the customer. This number may be provided when the order is placed and then referenced on the order confirmation or the bill.
<i>rRefs</i> ? <a href="#">Deprecated in JDF 1.2</a>	IDREFS	Array of <i>IDs</i> of any elements that are specified as <i>ResourceRef</i> elements. In version 1.1 it was the IDREF of a <i>ContactRef</i> . In JDF 1.2 and beyond, it is up to the implementation to maintain references.
<i>Company</i> ? <a href="#">Deprecated in JDF 1.1</a>	refelement	Resource element describing the business or organization of the contact. In JDF 1.1 and beyond, <i>Company</i> affiliation of <i>Contacts</i> is specified in <i>Contact</i> .
<i>Contact</i> * <a href="#">New in JDF 1.1</a>	refelement	Resource element describing contacts associated with the customer. There should be one <i>Contact</i> which has <i>ContactTypes</i> including “ <i>Customer</i> ”. The <i>Contact</i> with <i>ContactTypes</i> including “ <i>Customer</i> ” specifies the name, address etc. of the primary customer.
<i>CustomerMessage</i> * <a href="#">New in JDF 1.2</a>	element	Element that describes messages to the customer.

## Structure of the CustomerMessage Element

[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 3-8: Contents of the CustomerMessage element

Name	Data Type	Description
<b>ComChannel</b> *	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 communication channels.
<i>Language</i> ?	language	Language to be used for the CustomerMessage.
<i>MessageEvents</i>	NMTO-KENS	Defines the set of events that trigger a message that is defined or specified by the system. A list of predefined values is provided in “MessageEvents Values” on page 623.
<i>ShowList</i> ?	NMTO-KENS	List of parameters to display in the CustomerMessage. Values include: <i>Amount</i> – Amount of the product that was produced. <i>DeviceID</i> – <i>ID</i> of the device. This is a unique name within the workflow. <i>EndTime</i> – Actual <i>EndTime</i> of the job. <i>Error</i> – Errors that happened during the job. <i>FriendlyName</i> – <i>FriendlyName</i> of the device. <i>JobName</i> – <i>DescriptiveName</i> of the node that is executing. <i>JobRecipientName</i> – Name of the recipient of the job. <i>JobSubmitterName</i> – Name of the submitter of the job. <i>StartTime</i> – Actual <i>StartTime</i> of the job. <i>MediaBrand</i> – <i>Brand</i> of the media that is being printed. <i>MediaType</i> – <i>DescriptiveName</i> of the media that is being printed. <i>Operator</i> – Name of the Operator. <i>Resolution</i> – Output resolution. <i>ResolutionX</i> – Output resolution in X direction. <i>ResolutionY</i> – Output resolution in Y direction. <i>ScreeningFamily</i> – Name of the screening family of the output. <i>UserText</i> – User defined text as defined in a <b>Comment</b> with <b>Comment/@Name="UserText"</b> . <i>Warning</i> – Warnings that happened during the job.

### 3.4 Node Information in NodeInfo

The NodeInfo element contains information about planned scheduling and message routing. It allows MIS to plan, schedule and invoice jobs or job parts. Table 3-9 below describes the contents of the NodeInfo element.

Table 3-9: Contents of the NodeInfo element

Name	Data Type	Description
<i>CleanupDuration</i> ?	duration	Estimated duration of the clean-up phase of the process.
<i>DueLevel</i> ?	enumeration	Description of the severity of a missed deadline. Possible values are: <i>Unknown</i> – Consequences of missing the deadline are not known. <a href="#">Deprecated in JDF 1.2</a> <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 may end.
<i>FirstStart</i> ?	dateTime	Earliest date and time at which the process may begin.
<i>IPPVersion</i> ? <a href="#">New in JDF 1.1</a>	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" <a href="#">New in JDF 1.1</a>	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 should be printed before any lower priority ones. If one of the deadline oriented attributes (e.g., <i>FirstStart</i> or <i>LastEnd</i> and <i>JobPriority</i> are specified), the deadline oriented attributes must be honored before considering <i>JobPriority</i> .
<i>LastEnd</i> ?	dateTime	Latest date and time at which the process may end. This is the deadline to which <i>DueLevel</i> refers.
<i>LastStart</i> ?	dateTime	Latest date and time at which the process may begin.
<i>NaturalLang</i> ? <a href="#">New in JDF 1.1</a>	language	Language selected for communicating attributes. If not specified, the operating system language is assumed.
<i>MergeTarget</i> ? <a href="#">Deprecated in JDF 1.1</a>	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. <b>Note:</b> <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 should execute this node. If <i>Route</i> is not specified, the routing controller must determine a potential controller or device independently. For details, see Section 4.2, Process Routing. Note that <i>Route</i> must not be evaluated by the receiving Device, to determine whether the node should be executed. Selecting a Device for execution is specified by defining an input <b>Device</b> resource.
<i>rRefs</i> ? <a href="#">Deprecated in JDF 1.2</a>	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 <i>Employee</i> . 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.

Table 3-9: Contents of the *NodeInfo* element

Name	Data Type	Description
<i>TargetRoute</i> ?	URL	The URL where the JDF should 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 should be sent to the processor default output URL. JMF/QueueSubmissionParams/@ReturnURL takes precedence over NodeInfo/@TargetRoute of the JDF that is processed.
<i>TotalDuration</i> ?	duration	Estimated total duration of the process, including setup and cleanup.
<i>BusinessInfo</i> ?	element	Container for business related information. It is expected that JDF will be utilized in conjunction with other E-commerce standards, and this container is provided to store the E-commerce information within JDF in case a workflow with JDF as the root level document is desired. When JDF is used as part of an E-commerce solution such as PrintTalk, the information given in the envelope document overrides the information in <i>BusinessInfo</i> .
<i>Employee</i> ?	refelement	The internal administrator or supervisor that is responsible for the product or process defined in this node.
JMF *	element	Represents JMF query messages that set up a persistent channel, as described in Section 5.2.2.3, <i>Persistent Channels</i> . These message elements define the receiver that is designated to track jobs via JMF messages. These message elements should be honored by any JMF-capable controller or device that executes this node. When these messages are honored, a persistent communication channel is established that allows devices to transmit, (e.g., the status of the job as JMF Signals).
<i>MISDetails</i> ? <a href="#">New in JDF 1.2</a>	refelement	Definition how the costs for the execution of this node are to be charged.
<i>NotificationFilter</i> *	element	Defines the set of <i>Notification</i> elements that should be logged in the <i>AuditPool</i> . This provides a logging method for devices that do not support JMF messaging. For details of the <i>NotificationFilter</i> element, see Section 5.5.1.1, <i>Events</i> .

### 3.5 StatusPool

The *StatusPool* describes the *Status* of a JDF node that processes partitioned resources. *StatusPool* elements are only valid if the node's *Status* = *Pool*, otherwise the node's *Status* is valid for all parts, regardless of the contents of *StatusPool*. It may contain *PartStatus* elements that define the node's status with respect to specific partitions. It is an error to define *PartStatus* elements that reference identical or overlapping parts within one *StatusPool*. Partitioned resources are described in Section 3.8.2, *Description of Partitionable Resources*.

Table 3-10: Contents of the *StatusPool* element

Name	Data Type	Description
<i>Status</i> ?	enumeration	Identifies the <i>Status</i> of the node when JDF/@Status="Pool". Individual <i>PartStatus</i> 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 <i>PartStatus/@Status</i> . Possible values are all valid <i>Status</i> attributes of a JDF node except <i>Pool</i> are valid as defined in Table 3-4, "Contents of a JDF node," on page 38, <i>Status</i> .
<i>StatusDetails</i> ? <a href="#">New in JDF 1.2</a>	string	Identifies the <i>StatusDetails</i> of the node when JDF/@Status="Pool". Individual <i>PartStatus</i> 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 <i>PartStatus/@StatusDetails</i> . For a list of supported values, see "StatusDetails Supported Strings" on page 615.
<i>PartStatus</i> *	element	Element that defines the node's status for a set of parts.



The following table describes the **PartStatus** element.

Table 3-11: Contents of the **PartStatus** element

Name	Data Type	Description
<b>Status</b> ?	enumeration	Identifies the status of an individual part of the node. If not specified, defaults to <b>StatusPool/@Status</b> . Possible values are identical to those defined in defined in Table 3-4, “Contents of a JDF node,” on page 38, <b>Status</b> .
<b>StatusDetails</b> ? <a href="#">New in JDF 1.2</a>	string	Description of the status that provides details beyond the enumerative values given by the <b>Status</b> attribute. If not specified, defaults to <b>StatusPool/@StatusDetails</b> . For a list of supported values, see “StatusDetails Supported Strings” on page 615.
<b>Part</b> <sup>a</sup> <a href="#">Modified in JDF 1.2</a>	element	Specifies the selected part that the <b>PartStatus</b> 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 <b>Part</b> may contain either <i>Side</i> only or <i>Side</i> and <i>Separation</i> , but not <i>Separation</i> only. See “Contents of the Part element” on page 79 for details of the <b>Part</b> element. For details on partitioned resources, see “Description of Partitionable Resources” on page 74.

- a. The cardinality of **Part** in **PartStatus** has been changed from \* to none, (e.g., exactly one element) in version 1.1 of the JDF specification.

### 3.6 Resources

Resources represent the “things” that are produced or consumed by processes. They may be physical items such as inks, plates, or glue; electronic items such as files or images; or conceptual items such as parameters and device settings. Processes describe what resources they input or output through ResourceLinks, discussed in Section 3.7, Resource Links. By examining the input and outputs of a set of processes, it is possible to determine process dependencies, and therefore job routing.

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

Table 3-12: Contents of the **ResourcePool** element

Name	Data Type	Description
<b>Resource</b> *	element	List of <b>Resource</b> elements. The <b>Resource</b> elements are abstract and serve as placeholders for any resource type.

Like the *Type* attribute in abstract JDF nodes, the *Class* attribute in **Resource** elements helps to identify how particular resources should be used. These values are listed in Table 3-13, “Contents of the abstract Resource element,” on page 53, below, and are described in greater detail in the sections that follow.

Table 3-13: Contents of the abstract Resource element

Name	Data Type	Description
<b>AgentName</b> ? <a href="#">New in JDF 1.2</a>	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.
<b>AgentVersion</b> ? <a href="#">New in JDF 1.2</a>	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.
<b>Author</b> ? <a href="#">New in JDF 1.2</a>	string	Text that identifies the person who generated the resource.

Table 3-13: Contents of the abstract Resource element

Name	Data Type	Description
<i>CatalogID</i> ?	string	Identification of the resource, (e.g., in a catalog environment). Defaults to the value of <i>ProductID</i> .
<i>CatalogDetails</i> ?	string	Additional details of a resource in a catalog environment.
<i>Class</i>	enumeration	Defines the abstract resource type. For details, see the sections that follow. Possible values are: <i>Consumable</i> <i>Handling</i> <i>Implementation</i> <i>Intent</i> <i>Parameter</i> <i>Placeholder</i> <i>Quantity</i> <i>Class</i> must be specified in the resource root and must not be overwritten in a resource leaf.
<i>ID</i>	ID	Unique identifier of a resource. <i>ID</i> must be specified in the resource root and must not be overwritten in a resource leaf.
<i>Locked</i> = "false"	boolean	If <i>true</i> , the resource is spawned in read-only mode or referenced by an <i>Audit</i> and must not be modified without invalidating the <i>Audit</i> .
<i>PartUsage</i> = "Explicit" <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	enumeration	Description of the interpretation of partitions. One of: <i>Explicit</i> – Require explicit partition matches. All referenced partitions referenced in <i>Part</i> must exist, otherwise it is an error. <i>Implicit</i> – Allow sparse overrides of default values. The closest matching partition with no non-matching partition keys is returned. <i>PartUsage</i> must only be specified in the root of a resource. For details on <i>PartUsage</i> and partitioning, see Section 3.8.3.2, Implicit and Explicit <i>PartUsage</i> in Partitioned Resources. <i>PartUsage</i> was moved to this table from Table 3-27 on page 78 in JDF 1.2.
<i>PipeID</i> ?	string	If this attribute exists, the resource is a pipe. <i>PipeID</i> is used by JMF pipe-control messages to identify the pipe. For more information, see “Overlapping Processing Using Pipes” on page 114.
<i>PipeProtocol</i> ? <a href="#">New in JDF 1.2</a>	NMTOKEN	Defines the protocol use for pipe handling. <i>JMF</i> and <i>Internal</i> are the only non-proprietary piping protocols that are supported. Proprietary pipe protocols may be specified in addition to those defined below but will not necessarily be interoperable. Allowed values include: <i>Internal</i> – Internal or virtual pipe used within a combined process. <i>JMF</i> – JMF-based PipePush / PipePull messages. <i>None</i> – No pipe support.
<i>PipeURL</i> ? <a href="#">New in JDF 1.2</a>	URL	Pipe request URL. Dynamic pipe requests to this resource should be made <i>to</i> this URL. <sup>a</sup> Note that this URL is only used for initiating pipe requests. Responses to a pipe request are issued to the URL that is defined in the PipePush or PipePull message. For details on using <i>PipeURL</i> , see Section 4.3.3, Overlapping Processing Using Pipes.
<i>ProductID</i> ?	string	An ID of the resource as defined in the MIS system. For instance item codes or article numbers or identifiers on semi-finished products or handling resources.
<i>rRefs</i> ? <a href="#">Deprecated in JDF 1.2</a>	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.

Table 3-13: Contents of the abstract Resource element

Name	Data Type	Description
<a href="#">SpawnIDs ?</a> <a href="#">New in JDF 1.1</a>	NMTOKENS	List of SpawnIDs. This is used as a reference count for how often the resource has been spawned.
<a href="#">SpawnStatus</a> ="NotSpawned"	enumeration	The spawn status of a resource indicates whether or not a resource has been spawned, and under what circumstances. The list of possible values is assumed to be ordered, so that the <i>SpawnStatus</i> of a resource that has <b>ResourceRef</b> elements is defined as the maximum <i>SpawnStatus</i> of all recursively linked resources. Possible values, ordered from lowest to highest are: <i>NotSpawned</i> — Indicates that the resource has not been copied to another process. <i>SpawnedRO</i> — Indicates that the resource has been copied to another process where it cannot be modified. RO stands for read-only. <i>SpawnedRW</i> — Indicates that the resource has been copied to another process where it can be modified. RW stands for read/write.
<a href="#">Status</a> <a href="#">Modified in JDF 1.2</a>	enumeration	The status of a resource indicates under what circumstances it may be processed or modified. The list of possible values is assumed to be ordered, so that the <i>Status</i> of a resource that references further resources is defined as the minimum <i>Status</i> of all recursively linked resources. Possible values, ordered from lowest to highest, are: <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. <i>Rejected</i> — Indicates that the resource has been rejected by an <b>Approval</b> process. The metadata is valid. <a href="#">New in JDF 1.2</a> <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. <i>InUse</i> — Indicates that the resource exists, but is in use by another process. Also used for active pipes (see Section 3.6.3, Pipe Resources and Section 4.3.3, Overlapping Processing Using Pipes). <i>Draft</i> — Indicates that the resource exists in a state that is sufficient for setting up the next process but not for production. <i>Complete</i> — Indicates that the resource is completely specified and the parameters are valid for usage. A physical resource with <i>Status</i> = "Complete" is not yet available for production, although it is sufficiently specified for a process that references it through a <b>ResourceRef</b> from a parameter resource to commence execution. <i>Available</i> — Indicates that the whole resource is available for usage.
<a href="#">UpdateID ?</a> <a href="#">New in JDF 1.1</a>	NMTOKEN	Unique ID that identifies the <b>Resource</b> or <b>Resource</b> partition. Note that only one <b>Resource</b> , <b>Resource</b> partition or <b>ResourceUpdate</b> with a given value of <i>UpdateID</i> may occur per JDF document, even though the scope of the <b>ResourceUpdate</b> is local to the resource that it is defined in.
<a href="#">QualityControlResult *</a> <a href="#">New in JDF 1.2</a>	refelement	Results of quality measurements which were performed during or after the production of this resource.

- a. Note that in most cases this is the URL of the controller of the *other end* of the pipe. This may 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.

Figure 3.4 shows the structure of the abstract resource classes defined above. Arrows define inheritance relations and the thin orthogonal lines describe containing relations.

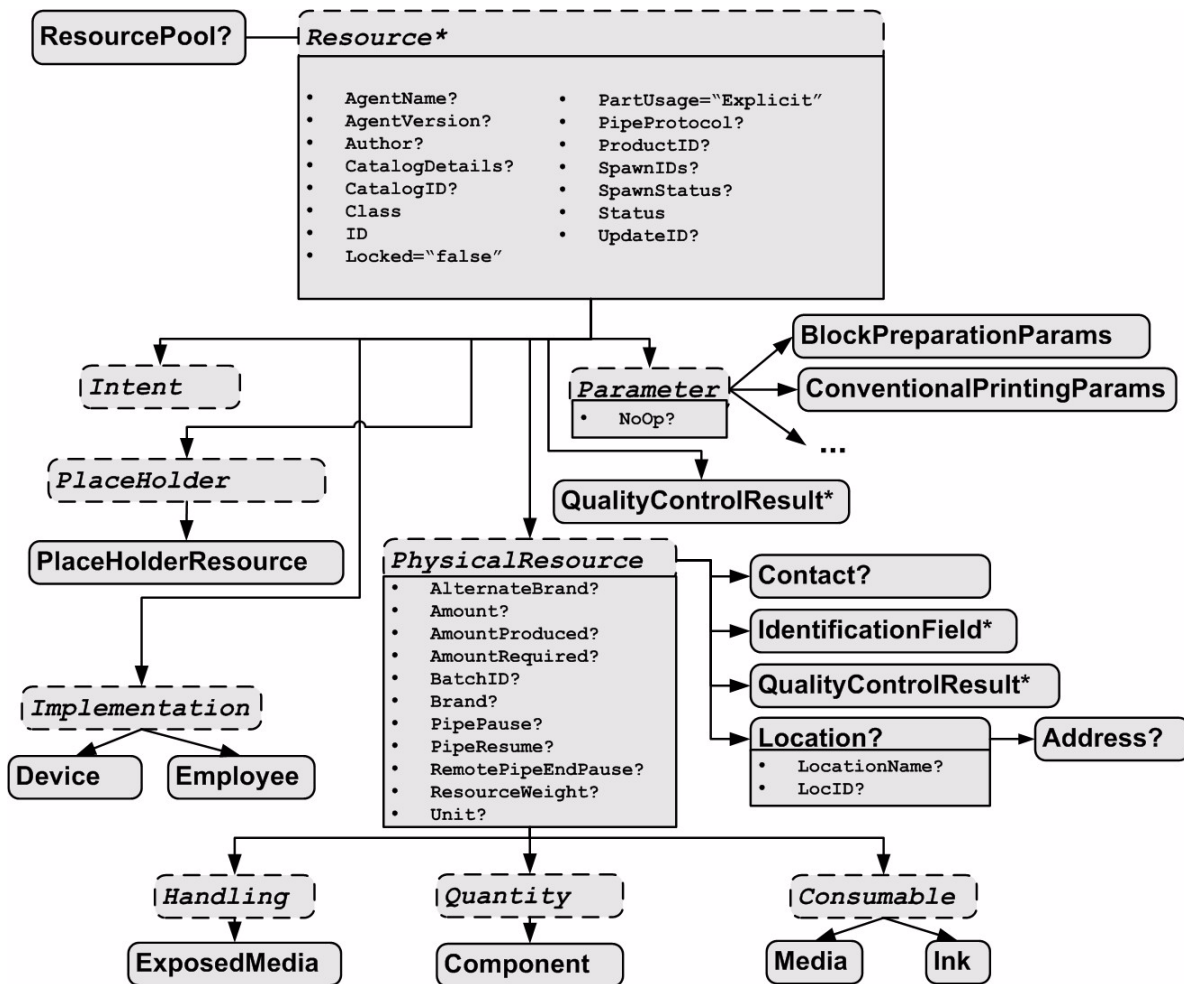


Figure 3.4: Structure of the abstract resource types

### 3.6.1 Resource Classes

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

#### 3.6.1.1 Parameter Resources

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



#### Parameter & Intent Resources

Parameter and Intent Resources are *information* about the job. Intent resources may originate in the customer’s RFQ and may 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 may be transformed into parameters for production process.

Table 3-14: Additional contents of the abstract parameter Resource elements

Name	Data Type	Description
<i>NoOp</i> = "false" <a href="#">New in JDF 1.1</a> <a href="#">Clarified in JDF 1.2</a>	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., <b>RunList</b> or <b>Component</b> ).

### 3.6.1.2 Intent Resources

*Intent* resources define the details of products to be produced without defining the process to produce them. In addition, they provide structures to define sets of allowable options and to match these selections with prices. The details of all intent resources are described in Section 7.1, *Intent Resources*. The abstract *Intent* resource element contains no attributes or elements besides those contained in the abstract *Resource* element.

### 3.6.1.3 Implementation Resources

*Implementation* resources define the devices and operators that execute a given node. Only two implementation resource types are defined: **Employee** (see Section 7.2.61, *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 3.7.2, *Links to Implementation Resources*.


Note that it is not recommended to specify the capabilities of a **Device** that is linked to a process to specify that it should execute the given process.

### 3.6.1.4 Physical Resources (Consumable, Quantity, Handling)

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

- *Consumable* resources are consumed during a process. Examples include **Ink** and **Media**. They are the unmodified inputs in a process chain.
- *Quantity* resources have been created by a process from either a *Consumable* resource or an earlier *Quantity* resource. For example, printed sheets are cut and a pile of cut blocks is created. **Component** resources are an example of *Quantity* resources.
- A *Handling* resource is used during a process, but is not destroyed by that process. **ExposedMedia** and **Tool** are examples of such a resource, although it does describe various kinds of items such as film and plates. A *Handling* resource may be created from a *Consumable* resource.

Table 3-15, "Additional contents of the abstract physical Resource elements," on page 58, defines the additional attributes and elements that may 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 should be consumed. Table 3-16 then describes the contents of the **Location** subelement of physical resource elements.



**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 3-15: Additional contents of the abstract physical Resource elements

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 resource links. For details on amount handling, see Section 3.8.1, Resource Amount.
<i>AmountProduced</i> ? <a href="#">New in JDF 1.2</a>	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>Actualmount</i> values of output resource links of leaf JDF nodes with <i>Status</i> = "Completed" that reference this resource
<i>AmountRequired</i> ?	double	Total amount of the resource that is referenced by all nodes that will consume this resource. This corresponds to the sum of all <i>Amount</i> values of input resource links that reference this resource.
<i>BatchID</i> ?	string	ID of a specific batch of the physical resource
<i>Brand</i> ? <a href="#">Clarified in JDF 1.2</a>	string	Information, such as the manufacturer, model, part number, and/or type, about the resource being used. Some examples are as follows. <ul style="list-style-type: none"> <li>• XYZ Premium InkProp Glossy 6x642A</li> <li>• ZYX Premium Multipurpose 1234, 88 Bright 24 lb. Bond, 8-1/2 x 11, White Copy Paper Reorder 4711</li> </ul>
<i>ResourceWeight</i> ? <a href="#">New in JDF 1.1</a>	double	Weight of a single component of the resource in grams.
<i>Unit</i> ?	NMTO-KEN	Unit of measurement for the values of <i>Amount</i> and <i>AmountRequired</i> . Note that it is strongly discouraged to specify units other than those that are defined in Section 1.6, Units.
<i>Contact</i> ?	refelement	If this element is specified, it describes the owner of the resource.
<i>IdentificationField</i> * <a href="#">New in JDF 1.1</a>	refelement	If this element is specified, a bar code or label is associated with this physical resource.
<i>Location</i> ?	refelement	Description of details of the resource location. Note, in order to describe multiple locations, resources may be partitioned by the <i>Location</i> -key as described in Section 3.8.2, Description of Partitionable Resources.

### Structure of Location Subelement

Table 3-16: Contents of the Location element

Name	Data Type	Description
<i>LocationName</i> ? <a href="#">New in JDF 1.1</a>	string	Name of the location, (e.g., in MIS). This part key allows the user to describe distributed resources.
<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.

### 3.6.1.5 Placeholder Resources

*Placeholder* resources, unlike physical resources, do not describe any logical or physical entity. Rather, they define process linking and help to define process ordering when the exact nature of interchange resources is still unknown. In essence, they serve as placeholders that stand in for defined resources. Using *Placeholder* resources, a processing skeleton can be constructed that gives a basic shape to a job. The appropriate resources can be substituted for *Placeholder* resources when they become known.

This kind of resource should only be used to link nodes of *Type* = *ProcessGroup*, since process leaf nodes have well defined resources that should be used in preference. The only resource whose *Class* = *Placeholder* is called **PlaceholderResource**.

Like *Implementation* resources, *Placeholder* resources contain no attributes besides those contained in the abstract **Resource** element.

### 3.6.2 Position of Resources within JDF Nodes

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

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

A process that needs additional detailed process information specifying the creation of a resource must infer this information by explicitly linking to the appropriate parameter resource.

### 3.6.3 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 may 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.

### 3.6.4 ResourceUpdate Elements

[New in JDF 1.1](#)

**ResourceUpdate** elements are an abstract element class that optionally contains any of the attributes and elements valid for the **Resource** that they reside in. The naming convention for **ResourceUpdate** elements is to add the suffix "Update" to the resource name. Required attributes and elements of resources are optional in the respective **ResourceUpdate**. In addition, a **ResourceUpdate** defined within a **Resource** must contain a unique *UpdateID* of type NMToken. Only devices that process the resource as input can reference the *UpdateID* of a **ResourceUpdate**. Such references to **ResourceUpdate** elements must update the current state of the device.

When a **ResourceUpdate** is referenced from a device (e.g., from a PPML TicketRef element [PPML]), said device will update **ONLY** those elements that are explicitly specified within the **ResourceUpdate**. No attributes are inherited from the **Resource** that contains the **ResourceUpdate**.

**ResourceUpdate** elements are useful for process input resources only and must not be applied to product intent resources.

Table 3-17: Contents of the abstract ResourceUpdate Element

Name	Data Type	Description
<a href="#">UpdateID</a> <a href="#">New in JDF 1.1</a>	NMTO-KEN	Unique ID that identifies the <b>ResourceUpdate</b> . Note that only one <b>Resource</b> , <b>Resource</b> partition, or <b>ResourceUpdate</b> with a given value of <i>UpdateID</i> may occur per JDF document, even though the scope of the <b>ResourceUpdate</b> is local to the resource that it is defined in.

### Example:

The following example shows **ResourceUpdate** elements in **highlight**.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="MyCombinedProcessNode"
Status="Ready" Type="Combined" Types="Interpreting Rendering DigitalPrinting"
Version="1.2">
  <ResourceLinkPool>
    <InterpretingParamsLink CombinedProcessIndex="0" Usage="Input" rRef="PDFIPParams"/>
    <RenderingParamsLink CombinedProcessIndex="1" Usage="Input" rRef="RParams"/>
    <DigitalPrintingParamsLink CombinedProcessIndex="2" Usage="Input" rRef="DPPParams"/>
    <MediaLink CombinedProcessIndex="2" Usage="Input" rRef="White"/>
    <MediaLink CombinedProcessIndex="2" Usage="Input" rRef="Yellow"/>
    <RunListLink CombinedProcessIndex="0" Usage="Input" rRef="RunList"/>
    <ComponentLink Usage="Output" rRef="OutComp"/>
  </ResourceLinkPool>
  <ResourcePool>
    <Media Class="Consumable" ID="White" Status="Available"/>
    <Media Class="Consumable" ID="Yellow" Status="Available"/>
    <InterpretingParams Class="Parameter" ID="PDFIPParams" Polarity="Positive"
PrintQuality="High" Status="Available" UpdateID="SetPrintQualityDefault">
      <InterpretingParamsUpdate Polarity="Negative" UpdateID="SetNegativePolarity"/>
      <InterpretingParamsUpdate Polarity="Positive" UpdateID="SetPositivePolarity"/>
      <InterpretingParamsUpdate PrintQuality="Draft" UpdateID="SetPrintQDraft"/>
      <InterpretingParamsUpdate PrintQuality="Normal" UpdateID="SetPrintQNormal"/>
      <InterpretingParamsUpdate PrintQuality="High" UpdateID="SetPrintQualityHigh"/>
    </InterpretingParams>
    <RenderingParams Class="Parameter" ID="RParams" Status="Available">
      <AutomatedOverprintParams OverPrintBlackLineArt="true" OverPrintBlackText="true"/>
    </RenderingParams>
    <DigitalPrintingParams Class="Parameter" ID="DPPParams" PrintingType="SheetFed"
Status="Available">
      <MediaRef UpdateID="SetMediaDefault" rRef="White"/>
      <DigitalPrintingParamsUpdate UpdateID="SetMediaYellow">
        <MediaRef rRef="Yellow"/>
      </DigitalPrintingParamsUpdate>
    </DigitalPrintingParams>
    <RunList Class="Parameter" ID="RunList" Status="Available"/>
    <Component Class="Quantity" ID="OutComp" Status="Unavailable"/>
  </ResourcePool>
</JDF>
```



### 3.7 Resource Links

ResourceLink elements describe what resources a node uses, and how it uses them. They 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 resource links, ALink and BLink, in the ResourceLinkPool. The resource links are 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.

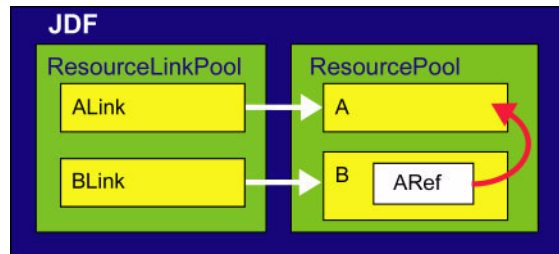


Figure 3.5: Resource Links and ResourceRefs

The previous section described resources used by the node in which it resides. This section describes how resources may 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.

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. 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 may in turn become inputs in another node, and a given node must not be executed before all required input resources are available.<sup>1</sup> 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.

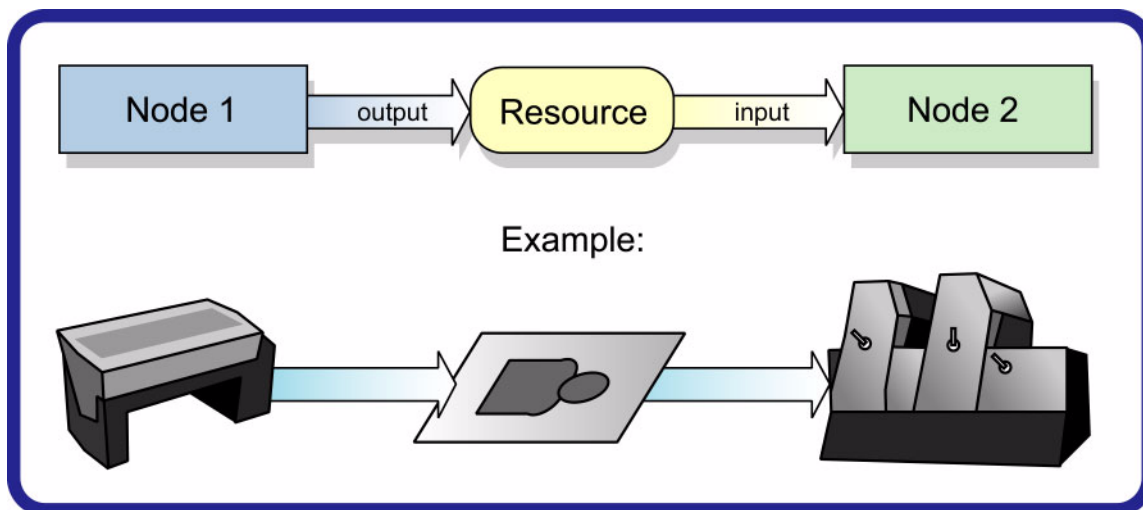


Figure 3.6: Nodes linked by a resource

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.

**ResourceLink** elements may also contain optional attributes to select a part of a resource, such as a single separation. A detailed description of resource partitioning is given in Section 3.8.2, *Description of Partitionable Resources*.

*ProcessGroup* and *Product* nodes may 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 may 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 may remove the **ResourceLink** elements from the intermediate node. Removing the **ResourceLink** specifies that the intermediate node may execute; (i.e., it may be sent to the appropriate controller or department), even though the specific resources are not yet available. If the **ResourceLinks** are not removed, the intermediate node must not execute until the input resources that are linked are available.

Resource links may be used for process control. For example, if a proof input resource is required for a print process, a print run may only commence 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 information should be stored in an input or an output resource may 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 applied for the definition of input and output resources of processes as described in 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 are always input resources.
- *Quantity* and *Handling* resources are used both as input and output resources. Their usage is defined by the “natural” process usage. For example, a printing plate is described as an 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 defines the output resource of its parent process.
- In case of parallel processing, the sum of the outputs of all parallel subnodes defines the output of the parent node.

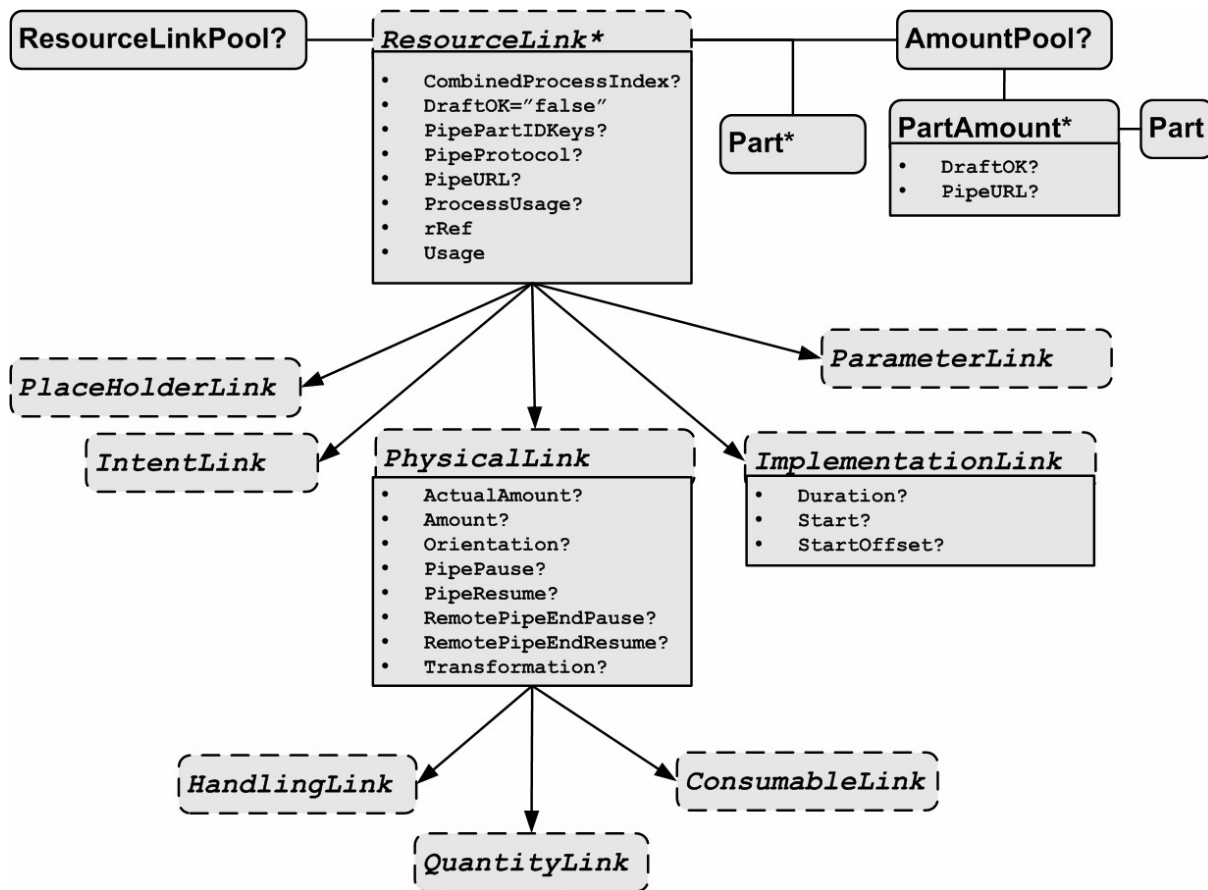


Figure 3.7: Structure of the abstract ResourceLink types

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 resource is entitled ScanParamsLink. The following seven abstract resource link classes exist:

- ConsumableLink
- HandlingLink
- ImplementationLink
- IntentLink
- ParameterLink
- PlaceholderLink
- QuantityLink

Each listed name is described in greater detail in the sections that follow. Figure Section 3.7, Structure of the abstract ResourceLink types shows the abstract resource link types derived from the ResourceLink type.

The following table lists the contents of a ResourceLinkPool element.

Table 3-18: Contents of the ResourceLinkPool element

Name	Data Type	Description
ResourceLink *	element	List of ResourceLink elements. The ResourceLink elements are abstract and are a placeholder for any resource link element.

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

Table 3-19: Contents of the abstract `ResourceLink` element

Name	Data Type	Description
<code>CombinedProcessIndex</code> ? <a href="#">New in JDF 1.1</a>	IntegerList	<code>Combined</code> and <code>ProcessGroup</code> nodes may contain resources from multiple process nodes. The <code>CombinedProcessIndex</code> attribute specifies the indices of individual processes in the <code>Types</code> attribute to which a <code>ResourceLink</code> in a <code>Combined</code> or <code>ProcessGroup</code> node belongs. Multiple entries in <code>CombinedProcessIndex</code> specify that the <code>ResourceLink</code> is used by the respective multiple processes in the <code>Combined</code> node. Must be specified when multiple resources of the same <code>Resource/@Type</code> and <code>ResourceLink/@Usage</code> are specified in one JDF node. If <code>CombinedProcessIndex</code> is not specified, even though multiple processes in the <code>Combined</code> or <code>ProcessGroup</code> node may link to the Resource, the <code>ResourceLink</code> applies to all of these processes.
<code>CombinedProcessType</code> ? <a href="#">Deprecated in JDF 1.1</a>	NMTO-KEN	<code>Combined</code> nodes contain input resources from multiple process nodes. The <code>CombinedProcessType</code> attribute specifies the name individual process to which a <code>ResourceLink</code> in a <code>Combined</code> node belongs. It must match one of the entries in the <code>Types</code> attribute of the node. It has been replaced by <code>CombinedProcessIndex</code> in JDF 1.1.
<code>DraftOK = "false"</code>	boolean	If <code>true</code> , the process may commence with a draft resource.
<code>PipePartIDKeys</code> ?	enumerations	Defines the granularity of a dynamic pipe for a partitioned resource. For instance, a resource may be partitioned by sheet, surface, and separation (resource attribute <code>PartIDKeys = "SheetName Side Separation"</code> ), but pipe requests should only be issued once per surface (resource link attribute <code>PipePartIDKeys = "SheetName Side"</code> ). The contents of <code>PipePartIDKeys</code> must be a subset of the <code>PartIDKeys</code> attribute of the resource that is linked by this <code>ResourceLink</code> . If <code>PipePartIDKeys</code> is not specified, it defaults to the implied or explicit value of <code>PipePartIDKeys</code> of the referenced resource.
<code>PipeProtocol</code> ? <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	NMTO-KEN	Defines the protocol use for pipe handling. <code>JMF</code> and <code>Internal</code> are the only non-proprietary piping protocols that are supported. Proprietary pipe protocols may be specified in addition to those defined below but will not necessarily be interoperable. Allowed values include:  <code>Internal</code> – Internal or virtual pipe used within a combined process. <a href="#">New in JDF 1.2</a> <code>JMF</code> – JMF-based <code>PipePush</code> / <code>PipePull</code> messages. <code>None</code> – No pipe support. If <code>PipeURL</code> is specified and <code>PipeProtocol</code> is not specified, <code>JMF</code> is assumed. If not specified, defaults to the value of the referenced <code>Resource/@PipeProtocol</code> .

Table 3-19: Contents of the abstract ResourceLink element

Name	Data Type	Description
<a href="#">PipeURL ?</a> <a href="#">Modified in JDF 1.2</a>	URL	Pipe request URL. Dynamic pipe requests from this end of a pipe should be made <i>to</i> this URL. <sup>a</sup> If not specified, defaults to the value of the referenced <b>Resource/@PipeURL</b> . 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 <b>PipePush</b> or <b>PipePull</b> message. For details on using <i>PipeURL</i> , see Section 4.3.3, <i>Overlapping Processing Using Pipes</i> .
<a href="#">ProcessUsage ?</a>	string	Identifies the resource usage in the process if multiple resources of the same type are required. For example, this attribute appears when two components—one <b>Cover</b> and one <b>BookBlock</b> —are used in <b>AdhesiveBinding</b> . The allowed values of <i>ProcessUsage</i> are defined in the appropriate process descriptions in Section 6, <i>Processes and Section 6.1..</i>
<a href="#">rRef</a>	IDREF	Link to the target resource.
<a href="#">rSubRef ?</a> <a href="#">Deprecated in JDF 1.2</a>	IDREF	Link to a subelement within the resource. In JDF 1.2 and beyond, resource links should only reference resources that are direct children in a <b>ResourcePool</b> .
<a href="#">Usage</a>	enumeration	Resource usage within this JDF node. Possible values are: <i>Input</i> – The resource is an input. <i>Output</i> – The resource is an output.
<a href="#">AmountPool ?</a> <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	element	Definition of partial amounts and pipe parameters for this <b>ResourceLink</b> . The allowed contents of the <b>AmountPool</b> are described for the various types of resource links in the sections below. If <i>AmountPool</i> is specified, the <b>ResourceLink</b> must not contain any of the amount related attributes defined in <b>AmountPool/PartAmount</b> .
<a href="#">Part *</a>	element	The <b>Part</b> elements identify the parts of a partitioned resource that are referenced by the <b>ResourceLink</b> . The structure of the <b>Part</b> element is defined in Table 3-28, “Contents of the Part element,” on page 79. For details on partitioned resources, see Section 3.8.2, <i>Description of Partitionable Resources</i> .

- a. Note that in most cases this is the URL of the controller of the *other end* of the pipe. This may 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.

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

Table 3-20: Contents of the AmountPool element

Name	Data Type	Description
<a href="#">PartAmount *</a> <a href="#">New in JDF 1.1</a>	element	Element that defines the amounts and pipe parameters for a partitioned resource. The contents of a <b>PartAmount</b> depends on the type of the <b>ResourceLink</b> .

The following table lists the generic contents of a **PartAmount** element. Further parameters of the **PartAmount** are described in the respective sections below (Table 3-22, “Contents of the abstract ImplementationLink or PartAmount element,” on page 66 and Table 3-23, “Additional contents of the abstract physical ResourceLink and PartAmount element,” on page 67). Note that **PartAmount** inherits values from its parent **ResourceLink**.

Table 3-21: General contents of the PartAmount element

Name	Data Type	Description
<a href="#">DraftOK ?</a> <a href="#">New in JDF 1.1</a>	boolean	If <i>true</i> , the process may commence with a draft resource partition.
<a href="#">PipeURL ?</a> <a href="#">New in JDF 1.1</a>	URL	Pipe request URL for this partition. Dynamic pipe requests from this end of a pipe should be made to this URL. <sup>a</sup> 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 <i>PipePush</i> or <i>PipePull</i> message. For details on using <i>PipeURL</i> , see Section 4.3.3, <i>Overlapping Processing Using Pipes</i> .
<a href="#">Part</a> <a href="#">New in JDF 1.1</a>	element	Specifies the selected part that the <i>PartAmount</i> is valid for. This must be a leaf partition of the resource.

- a. Note that in most cases this is the URL of the controller of the *other end* of the pipe. This may seem counterintuitive, but it allows parallel spawning and merging of processes that represent a dynamic pipe without having to include the node that describes the other end in the spawned file.

### 3.7.1 Links to Parameter Resources

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

### 3.7.2 Links to Implementation Resources

Implementation resources are linked by an instance of an *ImplementationLink* element. Since implementation *ResourceLinks* define the usage of a specific device during the course of a job, situations can arise where that resource is not required during the whole processing time. For instance, a forklift that only has to transport the completed components is not required to 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 may commence when implementation resources are still *InUse* by other processes if *Start* or *StartOffset* are specified. *ImplementationLink* elements always have a *Usage* of *Input*.

Table 3-22: Contents of the abstract ImplementationLink or PartAmount element

Name	Data Type	Description
<a href="#">Duration ?</a>	duration	Estimated duration during which the resource will be used.
<a href="#">Recommendation ?</a> <a href="#">Deprecated in JDF 1.2</a>	boolean	If <i>true</i> and the request cannot be fulfilled, the change may be logged as a Modified Audit and the job may 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.
<a href="#">Start ?</a>	dateTime	Time and date when the usage of the implementation resource starts.
<a href="#">StartOffset ?</a>	duration	Offset time when the resource is required after processing has begun. If both <i>Start</i> and <i>StartOffset</i> are specified, <i>Start</i> has precedence.

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

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

### 3.7.3 Links to Physical Resources

Just as physical resources inherit the contents of the abstract resource element, physical resource links inherit the contents of the abstract resource link element. They may, however, contain additional contents. These optional attributes are described in Table 3-23, below. The attributes in this table may occur either directly in the physical ResourceLink or in AmountPool and PartAmount elements of a resource link.

It is important to note that the order of occurrence of links to physical resources may be significant—most specifically with QuantityLinks. 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 components should occur in the new, gathered output component.

Table 3-23: Additional contents of the abstract physical ResourceLink and PartAmount element

Name	Data Type	Description
<a href="#">ActualAmount ?</a> <a href="#">New in JDF 1.2</a>	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 3.8.1, Resource Amount
<a href="#">Amount ?</a>	double	For a link with a <i>Usage</i> of "Input", specifies the amount of the resource that is required 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 3.8.1, Resource Amount). If not specified, ResourceLink/@Amount defaults to Resource/@Amount.
<a href="#">Orientation ?</a> <a href="#">New in JDF 1.1</a>	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 should manipulate the resource in such a way as to reflect the transformation. The coordinate system of the resource itself is <i>not</i> modified. Only 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.
<a href="#">PipePause ?</a>	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.
<a href="#">PipeResume ?</a>	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.
<a href="#">RemotePipeEnd-Pause ?</a>	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.
<a href="#">RemotePipeEnd-Resume ?</a>	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.
<a href="#">Transformation ?</a> <a href="#">New in JDF 1.1</a>	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 should manipulate the resource in such a way as to reflect the transformation. The coordinate system of the resource itself is <i>not</i> modified. Only 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.

The following example shows an InkLink with an AmountPool.

```

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

```

### 3.7.4 Links to Placeholder Resources

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

### 3.7.5 Links to Intent Resources

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

### 3.7.6 Inter-Resource Linking Using ResourceRef

[Modified in JDF 1.2](#)

In some cases, it is necessary to reference resource elements directly from other resources in order to reuse information. These links are abstract **ResourceRef** elements. The **ResourceRef**'s name is generated by appending the string "Ref" to the element name. Candidate elements for inter-resource linking have a data type of *relement* in the content description tables of this chapter and Section 7, Resources. The following table defines the attributes of the abstract **ResourceRef** element (see also Figure 3.4 and **ResourceElement** in Table 3-13, "Contents of the abstract Resource element," on page 53).

Table 3-24: Contents of the abstract ResourceRef element

Name	Data Type	Description
<i>rRef</i> <a href="#">Clarified in JDF 1.2</a>	IDREF	Reference to the resource. The linked resource must be a direct child of a <b>ResourcePool</b> .
<i>rSubRef?</i> <a href="#">Deprecated in JDF 1.2</a>	IDREF	Reference to a subelement of the resource. In JDF 1.2 and beyond, <b>ResourceRef</b> elements should only reference resources that are present in a <b>ResourcePool</b> .
<i>Part ?</i> <a href="#">New in JDF 1.1</a>	element	Definition of the partition that this <b>ResourceRef</b> references.



The sub element of a resource, also called **ResourceElement**, is defined in the following table: Table 3-25.

Table 3-25: Contents of the abstract *ResourceElement*

Name	Data Type	Description
<i>ID</i> ? <a href="#">Deprecated in JDF 1.2</a>	ID	Unique identifier of a resource element. In JDF 1.2 and beyond, <b>ResourceRef</b> and <b>ResourceLink</b> elements should only reference resources that are present in a <b>ResourcePool</b> . Therefore elements that are defined locally within a resource should not be referenced directly and should not contain an ID.

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

When a **ResourceRef** references a partitioned resource node that is not a resource leaf, the children of the referenced resource are ignored. Otherwise, the referenced structure would be a partitioned element and thus invalid when inlined. Thus the following example equivalence applies.

**ResourceRef** example with partition:

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

Valid inlined **ResourceRef** example with no inline partition:

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

Invalid inlined **ResourceRef** example with partition:

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

**ResourceRef** elements may also occur in the **NodeInfo** and **CustomerInfo** element of a JDF node. Resource elements that are referenced must reside in a **ResourcePool**. The restrictions on locations of resource elements described in Section 3.6.2, *Position of Resources within JDF Nodes* that apply to resource links similarly apply to referements.

### 3.7.6.1 Status of Resources That Contain rRef References

The *Status* of a resource that contains an rRef attribute is defined by the lowest *Status* of all recursively referenced resources. The ordering is defined in Table 3-13, “Contents of the abstract Resource element,” on page 53:

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 may be *Unavailable*, *Draft*, *Available*, etc.

### 3.7.6.2 Alignment of ResourceLink and ResourceRef

#### [New in JDF 1.1A](#)

**ResourceRef** elements must not contain any of the attributes and elements that may be specified in the **ResourceLink** as defined in Section 3.7, Resource Links. The value of these properties is implied from the value of the properties for the appropriate part in the **AmountPool** of the **ResourceLink**. The following example illustrates the alignment of a **MediaLink** and **MediaRef** in a **DigitalPrinting** node.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n20020626134204" Status="Waiting"
Type="DigitalPrinting" Version="1.2">
  <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>
  </ResourcePool>
  <ResourceLinkPool>
    <MediaLink Usage="Input" rRef="r0006">
      <!-- the AmountPool contains the ResourceLink partition details -->
      <AmountPool>
        <PartAmount Orientation="Flip180">
          <Part RunIndex="0 -1"/>
        </PartAmount>
        <PartAmount Orientation="Rotate0">
          <Part RunIndex="1~-2"/>
        </PartAmount>
      </AmountPool>
    </MediaLink>
    <DigitalPrintingParamsLink Usage="Input" rRef="r0007"/>
  </ResourceLinkPool>
</JDF>
```

## 3.8 Subsets of Resources

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 resource links and multiple output resource links exist that must 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.

### 3.8.1 Resource Amount

Yet another flexible feature of resources is that they may 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 resource links 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 physical resource link 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 resource links that reference them are consistent. The units used in the *Amount* attribute of a physical resource link 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 3.8.2, *Description of Partitionable 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.

#### 3.8.1.1 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 required 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 PhysicalResourceLink(*Input*)/AmountPool/PartAmount/@Amount -  
PhysicalResourceLink(*Input*)/AmountPool/PartAmount/@ActualAmount
- 5 PhysicalResourceLink(*Input*)/@Amount -  
PhysicalResourceLink(*Input*)/@ActualAmount
- 6 **PhysicalResource**(*Input*)/@Amount -  
PhysicalResourceLink(*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(Onput)/@Amount-ComponentLink(Input)/@ActualAmount`

`ComponentLinks` from intermediate nodes (`ProcessGroup` or `Product`) must be ignored when summing, since they redundantly link to the same resources without specifying an additional production amount.

### 3.8.1.2 Specifying Amount for a partially completed process

[New in JDF 1.2](#)

A process may be interrupted before the requested amount of output has been produced. When the job is resent from the controller to the Device, only the remaining *Amount* must be produced by the Device. The following figure shows the various processes, resources and `ResourceLinks` and their corresponding entries in Table 3-26 on page 73 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:

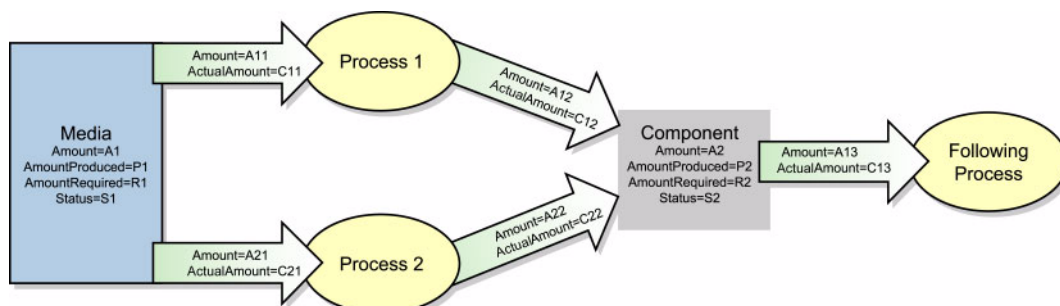


Figure 3.8: Amount Handling

Table 3-26: Example of Actual Amount and Amount Handling

Process Step	A1 P1 R1 S1	A11 C11  A21 C21	A12 C12  A22 C22	A2 P2 R2 S2	A13 C12
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	— —
<b>Consumption of the output by a subsequent process</b>					
A following process consumes 50,010 copies	390677 — 110000 Available	110000 109323 — —	100000 101234 — —	51224 101234 50000 Available	50000 50010
<b>Additional Copy Request</b>					
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

Table 3-26: Example of Actual Amount and Amount Handling

Process Step	A1 P1 R1 S1	A11 C11 A21 C21	A12 C12 A22 C22	A2 P2 R2 S2	A13 C12
<b>Parallel Production by a second device</b>					
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
<b>Consumption by the following process</b>					
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

### 3.8.2 Description of Partitionable 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 may 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 partitionable resource may contain nested elements, each with the same name as the resource. 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 may be overwritten by the equivalent attributes in descendent parts. If a leaf contains elements that may occur multiply, and additional elements with the same name exist in nodes that are closer to the root, only the elements in the leaf are valid for the respective part. For example, the following SeparationSpec is two color duo-tone (only *Black* and *SpotGreen*) in the part with *PageNumber* = "1".

```

<LayoutElement Class="Parameter" ID="ID1" PartIDKeys="PageNumber" Status="Available">
  <SeparationSpec Name="Cyan"/>
  <SeparationSpec Name="Magenta"/>
  <SeparationSpec Name="Yellow"/>
  <SeparationSpec Name="Black"/>
  <FileSpec/>
  <LayoutElement PageNumber="0"/>
  <LayoutElement PageNumber="1">
    <SeparationSpec Name="Black"/>
    <SeparationSpec Name="SpotGreen"/>
  </LayoutElement>
</LayoutElement>

```

### 3.8.2.1 Amount in Partitionable Resources

[New in JDF 1.2](#)

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

The following example defines two sets of 12 plates for two sheets with three surfaces. Each has a common brand attribute called “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, may remain *Available*.

```

<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" Side="Front">
    <ExposedMedia ProductID="S2FCPlateJ42" Separation="Cyan"/>
    <ExposedMedia ProductID="S2FMPlateJ42" Separation="Magenta"/>
    <ExposedMedia ProductID="S2FYPlateJ42" Separation="Yellow"/>
    <ExposedMedia ProductID="S2FKPlateJ42" Separation="Black"/>
  </ExposedMedia>
</ExposedMedia>

```

### 3.8.2.2 Relating PartIDKeys and Partitions

[New in JDF 1.2](#)

The *PartIDKeys* attribute describes the partition keys that may 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.

### 3.8.2.2.1 Incomplete Partitions

#### [New in JDF 1.2](#)

Partitioned resources may be partitioned by a restricted subset of keys in the *PartIDKeys* list. Keys from the back of the list may be omitted in individual partitions. If a key is omitted all following keys must also be omitted. The following example demonstrates a legal incomplete partition:

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

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

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

### 3.8.2.2.2 Multiple Keys per Partitioned Leaf or Node

#### [New in JDF 1.2](#)

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

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

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

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

### 3.8.2.2.3 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. The following example specifies that only "c1" must be folded:

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

The following example is NOT valid:

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



### 3.8.2.3 Partitioning of Resource sub-Elements

#### [New in JDF 1.2](#)

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

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

In this valid but incomplete example #2, the **Media** in the leaves is not complete because it does not contain the **MediaType** attribute. **MediaType** is *not* be inherited from the **Media** element in the root resource.

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

In this invalid example #3, **Media** is a subelement that must *not* be partitioned.

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

Partitioning may be combined with inter-resource links, (i.e., **RefElements**.) In the following valid example #4, each **MediaRef** is equivalent to an in-lined leaf with the explicit **Part** elements to define the partition, (i.e., it is equivalent to the valid example #1.)

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

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

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

### 3.8.2.4 Additional Attributes for use with partitioned Resources

[New in JDF 1.2](#)

In addition to the usual resource attributes and elements, the partitionable **Resource** element has partition-specific attributes and elements in its root. Specifying *PartIDKeys* in the root defines a partitioned resource. Further attributes are listed in the following table.

Table 3-27: Contents of the Partitionable Resource Element

Name	Data Type	Description																																				
<a href="#">PartIDKeys ?</a> <a href="#">Modified in JDF 1.2</a>	enumerations	<p>List of attribute names that are used to separate the individual parts. <i>PartIDKeys</i> also defines the sequence from root to leaf in which the <i>PartIDKeys</i> must occur in the partitioned resource. Each entry in the <i>PartIDKeys</i> list must occur only once. <i>PartIDKeys</i> must not be specified below the root of a partitioned resource. <b>Note:</b> <i>PartIDKeys</i> enumerations are often referred to as “partition keys” or “part keys” throughout this document and in common practice and discussions. Possible values are:</p> <table border="1"> <tr> <td><i>Bindery-</i></td> <td><i>LayerIDs</i></td> <td><i>SectionIndex</i></td> </tr> <tr> <td><i>SignatureName</i></td> <td><i>Location</i></td> <td><i>Separation</i></td> </tr> <tr> <td><i>BlockName</i></td> <td><i>Option</i></td> <td><i>SetDocIndex</i></td> </tr> <tr> <td><i>BundleItemIndex</i></td> <td><i>PageNumber</i></td> <td><i>SetIndex</i></td> </tr> <tr> <td><i>CellIndex</i></td> <td><i>PartVersion</i></td> <td><i>SetRunIndex</i></td> </tr> <tr> <td><i>Condition</i></td> <td><i>PreflightRule</i></td> <td><i>SetSheetIndex</i></td> </tr> <tr> <td><i>DocCopies</i></td> <td><i>PreviewType</i></td> <td><i>SheetIndex</i></td> </tr> <tr> <td><i>DocIndex</i></td> <td><i>RibbonName</i></td> <td><i>SheetName</i></td> </tr> <tr> <td><i>DocRunIndex</i></td> <td><i>Run</i></td> <td><i>Side</i></td> </tr> <tr> <td><i>DocSheetIndex</i></td> <td><i>RunIndex</i></td> <td><i>SignatureName</i></td> </tr> <tr> <td><i>FountainNumber</i></td> <td><i>RunTags</i></td> <td><i>TileID</i></td> </tr> <tr> <td><i>ItemNames</i></td> <td><i>RunPage</i></td> <td><i>WebName</i></td> </tr> </table> <p>For details, see Table 3-28, “Contents of the Part element,” on page 79. Note that <i>Part/@Sorting</i> and <i>Part/@SortAmount</i> are not valid entries in <i>PartIDKeys</i>, although they are valid <i>Part</i> attributes.</p>	<i>Bindery-</i>	<i>LayerIDs</i>	<i>SectionIndex</i>	<i>SignatureName</i>	<i>Location</i>	<i>Separation</i>	<i>BlockName</i>	<i>Option</i>	<i>SetDocIndex</i>	<i>BundleItemIndex</i>	<i>PageNumber</i>	<i>SetIndex</i>	<i>CellIndex</i>	<i>PartVersion</i>	<i>SetRunIndex</i>	<i>Condition</i>	<i>PreflightRule</i>	<i>SetSheetIndex</i>	<i>DocCopies</i>	<i>PreviewType</i>	<i>SheetIndex</i>	<i>DocIndex</i>	<i>RibbonName</i>	<i>SheetName</i>	<i>DocRunIndex</i>	<i>Run</i>	<i>Side</i>	<i>DocSheetIndex</i>	<i>RunIndex</i>	<i>SignatureName</i>	<i>FountainNumber</i>	<i>RunTags</i>	<i>TileID</i>	<i>ItemNames</i>	<i>RunPage</i>	<i>WebName</i>
<i>Bindery-</i>	<i>LayerIDs</i>	<i>SectionIndex</i>																																				
<i>SignatureName</i>	<i>Location</i>	<i>Separation</i>																																				
<i>BlockName</i>	<i>Option</i>	<i>SetDocIndex</i>																																				
<i>BundleItemIndex</i>	<i>PageNumber</i>	<i>SetIndex</i>																																				
<i>CellIndex</i>	<i>PartVersion</i>	<i>SetRunIndex</i>																																				
<i>Condition</i>	<i>PreflightRule</i>	<i>SetSheetIndex</i>																																				
<i>DocCopies</i>	<i>PreviewType</i>	<i>SheetIndex</i>																																				
<i>DocIndex</i>	<i>RibbonName</i>	<i>SheetName</i>																																				
<i>DocRunIndex</i>	<i>Run</i>	<i>Side</i>																																				
<i>DocSheetIndex</i>	<i>RunIndex</i>	<i>SignatureName</i>																																				
<i>FountainNumber</i>	<i>RunTags</i>	<i>TileID</i>																																				
<i>ItemNames</i>	<i>RunPage</i>	<i>WebName</i>																																				
<a href="#">PipePartIDKeys ?</a> <a href="#">New in JDF 1.2</a>	enumerations	<p>Defines the granularity of a dynamic pipe for a partitioned resource. For instance, a resource may be partitioned by sheet, surface and separation (resource attribute <i>PartIDKeys</i> = “<i>SheetName Side Separation</i>”), but pipe requests should only be issued once per surface (resource link attribute <i>PipePartIDKeys</i> = “<i>SheetName Side</i>”). The contents of <i>PipePartIDKeys</i> must be a subset of the <i>PartIDKeys</i> attribute of the resource that is linked by this <i>ResourceLink</i>. If <i>PipePartIDKeys</i> is not specified, it defaults to <i>PartIDKeys</i>, (i.e. maximum granularity.) For details on partitioned resources, see “Description of Partitionable Resources” on page 74.</p>																																				
<b>Resource *</b>	element	<p>Nested resource elements that contain the appropriate part ID(s). These elements must be of the same name and type as the root <b>Resource</b> element. They represent the individual parts or groups of parts.</p>																																				

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

The following table lists the content of a **Part** element, which contains a set of attributes that have a well described meaning. Each of the attributes, except *Sorting*, may be used in the nested resource elements of partitionable resources as the part ID 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 only lower level partition keys and thus implicitly select multiple partitions leaves or nodes from a partitioned resource. If multiple Part elements are defined, the result is a Boolean OR of the multiple parts.

Table 3-28: Contents of the Part element

Name	Data Type	Description
<i>BinderySignatureName</i> ? <a href="#">New in JDF 1.2</a>	NMTOKEN	Name of the BinderySignature used in a <b>StrippingParams</b> description.
<i>BlockName</i> ? <a href="#">New in JDF 1.1</a>	NMTOKEN	Identifies a CutBlock from a <b>Cutting</b> process. The value of this attribute must match the value of the <i>BlockName</i> attribute of a CutBlock.
<i>BundleItemIndex</i> ? <a href="#">New in JDF 1.2</a>	IntegerRange-List	The BundleItemIndex attribute selects a set of BundleItems from a <b>Component</b> resource.
<i>CellIndex</i> ? <a href="#">New in JDF 1.2</a>	IntegerRange-List	Index of SignatureCells in a <b>StrippingParams</b> or <b>BinderySignature</b> .
<i>Condition</i> ? <a href="#">New in JDF 1.2</a>	NMTOKEN	<p>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 of <i>Condition</i> may include:</p> <p><i>Good</i> – All correct components.</p> <p><i>Waste</i> – General waste.</p> <p><i>Overrun</i> – Excess <b>Component</b>(s) that were produced by running the device after the specified amount has been produces.</p> <p><i>xxxGood</i> – Like <i>Good</i> above, but where “xxx” can be the name of any JDF process, (e.g., “<i>FeedingGood</i>”, “<i>TrimmingGood</i>”, etc.). In the case of a combined process or process group the name of the last JDF process in the process chain is used.</p> <p><i>xxxWaste</i> – Like <i>Waste</i> above, but where “xxx” can be the name of any JDF process, (e.g., “<i>FeedingWaste</i>”, “<i>TrimmingWaste</i>”, etc.). In the case of a combined process or process group the name of the last JDF process in the process chain is used.</p> <p><i>BindingQualityTestFailed</i> – Failed binding quality test. <b>Component</b>(s) with this <i>Condition</i> belong to the batch of <b>Component</b>(s) that did not pass the test.</p> <p><i>BindingQualityTestPassed</i> – Passed binding quality test. <b>Component</b>(s) with this <i>Condition</i> belong to the batch of <b>Component</b>(s) that passed the test but were not destroyed in the process.</p> <p><i>BindingQualityTestWaste</i> – Passed binding quality test. <b>Component</b>(s) with this <i>Condition</i> belong to the batch of <b>Component</b>(s) that passed the test but were destroyed in the process.</p> <p><i>CaliperWaste</i> – Waste by caliper on gathering / collecting.</p> <p><i>DoubleFeedWaste</i> – Waste by DoubleFeed on feeders.</p> <p><i>IncorrectComponentWaste</i> – Waste by the attempted use of an incorrect components, (for example on a feeder.)</p> <p><i>BadFeedWaste</i> – Waste caused by a bad feed.</p> <p><i>ObliqueSheetWaste</i> – Waste by oblique sheets on gathering / collecting chains.</p> <p><i>PaperJamWaste</i> – Waste by paper or other media jam.</p> <p><i>WhitePaperWaste</i> – White paper waste.</p>

Table 3-28: Contents of the Part element

Name	Data Type	Description
<i>DocCopies</i> ?	IntegerRange-List	Identifies a set of document copies to which the partition applies. <i>DocCopies</i> is a logical reference that may be independent of the <b>RunList</b> structure and must not be used as an explicit partition key for <b>RunList</b> resources.
<i>DocIndex</i> ?	IntegerRange-List	The <i>DocIndex</i> attribute selects a set of logical instance documents from a <b>RunList</b> resource. <i>DocIndex</i> is a logical reference that may be independent of the <b>RunList</b> structure and must not be used as an explicit partition key for <b>RunList</b> resources.
<i>DocRunIndex</i> ?	IntegerRange-List	The <i>DocRunIndex</i> attribute selects a set of logical pages from instance documents of a <b>RunList</b> 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 may be independent of the <b>RunList</b> structure and must not be used as an explicit partition key for <b>RunList</b> resources. The index always refers to entries of the entire <b>RunList</b> and must not be modified if only a part of the <b>RunList</b> is spawned. Specifying <i>DocRunIndex</i> does not modify the index of a <b>RunList</b> entry and therefore does not reposition pages on a <b>Layout</b> .
<i>DocSheetIndex</i> ?	IntegerRange-List	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 may be independent of the <b>RunList</b> structure and must not be used as an explicit partition key for <b>RunList</b> resources. The index always refers to entries of the entire <b>RunList</b> and must not be modified if only a part of the <b>RunList</b> is spawned. Specifying <i>DocSheetIndex</i> does not modify the index of a <b>RunList</b> entry and therefore does not reposition pages on a <b>Layout</b> .
<i>FountainNumber</i> ?	integer	Zero-based position index of the fountain. Used to partition fountains along the axis of a roller; may be used for web printing.
<i>ItemNames</i> ? <a href="#">New in JDF 1.2</a>	NMTOKENS	List of items to select from a <b>Bundle</b> . If not specified, all <b>BundleItems</b> are processed.
<i>LayerIDs</i> ? <a href="#">New in JDF 1.1</a>	IntegerRange-List	The <i>LayerIDs</i> attribute selects a set layers that are defined by LayerID. If not specified, all layers are processed.
<i>Location</i> ?	string	Name of the location, (e.g., in MIS). This part key allows to describe distributed resources. Note that this name does not define the location by itself. See Section 3.8.2.6, Locations of Physical Resources for details on specifying locations.
<i>Option</i> ?	string	Option of an RFQ. Used mainly in Intent resources.
<i>PageNumber</i> ?	IntegerRange-List	Page number in a <b>Component</b> or document, (e.g., <b>FileSpec</b> that is not described as a <b>RunList</b> ).
<i>PartVersion</i> ?	string	Version identifier, (e.g., the language version of a catalog).
<i>PreflightRule</i> ? <a href="#">New in JDF 1.2</a>	string	Definition of the specific parts of a <b>PreflightReportRulePool</b> / <b>PRRule</b> used in preflight applications.

Table 3-28: Contents of the Part element

Name	Data Type	Description
<a href="#">PreviewType ?</a> <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	enumeration	<p>Type of the preview. Possible 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.<a href="#">New in JDF 1.2</a></p> <p><i>ThumbNail</i> – Very low resolution RGB preview.</p> <p><i>Viewable</i> – RGB preview in medium resolution.</p> <p>If both <i>PreviewType</i> and <b>Preview/@PreviewUsage</b> or <b>PreviewGenerationParams/@PreviewUsage</b> are specified, they must match.</p>
<a href="#">RibbonName ?</a>	string	A string that uniquely identifies each ribbon. Multiple ribbons are created out of one web after dividing in case of web printing.
<a href="#">Run ?</a>	string	The <i>Run</i> attribute selects an individual <b>RunList</b> partition from a <b>RunList</b> resource.
<a href="#">RunIndex ?</a>	IntegerRange-List	The <i>RunIndex</i> attribute selects a set of logical pages from a <b>RunList</b> resource in a manner that is independent from the internal structure of the <b>RunList</b> . It contains an array of mixed ranges and individual indices separated by whitespace. Each range consists of two indices connected with a tilde (~) and no whitespace. For example, <i>RunIndex</i> = "2~5 8 10 22~-1". Negative numbers reference pages from the back of a file in base-1 counting. In other words, -1 is the last page, -2 the second to last, etc. Thus <i>RunIndex</i> = "0~-1" refers to a complete range of pages, from first to last. <i>RunIndex</i> is a logical reference that is independent of the <b>RunList</b> structure and must not be used as an explicit partition key for <b>RunList</b> resources. The index always refers to entries of the entire <b>RunList</b> and must not be modified if only a part of the <b>RunList</b> is spawned. Specifying <i>RunIndex</i> does not modify the index of a <b>RunList</b> entry and therefore does not reposition pages on a <b>Layout</b> .
<a href="#">RunTags ?</a> <a href="#">New in JDF 1.1</a>	NMTOKENS	List of names in a named <b>RunList</b> . Used to partition resources that are linked from processes that also have a <b>RunList</b> as input when the sequence of the <b>RunList</b> is undefined. The partition is selected if the explicit or implied (e.g., from the PDL) value of <i>RunTag</i> of the <b>RunList</b> matches any of the entries in <i>RunTags</i> .
<a href="#">RunPage ?</a> <a href="#">New in JDF 1.1</a> <a href="#">Clarified in JDF 1.2</a>	integer	<p>Zero-based page number. Used when a document/file-based <b>RunList</b> is broken down into a page based <b>RunList</b>. For instance, a 2-page document <b>RunList</b>:</p> <pre>&lt;RunList URL="doc.pdf" (...)/&gt;</pre> <p>is split into:</p> <pre>&lt;RunList PartIDKeys="RunPage" (...)&gt;   &lt;RunList URL="doc_page0.pdf" RunPage="0" (...)/&gt;   &lt;RunList URL="doc_page1.pdf" RunPage="1" (...)/&gt; &lt;/RunList&gt;</pre>
<a href="#">SectionIndex ?</a> <a href="#">New in JDF 1.2</a>	IntegerRange-List	List of sections in a <b>StrippingParams</b> .

Table 3-28: Contents of the Part element

Name	Data Type	Description
<i>Separation</i> ?	string	<p>Identifies the separation name. Possible values include:</p> <ul style="list-style-type: none"> <li><i>Composite</i> – Non-separated resource.</li> <li><i>Separated</i> – The resource is separated, but the separation definition is handled internally by the resource, such as a PDF file that contains <i>SeparationInfo</i> dictionaries.</li> <li><i>Cyan</i> – Process color.</li> <li><i>Magenta</i> – Process color.</li> <li><i>Yellow</i> – Process color.</li> <li><i>Black</i> – Process color.</li> <li><i>Red</i> – Additional process color.</li> <li><i>Green</i> – Additional process color.</li> <li><i>Blue</i> – Additional process color.</li> <li><i>Orange</i> – Additional process color.</li> <li><i>Spot</i> – Generic spot color. Used when the exact nature of the spot color is unknown.</li> <li><i>Varnish</i> – Varnish.</li> </ul> <p>Other values may be any separation name defined in the <i>Name</i> attribute of a <b>Color</b> element in the <b>ColorPool</b>.</p> <p>When <i>Separation</i> is applied to a <b>ColorantControlLink</b>, it defines an implicit partition that selects a subset of separations for the process that is described by the <b>ColorantControl</b>. For details, see “ColorantControl” on page 303.</p>
<i>SetDocIndex</i> ? <a href="#">New in JDF 1.2</a>	IntegerRange-List	<p>The <i>SetDocIndex</i> attribute selects a set of logical instance documents from instance document sets of a <b>RunList</b> 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 may be independent of the <b>RunList</b> structure and must not be used as an explicit partition key for <b>RunList</b> resources. The index always refers to entries of the entire <b>RunList</b> and must not be modified if only a part of the <b>RunList</b> is spawned. Specifying <i>SetDocIndex</i> does not modify the index of a <b>RunList</b> entry and therefore does not reposition pages on a <b>Layout</b>.</p>
<i>SetIndex</i> ? <a href="#">New in JDF 1.1</a>	IntegerRange-List	<p>The <i>SetIndex</i> attribute selects a set of logical instance document sets from a <b>RunList</b> resource. <i>SetIndex</i> is a logical reference that may be independent of the <b>RunList</b> structure and must not be used as an explicit partition key for <b>RunList</b> resources. The index always refers to entries of the entire <b>RunList</b> and must not be modified if only a part of the <b>RunList</b> is spawned. Specifying <i>SetIndex</i> does not modify the index of a <b>RunList</b> entry and therefore does not reposition pages on a <b>Layout</b>.</p>
<i>SetRunIndex</i> ? <a href="#">New in JDF 1.2</a>	IntegerRange-List	<p>The <i>SetRunIndex</i> attribute selects a set of logical pages from instance document sets of a <b>RunList</b> 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 may be independent of the <b>RunList</b> structure and must not be used as an explicit partition key for <b>RunList</b> resources. The index always refers to entries of the entire <b>RunList</b> and must not be modified if only a part of the <b>RunList</b> is spawned. Specifying <i>SetRunIndex</i> does not modify the index of a <b>RunList</b> entry and therefore does not reposition pages on a <b>Layout</b>.</p>

Table 3-28: Contents of the Part element

Name	Data Type	Description
<a href="#"><i>SetSheetIndex</i> ?</a> <a href="#">New in JDF 1.2</a>	IntegerRange-List	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 may be independent of the <b>RunList</b> structure and must not be used as an explicit partition key for <b>RunList</b> resources. The index always refers to entries of the entire <b>RunList</b> and must not be modified if only a part of the <b>RunList</b> is spawned. Specifying <i>SetSheetIndex</i> does not modify the index of a <b>RunList</b> entry and therefore does not reposition pages on a <b>Layout</b> .
<i>SheetIndex</i> ?	IntegerRange-List	The <i>SheetIndex</i> attribute selects a set of logical sheets from a <b>RunList</b> resource. In 1-up simplex printing, it is identical to <i>RunIndex</i> . <i>SheetIndex</i> is a logical reference that is independent of the <b>RunList</b> structure and must not be used as an explicit partition key for <b>RunList</b> resources.
<i>SheetName</i> ?	string	A string that uniquely identifies each sheet. The value of this attribute must match the value of the <i>Name</i> attribute of a <b>Sheet</b> . (See "Sheet" on page 469.)
<i>Side</i> ?	enumeration	Denotes the side of the sheet. Possible values are: <i>Front</i> <i>Back</i> If <i>Side</i> is specified, the <b>Part</b> element refers to one surface of the sheet. If it is not specified, it refers to both sides. In case of web printing, <i>Front</i> is a synonym for the upper side and <i>Back</i> for the down side of the web.
<i>SignatureName</i> ?	string	A string that uniquely identifies the signature within the partitionable resource. The value of this attribute must match the value of the <i>Name</i> attribute of a <b>Signature</b> . (See <b>Layout/Signature/@Name</b> in "Layout" on page 405.)
<i>Sorting</i> ?	IntegerRange-List	Mapping from the implied partitionable resource order to a process order. The indices refer to the elements of the complete partitionable resource, not to the index in the selection of parts defined by the <b>Part</b> element. <sup>a</sup> If not specified the part order is the same as the sorting order. <i>Sorting</i> must not be used as a partition key.
<i>SortAmount</i> ?	boolean	If a sorted resource has an <i>Amount</i> attribute and <i>SortAmount</i> = <i>true</i> , each resource must be processed completely. If <i>SortAmount</i> = <i>false</i> (the default), each <b>Part</b> element must be processed the number of times specified in the <i>Amount</i> attribute before starting the next <b>Part</b> . <i>SortAmount</i> must not be used as a partition key.
<i>TileID</i> ?	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. <b>Tile</b> resources are described in detail in the Section 7.2.161, <b>Tile</b> . May also be used to identify multiple plates per cylinder. Then the x-index corresponds to a zero-based position index along the axis of a roller and the y-value to a zero-based position index along the circumference of a roller.
<i>WebName</i> ?	string	A string that uniquely identifies each web.

- a. Note that *Sorting* and *SortAmount* are semantically different from the other attributes in this table as they define the ordering of parts, whereas the other attributes define the selection of parts.

### 3.8.2.5 Options in Intent Resources

JDF defines *Option* as a part 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.

### 3.8.2.6 Locations of Physical Resources

Unlike other kinds of resources, physical resources may be stored at multiple, distributed locations. This is specified by including a *Location* element in the resource element. A *Location* partition key is provided to define multiple locations of one resource. The partition key carries no semantic meaning and does not by itself define the name of a location. The following example describes a set of plates that are distributed over two locations. (Note: See “Input Tray and Output Bin Names” on page 633 for additional detail on locating physical resources.)

```
<ResourcePool>
  <ExposedMedia Class="Handling" ID="L1" PartIDKeys="Location" Status="Available">
    <ExposedMedia Amount="42" Location="dd1">
      <Location LocID="PP_01234" LocationName="Desk Drawer 1">
        <Address/>
      </Location>
    </ExposedMedia>
    <ExposedMedia Amount="100" Location="dd2">
      <Location LocID="PP_01235" LocationName="Desk Drawer 2">
        <Address/>
      </Location>
    </ExposedMedia>
  </ExposedMedia>
  <Media/>
</ResourcePool>

<ResourceLinkPool>
  <ExposedMediaLink Amount="50" Usage="Input" rRef="L1">
    <Part Location="dd2"/>
    <!-- Note that @Location may but is not required to match Location/@LocationName -->
  </ExposedMediaLink>
</ResourceLinkPool>
```

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

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



### 3.8.3 Linking to Subsets of Resources

An agent can link to a subset of a resource by including a set of **Part** elements in a **ResourceLink** element in order to define a specific subset of a resource. For details of the **Part** element, please refer to Table 3-28, “Contents of the Part element,” on page 79.

Partitionable hierarchies define an implied ordering of the individual parts. In the example in Section 3.8.2, Description of Partitionable Resources, the first element has a *ProductID* = *S1FCPlateJ42* and the last has a *ProductID* = *S2FKPlateJ42*. If process ordering of a partitionable resource is important, the **Part** element of the **ResourceLink** must specify a *Sorting* attribute. If *Sorting* is not specified, process ordering is arbitrary. If *Sorting* is specified multiple times, the resolution of the sorting must be unambiguous.

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

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

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

#### 3.8.3.1 Handling Amount in a ResourceLink to a Partitioned Resource

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

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

#### 3.8.3.2 Implicit and Explicit PartUsage in Partitioned Resources

[New in JDF 1.2](#)

The *PartUsage* attribute defines how over-specialized **ResourceLinks** are resolved. If *PartUsage* = “*Explicit*”, **ResourceLinks** that do not point to an explicitly defined partition of a resource are an error. If *PartUsage* = “*Implicit*”, **ResourceLinks** that do not point to an explicitly defined partition of a resource refer to the closest matching resource partition.

```
<ResourceLinkPool>
  <ExposedMediaLink Usage="Input" rRef="XM_ID">
    <Part Separation="z" SheetName="x" Side="Front"/>
  </ExposedMediaLink>
```

```

</ResourceLinkPool>
<ResourcePool>
  <ExposedMedia Brand="Goocy" Class="Handling" ID="XM_ID" PartIDKeys=" SheetName Side
Separation" 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>
      <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>
    </ExposedMedia>
  <ExposedMedia ProductID="S2F" 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>
</ResourcePool>

```

The following table shows the *ProductID* of the Resource Partition that is selected for various values of *SheetName*, *Side*, and *Separation* for *PartUsage* = *Implicit* and *Explicit* respectively.

Table 3-29: *PartUsage* example uses

SheetName	Side	Separation	Implicit	Explicit
—	—	—	<i>Root</i>	<i>Root</i>
<i>S1</i>	—	—	<i>S1</i>	<i>S1</i>
<i>S2</i>	—	—	<i>S2</i>	<i>S2</i>
<i>S3</i>	—	—	<i>Root</i>	—
<i>S2</i>	<i>Back</i>	<i>Cyan</i>	<i>S2</i>	—
<i>S1</i>	<i>Back</i>	<i>Cyan</i>	<i>S1BC</i>	<i>S1BC</i>
<i>S1</i>	<i>Back</i>	<i>Orange</i>	<i>S1B</i>	—
<i>S1</i>	—	<i>Cyan</i>	<i>S1BC, S1FC</i>	<i>S1BC, S1FC</i>

### 3.8.3.3 Referencing Partitioned Resources from Nodes That Allow Multiple ResourceLinks

Some processes (e.g., *Collecting*, *Gathering*) allow multiple input resources of the same type. These multiple input resources may be represented by multiple individual resources or by partitioned resources or by a mixture of both. If ordering is significant, the order of the leaves in a partitioned resource defines said ordering. The following examples of gathering three input sheets are equivalent.

### Explicit reference of ordered partitioned resources

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Link0037" Status="Waiting"
Type="Gathering" Version="1.2">
  <ResourcePool>
    <GatheringParams Class="Parameter" ID="Gath01" Locked="false" Status="Available"/>
    <Component Class="Quantity" ComponentType="Sheet" DescriptiveName="printed insert
sheets" ID="Sheets01" PartIDKeys="SheetName" Status="Available">
      <Component SheetName="Sheet1"/>
      <Component SheetName="Sheet2"/>
      <Component SheetName="Sheet3"/>
    </Component>
  </ResourcePool>
  <ResourceLinkPool>
    <GatheringParamsLink Usage="Input" rRef="Gath01"/>
    <!--three ComponentLink explicitly reference individual parts -->
    <ComponentLink Usage="Input" rRef="Sheets01">
      <Part SheetName="Sheet1"/>
    </ComponentLink>
    <ComponentLink Usage="Input" rRef="Sheets01">
      <Part SheetName="Sheet2"/>
    </ComponentLink>
    <ComponentLink Usage="Input" rRef="Sheets01">
      <Part SheetName="Sheet3"/>
    </ComponentLink>
  </ResourceLinkPool>
</JDF>
```

### Implicit reference of ordered partitioned resources

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Link0037" Status="Waiting"
Type="Gathering" Version="1.2">
  <ResourcePool>
    <GatheringParams Class="Parameter" ID="Gath01" Locked="false" Status="Available"/>
    <Component Class="Quantity" ComponentType="Sheet" DescriptiveName="printed insert
sheets" ID="Sheets01" PartIDKeys="SheetName" Status="Available">
      <Component SheetName="Sheet1"/>
      <Component SheetName="Sheet2"/>
      <Component SheetName="Sheet3"/>
    </Component>
  </ResourcePool>
  <ResourceLinkPool>
    <GatheringParamsLink Usage="Input" rRef="Gath01"/>
    <!--the ComponentLink implicitly references all three parts -->
    <ComponentLink Usage="Input" rRef="Sheets01"/>
  </ResourceLinkPool>
</JDF>
```

## 3.8.4 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, each of which is illustrated in Figure 3.9, below. Both Case A and Case B in Figure 3.9 represent workflows that use the *Amount* attribute of their resource links to share resources. This method is practical when one controller controls all aspects of resource consumption or production. In Case A, the resource amount is split between subsequent processes. In Case B, 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 required. To create independent resources from one resource, the *Split* process is used, as shown in Case C (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 may therefore run in parallel. Case D 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.

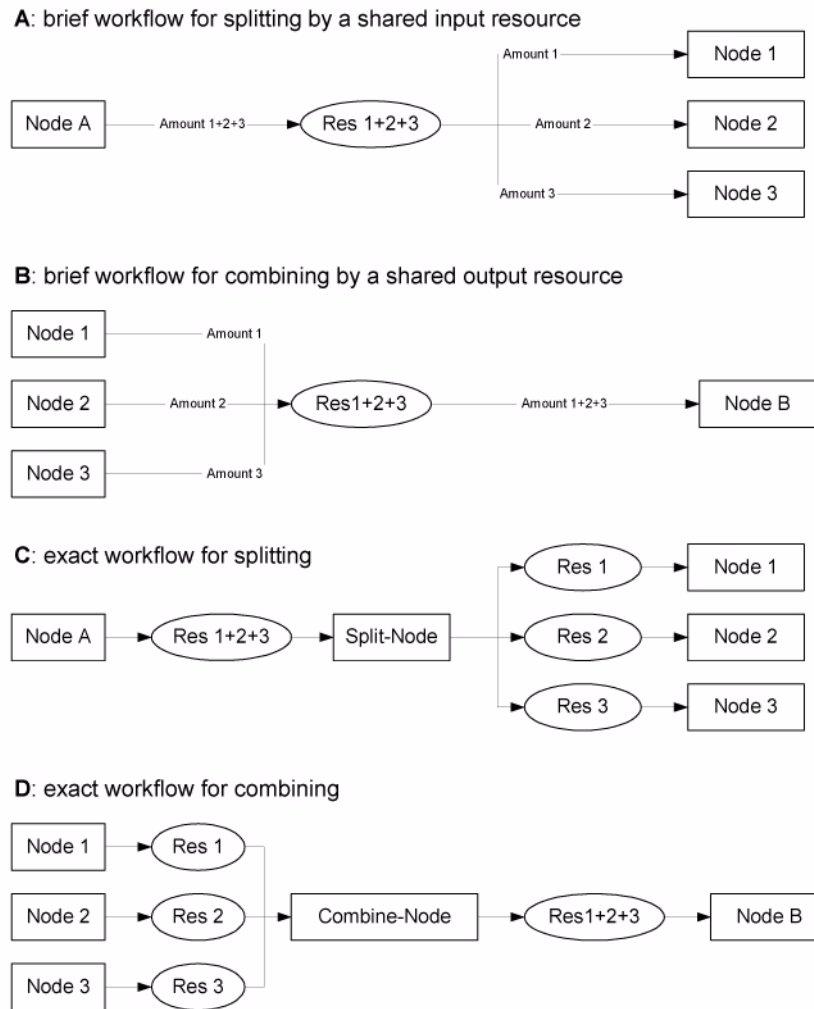


Figure 3.9: Splitting and combining physical resources

### 3.9 AuditPool

Audit elements contain the post-facto recorded results of a process such as the execution of a JDF node or modification of the JDF itself. Audit elements become static after a process has been finished. They cannot ever be modified after the process has been aborted or completed. Therefore, if 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 may 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.



#### Audit Pools

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

- 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 may be used by MIS for operations such as evaluation or invoicing. Figure 3.10 depicts the structure of the **AuditPool** and **Audit** element types derived from the abstract audit type. **Audit** entries are ordered chronologically, with the last entry in the **AuditPool** representing the newest. A **ProcessRun** element containing the scheduling data finalizes each process run. All subsequent entries belong to the next run. The following table defines the contents of the **AuditPool** element.

*Table 3-30: Contents of the AuditPool element*

Name	Data Type	Description
rRefs ? <a href="#">Deprecated in JDF 1.2</a>	IDREFS	List of all resources that are referenced from within the <b>AuditPool</b> . In JDF 1.2 and beyond, it is up to the implementation to maintain references.
Audit *	element	Chronologically ordered list of <b>Audit</b> elements. The <b>Audit</b> elements are abstract and serve as placeholders for any audit. <b>Audit</b> elements are described in the sections that follow.

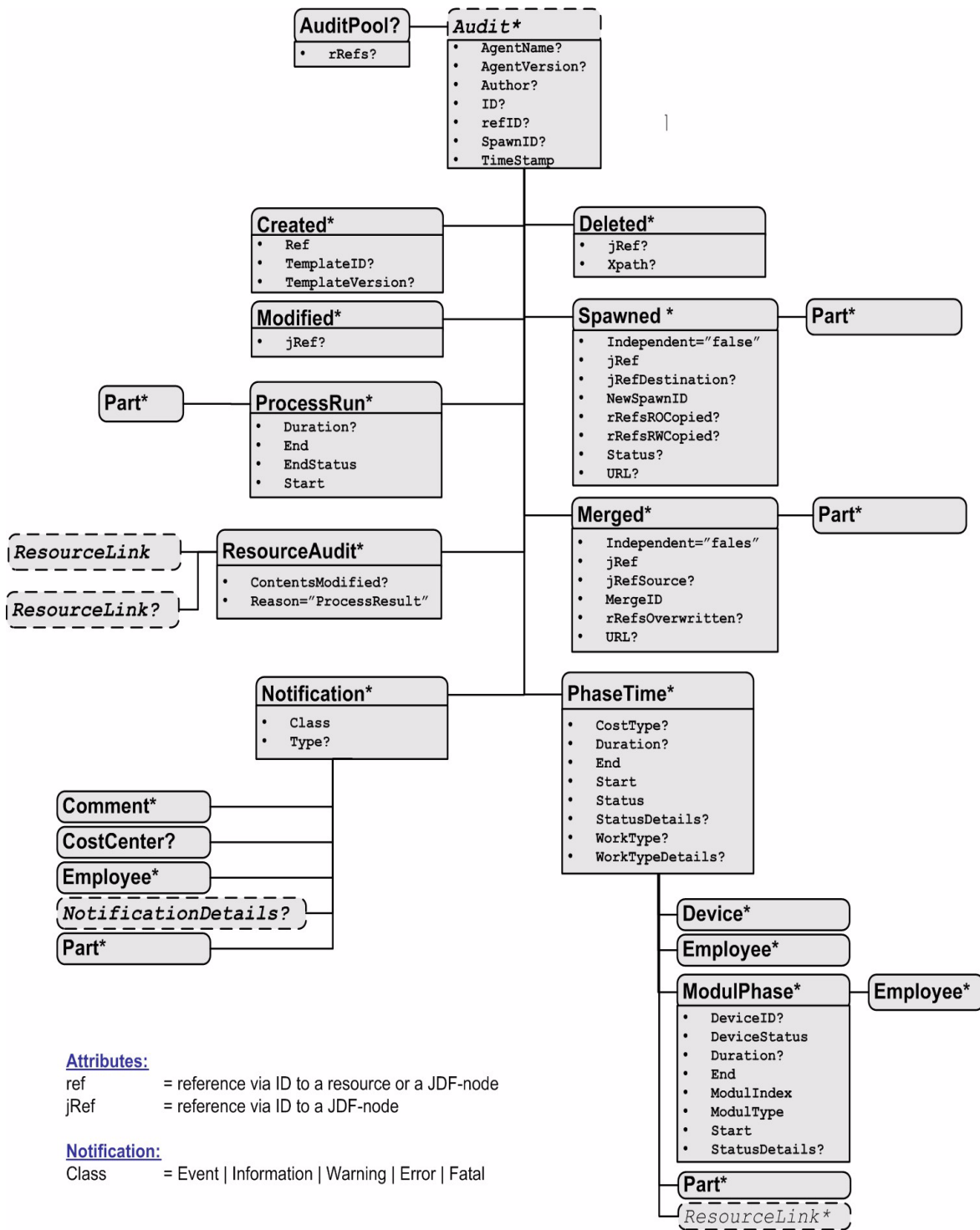


Figure 3.10: Structure of Audit element types derived from the abstract Audit type

### 3.9.1 Audit Elements

[Modified in JDF 1.2](#)

All **Audit** elements inherit the content from the abstract **Audit** data type, described in the following table.

Table 3-31: Contents of the abstract *Audit* type

Name	Data Type	Description
<i>AgentName</i> ? <a href="#">New in JDF 1.2</a>	string	The name of the agent application that added the audit element to the audit pool (and was responsible for the creation or modification). Both the company name and the product name may appear, and should be consistent between versions of the application.
<i>AgentVersion</i> ? <a href="#">New in JDF 1.2</a>	string	The version of the agent application that added the audit element to the audit pool (and was responsible for the creation or modification). The format of the version string may vary from one application to another, but should be consistent for an individual application.
<i>Author</i> ? <a href="#">Modified in JDF 1.2</a>	string	Text that identifies the person who made the entry. Prior to JDF 1.2, <i>Author</i> also contained information that is now encoded in <i>AgentName</i> and <i>AgentVersion</i> .
<i>ID</i> ? <a href="#">New in JDF 1.2</a>	ID	<i>ID</i> of the audit. <i>ID</i> must be specified to subsequently create correction <b>Audit</b> elements.
<i>refID</i> ? <a href="#">New in JDF 1.2</a>	IDREF	Reference to a previous <b>Audit</b> that this <b>Audit</b> corrects. The referenced <b>Audit</b> must reside in the same <b>AuditPool</b> .
<i>SpawnID</i> ? <a href="#">New in JDF 1.1</a>	NMTO-KEN	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 <b>Audit</b> at the time the <b>Audit</b> is generated.
<i>TimeStamp</i>	dateTime	In case of the audits <b>Created</b> , <b>Modified</b> , <b>Spawned</b> , <b>Merged</b> , and <b>Notification</b> , this attribute records the date and time when the related event occurred. In case of the audits <b>PhaseTime</b> , <b>ProcessRun</b> , and <b>ResourceAudit</b> , the attribute describes the time when the entry was appended to the audit pool.

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

#### 3.9.1.1 ProcessRun

This element serves two related functions. Its first 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, optionally, the process duration of the process run. These attributes are described in Table 3-32.

Table 3-32: Contents of the *ProcessRun* element

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.
<i>EndStatus</i>	enumeration	The <i>Status</i> of the process at the end of the run. For a description of process states, see Table 3-4, “Contents of a JDF node,” on page 38. Possible 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 may commence at a later time, (e.g., on another device).
<i>Start</i>	dateTime	Date and time at which the process started.

Table 3-32: Contents of the ProcessRun element

Name	Data Type	Description
Part * <a href="#">New in JDF 1.1</a>	element	Describes which parts of a process this ProcessRun belongs to. If Part is not specified for a ProcessRun, it refers to all parts. For example, imagine a print job that should produce three different sheets. All sheets are described by one partitioned resource. The Part elements define, unambiguously, the processing of the sheet to which the ProcessRun refers.

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

### 3.9.1.2 Notification

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

Table 3-33: Contents of the Notification element

Name	Data Type	Description
Class	enumeration	Class of the notification. Possible values, in order of severity from lowest to highest, are: <i>Event</i> – Indicates that a pure event due to any activity has occurred, (e.g., machine events, operator activities, etc.). This class is used for the transfer of conventional event messages. In case of <i>Class</i> = "Event", further event information should be provided by the <i>Type</i> attribute and <i>NotificationDetails</i> element. <i>Information</i> – Any information about a process which cannot be expressed by the other classes. No user interaction is required. <i>Warning</i> – Indicates that a minor error has occurred, and an automatic fix was applied. Execution continues. <i>Error</i> – Indicates that an error has occurred that requires user interaction. Execution cannot continue. <i>Fatal</i> – Indicates that a fatal error led to abortion of the process.
Type ?	NMTO-KEN	Identifies the type of notification. Also defines the name of the abstract <i>NotificationDetails</i> element. <sup>a</sup> A list of predefined Notification types is compiled in "NotificationDetails" on page 621.
Comment *	telem	The Notification element may contain Comment elements with 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> , it should provide at least one Comment element. In case of <i>Class</i> = "Event", Comment elements are optional.
CostCenter ?	element	The cost center to which this event should be charged.
Employee *	refelement	The employee associated with this event.
NotificationDetails ?	element	Abstract element which is a placeholder for additional structured information. It provides additional information beyond the <i>Class</i> and <i>Type</i> attribute and beyond the Comment element. For a list of supported NotificationDetails elements, see "NotificationDetails" on page 621.
Part * <a href="#">New in JDF 1.1</a>	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 should produce three different sheets. All sheets are described by one partitioned resource. The Part elements define, unambiguously, the sheet to which the audit refers.

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



### 3.9.1.2.1 NotificationDetails

The abstract NotificationDetails element is a placeholder only with no additional attributes. For a list of supported NotificationDetails elements, see “NotificationDetails” on page 621.

### 3.9.1.3 PhaseTime

This element contains audit information about the start and end times of any process states and substates, denoted as phases. Phases may 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 necessity 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.

The times specified in the PhaseTime elements should not overlap with each other and should cover the complete time range defined in the ProcessRun element that identifies the end of the run.

Table 3-34: Contents of the PhaseTime element

Name	Data Type	Description
<i>End</i>	dateTime	Date and time of the end of the phase.
<i>Start</i>	dateTime	Date and time of the beginning of the phase.
<i>Status</i>	enumeration	Status of the phase. Possible values of JDF node states are: <i>TestRunInProgress</i> <i>Setup</i> <i>InProgress</i> <i>Cleanup</i> <i>Spawned</i> <i>Stopped</i> The states listed above are a subset of the possible states of a JDF node. For all possible states of a JDF node see Table 3-4, “Contents of a JDF node,” on page 38. 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. For a list of supported values, see “StatusDetails Supported Strings” on page 615.
<i>Device</i> *	refelement	Links to <i>Device</i> resources that are working during this phase.
<i>Employee</i> *	refelement	Links to <i>Employee</i> resources that are working during this phase.
<b>MISDetails</b> ? <a href="#">New in JDF 1.2</a>	refelement	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 should 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 audit refers.
<i>ResourceLink</i> * <a href="#">New in JDF 1.1</a>	element	These resource links 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.

It is possible to monitor the states of individual modules of a complex device, such as a printer 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. In contrast to **PhaseTime** audit elements, **ModulePhase** elements are allowed to overlap in time with one another. **ModulePhase** elements are defined in the following table.

Table 3-35: Contents of the *ModulePhase* element

Name	Data Type	Description
<i>DeviceID</i>	string	Name of the device. This must be the <i>DeviceID</i> attribute of one of the <b>Device</b> elements specified in the <b>PhaseTime</b> audit.
<i>DeviceStatus</i>	enumeration	Status of the device module. Possible values are: <i>Unknown</i> – The module status is unknown. <i>Idle</i> – The module is not used, (e.g., a color print module that is inactive during a black-and-white print). <i>Down</i> – The module cannot be used. It may 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 may be resumed later. This status may indicate any kind of break, including a pause, maintenance, or a breakdown, as long as running can be easily resumed. These states are analog to the device states of Table 5-61, “Contents of the <i>ModuleStatus</i> element,” on page 167.
<i>End</i>	dateTime	Date and time of the end of the module phase.
<i>ModuleIndex</i> <a href="#">Modified in JDF 1.2</a>	IntegerRange-List	0-based indices of the module or modules. The list is based on all modules of the Device. If multiple module types are available on one device, each must be unique in the scope of the device.
<i>ModuleType</i>	NMTOKEN	Module description. The allowed values depend on the type of device that is described. The predefined values are listed in “ <i>ModuleType Supported Strings</i> ” on page 617.
<i>Start</i>	dateTime	Date and time of the beginning of the module phase.
<i>StatusDetails</i> ?	string	Description of the module status phase that provides details beyond the enumerative values given by the <i>DeviceStatus</i> attribute. For a list of supported values, see “ <i>StatusDetails Supported Strings</i> ” on page 615.
<b>Employee</b> *	refelement	References to <b>Employee</b> 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> ).

### 3.9.1.4 ResourceAudit

The **ResourceAudit** element describes the usage of resources during execution of a node or the modification of the intended usage of a resource, (i.e., the modification of a resource link.) 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 resource link, 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 used to indicate that a change has been made.

Table 3-36: Contents of the ResourceAudit element

Name	Data Type	Description
<i>ContentsModified</i> ?	boolean	Specifies that a modification has occurred but that the original resource has been deleted.
<i>Reason</i> = "ProcessResult" <a href="#">New in JDF 1.1</a>	enumeration	Reason for the modification. One of: <i>OperatorInput</i> – Human update that corrects inconsistencies from automated data collection. <i>PlanChange</i> – The resource was modified due to a change of plan before actual processing. <i>ProcessResult</i> – The actual consumption.
ResourceLink	element	The first resource link specifies the actual consumption/usage or production of a resource.
ResourceLink ?	element	The second optional resource link logs the modification of a resource link and the modification of the resource it refers to. It holds the planned resource link 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 3.7, Resource Links. The partitioning of resources using Part elements is defined in Section 3.8.2, Description of Partitionable Resources.

#### 3.9.1.4.1 Logging Machine Data by Using the ResourceAudit

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

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

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

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
Type="ConventionalPrinting" Version="1.2">
  <ResourceLinkPool>
    <MediaLink Amount="400" Usage="Input" rRef="RLink"/>
  </ResourceLinkPool>
  <ResourcePool>
    <Media Amount="400" ID="RLink" Weight="80"/>
  </ResourcePool>
</JDF>
```

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

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
Type="ConventionalPrinting" Version="1.2">
  <ResourceLinkPool>
    <!-- Note that ActualAmount has been added to the ResourceLink -->
    <MediaLink ActualAmount="421" Amount="400" Usage="Input" rRef="RLink"/>
  </ResourceLinkPool>
  <ResourcePool>
    <Media Amount="400" ID="RPrev" Weight="80"/>
    <!--Copy of the original resource-->
    <Media Amount="421" ID="RLink" Weight="90"/>
    <!--modified resource-->
  </ResourcePool>
  <AuditPool>
    <ResourceAudit>
      <MediaLink ActualAmount="421" Amount="400" Usage="Input" rRef="RLink"/>
      <MediaLink Amount="400" Usage="Input" rRef="RPrev"/>
    </ResourceAudit>
  </AuditPool>
</JDF>
```

### 3.9.1.4.2 Logging Changes in Product Descriptions by Using the ResourceAudit

ResourceAudit elements may also be used to store the original intent resources of a product specification in a change order or request for quote. The mechanism is the same as above. The following example shows the structure of a MediaIntent with Option partitions, where a late change of options from Option1 (80 gram paper) to Option2 (90 gram paper) is requested.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="J1" Status="Waiting"
  Type="Product" Version="1.2">
  <ResourceLinkPool>
    <MediaIntentLink Usage="Input" rRef="id">
      <Part Option="Option2"/>
    </MediaIntentLink>
  </ResourceLinkPool>
  <ResourcePool>
    <MediaIntent PartIDKeys="Option">
      <!-- the common MediaIntent resource details -->
      <MediaIntent Option="Option1">
        <Weight Preferred="80"/>
      </MediaIntent>
      <MediaIntent Option="Option2">
        <Weight Preferred="90"/>
      </MediaIntent>
    </MediaIntent>
  </ResourcePool>
  <AuditPool>
    <ResourceAudit>
      <!-- the actual MediaIntent resource link -->
      <MediaIntentLink Usage="Input" rRef="id">
        <Part Option="Option2"/>
      </MediaIntentLink>
      <!-- the original MediaIntent resource link -->
      <MediaIntentLink Usage="Input" rRef="id">
        <Part Option="Option1"/>
      </MediaIntentLink>
    </ResourceAudit>
  </AuditPool>
</JDF>
```

### 3.9.1.5 Created

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

Table 3-37: Contents of the Created element

Name	Data Type	Description
<i>ref</i> ? <a href="#">Deprecated in JDF 1.2</a>	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> ? <a href="#">New in JDF 1.2</a>	string	Defines the template JDF that was used as the template to create the node.
<i>TemplateVersion</i> ? <a href="#">New in JDF 1.2</a>	string	Defines the version of template JDF that was used as the template to create the node.
<i>XPath</i> ? <a href="#">New in JDF 1.2</a>	XPath	Location of the created elements or attributes relative to the parent JDF node of the Created audit element.

### 3.9.1.6 Deleted

[New in JDF 1.2](#)

This element allows any deletions of a JDF node or element to be logged. If the corresponding **Created** Audit 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** Audit, otherwise it should reside in an ancestor of the deleted element.

Table 3-38: Contents of the Deleted Element

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

### 3.9.1.7 Modified

This element allows any modifications affecting a JDF node (e.g., changes made to the **NodeInfo** element or **CustomerInfo** element) to be logged. Changes that can be logged by other audit element types (e.g., 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 3-39: Contents of the Modified element

Name	Data Type	Description
<i>jRef</i> ? <a href="#">Deprecated in JDF 1.2</a>	IDREF	The ID of the modified node. The modified element resides in the modified node. Defaults to the ID of the local JDF node. Replaced with <i>XPath</i> in JDF 1.2 and beyond.
<i>XPath</i> ? <a href="#">New in JDF 1.2</a>	XPath	Location of the modified elements or attributes relative to the parent JDF node of the <b>Modified</b> audit element.

### 3.9.1.8 Spawned

This element allows a job that has been spawned to be logged in the **AuditPool** of the parent node of the spawned job-part 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 3-40: Contents of the Spawned element

Name	Data Type	Description
<i>Independent</i> = "false" ?	boolean	Declares that independent jobs that have previously been merged into a big job are spawned. If it is set to <i>true</i> , the attributes <i>jRefDestination</i> , <i>rRefsROCopied</i> and <i>rRefsRWCopied</i> have no meaning and should be omitted.
<i>jRef</i>	IDREF	ID of the JDF node that has been spawned.
<i>jRefDestination</i> ?	NMTOKEN	ID of the JDF node to which the job has been spawned. <sup>a</sup> This attribute must be specified in the parent of the original node if independent jobs are spawned.
<i>NewSpawnID</i> <a href="#">New in JDF 1.1</a>	NMTOKEN	Copy of the <i>SpawnID</i> of the newly spawned node. Note that a <b>Spawned</b> audit may also contain a <i>SpawnID</i> attribute, which is the <i>SpawnID</i> of the node that this audit is being placed into prior to spawning.
<i>rRefsROCopied</i> ?	IDREFS	List of IDs separated by whitespace. Identifies the resources copied to the <b>ResourcePool</b> element of the spawned job during spawning. These resources should not be modified by the spawned job.
<i>rRefsRWCopied</i> ?	IDREFS	List of IDs separated by white spaces. Identifies the resources copied to the <b>ResourcePool</b> element of the spawned job during spawning. These resources may be modified by the spawned job 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.

Table 3-40: Contents of the Spawned element

Name	Data Type	Description
<i>Status</i> ? <a href="#">New in JDF 1.1</a>	enumeration	<i>Status</i> of the spawned node at the time of spawning. Allowed values are defined in <i>Status</i> in Table 3-4, “Contents of a JDF node,” on page 38.
<i>URL</i> ? <a href="#">New in JDF 1.1</a>	URL	Locator that specifies the location where the spawned node was stored by the spawning process.
Part *	element	Identifies the parts that were selected for spawning in case of parallel spawning of partitionable resources. See Section 3.8.2, Description of Partitionable Resources.

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

### 3.9.1.9 Merged

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

Table 3-41: Contents of the Merged element

Name	Data Type	Description
<i>Independent</i> = "false"	boolean	Declares that independent jobs are merged into a big job for common production. If it is set to <i>true</i> , the attributes <i>jRefSource</i> and <i>rRefsOverwritten</i> have no meaning and should be omitted.
<i>jRef</i>	IDREF	ID of the JDF node that has been returned or merged.
<i>jRefSource</i> ?	NMTO- KEN	ID of the JDF root node of the big job from which the spawned structure has been returned. <sup>a</sup>
<i>MergeID</i> <a href="#">New in JDF 1.1</a>	NMTO- KEN	Copy of the <i>SpawnID</i> of the merged node. Note that a <i>Merged</i> audit may also contain a <i>SpawnID</i> attribute, which is the <i>SpawnID</i> of the node that this audit is being placed into prior to 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> ? <a href="#">New in JDF 1.1</a>	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 3.8.2, Description of Partitionable Resources.

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

## 3.10 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.

### 3.10.1 Namespaces in XML

JDF Extensibility is implemented using XML Namespaces [XMLNS]. XML namespaces are defined by *xmlns* attributes. A general example is provided below. The example illustrates how private namespaces are declared and used to extend an existing JDF resource by adding private attributes and a private element.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1"
xmlns:foo="fooschema URI" ... >
. . .
<SomeJDFDefinedResource name="abc"
foo:specialname="cba">
```



#### 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 may be defined either in the default namespace or in a qualified namespace.

```

. . .
  <foo:PrivateStuff type=""/>
. . .
</SomeJDFDefinedResource>
. . .
</JDF>

```

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 standard `xmlns:xxx` attributes.

### 3.10.1.1 JDF Namespace

The official namespace URI for JDF Version 1.0 is: [http://www.CIP4.org/JDFSchema\\_1](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](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.

### 3.10.1.2 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](http://www.CIP4.org/JDFSchema_1_1_X). It is strongly recommended to use a prefix of “JDFX” as the JDF extension namespace prefix.

## 3.10.2 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 3.1.5, Combined Process Nodes) can create JDF process nodes of their own type. Then the content of the *Type* attribute may be specified with a prefix that identifies the organization. The prefix and name must be separated by a single colon (:) as shown in the following example.

```

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

```

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



### EXTENSIBILITY CAUTION

JDF “Extensibility” simply means that you can add your own XML elements, attributes, and enumerations to a JDF application. Although JDF is quite extensive, odds are you’ll find that your current databases and workflow systems use information elements that are unique to your client market or company ... *they may have even been defined by your internal MIS staff*. CIP4 acknowledges that it can’t define everything, nor should 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 should 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 may not be disclosed to third parties for integration purposes, should be viewed skeptically.

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

### 3.10.3 Extending Existing Resources

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

This does not mean that duplicate functionality may be added into these resource types. You must make sure to use the JDF-defined attributes and elements where possible and extend them with additional information that cannot be described using JDF-defined constructs. For example, it is not allowed to extend the RIP resource that controls the resolution with a `foo:Resolution` or `foo:Res` attribute that overrides the JDF defined resolution parameter (see attribute *Resolution* of resource **RenderingParams** in Section 7.2.135, `RenderingParams`).

### 3.10.4 Extending NMTOKEN Lists

[Clarified in JDF 1.2](#)

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 ":"). The namespace prefix that is used should be defined in the JDF ticket with the standard `xmlns:Prefix="URI"` notation, even if no extension elements or attributes from that namespace occur in the JDF ticket. Implementations that find an unknown NMTOKEN prefixed by a namespace prefix may then attempt to use the default value of that attribute if the value of *SettingsPolicy* in effect is *BestEffort*. For instance, if a JDF instruction contains the following text.

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

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

Table 3-42: Example from `TrappingParams`

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

### 3.10.5 Creating New Resources

There are certain process implementations that have functionality that cannot be specified by the predefined Resource types. In these cases, it is necessary to create a new Resource-type element, which must be clearly specified using its own namespace. These resource types may only be linked to custom-type JDF process nodes.

### 3.10.6 Future JDF Extensions

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



### 3.10.7 Maintaining Extensions

Given the mix of vendors that will use JDF, it is likely that there will be a number of private extensions. Therefore, JDF controllers must be prepared to receive JDF files that have extensions. These controllers can and 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 may 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.

### 3.10.8 Processing Unknown Extensions

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

If *SettingsPolicy* = *BestEffort*, a Notification audit element with *Class* = *warning* should be logged.

If *SettingsPolicy* = *MustHonor*, the process must not continue and a Notification audit element with *Class* = *error* should be logged.

If *SettingsPolicy* = *OperatorIntervention*, the process must stop and wait for an operator intervention and a Notification audit element with *Class* = *warning* should be logged.

### 3.10.9 Derivation of Types in XMLSchema

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 but not required.

## 3.11 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.

#### 3.11.1 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 will NOT always support all versions of all devices that are described in the JDF.
  - Customers may 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.

### 3.11.2 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 linked via references.

### 3.11.3 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 should also interpret these policies to apply to JMF as well. Version policies include three areas of application: JDF specification rules, JDF schema definition rules, and JDF application behavior. The policies are applicable to the transition from JDF 1.1/1.1a to JDF 1.2, as well as future versions of JDF, but are not applicable to JDF 1.0.

#### 3.11.3.1 JDF Specification Version Policies

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

- Changes to the JDF specification must be backwards compatible.
  - Extension elements or attributes must not be 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 must not be removed.
    - Deprecated elements or attributes are still valid in all versions of JDF 1.x
  - Data type changes must be extensions of existing data types. In other words the datatype of an extended attribute must be a complete superset of the existing datatype. 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 will not be altered.
  - No new attributes or elements will be introduced that conditionally modify the semantics of existing attributes and elements.
  - Semantics will 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 will be clearly marked in the specification.
- The default values of attributes and elements will not be altered.
  - The default behavior that is specified when an attribute or element is missing will not be altered.

#### 3.11.3.2 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 be backwards compatible.
  - JDF 1.x documents must validate against JDF 1.(x+n) schemas.
- Only one JDF schema namespace will be defined for all versions of JDF 1.x.
  - The namespace is [http://www.CIP4.org/JDFSchema\\_1\\_1](http://www.CIP4.org/JDFSchema_1_1).
- The *xs:version* attribute will 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 can 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" and "1.2" for the JDF 1.2 version of the schema). The schema data type of a JDF of JMF version is `JDFJMFVersion`.
  - This allow schema validators to detect incompatible versions when parsing a local legacy schema.
- The version annotations in the schema will be maintained wherever possible.
- Explicit copies of published legacy schema versions will be available on the CIP4 website.
- The schema default values of deprecated attributes will be removed from the schema. Deprecated attributes will still be valid but not explicitly defaulted in the schema.

### 3.11.3.3 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.

#### 3.11.3.3.1 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 should 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 3.11.2, JDF Version Definition.
- The `JDF/@Version` of a spawned JDF node is identical to the `JDF/@Version` of that node in a complete JDF.

#### 3.11.3.3.2 JDF Device/Controller Version Policies

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

- Implementations are strongly encouraged to 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 are strongly encouraged to 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 may 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 may 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 may create a product node that specifies the product intent of a series of pages. This product node may include information about the number of pages and the paper properties. Another node may then be inserted that includes a resource describing how the pages should be RIP'd. Later, another agent may provide more detail about the RIP'ing process by appending optional information to the RIP parameter resource.

Regardless of where in the life cycle they are written, nodes and their required 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 may 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 **FoldingIntent**, **ColorIntent**, **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 may be extremely vague, detailing only the general goal, or it may be very specific, stipulating the specific requirements inherent in meeting that goal. Product intent may 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 may share some parts.

Product intent is defined by the initiating agent of a job. It is not required, however. Many JDF jobs are written with full knowledge of the necessary processes, and are therefore comprised entirely of the various kinds of process nodes described in Section 3.1.3, Section 3.1.4, and Section 3.1.5. Any job that specifies product intent, however, 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 may initiate with the customer and in rather vague terms, and they may be later fleshed out or completed by a printer’s customer service representative, estimating department, or production planners.

### 4.1.1.1 Representation of Product Intent

The product description of a job is a hierarchy of *Product* nodes, and the bottom-most level of the product hierarchy represents portions of the product that are each homogeneous in terms of their materials and formats. All nodes below these *Product* nodes begin specifying the processes required to produce the products.

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

In some cases, more than one high level product node will use the output of a product node. These high level nodes represent the combination of homogeneous product parts. In this case, the *Amount* attribute of the **ResourceLinks** that connect the nodes will identify how the lower level product is shared.

### 4.1.1.2 Representation of Product Binding

Some product intent nodes, such as **BindingIntent**, 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 resource links. For more information about product intent, see Section 3.1.3, 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 JDF/NodeInfoBusinessInfo element, and the customer related data may be placed into the root JDF/CustomerInfo element.



### PrintTalk Implementation

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

This section sketches the usage of JDF in an E-commerce environment using the business object model that was defined by the PrintTalk [www.PrintTalk.org](http://www.PrintTalk.org) consortium.

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

Object Type	Description	References	Direction
RFQ (Request for Quote)	Initiated by a buyer to a print supplier. It may instigate a new product process or it may 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 may supersede an existing Quote before the Print Buyer has answered with a RFQ or an Order.	RFQ, PO, Confirmation	B←P
Purchase Order	Typically sent as a response to a quote but may 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 may supersede an existing Order prior to the Print Provider having confirmed it.	None, Quote, Confirmation	B→P

Table 4-1: Business Objects as defined by PrintTalk

Object Type	Description	References	Direction
(Order) Confirmation	Sent by the print supplier to the buyer acknowledging receipt of the purchase order. It may 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. If only parts of a job should be cancelled, one must send a new RFQ, Quote, or PO. In case of canceling parts of a confirmed order, the Change Order variations of these Business Objects must be 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. May 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 at [www.printtalk.org](http://www.printtalk.org) for a complete picture.

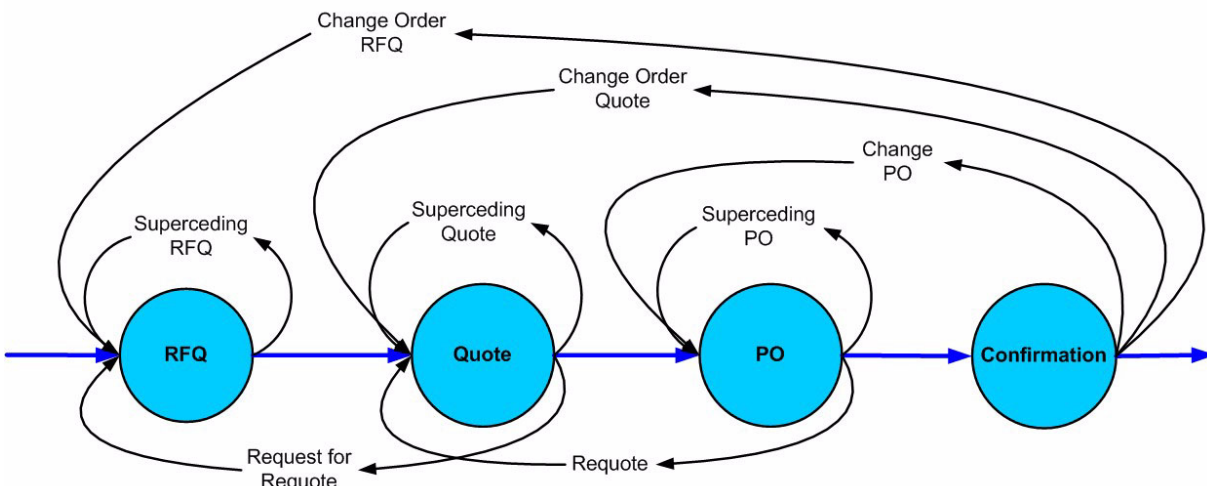


Figure 4.1: Simplified PrintTalk workflow (negotiation phase)

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 ResourceLinks is *Input*, its *Type* is *Product* and the Business object is an RFQ.

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

### PrintTalk example

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

### Equivalent pure JDF Example

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

#### 4.1.3 Specification of Delivery of End Products

A job may define one or more products and specify a set of deliveries of end products. To accomplish this, a node of *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 Drops. Each Drop has a common delivery address and time, and a set of DropItems that specifies the amount of individual **Components** that must be delivered 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 may 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.

Individual JDF process or ProcessGroup nodes may be inserted into a product intent node. These nodes will contain the requested process resource definitions as input resource links. The *Status* attribute of these resources should be *Incomplete*. No output resources should be defined. In other words, the actual specification of the process workflow should be left undefined. The application that sets up the actual workflow can then use these resource templates as a starting point for defining the process. It is recommended to specify a ProcessGroup node that does not define the process granularity. For details see "Use of the Types attribute in ProcessGroup nodes" on page 44. The following example shows how an ellipse spot function is requested within a simple product description. The JDF node in **highlight** defines the screening parameters of the product.



```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="Job1" JobID="J1" Status="Waiting"
Type="Product" Version="1.2">
  <ResourcePool>
    <Component Amount="10000" Class="Quantity"
    DescriptiveName="Complete 16-page Brochure" ID="Link0003" Status="Unavailable"/>
    <LayoutIntent Class="Intent" ID="Link0004" Status="Available">
      <Dimensions DataType="XYPairSpan" Preferred="612 792" Range="576 720~648 864"/>
      <Pages DataType="IntegerSpan" Preferred="16"/>
    </LayoutIntent>
    <MediaIntent Class="Intent" ID="Link0005" PartIDKeys="Option" Status="Available">
      <FrontCoatings DataType="NameSpan" Preferred="None"/>
      <MediaIntent Option="1">
        <FrontCoatings DataType="NameSpan" Preferred="Glossy"/>
      </MediaIntent>
      <BackCoatings DataType="NameSpan" Preferred="None"/>
    </MediaIntent>
  </ResourcePool>
  <ResourceLinkPool>
    <ComponentLink Usage="Output" rRef="Link0003"/>
    <LayoutIntentLink Usage="Input" rRef="Link0004"/>
    <MediaIntentLink Usage="Input" rRef="Link0005"/>
  </ResourceLinkPool>
  <JDF ID="Link0006" Status="Waiting" Type="ProcessGroup" Types="Screening">
    <ResourcePool>
      <ScreeningParams Class="Parameter" ID="ScreenID" Status="Incomplete">
        <ScreenSelector ScreeningFamily="My favorite screen" SpotFunction="Ellipse"/>
      </ScreeningParams>
    </ResourcePool>
    <ResourceLinkPool>
      <ScreeningParamsLink Usage="Input" rRef="ScreenID"/>
    </ResourceLinkPool>
  </JDF>
</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 may be a non-trivial task. Systems with a centralized, smart master controller may want to route jobs dynamically by sending them to the appropriate locations. Simple systems, on the other hand, may have a static, well defined routing path. Such a system may, for example, pass the job from hot folder to hot folder. Both of these extremes are valid examples of JDF systems that have no need for additional routing metadata.

In order to accommodate systems between these extremes, the **NodeInfo** element of a node contains optional *Route* and *TargetRoute* attributes that let an agent define a static process route on a node-by-node basis. **JMF/QueueSubmissionParams/@ReturnURL** takes precedence over **NodeInfo/@TargetRoute** of the JDF 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

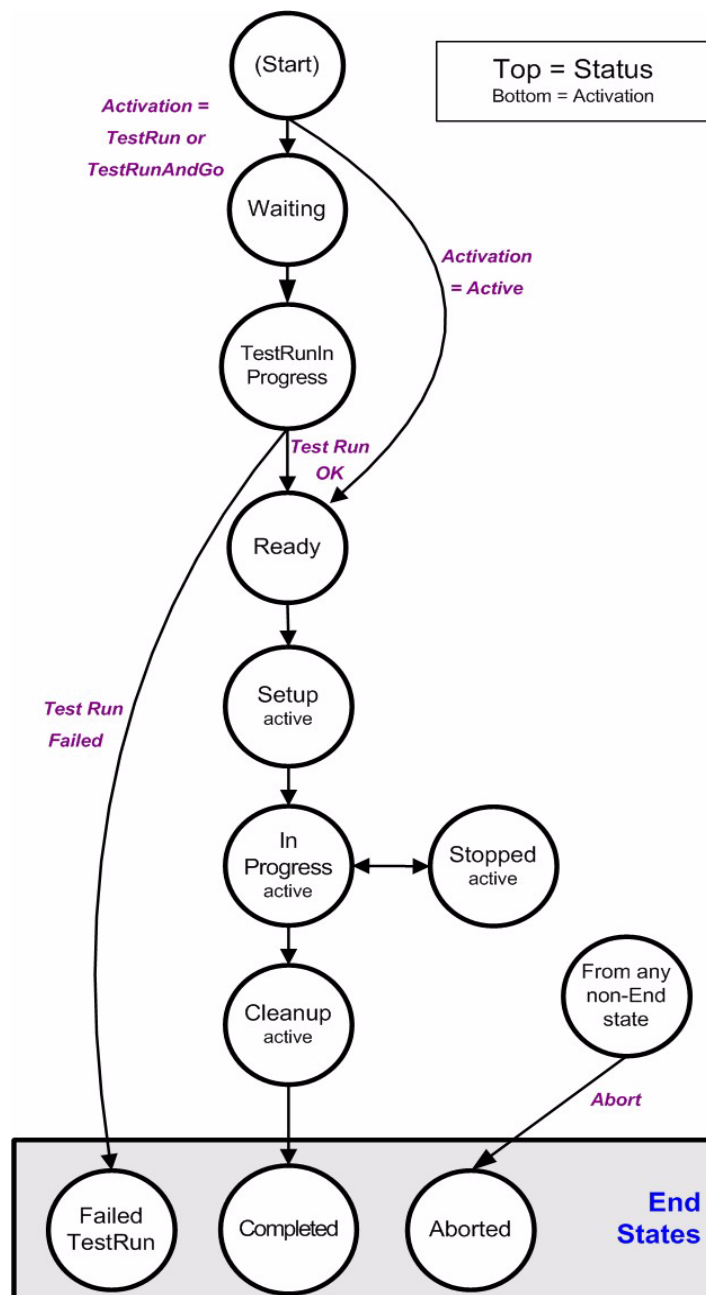


Figure 4.2: Life Cycle of a JDF node

### 4.2.2 Distributing Processing to Work Centers or Devices

JDF syntax supports two means of distributing processes to work centers or devices. Its first option is to use a “smart” controller that has the ability to parse a JDF job and identify individual processes or process groups that may 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.

In order to determine which node should be executed, the controller/device uses the following procedures.

1 It searches the JDF document for node types it can execute by comparing the *Type* and *Types* attributes and optionally the *Category* attribute of the node to its own capabilities and by determining the *Activation* of the nodes. It should also verify that the *Status* of the node is either *Waiting* or *Ready*. If a **Device** resource is specified as input to the node, the resource must match the controller/device. Devices may opt to limit the scope of the node search. The limitations should be specified in the device capability description by appropriately setting `DeviceCap/@ExecutionPolicy`.

2 The controller/device may 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 optionally skip these checks and expect the lower level controller or device that it controls to perform this step and return with an error if it fails.

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

```
NotificationNotificationClass
"Error" AuditPool Notification
ErrorReturnCode:"102"
```

The node will go through various states during its life time as is described in Figure 4.2.

### 4.2.3 Device / Controller Selection

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

Knowledge of the capabilities of lower level controllers/devices either may be hard-wired into the system or gained using the *KnownDevices* message. Since JDF does not yet provide mechanisms to determine if a given device is capable of processing a node without actually performing a test run, a controller must either have a prior knowledge of the detailed capabilities of devices that it controls or it must perform a test run to determine if a device is capable of executing a node. Furthermore, in addition to the explicit routing information in the *Route* attribute of the *NodeInfo* element of a node, JDF may contain implicit routing information in the form of **Device** implementation resources.

JMF defines the *KnownControllers* query to find controllers and the *KnownDevices* query to find devices that are controlled by a controller. The information provided by these queries can be used by a controller to infer the appropriate routing for a node. In a system that does not support messaging, this information must be provided outside of JDF.

## 4.3 Execution Model

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

The processing model of JDF is based on a producer/consumer model, which means that the sequencing of events is controlled by the availability of input resources. As has been described, nodes act both as producers and consumers of resources. When all necessary inputs are available in a given node, and not before, the process may 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.

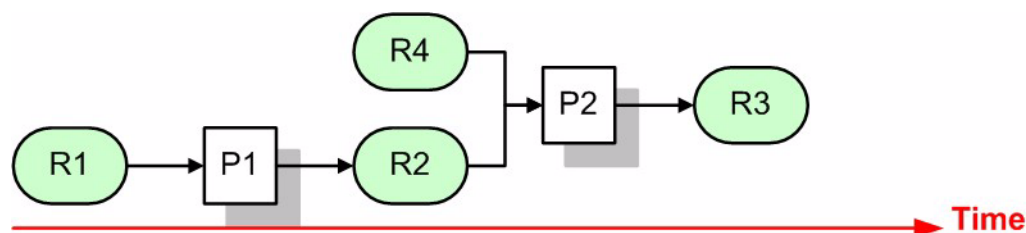


Figure 4.3: Example of a simple process chain linked by resources

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 may 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 3-13, “Contents of the abstract Resource element,” on page 53. For more information about the values of the *Status* attribute of processes, see Table 3-4, “Contents of a JDF node,” on page 38.

Table 4-2: Examples of resource and process states in the case of simple process routing

Object Status	before running P1	during running P1	after running P1, before P2	during P2	after P2
resource R1	<i>Available</i>	<i>InUse</i>	<i>Unavailable</i>	<i>Unavailable</i>	<i>Unavailable</i>
resource R2	<i>Unavailable</i>	<i>Unavailable</i>	<i>Available</i>	<i>InUse</i>	<i>Unavailable</i>
resource R3	<i>Unavailable</i>	<i>Unavailable</i>	<i>Unavailable</i>	<i>Unavailable</i>	<i>Available</i>
resource R4	<i>Available</i>	<i>Available</i>	<i>Available</i>	<i>InUse</i>	<i>Available</i>
process P1	<i>Waiting or Ready</i>	<i>InProgress</i>	<i>Completed</i>	<i>Completed</i>	<i>Completed</i>
process P2	<i>Waiting or Ready</i>	<i>Waiting or Ready</i>	<i>Waiting or Ready</i>	<i>InProgress</i>	<i>Completed</i>

If a process aborts before completion, its output resources are *Unavailable* unless the output has been partially produced in which case the device may update the amount and set the output to *Available*.

When the attribute *Amount* is used in connection with the quantifiable resources R1, R2, or R3 and their links, then the controller must decide whether or not a resource is available by comparing the individual values. If the amounts are used to define the availability, then the resource *Status* may be set to *Available* for all *Quantity* resources. Note that when the value of the *Status* attribute of the resource is *Unavailable*, the resource is not available even if a sufficient *Amount* is specified.

If amounts are specified in the resource element, they represent the actual available amount. If they are not specified, the actual amount is unknown, and it is assumed that the process will consume the entire resource. Amounts of *PhysicalLink* elements must be specified for output resources that represent the intended production amount. The specification of the *Amount* attribute for input resources is not required, although it can be specified. For details, see “Resource Amount” on page 71. 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 may be available. If this occurs, the actual produced amount is logged into the audit pool by a resource audit element.

### 4.3.2 Partial Processing of Nodes with Partitioned Resources

[New in JDF 1.2](#)

JDF nodes themselves may not be partitioned, although the input and output resources may. If the input and output *ResourceLinks* 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 may 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 illustrate the restriction algorithms:

Input Partition 1	Input Partition 2	Output Partition	Node Partition	Description
<i>SheetName</i> = "S1"	—	—	<i>SheetName</i> = "S1"	If only the input is partitioned, the node partition is defined by the input.
<i>SheetName</i> = "S1" <i>Separation</i> = "Cyan"	—	—	<i>SheetName</i> = "S1" <i>Separation</i> = "Cyan"	If only the input is partitioned, the node partition is defined by the input.
<i>SheetName</i> = "S1" <i>Separation</i> = "Cyan"	<i>Separation</i> = "Cyan" + <i>Separation</i> = "Black" ( <i>PartUsage</i> = "Implicit")	—	<i>SheetName</i> = "S1" <i>Separation</i> = "Cyan" + <i>SheetName</i> = "S1" <i>Separation</i> = "Black"	The first input is partitioned by <i>SheetName</i> and <i>Separation</i> which defines the partition key granularity. The second input is partitioned by <i>Separation</i> only but has an implied <i>SheetName</i> and has a larger but overlapping set of separation values. The separation value set is therefore defined by the second key.
<i>SheetName</i> = "S1"	—	<i>SheetName</i> = "S1" <i>Separation</i> = "Cyan"	<i>SheetName</i> = "S1" <i>Separation</i> = "Cyan"	The input and output base partitions are identical. The output further restricts the partition.
<i>SheetName</i> = "S1"	—	<i>SheetName</i> = "S2" <i>Separation</i> = "Cyan"	Error	Input and output are not overlapping. This specifies the null set.
<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 <b>PlacedObject</b> elements with an <i>Ord</i> in the range of 0 to 7. This special case is important when <b>RunList</b> 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 resource links 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 may 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. All that is required is that the *PipeID* attribute be specified 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 3-23, “Additional contents of the abstract physical ResourceLink and PartAmount element,” on page 67). If none of them are specified, any produced *Quantity* may be immediately consumed by the consuming end of the pipe. The unit of the buffers is defined by the *Unit* attribute of the resource.



#### PIPE RESOURCES

A pipe resource is simply an input to a process that can be exhausted and may be replenished. Examples may 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.

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 an active pipe, represented here as R2, is defined to be *Status = InUse* (see also Table 3-13, “Contents of the abstract Resource element,” on page 53).

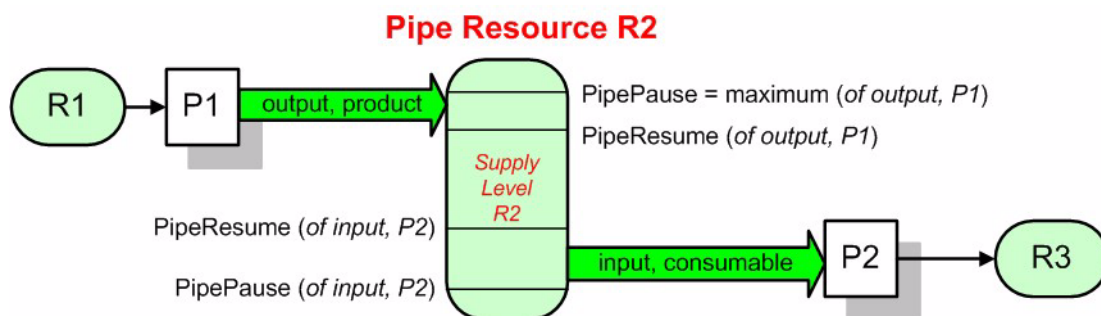


Figure 4.4: Example of a Pipe resource linking two processes

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.

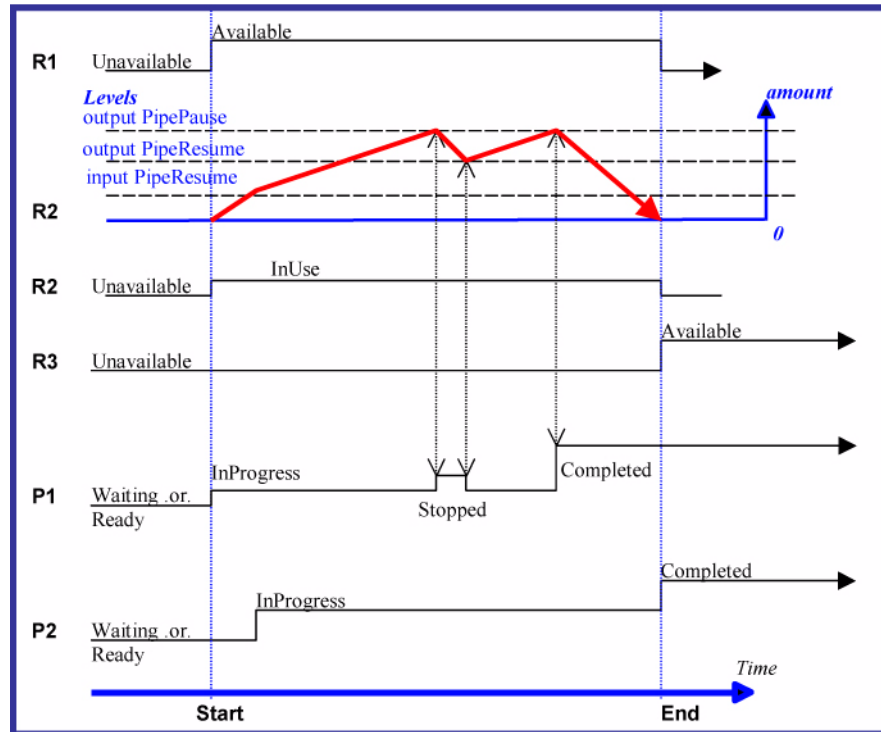


Figure 4.5: Example of status transitions in case of overlapping processing

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 link. Whenever the level representing the available quantity of the pipe resource exceeds the *PipePause* level of the output link, 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 link, 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 links, the *PipeResume* value must be smaller than the *PipePause* value, whereas in the case of input links, the *PipeResume* value must be greater than the *PipePause* value. If *PipePause* is specified for an input or an output link and *PipeResume* is not specified, the related process may 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 5.5.3, Pipe Control). If the attributes *PipeResume* or *PipePause* of links 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. To accomplish this, the *PipePartIDKeys* attribute may be specified in the appropriate **ResourceLink** element. For instance, a partitioned **ImageSetting** process may be defined for multiple sheet separations, but a complete set containing all separations of both sides of a single sheet should be sent to the pressroom as one pipe request. In this case, the value of the *PartIDKeys* attribute of the **ExposedMedia** resource would be *SheetName Side Separation* and the value of the *PipePartIDKeys* attribute of the resource link to 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 required 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 required during a press run because of quality deterioration. The exact point in time where quality becomes unacceptable is not predetermined and may 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 5.5.3, Pipe Control) sent to another controller whose URL address is specified by the *PipeURL* attribute of the respective resource link. Depending on the values of the resource link's *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 may control the process that contains the resource link, 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 5.5.3, Pipe Control are designed to establish communication between processes at both ends of dynamic pipe, even if the corresponding processes are spawned separately.

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

Table 4-3: Actions generated when a dynamic-pipe buffer passes various levels

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 may notify an off-line finishing process when a certain amount is ready by sending a `PipePush` message, or the printing process may request a new plate by sending a `PipePull` message.

#### 4.3.3.3 Comparison of Non-Dynamic and Dynamic Pipes

The resource link between non-dynamic pipes provides the buffering parameters for the process to which the link 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 resource link control the process at the other end. In the case of dynamic pipes, no master controller is required to control the pipe. Control is accomplished by sending pipe messages. If pipe resources are linked to multiple consumers or producers, such as two finishing lines that consume the output of one press one palette at a time, it is up to implementation to ensure consistency of the processes.

When using pipe resources, it is recommended that scheduling data for the process be specified only in the `NodeInfo` element of the parent node of the processes linked by pipe resources in order to avoid scheduling deadlocks. In Figure 4.5, for instance, the actual start and end time of the corresponding parent of P1 and P2 are marked on the time axis.

#### 4.3.4 Parallel Processing

While serial processing assumes that all resources will be produced and consumed in a linear fashion, and while overlapping processing uses multiple processes that work together to use and create resources, there are times when it makes sense to run more than one process simultaneously, creating a multi-pronged workflow. This kind of process routing is known as parallel processing. Subsections of jobs are spawned off so that nodes may 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 may be produced independently of a cover, and both will be bound together later.

In parallel processing, processes can be run in a coordinated parallel fashion by using independent resources. An independent resource is a resource that is not shared between multiple processes. Implementation resources, for example, cannot be shared and are therefore always independent, and *Consumable* and *Quantity* resources can each be split to function as independent resources. Individual partitions of partitionable resources are independent and may 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 may 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 may split consumable resources and create independent process chains in which each chain consumes its own resource part. Afterwards, the agent can 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 must 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** that links to a draft input resource must include a *DraftOK* attribute 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 must not be used for printing until it is *Available* and no longer designated as a *Draft*.

Iterative processes may 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 required resources 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 may 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

[Modified in JDF 1.2](#)

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 may only be set as *Available* if 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 job 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 subjobs that have already been spawned may in turn spawn other sub-subjobs, and so on. This does not mean, however, that a node may be re-spawned. If a node is 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 job. Furthermore, in order to avoid inconsistent JDF states after merging, each merging should be logged, or the appropriate spawn audit 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 job has been merged back into the original job.

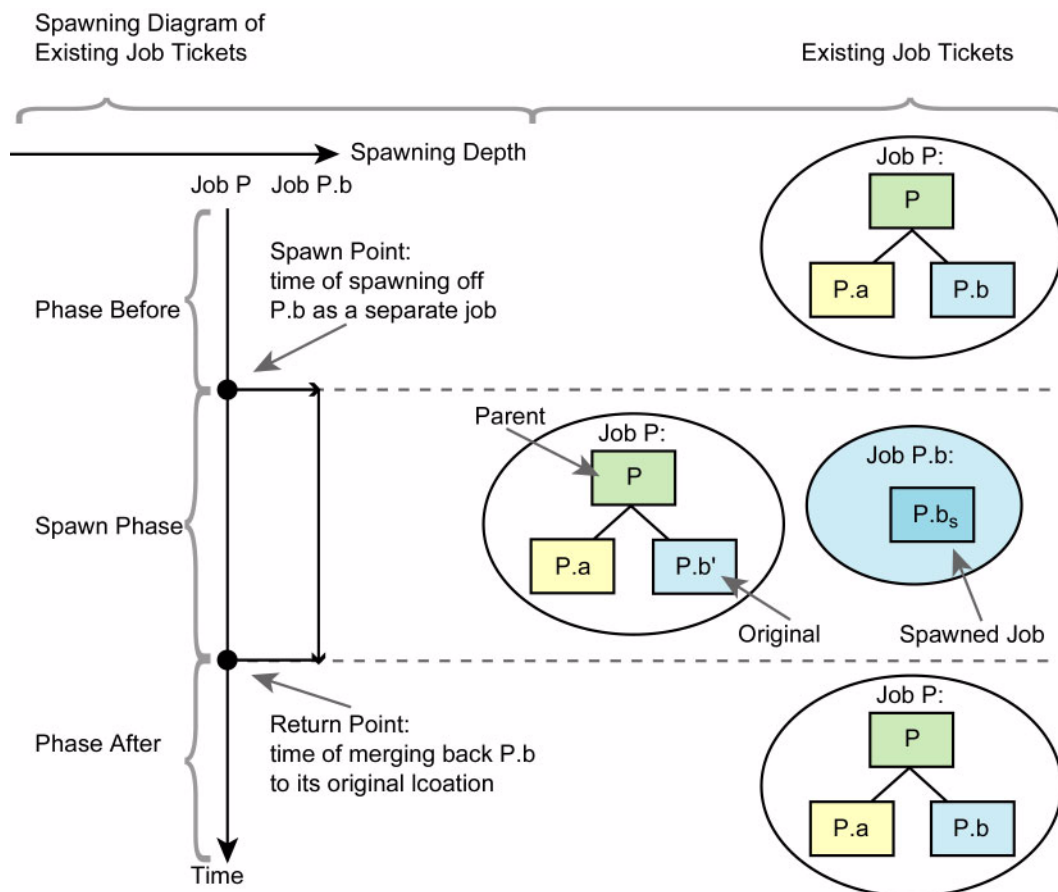


Figure 4.6: The spawning and merging mechanism and its phases

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** job part (P.b'), also denoted as a subjob) that has been spawned; and the **spawned job** (P.b<sub>s</sub>) 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 required 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 may be invented, begin with these procedures.

##### Spawning

To indicate that a process has been spawned, the *Status* attribute of the original JDF node must be set to the value *Spawned* (see Table 3-4, “Contents of a JDF node,” on page 38). 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. This simplifies bookkeeping of audits 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 job. 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 job must be copied into the *AncestorPool* element of the newly spawned job before the ancestor information of the previously spawned job is appended to the *AncestorPool* element of the newly spawned job. The last *Ancestor* element in each *AncestorPool* is the parent, the second-to-last the grandparent, and so on. *NodeInfo* and *CustomerInfo* elements may optionally be copied into the respective *Ancestor* elements. The following code is an example of a family tree:

```
<AncestorPool>
  <Ancestor FileName="file:///grandparent.jdf" NodeID="p_01"/>
  <Ancestor FileName="file:///parent.jdf" NodeID="p_02"/>
</AncestorPool>
```

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

When a job is spawned, the action must be logged in the parent node of the spawned node in the original job. This is accomplished by creating a *Spawned* element with the *jRef* attribute set to the ID of the spawned JDF node. This *Spawned* element must be appended to the *AuditPool* container of the original parent node. If no *AuditPool* container exists in the parent node, one must be created for the purpose.

##### Merging

After processing, the spawned job must be merged back to its original location. Before this can occur, however, duplicate information contained in any elements that are not required for further processing (such as *CustomerInfo* or *NodeInfo*) may optionally 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 to the *AuditPool* (see Section 3.9, *AuditPool*). The *MergeID* of the *Merged* audit 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 resource pool of an ancestor node, denoted as node A. This node is the parent node.

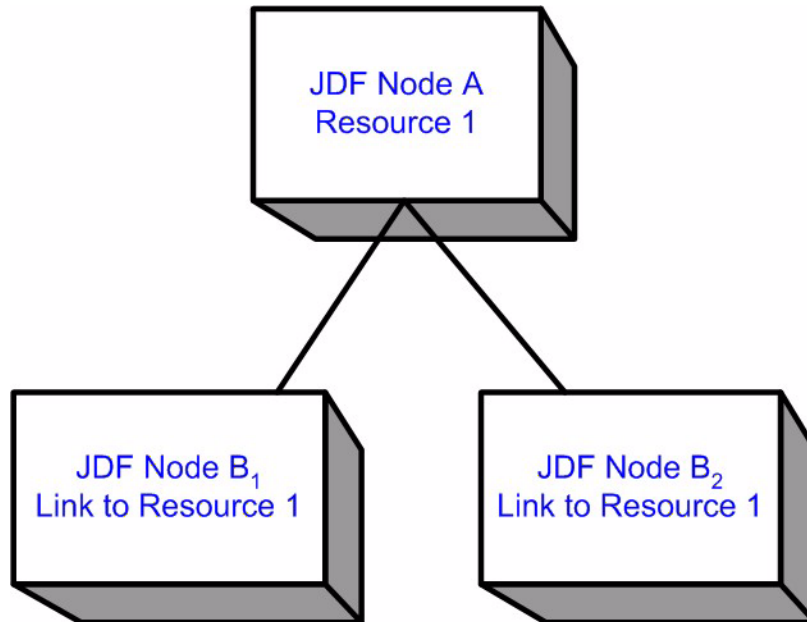


Figure 4.7: JDF node structure that requires resource copying during spawning and merging

When node  $B_1$  is spawned, its resources must also be duplicated. To accomplish this, the affected resources must be copied to the spawned job and purged during merging, a process that is described below.

##### 4.4.2.1 Spawning of Resources with Inter-Resource Links

Resources may be linked to a node by three mechanisms.

- Explicit links defined by a `ResourceLink` in the `ResourceLinkPool` of the node.
- Implicit links defined by the `ResourceRef` elements of linked Resources (implicit links are recursive).
- Implicit links defined by the `ResourceRef` elements of the `AuditPool`, `CustomerInfo`, or `NodeInfo` element of the node.

A spawning or merging agent must resolve all of these links by copying any non-local resources into the local `ResourcePool`.

#### Spawning

Spawning begins as it did in Case 1. The affected resources must then be copied to the resource pool of the spawned job. 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 one resource, or for read/write access, where a resource may only be spawned one time. 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 job. Read/write access effectively locks the original resources, just as if the attribute *Locked* = *true*<sup>1</sup> were present. If a resource is spawned as read-only, it is not a good idea to modify the orig-

1. Usually resources become locked (*Locked* = *true*) if they are referenced by audit elements (see also Section 3.9, `AuditPool`).

inal resource that remains in the parent job ticket as this may lead to inconsistencies. The *Locked* attribute of spawned resources that are copied read-only should 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. Then, if resources have been copied for spawning, they must be purged after merging. Read-only resources may simply be deleted in the spawned node before merging. If the original resource and the spawned resource are not identical, however, a JDF content error should be logged by a *Notification* element of *Class = Error* (see Section 4.6, *Error Handling*). Read/write resources must be copied into their original location, completely overwriting the original resource. 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 *ResourceLinks* 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 *ResourceLinks* of the parent node or by modifying the copies of existing *Part* elements. *Part* elements must be included in all *ResourceLinks* that point to resources that are spawned with write access. *Part* elements may be included in *ResourceLinks* that point to resources that are spawned with read only access, (e.g., physical resources where only a part is provided to a process as input). In addition, copies of the *Part* elements are appended to the *Spawned* audit element. The *Status* of any partitioned resource is defined individually for each partition. The *Status* of the parent node is set to “*Pool*” and a *StatusPool* is generated with the appropriate information. The *PartStatus* that describes the newly spawned node is set to “*Spawned*”.

Exactly one *Part* element that contains the partition keys of this spawn and all partition keys of previous spawns must be present in the *AncestorPool* of the spawned JDF node.

The spawning procedure described in this section can be performed iteratively for multiple parts, effectively generating one *Spawned* audit element and one *PartStatus* in the *StatusPool* per part. The *Spawned* and *Merged* audit elements are not placed in the parent node of the node to be spawned, but rather in the node itself.

An Agent that receives a JDF node that has been spawned in parallel and thus has a *Part* element in the *AncestorPool* must not modify any elements except for:

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

Synchronizing multiple *NodeInfo*, *CustomerInfo* elements, or newly inserted sub JDF nodes in spawned JDF nodes is not required or supported.

#### 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 is appended to the *Merged* element and any read/write resources are merged into their appropriate parts. The *Status* of the spawned node is copied into the appropriate *PartStatus* in the *StatusPool*.

An example of partitioned Spawning and Merging can be found in “Spawning and Merging” on page 672.

### 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 job, with B as root node, has already been deleted.

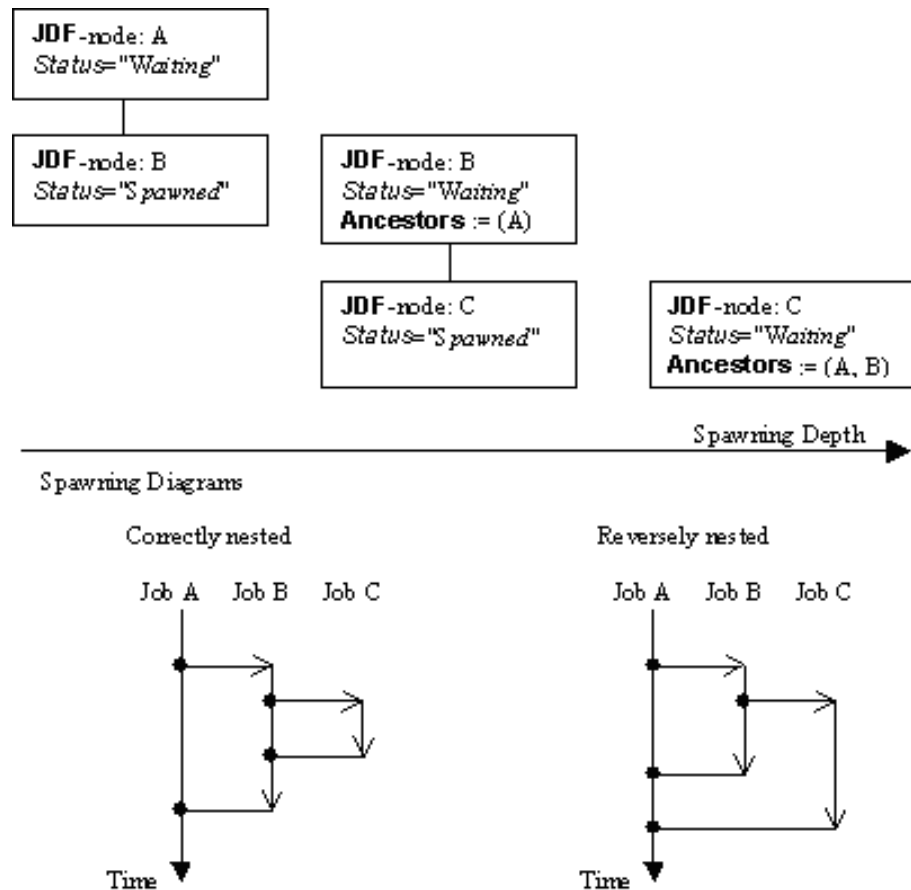


Figure 4.8: Example for a JDF node structure with nested spawning

#### 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 may 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 may exist already or which may 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 may then be exe-

cutted 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 may be related to other nodes in various ways.

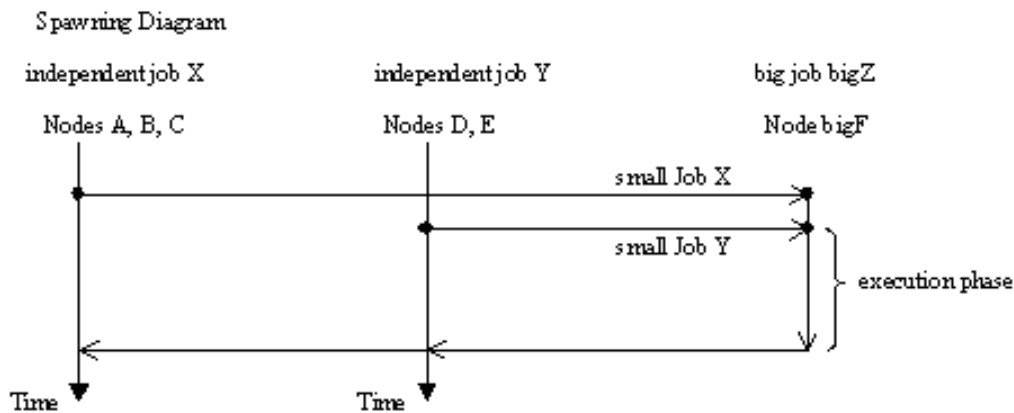
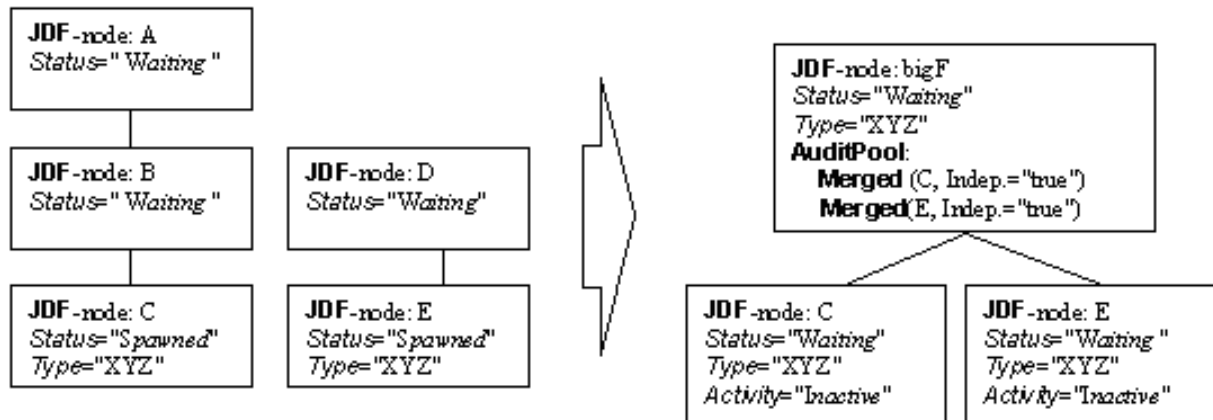


Figure 4.9: Example of the spawning and merging of independent jobs

### 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 job to define the ancestry (as was described above). A *Merged* element containing information about the spawned independent jobs and when they have been received is added to the big job.

In the original jobs, the *Status* of the process is designated as *Spawned*, and a *Spawned* element with the optional attribute *jRefDestination* specified is added to the parent of the original job. The attribute *jRefDestination* contains the ID of the big job beneath which the spawned process has been placed. The changes in the parent are the equivalent of those described in Case 1 except for the specification of the attribute *jRefDestination* in the *Spawned* element.

Where necessary, resource instances must be copied and logged as in Case 2 by appending the IDs to the appropriate attribute (*rRefsROCopied* or *rRefsRWCopied*) of the *Spawned* element in the parent of the original job. This is required in single spawning and merging. Furthermore, the *ResourceLink* elements of the spawned process must be copied to the *ResourceLinkPool* of the active, big process node. In this way, the input resources and the resources to be produced are linked to the big job.



## Merging

For each of the spawned small jobs, the return procedure is performed as it was in the preceding cases. Once the process explained in Case 1 is performed, the completed job is copied back to its original location and the attribute *Activation* is restored by setting it to the activation of the big-job node after completion.

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

If the big job is retained, a *Spawned* element with the attribute *Independent* = *true* must be appended to the *AuditPool* of the big job. For instance, saving the finished big job may be desirable if the audit information contained in the big job should be available for individual invoicing. Finally, the newly created big JDF should be deleted to avoid the double existence of nodes.

### 4.4.6 Case 6: Simultaneous Spawning and Merging of Multiple Nodes

It is not possible to explicitly spawn multiple nodes simultaneously. The nodes must be grouped into a single *ProcessGroup* node. This node can then be spawned and merged as described in the previous sections.

## 4.5 Node and Resource IDs

[Clarified in JDF 1.2](#)

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 job, 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 job, 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 A<sub>1</sub> with some IDs: *ID* = "A" *ID* = "A.A" *ID* = "A.B" where the first is the ID of the root node.
- Receives small job A<sub>2</sub> with some IDs: *ID* = "A" *ID* = "A.A" *ID* = "anything" ...
- The agent creates locally unique pure IDs: *ID* = "A1" and *ID* = "A2" each prepended to all IDs of each received small job; the IDs of the small job A<sub>1</sub> become: *ID* = "A1.A" *ID* = "A1.A.A" *ID* = "A1.A.B", and the IDs of the small job A<sub>2</sub> 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<sub>1</sub> 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 prepended 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 may be logged using **Notification** elements. **Notification** elements may also be sent in JMF **Signal** messages. The content of the **Notification** element is described in Table 3-33, “Contents of the Notification element,” on page 92. For a list of predefined error codes, see “Supported Error Codes in JMF and Notification elements” on page 619. Further details about error handling are provided in the next four sections.

### 4.6.1 Classification of Notifications

Notification elements are classified by the attribute *Class*. Every workflow implementation must associate a class with all events on an event-by-event basis. The following list shows the possible values for *Class*.

<i>Event</i>	Indicates a <b>pure event</b> which occurred due to a certain operation-related action, (e.g., machine events, operator activities, etc.). This class is used for messaging.
<i>Information</i>	Indicates not an error, but rather any information about a process that cannot be expressed by the other classes, (e.g., the beginning of execution).
<i>Warning</i>	Indicates that a minor error has occurred, and an automatic fix was applied. Execution continues. The node’s <i>Status</i> is unchanged. This appears in situations such as A4-Letter substitutions when toner is low or when unknown extensions are encountered in a required resource
<i>Error</i>	Indicates that an error has occurred that requires user interaction. Execution cannot continue until the problem has been fixed. The node’s <i>Status</i> is <i>Stopped</i> . This value appears in situations such as when resources are missing, when major incompatibilities are detected, or when the toner is empty.
<i>Fatal</i>	Execution must be aborted. The node’s <i>Status</i> is <i>Aborted</i> . This value is seen with most protocol errors or when major device malfunction has occurred.

### 4.6.2 Event Description

A description of the event is given by a generic **Comment** element, which is required for the notification classes *Information*, *Warning*, *Error*, or *Fatal*. For example, after a process is aborted, error information describing a device error may be logged in the **Comment** element of the **Notification** element. If phase times are logged, the **PhaseTime** element that logged the transition to the *Aborted* state may also contain a local **Comment** element that describes the cause of the process abortion. **PhaseTime** and **Notification** elements are optional subelements of the **AuditPool**, which is described in Section 3.9, **AuditPool**.

### 4.6.3 Error Logging in the JDF File

A JDF-compliant controller/agent should log an error by inserting a **Notification** element in the **AuditPool** of the node that generated the error. The **NodeInfo** element may contain **NotificationFilter** elements to define the notification events (or, more specifically, errors) that should be logged.

### 4.6.4 Error Handling via Messaging (JMF)

A JMF **Signal** message with a **Notification** element in the message body should be sent through all persistent channels that subscribed events of class *error*. How to subscribe error events via JMF, see Section 5.2.2.3, **Persistent Channels** and Section 5.5.1.1, **Events**. Note that this is different from the **NotificationFilter** elements of the **NodeInfo** element, which is defined for logging events by **Notification** elements to the **AuditPool**.

## 4.7 Test Running

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

The ability to perform a test run may be built into individual devices or controllers. Alternatively, a controller implementation may perform test runs on behalf of its devices. A test run may be routed through all of the different devices and controllers in a workflow, just as if the test run were a standard execution run. For the routing of jobs and nodes through different devices and controllers for a test, the spawning and merging mechanism may also be applied. The devices/controllers receiving a job read and analyze it WITHOUT initiating execution. Rather, they investigate the content of the node they would execute. A device/controller with agent capabilities may record results into the audit pool 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 inputs that have been specified as required are actually the inputs that are required, and that none are missing or in error. For example, an input requirement may 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 may request a test of certain nodes by setting the JDF attribute *Activation* to *TestRun* (see Table 3-4, “Contents of a JDF node,” on page 38), which is inherited by all descendent nodes that are not inactive (*Activation* = *Inactive*). If a device or controller<sup>1</sup> 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 5.4, *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 is required to 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 audits, see Section 3.9, *AuditPool*.

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

#### 4.7.1 Resource Status During Testrun

In order to test run a complete set of nodes, it is sometimes necessary to imply the *Status* of resources that are produced by prior nodes. Successful test running does *not* set the *Status* attribute of a resource to *Available* unless the resource actually is available. Nodes that require an output resource from a node that has completed test running for purposes of test running itself may 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*.

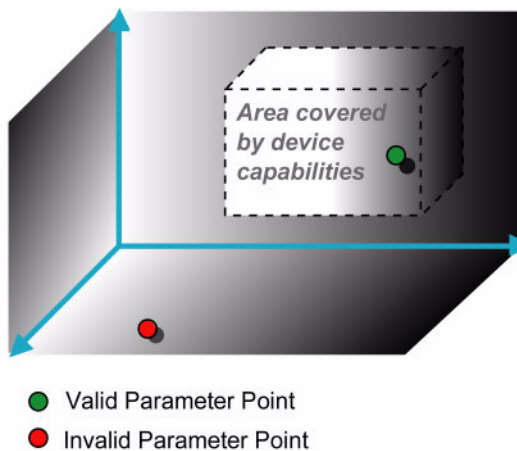


Figure 4.10: *Parameter Space in device Capabilities*<sup>a</sup>

- 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](#)

[Modified in JDF 1.2](#)

While the JDF schema describes the structure of all JDF, 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 may 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. 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 job 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 502.

It is assumed that **Device** elements that describe capabilities will be transported in JMF **KnownDevices** messages. It is not recommended to specify the capabilities of a device that is linked to a process to specify that it should execute the given process.

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. 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 must 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 may 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, optionally, the digital assets to which the JDF job tickets refer into a single package. See “JDF Packaging” on page 560

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 must 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: Contents of the JMF element

Name	Data Type	Description
<i>DeviceID</i> ?	string	Identifies the recipient device or controller. The envelope of the message contains the URL address of the controller that receives the message via HTTP. Therefore, if <i>DeviceID</i> does not specify a recipient, that controller is assumed to be the recipient.
<i>ResponseURL</i> ? <a href="#">New in JDF 1.2</a> <i>Unidirectional</i>	URL	URL of the direct response to this JMF. 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>ResponseTypeObject</i> is required for the <i>Message</i> . The <i>Response</i> may be empty. Must not be present when a bidirectional protocol is used, (e.g., in HTTP). The URL must be an explicit locator. It is up to the sending agent to generate a unique locator for the response. Example: <code>"file://master/JMFResponseFolder/Rip1/r12345.jmf"</code>
<i>SenderID</i>	string	String that identifies the sender device, controller or agent.
<i>TimeStamp</i>	dateTime	Time stamp that identifies when the message was created.
<i>Version</i> <a href="#">Modified in JDF 1.2</a>	JDFJMFVersion	Text that identifies the version of the JMF message. The current version of this specification are "1.1" and "1.2". 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 101.  Note that <i>Version</i> was optional below 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> will default to "1.1".
<i>xmlns</i> ? <a href="#">New in JDF 1.1</a>	URI	JDF supports use of XML namespaces. The namespace must be declared. For details on using namespaces in XML, see <a href="http://www.w3.org/TR/REC-xml-names/">http://www.w3.org/TR/REC-xml-names/</a> .
<i>Message</i> +	element	Abstract message element(s). Note that while a JMF instance may include multiple messages, the order of execution of the <i>Message</i> elements within a JMF is not deterministic.

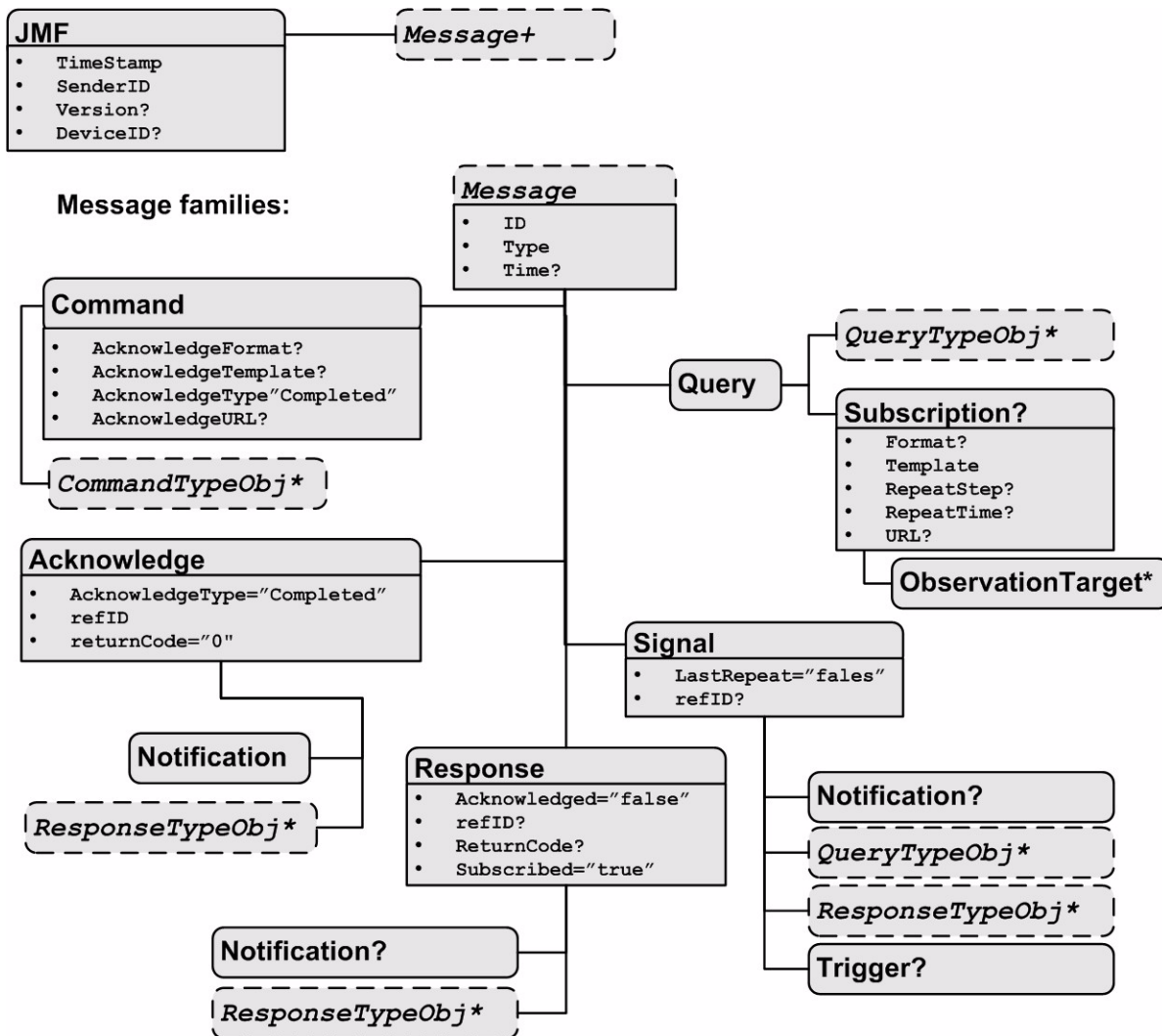
The following table describes the contents of the abstract *Message* element. All messages contain an *ID* and a *Type* attribute.

Table 5-2: Contents of the abstract Message element

Name	Data Type	Description
<i>ID</i>	ID	Identifies the message.
<i>Time</i> ?	dateTime	Time at which the message was generated. This attribute is only required if 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 Section 5.6 and Section 5.7.
<i>xsi:type</i> ? <a href="#">New in JDF 1.2</a>	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 5.5, Standard Messages. If omitted then a general definition for the JMF message will be used. See "JDF Nodes" on page 35

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

Figure 5.1: Contents of a JMF root element and the message families



## 5.2 JMF Semantics

JMF encodes messages of several types. The first part of this section describes message elements that contain and convey content, while the second describes the way in which these element types can be used to establish communication.

### 5.2.1 Message Families

A message contains one or more of the following five high level elements, referred to as **message families**, in the root node. These families are Query, Command, Response, Acknowledge, and Signal. An explanation of each family is provided in the following sections, along with an encoding example.



### Response & Acknowledgement

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

#### 5.2.1.1 Query

A Query element is used as a message that retrieves information from a controller without changing the state of that controller. A query is sent to a controller. After a Query is sent, a Response is returned. If the Query included a Subscription, Signals are sent to the designated URL until a StopPersistentChannel Command is sent.

#### Query with Subscription

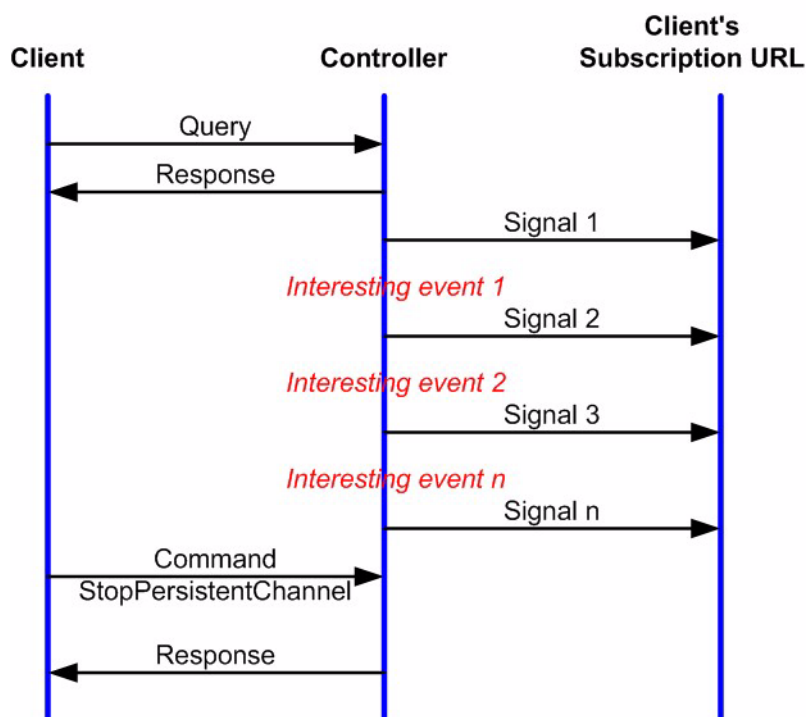


Figure 5.2: Interaction of Messages with a subscription

The Query contains an *ID* attribute and a *Type* attribute, which it inherits from the abstract message type described in Table 5-2, “Contents of the abstract Message element,” on page 130. JMF supports a number of well defined query types, and each query type can contain additional descriptive elements, which are described in Section 5.6 and Section 5.7. The following table shows the content of a Query message element.

Table 5-3: Contents of the Query message element

Name	Data Type	Description
QueryTypeObj *	element	Abstract element that is a placeholder for any descriptive elements that provide details required for the query. The element type of QueryTypeObj is defined by the <i>Type</i> attribute of the abstract Message element.
Subscription ?	element	If specified creates a persistent channel. For the structure of a Subscription element, see Section 5.2.2.3, Persistent Channels.



The following is an example of a query message:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Controller-1"
  TimeStamp="2000-07-25T11:38:23.3+02:00" Version="1.2">
  <Query ID="M007" Type="KnownDevices"/>
</JMF>
```

### 5.2.1.2 Response

A **Response** element is used to reply to a **Query** or a **Command** and is always a direct answer of a **Query** or a **Command**. A **Response** is returned from a controller to the controller that submitted the **Query** or **Command**; however, **Response(s)** are not acknowledged themselves.

A command response indicates that the command has been received and interpreted. The response of commands with short latency also includes the information about the execution. Commands with long latency may additionally generate a separate **Acknowledge** message (see Section 5.2.1.5, **Acknowledge**) to broadcast the execution of the command. Command responses should contain a **Notification** element that describes the return status in text, if **ReturnCode** is greater than 0. Responses contain an attribute called **refID**, which identifies the initiating query or command. The following table shows the content of a **Response** message.

Table 5-4: Contents of the Response message element

Name	Data Type	Description
<b>Acknowledged</b> = "false"	boolean	Used only in responses to command messages. Indicates whether the command will be acknowledged separately. If "true", an <b>Acknowledge</b> message will be supplied after command execution. If "false", no <b>Acknowledge</b> message will be supplied.
<b>refID</b> ? <a href="#">Modified in JDF 1.2</a>	NMTOKEN	Copy of the <b>ID</b> attribute of the initiating query or command message to which the response refers. If not specified, the response refers to the entire JMF message, (e.g., if the JMF was not parsable). <b>Response/@Type</b> is set to "Notification" if the <b>Type</b> of the incoming <b>Message</b> is corrupted or unknown.
<b>ReturnCode</b> = "0"	integer	Describes the result. "0" indicates success. For all other possible codes see "Supported Error Codes in JMF and Notification elements" on page 619.
<b>Subscribed</b> ?	boolean	If a <b>Subscription</b> 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 5.2.2.3, <b>Persistent Channels</b> .
<b>Notification</b> ?	element	Additional information including textual description of the return code. The <b>Notification</b> element should be provided if the <b>ReturnCode</b> is greater than 0, which indicates that an error has occurred.
<b>ResponseTypeObj</b> *	element	Abstract element that is a placeholder for any descriptive elements that provide details queried for or details about command execution.

An example of a response on a command is provided in the Section 5.2.1.4, **Command**. The encoding example for the query, shown above, might generate the following response:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="RIP-1" TimeStamp="2000-07-25T11:38:25+02:00" Version="1.2">
  <Response ID="M107" Type="KnownDevices" refID="M007">
    <DeviceList>
      <DeviceInfo DeviceStatus="Unknown">
        <Device DeviceID="Rip1"/>
      </DeviceInfo>
      <DeviceInfo DeviceStatus="Unknown">
        <Device DeviceID="Rip2"/>
      </DeviceInfo>
    </DeviceList>
  </Response>
</JMF>
```

### 5.2.1.3 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 may be used to automatically broadcast status changes.

Controllers can get signal messages in one of three ways. The first way is to subscribe for them with an initiating query transmitted via a message channel that includes a **Subscription** element. The second way is to subscribe for them with an initiating query defined in the **NodeInfo** element of a JDF node that also includes a **Subscription** element (see JMF elements in Table 3-9, “Contents of the NodeInfo element,” on page 51). The first query is transmitted separately via a mechanism such as HTTP, whereas the second is read together with the corresponding JDF node. Once the subscription has been established, signals are sent to the subscribing controllers via persistent channels. In both cases, however, the **Signal** message contains a *refID* attribute that refers to the persistent channel. The value of the *refID* attribute identifies the persistent channel that initiated the **Signal**.

The third way in which a controller may 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 subcontrollers that are newly connected to a network want to inform other controllers about their capabilities. Hard-wired signals, however, must not have a *refID* attribute. If no *refID* is specified, the corresponding query parameters must be specified instead.

Table 5-5: Contents of the Signal message element

Name	Data Type	Description
<i>LastRepeat</i> = "false"	boolean	If <i>true</i> , the persistent channel is being closed by the Device and no further messages will be generated that fulfill the persistent channel criteria. If <i>false</i> , further signals will be sent. For further details, see Section 5.2.2.3, 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 <b>Notification</b> 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. For details about subscribing pure events see Section 5.5.1.1, Events.
QueryTypeObj * <a href="#">Modified in JDF 1.2</a>	element	If no <i>refID</i> is specified, the corresponding query parameters must be specified instead by providing this element. This element is an abstract element and a placeholder for any descriptive elements that provide details for the virtual <b>Query</b> , which, if sent, would convey the same <b>ResponseTypeObj</b> elements. The element type of <b>QueryTypeObj</b> is defined by the <i>Type</i> attribute of the abstract <b>Message</b> element.
ResponseTypeObj *	element	Abstract element that is a placeholder for any descriptive elements that provide details subscribed. These element types are the same as in the <b>Response</b> message element.
Trigger ?	element	Describes the trigger event which caused this signal. The <b>Trigger</b> element recalls some information provided during the subscription of the signal messages. For details on subscribing signals see Section 5.2.2.3, Persistent Channels.

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

Table 5-6: Table 5-6 Contents of the 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 5-13.
<i>RepeatTime</i> ?	double	Recalls the <i>RepeatTime</i> attribute specified during subscription of the signal. For details see Table 5-13.

Table 5-6: Table 5-6 Contents of the Trigger element

Name	Data Type	Description
<a href="#">Added ?</a> <a href="#">Deprecated in JDF 1.2</a>	element	A pool that contains the description of trigger events caused by the adding of elements like services, controllers, devices, or messages. Replaced by <b>ChangedPath</b> in JDF 1.2 and above.
<a href="#">ChangedAttribute *</a> <a href="#">Deprecated in JDF 1.2</a>	element	If a change of an attribute triggered this signal, this element describes the attribute that changed. Replaced by <b>ChangedPath</b> in JDF 1.2 and above.
<a href="#">ChangedPath *</a> <a href="#">New in JDF 1.2</a>	element	If a change of an attribute or element triggered this signal, this element describes the details of the element or attribute that changed.
<a href="#">Removed ?</a> <a href="#">Deprecated in JDF 1.2</a>	element	A pool that contains the description of trigger events caused by the removal of elements like services, controllers, devices, or messages. Replaced by <b>ChangedPath</b> in JDF 1.2 and above.

[New in JDF 1.2](#)

The following describes the structure of the **ChangedPath** element. **ChangedPath** replaces the **ChangedAttribute**, **Added** and **Removed** elements.

Table 5-7: Contents of the ChangedPath element

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

[Deprecated in JDF 1.2](#)

The following describes the structure of the **ChangedAttribute** element.

Table 5-8: Contents of the 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 has to be cast to the appropriate data type that depends on the attribute's data type.
<i>NewValue</i>	string	New value of the attribute.

[Deprecated in JDF 1.2](#)

The following describes the structure of the **Added** element referenced in Table 5-6.

Table 5-9: Contents of the Added element

Name	Data Type	Description
AddedElement *	element	If the appending of an element like a service, controller, device, or message triggered this signal, this element describes which service, controller, device, message, etc. has been added.  This is an abstract element. It is a placeholder for a ResponseTypeObj like NotificationDef, a JDFController, a Device, a JDFService, or a MessageService.  For details on these elements see Section 5.5.1, Controller Registration and Communication Messages.

[Deprecated in JDF 1.2](#)

The following describes the structure of the **Removed** element referenced in Table 5-6.

Table 5-10: Contents of the Removed element

Name	Data Type	Description
RemovedElement *	element	If the removal of an element like a service, controller, device, or message triggered this signal, this element describes which service, controller, device, message, etc. has been removed.  This is an abstract element. It is a placeholder for a ResponseTypeObj like NotificationDef, a JDFController, a Device, a JDFService, or a MessageService.  For details on these elements see Section 5.5.1, Controller Registration and Communication Messages.

The following is an example of a signal message:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Press 45" TimeStamp="2000-07-25T12:28:01+02:00" Version="1.2">
  <Signal ID="s123" Type="Status">
    <StatusQuParams JobID="42" JobPartID="66"/>
    <DeviceInfo DeviceStatus="Setup"/>
  </Signal>
</JMF>
```

#### 5.2.1.4 Command

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

Table 5-11: Contents of the Command message element

Name	Data Type	Description
<a href="#">New in JDF 1.2</a> <i>AcknowledgeFormat ?</i> <i>Unidirectional</i>	string	A formatting string used with the <i>AcknowledgeTemplate</i> attribute to define a sequence of generated URLs. <i>AcknowledgeFormat</i> and <i>AcknowledgeTemplate</i> are used in an analogous manner to <i>FileFormat</i> and <i>FileTemplate</i> attributes of the <b>FileSpec</b> resource. (See "FileSpec" on page 359) Only one of <i>AcknowledgeFormat</i> and <i>AcknowledgeTemplate</i> or <i>AcknowledgeURL</i> must be specified.
<a href="#">New in JDF 1.2</a> <i>AcknowledgeTemplate ?</i> <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.

Table 5-11: Contents of the Command message element

Name	Data Type	Description
<a href="#">AcknowledgeURL ?</a> <a href="#">Modified in JDF 1.2</a> <i>Bidirectional</i>	URL	URL of the recipient of any Acknowledge. 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.
<i>AcknowledgeType</i> = "Completed" <a href="#">New in JDF 1.1</a>	enumerations	Defines the actions that should be acknowledged. This is necessary mainly for device-machine pairs where the machine is not accessible online. <i>Received</i> – The Command has been received and understood, (e.g., by an operator). <i>Applied</i> – The Command has been applied to the machine, (e.g., by an operator). <i>Completed</i> – The Command has been executed.
<i>CommandTypeObj</i> *	element	Abstract element that is a placeholder for any descriptive elements that provide details of the command.

The following example demonstrates how a **ResumeQueueEntry** command may cause a job in a queue to begin executing:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="A3 Printer" SenderID="MIS
master A" TimeStamp="2000-07-25T12:32:48+02:00" Version="1.2">
  <Command ID="M009" Type="ResumeQueueEntry">
    <QueueEntryDef QueueEntryID="job-0032"/>
  </Command>
</JMF>
```

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

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="A3 Printer" SenderID="A3
Printer" TimeStamp="2000-07-25T12:32:48+02:00" Version="1.2">
  <Response ID="M109" Type="ResumeQueueEntry" refID="M009">
    <Queue DeviceID="A3 Printer" Status="Full">
      <QueueEntry JobID="job-0032" QueueEntryID="job-0032" Status="Running"/>
    </Queue>
  </Response>
</JMF>
```

### 5.2.1.5 Acknowledge

An **Acknowledge** element is a message that is an asynchronous answer to a **Command** issued by a controller. Each **Acknowledge** message is unidirectional and similar to a **Response**, and the *refID* attribute of each refers to the initiating command. **Acknowledge** messages are generated if commands with long latency have been executed in order to inform the command sender about the results. **Acknowledge** messages are only generated if the initiating command has specified the attribute *AcknowledgeURL*.

#### Command with Acknowledge

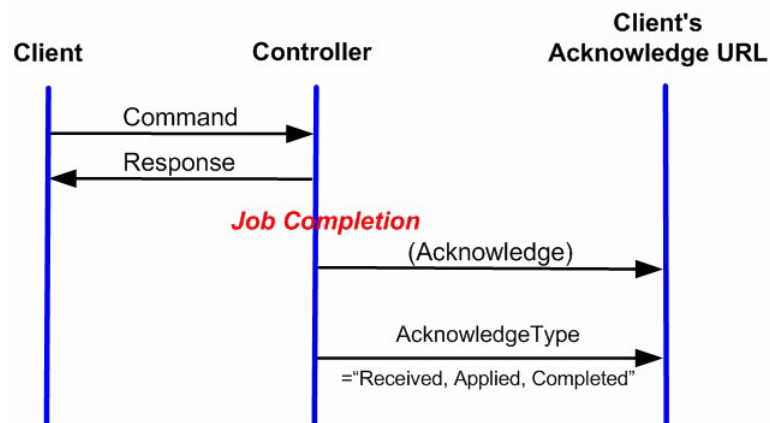


Figure 5.3: Interaction of Command and Acknowledge Messages

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

Table 5-12: Contents of the Acknowledge message element

Name	Data Type	Description
<i>AcknowledgeType</i> = "Completed" <a href="#">New in JDF 1.1</a>	enumerations	Defines the context of this message. This is necessary mainly for device-machine pairs where the machine is not accessible online. <i>Received</i> – The initiating <b>Command</b> has been received and understood, (e.g., by an operator). <i>Applied</i> – The initiating <b>Command</b> has been applied to the machine, (e.g., by an operator). <i>Completed</i> – The initiating <b>Command</b> 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 <b>Acknowledge</b> 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 619.
<b>Notification</b> ? <a href="#">Modified in JDF 1.1A</a>	element	Textual description of the command execution.
<b>ResponseTypeObj</b> *	element	Abstract element that is a placeholder for any descriptive elements that provide details about command execution. Delayed <b>Acknowledge</b> messages contain the same <b>ResponseTypeObj</b> elements as direct <b>Response</b> messages.

The following is an example of an **Acknowledge** message:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" DeviceID="A3 Printer" SenderID="A3 Printer" TimeStamp="2000-07-25T12:32:48+02:00" Version="1.2">
  <Acknowledge ID="M109" Type="PipePush" refID="M010">
    <JobPhase JobID="J1" JobPartID="1" Status="InProgress"/>
  </Acknowledge>
</JMF>
```

## 5.2.2 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.

### 5.2.2.1 Single Query/Command Response Communication

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

### 5.2.2.2 Signal

JMF signal messages are "fire and forget." In other words, no acknowledgment is sent by the receiver besides the standard protocol HTTP response that is sent when a communication link is sought.

### 5.2.2.3 Persistent Channels

Queries may be made persistent by including a **Subscription** element that defines the persistent channel-receiving end (see also Figure 5.1). The responding controller should initially send a **Response** to the subscribing controller. Then the responding controller should send **Signal** messages whenever the condition specified by one of the attributes in the following table is true. This is referred to as a **persistent channel**. The *refID* attribute of the **Signal** is defined by the *ID* attribute of the **Query**. In other words, the *refID* of the signal identifies the persistent channel.

Any **Query** may be set up as a persistent channel, although in some cases this may not make sense. Whether or not a responding controller implements a JDF Persistent Channel as an HTTP/1.1 [RFC2616] persistent connection depends on implementation.

Table 5-13: Contents of the Subscription element

Name	Data Type	Description
<i>Format</i> ? <a href="#">New in JDF 1.2</a> <i>Unidirectional</i>	string	A formatting string used with the <i>Template</i> attribute to define a sequence of generated URLs. <i>Format</i> and <i>Template</i> are used in an analogous manner to <i>FileFormat</i> and <i>FileTemplate</i> attributes of the <b>FileSpec</b> resource. (See “FileSpec” on page 359) Only one of <i>Format</i> and <i>Template</i> or <i>URL</i> must be specified.
<i>Template</i> ? <a href="#">New in JDF 1.2</a> <i>Unidirectional</i>	string	A template, used with <i>Format</i> , to define a sequence of generated URLs.
<i>RepeatStep</i> ?	integer	Requests an update signal whenever the <i>Amount</i> associated with the query is an integer multiple of <i>RepeatStep</i> . If not specified, it is up to the sending controller to generate signals.
<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>URL</i> ? <a href="#">Modified in JDF 1.2</a> <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.
ObservationTarget *	element	Requests an updating <b>Signal</b> message whenever the value of one of the attributes specified in <b>ObservationTarget</b> changes.

Table 5-14: Contents of the ObservationTarget element

Name	Data Type	Description
<i>Attributes</i> ? <a href="#">Deprecated in JDF 1.2</a>	NMTOKENS	Requests an update signal whenever the value of one of the attributes specified by <i>Attributes</i> is modified. A value of “*” denotes a message request for any attribute change which is the default. Replaced with <i>ElementPath</i> in JDF 1.2 and above.
<i>ElementType</i> ? <a href="#">Deprecated in JDF 1.2</a>	NMTOKEN	Name of the element that contains attributes that may change. Defaults to the abstract <b>ResponseTypeObj</b> of the message. Replaced with <i>ElementPath</i> in JDF 1.2 and above.
<i>ElementIDs</i> ? <a href="#">Deprecated in JDF 1.2</a>	NMTOKENS	IDs of the elements that contain attributes that may change. Used only in conjunction with a query of the state change of a certain resource or node which cannot uniquely be addressed by the other attributes of this element. Replaced with <i>ElementPath</i> in JDF 1.2 and above.
<i>ObservationPath</i> ? <a href="#">New in JDF 1.2</a>	XPath	XPath of the elements or attributes that are observed. If not specified, a <b>Signal</b> is emitted on any change in the abstract <b>ResponseType</b> of the message.

If a persistent signal channel has been set up and the device knows that this is the last time that the condition for signaling will be *true*, it should set the *LastRepeat* flag of the corresponding **Signal** message to *true*. In general, this will happen for a **Status** query, as when the job that has been tracked is completed. It may also happen when a device is shut down and will, therefore, not send any further updates. If a controller that does not support persistent channels is queried to set up a persistent channel, it must answer the query with a **Response**, set *Subscribed* to “*false*”, and set the *ReturnCode* to “111”.

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

### Creating Persistent Channels in a JDF Node

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

### Deleting Persistent Channels

A persistent channel may be deleted by sending a **StopPersistentChannel** command, as described in Section 5.5.1.6, **StopPersistentChannel**.

## 5.3 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 **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 bi-directional 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 5.6.4.8, **SubmissionMethods**.

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

## 5.4 Error and Event Messages

If a command or a query message is not successfully handled, a processor must reply with a standardized response that may contain a **Notification** element. **Notification** elements, described in detail in Section 3.9.1.2, **Notification**, convey a textual description. The information contained in the **Notification** element may be used by a user interface to visualize errors.

The response messages **Response** and **Acknowledge** contain a *ReturnCode* attribute. *ReturnCode* defaults to 0, which indicates that the response is successful. In case of success and in responses to commands an



### 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.



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 619. The following example uses a **Notification** element to describe an error:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer" TimeStamp="2000-07-25T12:32:48+02:00" Version="1.2">
  <Response ID="M109" ReturnCode="5" Type="ResumeQueueEntry" refID="M009">
    <Notification Class="Error" TimeStamp="2000-07-25T12:32:48+02:00" Type="Error">
      <Comment>StartJob unsuccessful - Device does not handle commands</Comment>
      <Error ErrorID="1234"/>
    </Notification>
  </Response>
</JMF>
```

### 5.4.1 Pure Event Messages

Notification elements are also used to signal usual events due to any activities of a device, operator, etc., (e.g., scanning a bar code). Such pure events can be subscribed to by the **Events** message described in Section 5.5.1.1, Events. These **Signals** always have a *Type* = "Notification":

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer" TimeStamp="2000-07-25T12:32:48+02:00" Version="1.2">
  <Signal ID="S1" Type="Notification">
    <Notification Class="Event" TimeStamp="2000-07-25T12:32:48+02:00" Type="Barcode">
      <Comment>Palette completed</Comment>
      <Barcode Code="99923AAA123"/>
    </Notification>
  </Signal>
</JMF>
```

## 5.5 Standard Messages

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 not required that every JDF-compliant application support every one of the signals and queries described in this list. It is, however, possible to discover which messages are supported in a workflow. A controller responds to the **KnownMessages** query by publishing a list of all the messages it supports (see Section , **KnownDevices**, below).

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

- Q: Query
- C: Command
- R: Response
- S: Signal

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

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

Table 5-15: Messaging table template

Object Type	Element name	Description
Abstract subelement of the <b>query</b> or <b>command</b> :	Name and type of the subelement that defines specifics of the query or command, followed by a cardinality symbol.	Short description of the subelement(s), if applicable.
Abstract subelement of the <b>response</b> to a query or command:	Name and type of subelement that contains specific information about the response to the query or command followed by cardinality symbol.	Short description of the subelement(s), if applicable.

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

### 5.5.1 Controller Registration and Communication Messages

The message types of the following table are defined in order to exchange metadata about controller or device abilities and for general communication.

Table 5-16: Process registration and communication messages

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 <a href="#">Deprecated in JDF 1.2</a>	QRS	Returns a list of services (JDF Node Types) that are defined in the JDF specification.
KnownMessages	QRS	Returns a list of all messages that are supported by the controller.
RepeatMessages	QR	Returns a set of previously sent messages that have been stored by the controller.
StopPersistentChannel	CR	Closes a persistent channel.

#### 5.5.1.1 Events

[Modified in JDF 1.2](#)

Table 5-17: Contents of the Events message

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

The `Events` message type is intended to be used to query for supported `Signal` messages and to subscribe for asynchronous, randomly occurring `Signals` of a device or controller. These events are described in Section 4.6.1, `Classification of Notifications` and can only be transmitted via `Signal` messages. If the query contains a `Subscription` element, a `NotificationFilter` element is combined by a logical AND operation with the `Subscription` element for selective subscriptions. An empty `Events` message (without a `Subscription` and `NotificationFilter` element) can be used to query for all events as described in “Pure Event Messages” on page 141, which are supported by a device or controller. If all signals are requested, a `NotificationFilter` with `SignalTypes` = “*all*” must be included in the query.

The controller that subscribes for `Events` messages receives `Signal` messages. In JDF 1.2, the `Events` message was enhanced to subscribe for all types of `Signals`, not only `Notification Signals`. The event type and values of `Notification` messages may then be provided by specifying a `Type` attribute and an abstract `NotificationDetails` element in the `Notification` element, as described in Section 3.9.1.2, `Notification`. Possible `NotificationDetails` elements are defined in “`NotificationDetails`” on page 621. Example of a subscription of all `Events` and the response, including the JDF 1.2 feature of subscribing for all messages by setting `NotificationFilter/@SignalTypes=“All”`:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer" TimeStamp="2000-07-25T12:32:48+02:00" Version="1.2">
  <Query ID="M170" Type="Events">
    <Subscription URL="http://www.anycompany.com/MIS/JMF/JobTracker"/>
    <NotificationFilter Classes="Event Warning Error Fatal" SignalTypes="All"/>
  </Query>
</JMF>
```

```

<JMF xmlns="http://www.CIP4.org/JDFSchemas_1_1" SenderID="A3 Printer" TimeStamp="2000-
07-25T12:32:48+02:00" Version="1.2">
  <Response ID="M1001" Type="Events" refID="M170">
    <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>

```

## Structure of the NotificationFilter Element

Table 5-18: Contents of the NotificationFilter element

Name	Data Type	Description
<i>DeviceID</i> ?	string	ID of the device whose messages are queried/subscribed. May be specified for device selection if the controller controls more than one device.
<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>QueueEntryID</i> ? <a href="#">New in JDF 1.2</a>	string	<i>QueueEntryID</i> of the job whose messages are queried/subscribed. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> , and <i>Part</i> are ignored. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> , or <i>QueueEntryID</i> are specified, <i>NotificationFilter</i> applies to all jobs.
<i>SignalTypes</i> = "Notification" <a href="#">New in JDF 1.2</a>	NMTOKENS	Possible <i>Signal/@Type</i> values of the subscribed messages. The special token "all" specifies that all Signals, regardless of <i>Type</i> are queried/subscribed.
<i>Types</i> ?	NMTOKENS	Possible <i>Notification/@Type</i> names are defined in "NotificationDetails" on page 621. Matching notification types are returned/subscribed. Defaults to all supported notification types.
<i>Classes</i> ?	enumerations	Defines the set of <i>Notification/@Classes</i> to be queried/subscribed for. Possible values are: <i>Event</i> <i>Information</i> <i>Warning</i> <i>Error</i> <i>Fatal</i> If not specified, all notification classes are subscribed to. If the values both <i>Classes</i> and <i>Types</i> are lists of values, the <i>NotificationFilter</i> defines an OR of all combinations.
<i>Part</i> * <a href="#">New in JDF 1.2</a>	element	<i>Part</i> elements that describe the partition of the job whose messages are queried/subscribed. See "Partial Processing of Nodes with Partitioned Resources" on page 112 for details on job partitions.

## Structure of the NotificationDef Element

Table 5-19: Contents of the NotificationDef element

Name	Data Type	Description
<i>Classes</i> ? <a href="#">Modified in JDF 1.2</a>	enumerations	Notification/@ <i>Class</i> of the Notification in a Signal. Possible values are: <i>Event</i> <i>Information</i> <i>Warning</i> <i>Error</i> <i>Fatal</i> <i>Classes</i> must not be specified unless <i>SignalType</i> = "Notification". For details, see Section 4.6.1, Classification of Notifications.
<i>SignalType</i> = "Notification" <a href="#">New in JDF 1.2</a>	NMTOKEN	Signal/@ <i>Type</i> value of the subscribed message.
<i>Type</i> ? <a href="#">Modified in JDF 1.2</a>	NMTOKEN	Notification type, that is the name of the element derived from the abstract NotificationDetails element. <i>Type</i> must not be specified unless <i>SignalType</i> = "Notification". For a list of predefined names see "NotificationDetails" on page 621.

### 5.5.1.2 KnownControllers

Table 5-20: Contents of the KnownControllers message

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

The KnownControllers query requests information about the controllers and devices that are known to the controller and may be directly accessed by JMF messaging. KnownControllers is designed to define a registration server. A processor that needs information about its system environment can query a registration server for a list of known controllers. A single controller that supports multiple URLs or protocols is defined using multiple JDFController elements with the same *ControllerID* attribute. This list can subsequently be iterated using the other process registration queries in this section. The URL of the master registration server must be defined using a method outside of JDF.

### JDFController

Table 5-21: Contents of the JDFController element

Name	Data Type	Description
<i>ControllerID</i> ? <a href="#">New in JDF 1.2</a>	string	String that identifies the Controller or Agent. The <i>ControllerID</i> is used as the <i>SenderID</i> of JMF messages that are produced by this Controller.
<i>URL</i>	URL	URL of the controller. If the URL scheme is "file:", <i>URL</i> must specify a directory where the JMF messages must be deposited.

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

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

## KnownDevices

Table 5-22: Contents of the 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.
<a href="#">ResponseTypeObj</a> <a href="#">Modified in JDF 1.1A</a>	DeviceList ?	The list of known devices. <sup>a</sup>

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

The KnownDevices query requests information about the devices that are controlled by a controller. If a high level controller controls lower level controllers, it should also list the devices that are controlled by these. The response is a list of **Device** resources (see Section 7.2.50, Device) controlled by the controller that receives the query, as demonstrated in the following example:

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

### Structure of the DeviceFilter Element

The DeviceFilter element refines the list of devices that should be returned. Only devices that match all parameters of one of the **Device** resources specified in the DeviceFilter element are included.

Table 5-23: Contents of the DeviceFilter element

Name	Data Type	Description
<i>DeviceDetails</i> = "None" <a href="#">New in JDF 1.1</a>	enumeration	Refines the level of provided information about the device. Possible values are: <i>None</i> – <i>Brief</i> – Provide all available device information except for Device elements. <i>Modules</i> – ModuleStatus elements should 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> <a href="#">New in JDF 1.2</a>	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.
<b>Device *</b>	element	Only devices that match the attribute values specified in one of these <b>Device</b> resources are included. Devices match the criteria if the attribute values specified here in the <b>Device</b> resource match the equivalent attribute values of the known devices. Unspecified attributes always match. If <b>Device</b> is not specified, all known <b>Devices</b> 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.

## Structure of the DeviceList Element

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

[New in JDF 1.1 a](#)

Table 5-24: Contents of the 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 5-59, “Contents of the DeviceInfo element,” on page 164 in the message description Section 5.5.2.8, Status.

### 5.5.1.3 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 772 for the details of this deprecated element.

### 5.5.1.4 KnownMessages

Table 5-25: Contents of the 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.

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

### KnownMsgQuParams

The flags of the KnownMsgQuParams element filter out the types of messages that should be included in the response list. Multiple flags are allowed.

Table 5-26: Contents of the KnownMsgQuParams element

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

### MessageService

The response is a list of MessageService elements, one for each supported message type. The flags of the MessageService response element are set in each MessageService entry. They define the supported usage of the message by the controller. Note that no *Response* attribute is included in the list, since the capability to process one of the other message families implies the capability to generate an appropriate *Response*. Multiple flags are allowed.

Table 5-27: Contents of the MessageService element

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

Table 5-27: Contents of the MessageService element

Name	Data Type	Description
<i>Query</i> = "false"	boolean	If <i>true</i> the message is supported as a Query.
<i>Signal</i> = "false"	boolean	If <i>true</i> the message is supported as a Signal.
<i>Type</i>	NMTOKEN	Type of the supported message. Extension types may be specified by stating the namespace in the value.
<i>DevCaps</i> * <a href="#">New in JDF 1.1</a>	element	Specifies the restrictions of the parameter space of the supported messages. For details on using DevCaps, see Section 7.3.1.2, Structure of the DevCaps Subelement.

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

```
<Response ID="M1" Type="KnownMessages" refID="Q1">
  <MessageService Query="true" Type="KnownMessages"/>
  <MessageService Persistent="true" Query="true" Signal="true" Type="Status"/>
</Response>
```

### 5.5.1.5 RepeatMessages

Table 5-28: Contents of the RepeatMessages message

Object Type	Element name	Description
QueryTypeObj	MsgFilter ?	A filter for the messages to be repeated. For details, see Section 5.5.1.1, Events.
ResponseTypeObj	Message *	The recent messages queried.

The RepeatMessages query returns a list of messages that have been previously sent by the controller. The optional MsgFilter element allows the list to be filtered. The list of JMF messages that fulfill the filter criteria may 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.

#### Structure of the MsgFilter Element

Table 5-29: Contents of the 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 required.
<i>Family</i> ?	enumeration	Filter for Message family. Possible values are: <i>Acknowledge</i> <i>Response</i> <i>Signal</i> <i>All</i> – Response, Signal, and Acknowledge messages are queried. <a href="#">Deprecated in JDF 1.2.</a>
<i>JobID</i> ? <a href="#">New in JDF 1.2</a>	string	<i>JobID</i> of the job whose messages are queried/subscribed.
<i>JobPartID</i> ? <a href="#">New in JDF 1.2</a>	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.

Table 5-29: Contents of the *MsgFilter* element

Name	Data Type	Description
<a href="#">QueueEntryID ?</a> <a href="#">New in JDF 1.2</a>	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>MsgFilter</i> applies to all jobs that will be processed by the receiver.
<a href="#">ReceiverURL ?</a>	URL	URL for which the messages are intended.
<a href="#">Part *</a> <a href="#">New in JDF 1.2</a>	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 112.

If the returned list is incomplete because the parameters supplied in the *MsgFilter* element cannot be fulfilled by the application, the *ReturnCode* may be 108 (empty list) or 109 (incomplete list) and should be flagged as a warning.

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

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer" TimeStamp="2000-07-25T12:32:48+02:00" Version="1.2">
  <Response ID="RepMsg" Type="RepeatMessages">
    <Response ID="R1" Time="2000-06-14T11:00+02:00" Type="Status"/>
    <Response ID="R2" Time="2000-06-14T10:50+02:00" Type="Occupation"/>
    <Signal ID="R3" Time="2000-06-14T08:20+02:00" Type="Resource"/>
    <Signal ID="R4" Time="2000-06-14T03:01+02:00" Type="Notification"/>
  </Response>
</JMF>
```

### 5.5.1.6 StopPersistentChannel

Table 5-30: Contents of the *StopPersistentChannel* message

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

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

#### Structure of the *StopPersChParams* Element

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

Table 5-31: Contents of the *StopPersChParams* element

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



Table 5-31: Contents of the *StopPersChParams* element

Name	Data Type	Description
<a href="#">QueueEntryID</a> ? <a href="#">New in JDF 1.2</a>	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> * <a href="#">New in JDF 1.2</a>	element	<i>Part</i> elements that describe the partition of the job whose messages are suppressed. For details on Node partitions, see “Partial Processing of Nodes with Partitioned Resources” on page 112.

## 5.5.2 Device/Operator Status and Job Progress Messages

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 7.2.50, *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 5-32: *Status and progress messages*

Message type	Family	Description
<a href="#">FlushResources</a> <a href="#">New in JDF 1.2</a>	CRS	Remove temporary resource from a Device.
<a href="#">NewJDF</a> <a href="#">New in JDF 1.2</a>	CQRS	Initiates or reports modifications of JDF nodes.
<a href="#">NodeInfo</a> <a href="#">New in JDF 1.2</a>	CQRS	Initiates or reports modifications of JDF node information, (e.g., scheduling).
<i>Occupation</i>	QRS	Queries the occupation of an employee.
<i>Resource</i>	CQRS	Queries and/or modifies JDF resources that are used by a device, such as device settings, or by a job. This message can also be used to query the level of consumables in a device.
<a href="#">ResourcePull</a> <a href="#">New in JDF 1.2</a>	CR	Creates a new <i>QueueEntry</i> from an already existing <i>QueueEntry</i> and submits it to the queue in order to be executed.
<a href="#">Shutdown</a> <a href="#">New in JDF 1.2</a>	CRS	Shuts down a device.
<i>Status</i>	QRS	Queries the general status of a device, controller or job.
<i>Track</i>	QRS	Queries the location of a given job or job part.
<a href="#">WakeUp</a> <a href="#">New in JDF 1.2</a>	CRS	Wakes up a device that is in standby mode.

### 5.5.2.1 FlushResources

[New in JDF 1.2](#)

Table 5.33: Contents of the FlushResources message

Object Type	Element Name	Description
CommandTypeObj	QueueFilter ?	Defines a filter for the returned Queue element in the FlushResources message.
	FlushResourceParams ?	Defines the resources to be removed.
ResponseTypeObj	FlushedResources ?	This element is a placeholder for future use.
For the definition of the Queue element, see “Queue-Handling Elements” on page 187.		

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

Table 5.34: Contents of the FlushResourceParams element

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

### 5.5.2.2 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. Both sides may initiate synchronization. A query or signal informs a Controller or MIS system that a JDF node has been created. A command initiates a modification.

#### Structure of the NewJDF Query Message

Table 5-35: Contents of the NewJDF query message

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

The NewJDF query is sent to a Device or Controller in order to extract information about previously unknown JDF nodes. For instance, an MIS that has received a JMF with an unknown JobPartID may query the JMF sender about details of the JDF with that JobPartID. When used as a Signal, the Signaling device specifies that it has created a new JDF with the properties defined by IDInfo, for instance when a Workflow Controller has instantiated an abstract ProcessGroup node with new sub-nodes. NewJDF is made selective by specifying a NewJDFDQuParams element.

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

## Structure of the NewJDFQuParams Element

Table 5-36: Contents of the NewJDFQuParams element

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

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

## Structure of the NewJDF Command Message

Table 5-37: Contents of the 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.

## Structure of the NewJDFCmdParams Element

Table 5-38: Contents of the NewJDFCmdParams element

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

## Structure of the IDInfo Element

Table 5-39: Contents of the IDInfo element

Name	Data Type	Description
<i>Category</i> ?	NMTOKEN	<b>JDF/@Category</b> of the JDF node.
<i>JobID</i> ?	string	Job ID of the JDF node.
<i>JobPartID</i> ?	string	Job part ID of the JDF node.
<i>ParentJobID</i> ?	string	<i>JobID</i> of the parent node of the JDF node. If not specified, it defaults to the value of <i>JobID</i> .
<i>ParentJobPartID</i> ?	string	Job part ID of the parent node of the JDF node.
<i>Type</i> ?	NMTOKEN	<b>JDF/@Type</b> of the JDF node.
<i>Types</i> ?	NMTOKENS	<b>JDF/@Types</b> of the JDF node.
<b>Device</b> ?	element	Description of the Device that the JDF is targeted for
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.

### 5.5.2.3 NodeInfo

#### [New in JDF 1.2](#)

The **NodeInfo** message can be used as a command or a query to modify or to query JDF **NodeInfo** elements. The query simply retrieves information about the **NodeInfo** without modifying it, while the command modifies those settings within the **NodeInfo** that is specified. Settings that are not specified remain unchanged.

#### Structure of the NodeInfo Query Message

Table 5-40: Contents of the NodeInfo query message

Object Type	Element Name	Description
QueryTypeObj	NodeInfoQuParams	Specifies the node queried.
ResponseTypeObj	NodeInfoResp *	Details of the NodeInfo elements

The **NodeInfo** query is made selective by specifying a **NodeInfoQuParams** element. The presence of the *JobID* attribute determines whether global device resources or job-related resources are returned. The query response returns a list of **NodeInfoResp** elements that contains the queried information concerning the resources. If the list is empty because the selective query parameters of the **NodeInfoQuParams** lead to a null selection, then the *ReturnCode* may be 103 (*JobID* unknown), 104 (*JobPartID* unknown), or 108 (empty list) and should be flagged as a warning.

#### Structure of the NodeInfoQuParams Element

Table 5-41: Contents of the 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, <i>ResourceQuParams</i> applies to all jobs.
<i>Part</i> *	element	<i>Part</i> elements that describe the partition of the job whose <b>NodeInfo</b> is modified. For details on Node partitions, see “Partial Processing of Nodes with Partitioned Resources” on page 112.

#### Structure of the NodeInfo Command Message

Table 5-42: Contents of the NodeInfo Command Message

Object Type	Element name	Description
CommandTypeObj	NodeInfoCmdParams	Specifies the <b>NodeInfo</b> elements to be modified.
ResponseTypeObj	NodeInfoResp *	Contains information about the <b>NodeInfo</b> and the <b>NodeInfo</b> after modification.

The **NodeInfo** command may be 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 presence of the *JobID* attribute determines whether global device resources or job-related resources are modified.

The **Response** contains a list of **NodeInfoResp** elements with a copy of **NodeInfo** after the changes have been applied. If the **NodeInfo** command was successful, the value of the *ReturnCode* attribute is “0”. If it is not successful, the value of *ReturnCode* may be one of those that have been described above in the section about the **NodeInfo** query message, “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.

## Structure of the NodeInfoCmdParams Element

Table 5-43: Contents of the 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, <i>NodeInfoCmdParams</i> applies to all jobs.
<i>UpdateMethod</i> = "Complete"	enumeration	Method how <i>NodeInfo</i> is applied to the JDF. Values are: <i>Complete</i> – The <i>NodeInfo</i> in the JDF is completely overwritten by <i>NodeInfo</i> in this message. <i>Incremental</i> – The <i>NodeInfo</i> in the JDF is incrementally updated by the values that are explicitly set in <i>NodeInfo</i> in this message.
<i>Part</i> *	element	<i>Part</i> elements that describe the partition of the job whose <i>NodeInfo</i> is modified. For details on Node partitions, see “Partial Processing of Nodes with Partitioned Resources” on page 112.
<i>NodeInfo</i> ?	element	<i>NodeInfo</i> to be uploaded to the Device.

## Structure of the NodeInfoResp Element

Table 5-44: Contents of the NodeInfoResp element

Name	Data Type	Description
<i>JobID</i>	string	Job ID of the JDF node that is being modified.
<i>JobPartID</i> ?	string	Job part ID of the JDF node that is being modified.
<i>QueueEntryID</i> ?	string	<i>QueueEntryID</i> of the job that is currently being executed. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> , and <i>Part</i> are ignored. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> , or <i>QueueEntryID</i> are specified, <i>NodeInfoResp</i> applies to all jobs.
<i>Part</i> *	element	<i>Part</i> elements that describe the partition of the job <i>NodeInfo</i> is modified. For details on Node partitions, see “Partial Processing of Nodes with Partitioned Resources” on page 112.
<i>NodeInfo</i> ?	element	<i>NodeInfo</i> after uploading to the controller.

The following is an example for retrieving *NodeInfo* settings:

```
<Query ID="Q1" Type="NodeInfo">
  <NodeInfoQuParams JobID="J1"/>
</Query>
```

The following is a possible response to the query above:

```
<Response ID="M1" Type="NodeInfo" refID="Q1">
  <NodeInfoResp JobID="J1" JobPartID="P1">
    <NodeInfo/>
  </NodeInfoResp>
  <NodeInfoResp JobID="J1" JobPartID="P2">
    <NodeInfo/>
  </NodeInfoResp>
</Response>
```

### 5.5.2.4 Occupation

Table 5-45: Contents of the Occupation message

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

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

#### Structure of the EmployeeDef Element

The Occupation query may be focused to certain employees specifying a EmployeeDef element. If no EmployeeDef element is specified, a list of all known employees is returned.

Table 5-46: Contents of the EmployeeDef element

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

#### Structure of the Occupation Element

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 5-47: Contents of the 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 should not exceed 100.
<i>JobID</i> ?	string	<i>JobID</i> of the JDF node that the employee is assigned to. If no <i>JobID</i> is specified but devices are, the employee is performing tasks not related to a job.
<i>JobPartID</i> ?	string	Job part ID of the JDF node that is currently being executed.
<i>QueueEntryID</i> ? <a href="#">New in JDF 1.2</a>	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, Occupation applies to all jobs.
<b>Device</b> *	element	Devices that the employee is currently assigned to.
<b>Employee</b>	element	Description of the employee being tracked.
<b>Part</b> * <a href="#">New in JDF 1.2</a>	element	<b>Part</b> 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 112

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

```
<Response ID="M1" Type="Occupation" refID="Q1">
  <!--Two jobs on one device with one operator-->
  <Occupation Busy="30" JobID="J1">
    <Employee PersonalID="P1234"/>
    <Device DeviceID="Press1"/>
  </Occupation>
  <Occupation Busy="70" JobID="J2">
    <Employee PersonalID="P1234"/>
    <Device DeviceID="Press1"/>
  </Occupation>
  <!--Another operator on job j2 -->
  <Occupation Busy="50" JobID="J2">
```

```

    <Employee PersonalID="P4321"/>
    <Device DeviceID="Press1"/>
  </Occupation>
  <!--No Job context -->
  <Occupation Busy="0">
    <Device DeviceID="Press2"/>
    <Employee PersonalID="P5678"/>
  </Occupation>
</Response>

```

### 5.5.2.5 Resource

The **Resource** message can be used as a command or a query to modify or to query JDF resources. In both cases (query and command), it is possible to address either global device resources, such as device settings, or job-specific resources. The query simply retrieves information about the resources without modifying them, while the command modifies those settings within the resource that are specified. Settings that are not specified remain unchanged.

#### Structure of the Resource Query Message

Table 5-48: Contents of the Resource query message

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

The **Resource** query may be made selective by specifying a **ResourceQuParams** element. The presence of the **JobID** attribute determines whether global device resources or job-related resources are returned. If no **ResourceQuParams** element is specified, only the global device resources are returned.

The query **Response** returns a list of **ResourceInfo** elements that contains the queried information concerning the resources. If the list is empty because the selective query parameters of the **ResourceQuParams** lead to a null selection of the known device/job resources, then the **ReturnCode** may be 103 (JobID unknown), 104 (Job-PartID unknown), or 108 (empty list) and should be flagged as a warning.

#### Structure of the ResourceQuParams Element

Table 5-49: Contents of the ResourceQuParams element

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. If <i>Classes</i> is not used or empty, then all classes known shall be queried. For possible resource class names, see the <i>Class</i> attribute in Table 3-13, "Contents of the abstract Resource element," on page 53.
<i>Exact</i> = "false"	boolean	Requests an exact description of the JDF resource. If <i>true</i> , the response should also return the requested JDF resource.
<i>JobID</i> ?	string	Job ID of the JDF node that is being queried. If no <i>JobID</i> is specified, global device settings are queried.
<i>JobPartID</i> ?	string	Job part ID of the JDF node that is being queried.
<i>Location</i> ?	string	Identifies the location of a resource, such as paper tray, ink container, or thread holder. The name is the same name used in the Partition-key <i>Location</i> of distributed resources (see also Section 3.8.2.6, Locations of Physical Resources). If not specified, the location will be selected by the device.

Table 5-49: Contents of the ResourceQuParams element

Name	Data Type	Description
<i>ProcessUsage</i> ?	string	Selects a resource in which the value of the <i>ProcessUsage</i> attribute of the resource link (see Table 3-19, “Contents of the abstract ResourceLink element,” on page 64) 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 <b>Component</b> input <b>ExposedMedia</b> of a <b>ConventionalPrinting</b> process can be distinguished by specifying <i>ProcessUsage</i> = “Plate” and <i>ProcessUsage</i> = “Proof”, respectively.  The <i>ResourceName</i> , <i>Usage</i> , and <i>ProcessUsage</i> attributes are combined by a logical AND conjunction to select the resource to be queried.
<i>ProductID</i> ? <a href="#">New in JDF 1.2</a>	string	<i>ProductID</i> of the resource that is queried.
<i>QueueEntryID</i> ? <a href="#">New in JDF 1.2</a>	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.
<i>ResourceName</i> ?	NMTOKEN	Name of the resource being queried. For possible resource names, see titles in Section 7, Resources.
<i>Usage</i> ?	enumeration	<i>Input</i> – The resource is an input. <i>Output</i> – The resource is an output.  Selects a resource in which the value of the <i>Usage</i> attribute of the resource link (see Table 3-19, “Contents of the abstract ResourceLink element,” on page 64) matches the token specified here in this attribute. Only necessary if a resource name is used both as input and output by one node.
<i>Part</i> * <a href="#">New in JDF 1.2</a>	element	<i>Part</i> elements that describe the resource whose messages are queried.

### Structure of the Resource Command Message

Table 5-50: Contents of the Resource command message

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

The **Resource** command may be used to modify either global device settings or a running job. It may be made selective by specifying the optional attributes in the **ResourceCmdParams** element. The presence of the *JobID* attribute determines whether global device resources or job-related resources are modified.

The **Response** contains a list of **ResourceInfo** elements with all resources and private extensions of the device after the changes have been applied. The type of the resource that is given as a response depends on the type of the resource given in the command.

If the **Resource** command was successful, the value of the *ReturnCode* attribute is “0”. If it is not successful, the value of *ReturnCode* may be one of those that have been described above in the section about the **Resource** query message, “200” (invalid resource parameters), or “201” (insufficient resource parameters). Partial application of the resource 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 resource in order to find the setting that could not be applied.



## Structure of the ResourceCmdParams Element

Table 5-51: Contents of the ResourceCmdParams element

Name	Data Type	Description
<b>Activation =</b> <i>"Active"</i> <a href="#">New in JDF 1.1</a>	enumeration	<p>Describes the activation status of the uploaded resource. Allows for a range of activity, including deactivation and test running. Possible values, in order of involvement from least to most active, are:</p> <p><i>Held</i> – Used for uploading a resource that requires operator intervention before being applied.</p> <p><i>TestRun</i> – Used for a test run check by the controller or a device. This does not imply that the update should be automatically applied when the check is completed.</p> <p><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.</p> <p><i>Active</i> – The update must be applied immediately.</p> <p>Note that the <i>Inactive</i> value defined in <i>JDF/@Activation</i> is not a valid value in this list.</p>
<b>Exact = "false"</b>	boolean	Requests an exact description of the JDF resource. If <i>"true"</i> , the response should also return the requested JDF resource.
<b>JobID ?</b>	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.
<b>JobPartID ?</b>	string	<i>JobPartID</i> of the JDF node that the resource being modified is linked to.
<b>ProcessUsage ?</b>	NMTOKEN	<p>Selects a resource in which the value of the <i>ProcessUsage</i> attribute of the resource link (see Table 3-19, "Contents of the abstract ResourceLink element," on page 64) matches the token specified here in this attribute.</p> <p>Only necessary if a resource name is used more than once by one node. For example, the <b>ExposedMedia</b> input resources of a <b>ConventionalPrinting</b> process can be distinguished by specifying <i>ProcessUsage = Plate</i> and <i>ProcessUsage = Proof</i>, respectively.</p> <p>The <i>ResourceName</i> and <i>ProcessUsage</i> attributes are combined by a logical AND conjunction to select the resource to be queried.</p>
<b>ProductID ?</b> <a href="#">New in JDF 1.2</a>	string	<i>ProductID</i> of the resource that is updated.
<b>ProductionAmount ?</b>	double	New amount of resource production. This value replaces the <i>Amount</i> in the resource link of the resource specified by <i>ResourceName</i> .
<b>QueueEntryID ?</b> <a href="#">New in JDF 1.2</a>	string	<i>QueueEntryID</i> of the job that is currently being executed. If <i>QueueEntryID</i> is specified, <i>JobID</i> , <i>JobPartID</i> , and <i>Part</i> are ignored. If none of <i>JobID</i> , <i>JobPartID</i> , <i>Part</i> , or <i>QueueEntryID</i> are specified, ResourceCmdParams applies to all jobs.
<b>ResourceName ?</b>	NMTOKEN	Name of the resource whose production amount will be modified. For possible resource names see titles in Section 7, Resources.
<b>Status ?</b> <a href="#">New in JDF 1.2</a>	enumeration	Updated <i>Status</i> of the selected resource. The list of possible values is defined in Table 3-13, "Contents of the abstract Resource element," on page 53.

Table 5-51: Contents of the ResourceCmdParams element

Name	Data Type	Description
<a href="#">UpdateIDs ?</a> <a href="#">New in JDF 1.1</a>	NMTOKENS	The <i>UpdateID</i> attributes of one or more ResourceUpdate that are defined in resources known to the recipient. The data type is NMTOKENS and not IDREFS because no matching IDs exist within this message. The order of tokens in defines the order in which the updates are applied.
<a href="#">MISDetails ?</a> <a href="#">New in JDF 1.2</a>	refelement	Definition how the costs for the production of the Resource are to be charged.
<a href="#">Part *</a> <a href="#">New in JDF 1.2</a>	element	Part elements that describe the partitions of the resource that is being modified. If not specified, the entire resource is selected.
<a href="#">Resource *</a>	element	Resources to be uploaded to the Device. They completely replace the original resources with the same ID.  The resources to be modified are identified by their <i>ID</i> values, which means that the <i>ID</i> attributes must be known to the controller that issued the Resource command.

### Structure of the ResourceInfo Element

Table 5-52: Contents of the ResourceInfo element

Name	Data Type	Description
<a href="#">ActualAmount ?</a> <a href="#">New in JDF 1.2</a>	double	Reflects the current accumulated amount of the resource that has been consumed (input) or produced (output) by the process. This corresponds to the value of the <i>ActualAmount</i> attribute in the corresponding resource link of the resource were it to be written now.
<a href="#">Amount ?</a>	double	Reflects the intended accumulated amount of the resource that should be consumed (input) or produced (output) by the process. This corresponds to the value of the <i>Amount</i> attribute in the corresponding resource link of the resource were it to be written now.
<a href="#">AvailableAmount ?</a>	double	Device-specific amount of the <i>Consumable</i> resource that is available in the device.
<a href="#">Level = "OK"</a>	enumeration	This attribute is device dependent. A device may specify the level status that describes a low or empty consumable level. Possible values are: <i>Empty</i> – Specification is left to the device manufacturer. <i>Low</i> – Specification is left to the device manufacturer. <i>OK</i> – Specification is left to the device manufacturer.
<a href="#">Location ?</a>	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 3.8.2.6, Locations of Physical Resources). If not specified, the location will be defined by the device.
<a href="#">ResourceName ?</a>	NMTOKEN	Name of the resource if <i>Exact = false</i> in the query. Only one of Resource or <i>ResourceName</i> must be specified.

Table 5-52: Contents of the ResourceInfo element

Name	Data Type	Description
<i>ProcessUsage</i> ?	NMTOKEN	Selects a resource in which the value of the <i>ProcessUsage</i> attribute of the resource link (see Table 3-19, "Contents of the abstract ResourceLink element," on page 64) 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 <b>ExposedMedia</b> input resources of a <b>ConventionalPrinting</b> process can be distinguished by specifying <i>ProcessUsage</i> = <i>Proof</i> and <i>ProcessUsage</i> = <i>Plate</i> , respectively. The <i>ResourceName</i> and <i>ProcessUsage</i> attributes are combined by a logical AND conjunction to select the resource to be queried.
<i>ProductID</i> ? <a href="#">New in JDF 1.2</a>	string	<i>ProductID</i> of the resource.
<i>Status</i> ? <a href="#">New in JDF 1.2</a>	enumeration	Updated <i>Status</i> of the selected resource. The list of possible values is defined in Table 3-13, "Contents of the abstract Resource element," on page 53.
<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 may be necessary due to technical considerations, such as when ink is specified in weight (g) and ink measurement is specified in volume (liter).
<i>CostCenter</i> ?	element	Cost center to which the resource consumption is allocated.
<i>MISDetails</i> ? <a href="#">New in JDF 1.2</a>	refelement	Definition how the costs for the production of the Resource are to be charged.
<i>Part</i> * <a href="#">New in JDF 1.2</a>	element	<i>Part</i> elements that describe the resource.
<i>Resource</i> ?	element	JDF description of the resource. If the query or command leading to this response element contains <i>Part</i> elements, the resource must only contain the appropriate matching partitions.

The following is an example for retrieving settings:

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

The following is a possible response to the query above:

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

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

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

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

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

### 5.5.2.6 ResourcePull

[New in JDF 1.2](#)

Table 5.53: Contents of the ResourcePull message

Object Type	Element	Description
CommandTypeObj	ResourcePullParams	Defines the parameters of the repeated job.
	QueueFilter	Defines a filter for the returned Queue element in the ResourcePull message.
ResponseTypeObj	QueueEntry	Provides the queue entry of the repeated job.
	Queue	Describes the state of the queue after the command has been executed.
Definition of the QueueEntry and Queue elements, see “Queue-Handling Elements” on page 187.		

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 required parameters. It may be used to regenerate the output of a QueueEntry or JDF node with any *Status*.

#### Workflow Integration with ResourcePull

When ResourcePull is submitted directly to a Device in a workflow that is monitored by an MIS system, the MIS system must be informed about the re-execution of the JDF node, so that it can update the state of the entire job appropriately.

**Note:** It is preferred to pull a resource from a Device in a workflow that is monitored by an MIS system by sending the ResourcePull message to the MIS. The MIS can then control the Device in the standard manner and also maintain consistency of its internal job representation.

#### Structure of the ResourcePullParams Element

The ResourcePullParams may contain queue-ordering attributes equivalent to those used by the SetQueueEntryPriority, and SetQueueEntryPosition messages. The optional list of Part elements refers to the output resource that is produced by the JDF node.

For example, if an **ImageSetting** process produces a partitioned set of plates, the following example message would request only the yellow plate of the *Front Surface* of *Sheet1*.

```
<Command ID="C2" Type="ResourcePull">
  <ResourcePullParams Priority="100" QueueEntryID="AllPlates">
    <Part Separation="Yellow" SheetName="Sheet1" Side="Front"/>
  </ResourcePullParams>
</Command>
```

Table 5.54: Contents of the ResourcePullParams element

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</i> = "false"	boolean	If "true", the entry is submitted as held.

Table 5.54: Contents of the ResourcePullParams element

Name	Data Type	Description
<i>NextQueueEntryID</i> ?	string	ID of the queue entry that should be ordered directly behind the entry.
<i>PrevQueueEntryID</i> ?	string	ID of the queue entry that should be ordered directly in front of the entry.
<i>JobID</i> ?	string	Job ID of the JDF node that creates the requested resource. If <i>QueueEntryID</i> is specified, <i>JobID</i> is ignored. One of <i>JobID</i> or <i>QueueEntryID</i> must be specified.
<i>Priority</i> = "1"	integer	Number from 0 to 100, where 0 is the lowest priority and 100 is the maximum priority.
<i>QueueEntryID</i> ?	string	<i>QueueEntryID</i> of the JDF node that creates the requested resource. If <i>QueueEntryID</i> is specified, <i>JobID</i> is ignored. One of <i>JobID</i> or <i>QueueEntryID</i> 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). One of: <i>Complete</i> – Restart from the original input resources, if they are available. The process may run based on intermediate resources if any original resources are not available. <i>CompleteOnly</i> – Restart from the original input resources. The process must not run if any original resources are not available. <i>Fast</i> – Reuse as many intermediate resources as possible, (e.g., restart ImageSetting from stored intermediate raster files and do not reRIP, if possible).
<i>ResourceID</i>	string	ID attribute of the Resource requested.
<i>ReturnURL</i> ?	URL	URL where the JDF file should be written when the job is completed or aborted. If not specified, the JDF should be placed in the default output hot folder of the queue controller. <i>ReturnURL</i> takes precedence when <i>NodeInfo/@TargetRoute</i> is specified in the previously submitted JDF.
<i>WatchURL</i> ?	URL	URL of the controller that should be notified when the status of the <i>QueueEntry</i> or the underlying job changes. Specifying <i>WatchURL</i> is equivalent to sending a subscription for an <i>Events</i> message with <i>SignalTypes="All"</i> .
<i>Part</i> *	element	The <i>Part</i> elements identify the parts of a partitioned output resource to be created by the JDF node. The structure of the <i>Part</i> element is defined in Table 3-28, "Contents of the Part element," on page 79. For details on partitioned resources, see Section 3.8.2, Description of Partitionable Resources. For details on Node partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 112.
<i>Disposition</i> ?	element	Definition how long the <i>QueueEntry</i> must be retained in the queue. If not specified, the <i>QueueEntry</i> must be removed from the queue immediately after process completion of the <i>QueueEntry</i> .
<i>MISDetails</i> ?	refelement	Definition how the costs for the production of the <i>Resource</i> are to be charged.

### 5.5.2.7 Shutdown

[New in JDF 1.2](#)

The `ShutDown` message shuts down a controller or device. Note that a Device that signals its own shutdown must use the `Status` message.

Table 5.55: Contents of the `ShutDown` message

Object Type	Element	Description
CommandTypeObj	ShutDownCmdParams	Defines the details of a shutdown.
	QueueFilter	Defines a filter for the returned Queue element in the <code>ShutDown</code> message.
ResponseTypeObj	DeviceInfo	Describes the device status as anticipated after the shutdown.
	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 “Queue-Handling Elements” on page 187s.

#### Structure of the `ShutDownCmdParams` Element

Table 5.56: Contents of the `ShutDownCmdParams` element

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

### 5.5.2.8 Status

Table 5-57: Contents of the `Status` message

Object Type	Element name	Description
QueryTypeObj	StatusQuParams	Refines the query to include various aspects of the device and job states.
ResponseTypeObj <a href="#">Modified in JDF 1.2</a>	DeviceInfo +	Describes the actual device status. If the queue handles multiple devices, one <code>DeviceInfo</code> must be specified for each device.
	Queue ?	Provides information about the queue and all its entries. This element will only be provided if the device has queue capabilities. The Queue element is described in Section 5.6.5, Queue-Handling Elements.

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 required 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.

#### Structure of the `StatusQuParams` Element

The various aspects of the device, queue, and job states may be 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 should 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 may 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 619).

Table 5-58: Contents of the *StatusQuParams* element

Name	Data Type	Description
<i>DeviceDetails</i> = "None"	enumeration	Refines the provided status information about the device. Possible values are: <i>None</i> <i>Brief</i> – Provide all available device information except for <i>Device</i> elements. <i>Modules</i> – <i>ModuleStatus</i> elements should 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 <i>Device</i> elements which represent details of the device. <i>Capability</i> – Provide <i>Device</i> 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 <i>Device</i> elements which represent details of the device.
<i>EmployeeInfo</i> = "false"	boolean	If <i>true</i> , <i>Employee</i> elements may be provided in the response. Those elements describe the employees which are associated to the device independent on any job.
<i>JobDetails</i> = "None" <a href="#">Modified in JDF 1.2</a>	enumeration	Refines the provided status information about the jobs associated with the device. Each higher entry includes the values specified in the lower entries. Possible values are: <i>None</i> – Specify only <i>JobID</i> , <i>JobPartID</i> and <i>Amount</i> , and/or <i>PercentCompleted</i> . <i>MIS</i> – Provide business with the relevant information contained in the <i>CostCenter</i> 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> . <a href="#">Deprecated in JDF 1.2</a> <i>Brief</i> – Provide all available status information including <i>JobPhase</i> elements except for JDF. <i>Full</i> – Provide maximum available status information. Includes an actual JDF which represents a snapshot of the current job state.
<i>JobID</i> ?	string	Job ID of the JDF node whose status is being queried. If not specified, list all known jobs.
<i>JobPartID</i> ?	string	JobPart ID of the JDF node whose status is being queried.
<i>QueueEntryID</i> ? <a href="#">New in JDF 1.2</a>	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 may be provided. This is analogous to a <i>QueueStatus</i> query (see Section 5.6.4.6, <i>QueueStatus</i> ).
<i>Part</i> * <a href="#">New in JDF 1.2</a>	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 112.

## Structure of the DeviceInfo Element

The response returns a DeviceInfo element for the queried device.

Table 5-59: Contents of the DeviceInfo element

Name	Data Type	Description
<i>CounterUnit</i> ?	string	The unit of the <i>ProductionCounter</i> , the <i>TotalProductionCounter</i> and numerator unit of <i>Speed</i> . The default unit is the default unit defined by JDF for the output resource of the node executed by the device. For example, in case of a sheet printer, it is the number of sheets; in case of a web printer, it is the length of printed web in meters.
<i>DeviceCondition</i> ? <a href="#">New in JDF 1.2</a>	enumeration	The general condition of a device. <i>OK</i> – The device is in working condition. <i>NeedsAttention</i> – The device is still in working condition but requires attention. <i>Failure</i> – The device is not in working condition. <i>OffLine</i> – The device is off line and its condition is unknown.
<i>DeviceOperationMode</i> ? <a href="#">New in JDF 1.2</a>	enumeration	<i>DeviceOperationMode</i> shows the operation mode that the device is in. It is used to show if the production of a device is aimed at producing good products or not. The latter case applies when a device is used to produce a job for testing, calibration, etc. without the intention to produce good output. <i>Productive</i> – The device is used to produce good product. Any times recorded in this mode should 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 should not be allocated against the job. <i>Maintenance</i> – The device is used without the intention to produce good product, e.g. to perform (preventative) maintenance.
<i>DeviceStatus</i>	enumeration	The status of a device. Possible values are: <i>Unknown</i> – No device is known or the device cannot provide a <i>DeviceStatus</i> . <i>Idle</i> – No job is being processed and the device is accepting new jobs. <i>Down</i> – No job is being processed and the device currently cannot execute a job. The device may 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, but running may be resumed later. This status may indicate any kind of break, including a pause, maintenance, or a breakdown, as long as execution has not been aborted.
<i>HourCounter</i> ?	duration	The total integrated time (life time) of device operation in hours.
<i>PowerOnTime</i> ?	dateTime	Date and time when the device was switched on.
<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 <i>Events</i> message of <i>Type</i> = <i>CounterReset</i> (see “NotificationDetails” on page 621).
<i>Speed</i> ?	double	The current machine speed. <i>Speed</i> is defined in the same units as <i>ProductionCounter</i> / hour.



Table 5-59: Contents of the DeviceInfo element

Name	Data Type	Description
<i>StatusDetails</i> ?	string	String that defines the device state more specifically. For a list of supported values, see “StatusDetails Supported Strings” on page 615.
<i>TotalProductionCounter</i> ?	double	The current total machine production counter since the machine was produced.
<b>Device</b> ?	element	A <b>Device</b> resource that describes details of the device.
<b>Employee</b> *	element	<b>Employee</b> resources that describe which employees are currently working at the device.
<b>JobPhase</b> * <a href="#">Clarified in JDF 1.2</a>	element	Describes the actual status of jobs in the device. All jobs that are active on the device must be specified. Supplying no <b>JobPhase</b> specifies that no job is currently active on the device.  Active jobs have <b>JDF/@Activation=Active</b> , <i>TestRun</i> or <i>TestRunAndGo</i> and <b>JDF/@Status</b> or <b>JDF/StatusPool/PartStatus@Status = TestRunInProgress</b> , <i>Setup</i> , <i>InProgress</i> , <i>Cleanup</i> or <i>Stopped</i> . For details on using <b>JobPhase</b> elements, see Table 5-60 on page 165.
<b>ModuleStatus</b> *	element	Status of individual modules. For details on using <b>ModuleStatus</b> elements, see Table 5-61 on page 167.

### Structure of the JobPhase Element

A **Status** response may provide **JobPhase** elements. The **JobPhase** element represents the actual state of a job. The **JobPhase** element is an analogue to the **PhaseTime** audit element described in Section 3.9.1.3, **PhaseTime**. The main difference between a **JobPhase** element and a **PhaseTime** audit element is that a **JobPhase** message element reflects a snapshot of the current job status whereas the **PhaseTime** audit reflects a time span bordered by two (sub-) status transitions. **JobPhase** elements must not overlap in time.

For exact information about the job phase, a **JobPhase** element may embed a copy of the current state of the job described as JDF. If **Part** elements are specified, all attributes in **JobPhase** applies only to the specified parts. If an actual JDF is not supported by the controller, the same rules apply for the **Status** response as those which apply for the **Resource** response.

Table 5-60: Contents of the JobPhase element

Name	Data Type	Description
<i>Activation</i> ? <a href="#">New in JDF 1.1</a>	enumeration	The activation of the JDF node. Possible values are the same as the possible values of a JDF node’s <i>Activation</i> attribute. For details, see Table 3-4, “Contents of a JDF node,” on page 38.
<i>Amount</i> ?	double	Total <i>Amount</i> that the node defined in this <b>JobPhase</b> 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 <b>DeviceInfo</b> .
<i>Deadline</i> ?	enumeration	Scheduling state of the job. Possible values are: <i>InTime</i> – The job or job part will probably not miss the deadline. <i>Warning</i> – The job or job part could miss the deadline. <i>Late</i> – The job or job part will miss the deadline. For more details on scheduling, see Section 3.4, Node Information in <b>NodeInfo</b> .
<i>JobID</i> ?	string	Job ID of the JDF node that the <b>JobPhase</b> belongs to.
<i>JobPartID</i> ?	string	Job part ID of the JDF node that the <b>JobPhase</b> belongs to.
<i>PercentCompleted</i> ?	double	Node processing progress in percent (%) completed.

Table 5-60: Contents of the JobPhase element

Name	Data Type	Description
<a href="#">PhaseAmount ?</a> <a href="#">New in JDF 1.2</a>	double	Amount that the node defined in this JobPhase produced during this JobPhase. 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 DeviceInfo.
<a href="#">PhaseStartTime ?</a> <a href="#">New in JDF 1.2</a>	dateTime	Time that this JobPhase started.
<a href="#">PhaseWaste ?</a> <a href="#">New in JDF 1.2</a>	double	Amount of waste that the node defined in this JobPhase produced during this JobPhase. The unit is specified in the <i>CounterUnit</i> attribute of the parent element DeviceInfo.
<a href="#">QueueEntryID ?</a>	string	If the job was submitted to a Queue, and the <i>QueueEntryID</i> is known, this attribute should be provided.
<a href="#">RestTime ?</a> <a href="#">New in JDF 1.1</a>	duration	Estimated duration required for finishing of processing of this node.
<a href="#">Speed ?</a>	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 DeviceInfo element.
<a href="#">StartTime ?</a> <a href="#">New in JDF 1.1</a>	dateTime	Time when execution of the node that is described by this JobPhase 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. Possible values are the same as the possible values of a JDF node's <i>Status</i> attribute. For details, see Table 3-4, "Contents of a JDF node," on page 38.
<a href="#">StatusDetails ?</a>	string	String that defines the job state more specifically. For a list of supported values, see "StatusDetails Supported Strings" on page 615.
<a href="#">TotalAmount ?</a> <a href="#">New in JDF 1.1</a>	double	Amount that will be produced when this job phase is 100% completed. The unit is specified in the <i>CounterUnit</i> attribute of the parent element DeviceInfo.
<a href="#">Waste ?</a> <a href="#">New in JDF 1.1</a>	double	Total <i>Amount</i> of waste that the node defined in this JobPhase produced since <i>StartTime</i> . The unit is specified in the <i>CounterUnit</i> attribute of the parent element DeviceInfo.
<a href="#">CostCenter ?</a>	element	The cost center that the job is currently being charged to. Defaults to the cost center specified in the DeviceInfo element.
<a href="#">JDF ?</a>	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. <i>JDF/@Activation</i> must be set to "Informative" in this JDF element.
<a href="#">MISDetails ?</a> <a href="#">New in JDF 1.2</a>	refelement	Definition how the costs for this JobPhase are to be charged.
<a href="#">Part *</a> <a href="#">Modified in JDF 1.1</a>	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 112.

## Structure of the ModuleStatus Element

The ModuleStatus element is identical to the ModulePhase element of the PhaseTime audit element (see Table 3-35, “Contents of the ModulePhase element,” on page 94), except that the attributes *Start* and *End* are missing. These attributes specify the time interval in the audit pendant ModulePhase and the *DeviceID* attribute, which is unnecessary here. The ModuleStatus element is described in the following table.

Table 5-61: Contents of the ModuleStatus element

Name	Data Type	Description
<i>DeviceStatus</i>	enumeration	Status of the module. Possible values are: <i>Unknown</i> – The module status is unknown. <i>Idle</i> – The module is not used. An example is a color print module that is inactive during a black-and-white print. <i>Down</i> – The module cannot be used. It may 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 may be resumed later. This status may indicate any kind of break, including a pause, maintenance, or a breakdown, as long as running can be easily resumed.
<i>ModuleIndex</i>	IntegerRange-List	0-based indices of the module or modules. If multiple module types are available on one machine, indices must also be unique.
<i>ModuleType</i>	NMTOKEN	Module description. The allowed values depend on the type of device that is described. The predefined values are listed in “ModuleType Supported Strings” on page 617.
<i>StatusDetails</i> ?	string	Description of the module status phase that provides details beyond the enumerative values given by the <i>DeviceStatus</i> attribute. For a list of supported values, see “StatusDetails Supported Strings” on page 615.
<b>Employee</b> *	element	Links to <b>Employee</b> resources that are working at this module (the module is specified by the attributes <i>ModuleIndex</i> and <i>ModuleType</i> ).

The following is an example of a response to a **Status** query. The device in this example holds one job and executes another job that is currently printed duplex (each side) on four-color modules for the front and three-color modules for the back, with one idle:

```
<Response ID="M1" Type="Status" refID="Q1">
  <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 DeviceStatus="Running" ModuleIndex="0~3 6~8"
ModuleType="PrintModule"/>
    <ModuleStatus DeviceStatus="Idle" ModuleIndex="4" ModuleType="PrintModule"/>
    <ModuleStatus DeviceStatus="Running" ModuleIndex="5"
ModuleType="PerfectingModule"/>
  </DeviceInfo>
</Response>
```

### 5.5.2.9 Track

Table 5-62: Contents of the Track message

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

The Track query requests information about the location of Jobs that are known by a controller. If a high level controller controls lower level controllers, it should also list the jobs that are controlled by these. The response is a list of TrackResult elements.

#### Structure of the TrackFilter Element

The TrackFilter element refines the list of TrackResults that should be returned. Only jobs that match all parameters specified are included.

Table 5-63: Contents of the TrackFilter element

Name	Data Type	Description
<i>JobID</i> ?	string	<i>JobID</i> of the JDF node that is being tracked. Defaults to list Job-Phase elements of all known nodes.
<i>JobPartID</i> ?	string	<i>JobPartID</i> of the JDF node that is being tracked.
<i>ProjectID</i> ? <a href="#">New in JDF 1.2</a>	string	<i>ProjectID</i> of the JDF node that is being tracked.
<i>QueueEntryID</i> ? <a href="#">New in JDF 1.2</a>	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 <b>JDF/@Status</b> of the jobs being tracked. Possible values are a combination of any of the possible values of a JDF node's <i>Status</i> attribute. When not known or specified, all enumerations are applicable. Possible values are: <i>Waiting</i> <i>Ready</i> <i>FailedTestRun</i> <i>Setup</i> <i>InProgress</i> <i>Cleanup</i> <i>Spawned</i> <i>Stopped</i> <i>Completed</i> <i>Aborted</i> For details, see Table 3-4, "Contents of a JDF node," on page 38.
<i>Part</i> * <a href="#">New in JDF 1.2</a>	element	<i>Part</i> elements that describe the partition of the job that is being tracked. For details on Node partitions, see "Partial Processing of Nodes with Partitioned Resources" on page 112.

## Structure of the TrackResult Element

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

Table 5-64: Contents of the 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> ? <a href="#">New in JDF 1.2</a>	string	<i>ProjectID</i> of the highest level node of the JDF node that is being tracked.
<i>QueueEntryID</i> ? <a href="#">New in JDF 1.2</a>	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 may be queried for details of the job.
<i>Part</i> * <a href="#">New in JDF 1.2</a>	element	<i>Part</i> elements that describe the partition of the job that is being tracked. For details on Node partitions, see “Partial Processing of Nodes with Partitioned Resources” on page 112.

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

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

### 5.5.2.10 WakeUp

[New in JDF 1.2](#)

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

Table 5.65: Contents of the 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.

## Structure of the WakeUpCmdParams Element

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

Table 5.66: Contents of the WakeUpCmdParams element

Name	Data Type	Description
—	—	—

### 5.5.3 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 5-67: Dynamic pipe messages

Message type	Family	Description
PipeClose	CR	Closes a pipe because no further resources are required. This is typically used to terminate the producing process.
PipePull	CR	Requests a new resource from a pipe.
PipePush	CR	Notifies that a new resource is available in a pipe.
PipePause	CR	Pauses a process if no further resources can be consumed or produced.

#### 5.5.3.1 PipeClose

Table 5-68: Contents of the PipeClose message

Object Type	Element name	Description
CommandTypeObj	PipeParams	Describes the pipe resource. The PipeParams element is described in Section 5.5.3.2, PipePull.
ResponseTypeObj	JobPhase	The status of the responding process. The JobPhase element is defined in Table 5-60 on page 165.

The PipeClose message notifies the process at the other end of a dynamic pipe that the sender of this message needs no further resources or will produce no further resources through the pipe. The PipeClose command response is equivalent to the PipePull and PipePush command responses described below.

#### 5.5.3.2 PipePull

Table 5-69: Contents of the 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 5-60 on page 165.

The PipePull message requests resources that are described in a JDF dynamic pipe (see Section 3.6.3, Pipe Resources and Section 4.3.3, Overlapping Processing Using Pipes). PipePull messages are the JMF equivalent of a dynamic input resource link, below, depicts the mode of operation of a PipePull message.

The PipePull command response returns a *ReturnCode* of 0 if the command has been accepted by the receiving controller. If not successful the *ReturnCode* may be one of the codes presented in Supported Error Codes in JMF and Notification elements. The Response may contain a Notification element. The JobPhase element (see Section 5.5.2.8, Status) returned should provide only the *Status* attribute that describes the job status of the responding process after receiving the command.

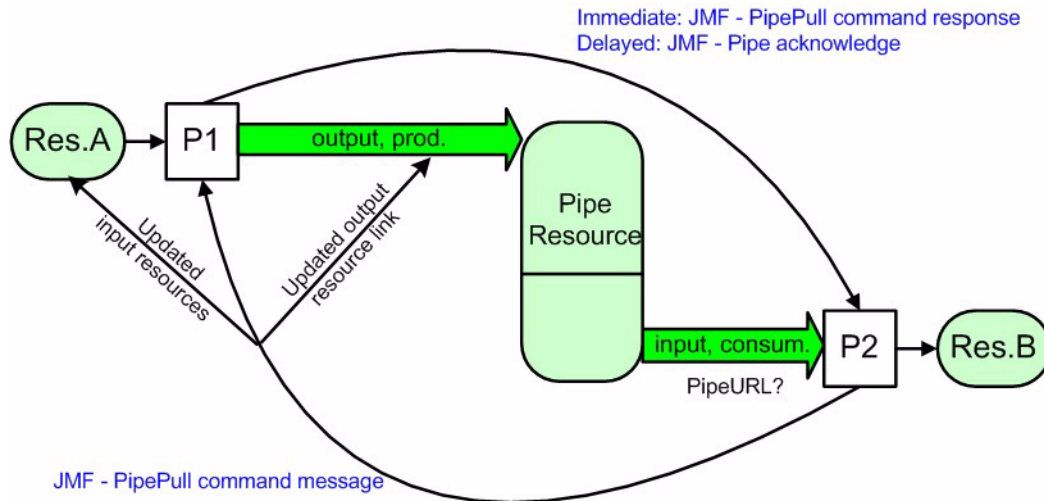


Figure 5.4: Mechanism of a PipePull message

### Structure of the PipeParams Element

The PipeParams element is also used by the messages PipeClose, PipePush, and PipePause. The URL where an optional Acknowledge should be sent when the pipe command has been executed may be 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 5-70: Contents of the PipeParams element

Name	Data Type	Description
<i>JobID</i> ? <a href="#">New in JDF 1.2</a>	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> ? <a href="#">New in JDF 1.2</a>	string	Specifies the <i>JobPartID</i> of the node at the receiving end of the message that links to the resource specified in <i>PipeID</i> .
<i>PipeID</i>	string	Pipe ID of the JDF resource that defines the dynamic pipe.
<i>Status</i> = "InProgress"	enumeration	Process status after the request. Possible values are defined in Table 3-4, "Contents of a JDF node," on page 38.
<i>UpdatedStatus</i> ?	enumeration	This value represents the actual status of the pipe resource and may be used by the receiving process for process termination control. For details see Section 4.3.5.2, Formal Iterative Processing. For possible values of the resource <i>Status</i> attribute see Table 3-13, "Contents of the abstract Resource element," on page 53.

Table 5-70: Contents of the PipeParams element

Name	Data Type	Description
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, the resources are ignored.
ResourceLink ?	element	Updated resource link to the pipe resource: PipePull (it is an output link), PipePush (it is an input link), and PipePause (depends on the pipe end). This resource link may be used by the process that links to the pipe resource.  The attributes <i>rRef</i> and <i>Usage</i> of a ResourceLink must not be modified by the agent that sends the Pipe message because these attributes are used by the JMF receiver to identify the ResourceLink that is to be modified. For details see Section 3.6.4, ResourceUpdate Elements. In the context of dynamic pipes these two attributes have no meaning.  In case of the PipeClose command, the resource link is ignored.

### 5.5.3.3 PipePush

Table 5-71: Contents of the PipePush message

Object Type	Element name	Description
CommandTypeObj	PipeParams	Describes the produced pipe resource. The PipeParams element is described in Section 5.5.3.2, PipePull.
ResponseTypeObj	JobPhase	The status of the responding process. The JobPhase element is defined in Table 5-60 on page 165.

The PipePush message notifies the availability of pipe resources that are described in a JDF dynamic pipe (see Section 3.6.3, Pipe Resources and Section 4.3.3, Overlapping Processing Using Pipes). PipePush messages are the JMF equivalent of a dynamic output resource link. Figure 5.5 depicts the mode of operation of a PipePush message. The PipePush command response is equivalent to the PipePull command response described above.

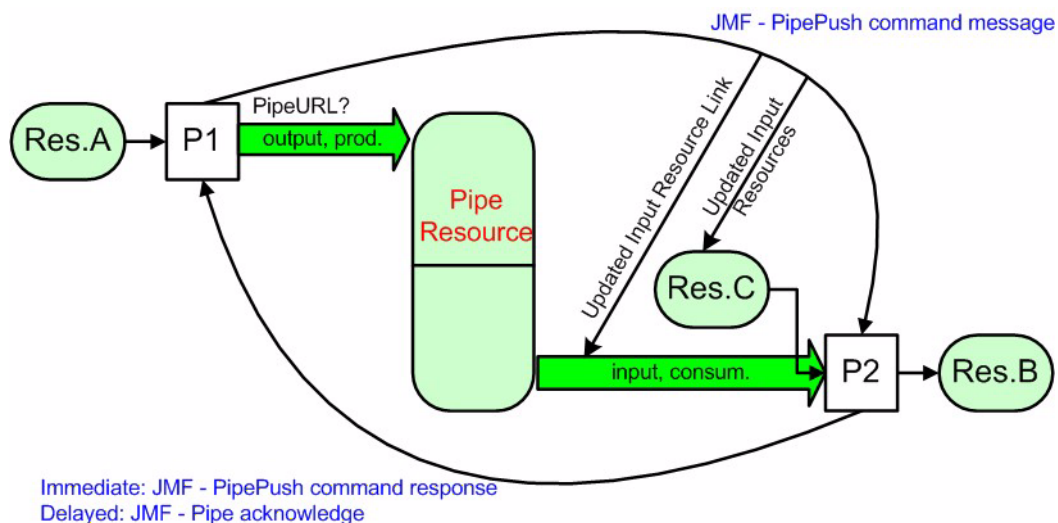


Figure 5.5: Mechanism of a PipePush message



### 5.5.3.4 PipePause

Table 5-72: Contents of the PipePause message

Object Type	Element name	Description
CommandTypeObj	PipeParams	Describes the pipe resource. The PipeParams element is described in Section 5.5.3.2, PipePull.
ResponseTypeObj	JobPhase	The status of the responding process. The JobPhase element is defined in Table 5-60 on page 165.

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

## 5.6 Queue Support

In JMF, a device is assumed to have one input queue that accepts submitted jobs. If a real device supports multiple queues, it is represented by multiple logical devices in JDF. 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 may 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 queue controller may decide to 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 5.5, Standard Messages. 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).

### 5.6.1 Queue Entry ID Generation

Queue entries are accessed using a *QueueEntryID* attribute, which is generated by the controller of the queue when the job is submitted and is returned in the SubmitQueueEntry response. This attribute must uniquely identify an entry within the scope of one queue. An implementation is free to choose the algorithm that generates *QueueEntryIDs*.

### 5.6.2 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 “Contents of the QueueFilter Element” on page 189) 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.

### 5.6.3 Queue Entry Handling Commands

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 5-73: QueueEntry handling messages

Message type	Family	Description
<a href="#">AbortQueueEntry</a> <a href="#">Modified in JDF 1.2</a>	CR	The QueueEntry is aborted and remains in the Queue with QueueEntry/@Status="Aborted".
HoldQueueEntry	CR	The entry remains in queue but is not executed until a Resume-QueueEntry command is received.
RemoveQueueEntry	CR	A job is removed from the queue.
<a href="#">RequestQueueEntry</a> <a href="#">New in JDF 1.2</a>	CR	A new job is requested by the device. This message is used to signal that a Device has processing resources available.
ResubmitQueueEntry	CR	Replaces a queue entry without affecting the entry's parameters. The command is used, for example, for late changes to a submitted JDF.
ResumeQueueEntry	CR	A held job is resumed. The job is re-queued at the position defined by its current priority. Submission time is set to the current time stamp.
<a href="#">ReturnQueueEntry</a> <a href="#">New in JDF 1.2</a>	CR	Returns a job that had been submitted with a SubmitQueueEntry to the queue that represents the controller that originally submitted the job.
SetQueueEntryPosition	CR	Queues a job behind a given position n, where n represents a numerical value. "0" = pole position. Priority is set to the priority of the job at position n.
SetQueueEntryPriority	CR	Sets the priority of a queued job to a new value. This does not apply to jobs that are already running.
SubmitQueueEntry	CR	A job is submitted to a queue in order to be executed.
<a href="#">SuspendQueueEntry</a> <a href="#">New in JDF 1.2</a>	CR	The entry is suspended if it is already running. It remains suspended until a ResumeQueueEntry command is received.

The following table specifies the status transitions for the respective queue entry handling messages. The error(n) indicates the ReturnCode which is returned on an illegal Status transition and the queue entry Status is unchanged. For details on error codes, see "Supported Error Codes in JMF and Notification elements" on page 619.

Table 5-74: Status transitions for QueueEntry handling messages

Old Status Message type	Non existent	Waiting	Held	Running	Suspended	Completed	Aborted
AbortQueueEntry	error(105)	Aborted	Aborted	Aborted	Aborted	error(114)	error(113)
HoldQueueEntry	error(105)	Held	error(113)	error(106)	error(106)	error(114)	error(114)
RemoveQueueEntry	error(105)	Removed	Removed	error(106)	error(106)	Removed	Removed
RequestQueueEntry	RequestQueueEntry is emitted by the controller of the queue and not sent to the queue. Therefore it is not applicable in this section.						
ResubmitQueueEntry	error(105)	Waiting	Held	error(107)	error(107)	error(114)	error(114)
ResumeQueueEntry	error(105)	error(113)	Waiting	error(113)	Running	error(114)	error(114)
ReturnQueueEntry	ReturnQueueEntry is emitted by the controller of the queue and not sent to the queue. Therefore it is not applicable in this section.						
SetQueueEntryPosition	error(105)	Waiting	Held	error(107)	error(107)	error(114)	error(114)
SetQueueEntryPriority	error(105)	Waiting	Held	error(107)	error(107)	error(114)	error(114)

Table 5-74: Status transitions for QueueEntry handling messages

Old Status Message type	Non existent	Waiting	Held	Running	Suspended	Completed	Aborted
SubmitQueueEntry	Waiting, Held, Running	A new QueueEntryID is generated by the queue owner on submission. Therefore these states are not applicable.					
SuspendQueueEntry	error(105)	error(115)	error(115)	Suspended	error(113)	error(114)	error(114)

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

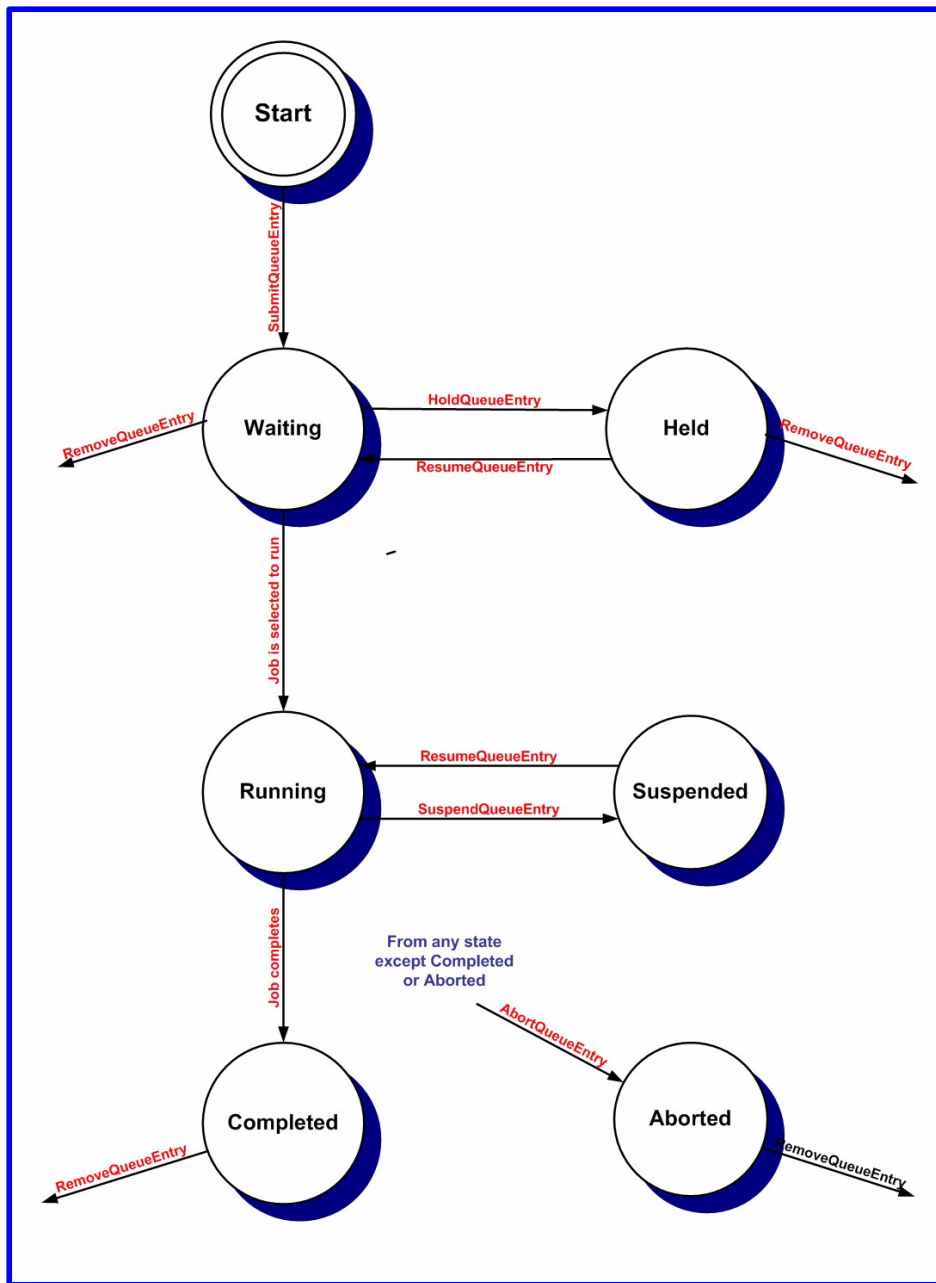


Figure 5.6: JMF QueueEntry Status Transition Diagram

### 5.6.3.1 AbortQueueEntry

[Modified in JDF 1.2](#)

Table 5-75: Contents of the AbortQueueEntry message

Object Type	Element name	Description
CommandTypeObj <a href="#">Modified in JDF 1.2</a>	QueueEntryDef	Defines the queue entry.
	QueueFilter ? <a href="#">New in JDF 1.2</a>	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 5.6.5, Queue-Handling Elements.		

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

The following example demonstrates how an AbortQueueEntry command may cause 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="2003-07-25T12:32:48+02:00" Version="1.2">
  <Command ID="M009" Type="AbortQueueEntry">
    <QueueEntryDef QueueEntryID="job-0032"/>
    <QueueFilter>
      <QueueEntryDef QueueEntryID="job-0032"/>
    </QueueFilter>
  </Command>
</JMF>
```

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

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="A3 Printer"
  TimeStamp="2003-07-25T12:32:48+02:00" Version="1.2">
  <Response ID="M109" Type="AbortQueueEntry" refID="M009">
    <Queue DeviceID="A3 Printer" Status="Running">
      <QueueEntry JobID="job-0032" QueueEntryID="job-0032" Status="Aborted"/>
    </Queue>
  </Response>
</JMF>
```

### 5.6.3.2 HoldQueueEntry

[Modified in JDF 1.2](#)

Table 5-76: Contents of the HoldQueueEntry message

Object Type	Element name	Description
CommandTypeObj <a href="#">Modified in JDF 1.2</a>	QueueEntryDef	Defines the queue entry.
	QueueFilter ? <a href="#">New in JDF 1.2</a>	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 5.6.5, Queue-Handling Elements.		

The entry specified by QueueEntryDef remains in the queue but is never executed. If its Status is "Waiting", its Status is set to "Held". The HoldQueueEntry command has no effect on jobs with a Status other than "Waiting". For details, see "Status transitions for QueueEntry handling messages" on page 174.

### 5.6.3.3 RemoveQueueEntry

[Modified in JDF 1.2](#)

Table 5-77: Contents of the RemoveQueueEntry message

Object Type	Element name	Description
CommandTypeObj <a href="#">Modified in JDF 1.2</a>	QueueEntryDef	Defines the queue entry.
	QueueFilter ? <a href="#">New in JDF 1.2</a>	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 5.6.5, Queue-Handling Elements.		

This command causes the entry specified by QueueEntryDef to be removed from the queue. It does not affect QueueEntries with a *Status* = "Running" or "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 174

### 5.6.3.4 RequestQueueEntry

[New in JDF 1.2](#)

Table 5-78: Contents of the RequestQueueEntry message

Object Type	Element name	Description
CommandTypeObj	RequestQueueEntryParams	Defines the specifics for the requested job.
ResponseTypeObj	—	The controller does not send any immediate response. Any job submission is handled using hot folders or the standard SubmitQueueEntry message.
For the definition of the elements listed above see, Section 5.6.5, Queue-Handling Elements.		

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.

#### Structure of the RequestQueueEntryParams Element

Name	Data Type	Description
JobID ?	string	JobID of the requested QueueEntry.
JobPartID ?	string	JobPartID of the requested QueueEntry.
QueueURL	URL	URL of the Queue controller that is requesting the QueueEntry and will accept Queue manipulation messages.
Part *	element	Partition Parts of the requested QueueEntry.
Queue ?	element	Representation of the current status of the device's Queue.

### 5.6.3.5 ResubmitQueueEntry

[Modified in JDF 1.2](#)

Table 5-79: Contents of the ResubmitQueueEntry message

Object Type	Element name	Description
CommandTypeObj <a href="#">Modified in JDF 1.2</a>	ResubmissionParams	Defines the job resubmission.
	QueueFilter ? <a href="#">New in JDF 1.2</a>	Defines a filter for the returned Queue element in the ResubmitQueueEntry message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.
For the definition of the Queue element, see Section Section 5.6.5, Queue-Handling Elements.		

A job is resubmitted to a queue using the ResubmitQueueEntry message. This allows late changes to be made to a job without affecting queue parameters and without exporting the internal structure of a queue. Resubmission overwrites the job specified in ResubmissionParams/@URL. The QueueEntry/@Status must be "Waiting" or "Held". Job resubmission does not affect other queue parameters as specified. For example, resubmission does not affect queue ordering. For details, see "Status transitions for QueueEntry handling messages" on page 174

#### Structure of the ResubmissionParams Element

Table 5-80: Contents of the ResubmissionParams element

Name	Data Type	Description
QueueEntryID	string	ID of the queue entry to be replaced.
URL	URL	Location of the JDF to be submitted. May be either a URL or, in the case of MIME/Multipart/Related, a CID.

### 5.6.3.6 ResumeQueueEntry

[Modified in JDF 1.2](#)

Table 5-81: Contents of the ResumeQueueEntry message

Object Type	Element name	Description
CommandTypeObj <a href="#">Modified in JDF 1.2</a>	QueueEntryDef	Defines the queue entry.
	QueueFilter ? <a href="#">New in JDF 1.2</a>	Defines a filter for the returned Queue element in the ResumeQueueEntry message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.
For the definition of the elements listed above, see Section 5.6.5, Queue-Handling Elements.		

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". For details, see "Status transitions for QueueEntry handling messages" on page 174.

### 5.6.3.7 ReturnQueueEntry

[New in JDF 1.2](#)

Table 5-82: Contents of the ReturnQueueEntry message

Object Type	Element name	Description
CommandTypeObj	ReturnQueueEntryParams	Defines the job being returned from Device to Controller after processing is completed or aborted.
ResponseTypeObj	-	
For the definition of the elements listed above, see Section 5.6.5, Queue-Handling Elements.		

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. JDF Packaging may be supported by a return queue and should be determined by use of a `SubmissionMethods` query. The `ReturnQueueEntryParams` element provides the required 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.

### Structure of the `ReturnQueueEntryParams` Element

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 5-83: Contents of the `ReturnQueueEntryParams` element

Name	Data Type	Description
<code>Aborted ?</code>	IDREFS	ID of the JDF nodes that have been executed and aborted or failed test running. If <code>Aborted</code> and <code>Completed</code> are empty, no executable node was found.
<code>Completed ?</code>	IDREFS	ID of the JDF nodes that have been executed and completed or succeeded in test run.
<code>Priority ?</code>	integer	The priority of the <code>QueueEntry</code> when it was executed on the device. The controller receiving this message may prioritize this job for continued processing based on this value.
<code>URL</code>	URL	Location of the JDF to be returned.

### 5.6.3.8 `SetQueueEntryPosition`

[Modified in JDF 1.2](#)

Table 5-84: Contents of the `SetQueueEntry` message

Object Type	Element name	Description
CommandTypeObj <a href="#">Modified in JDF 1.2</a>	<code>QueueEntryPosParams</code>	Defines the queue entry.
	<code>QueueFilter ?</code> <a href="#">New in JDF 1.2</a>	Defines a filter for the returned <code>Queue</code> element in the <code>SetQueueEntryPosition</code> message.
ResponseTypeObj	<code>Queue</code>	Describes the state of the queue after the command has been executed.

For the definition of the `Queue` element, see Section 5.6.5, Queue-Handling Elements.

The position of the queue entry is modified. The `QueueEntryPosParams` element provides the required parameters. The position of a queue entry must only be modified if `Status = "Waiting"` or `Status = "Held"`. For details, see "Status transitions for `QueueEntry` handling messages" on page 174.

### Structure of the `QueueEntryPosParams` Element

`QueueEntryID` specifies the queue entry to be moved. Jobs may 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. Only one of `NextQueueEntryID`, `PrevQueueEntryID` or `Position` may be specified.

Table 5-85: Contents of the `QueueEntryPosParams` element

Name	Data Type	Description
<code>NextQueueEntryID ?</code>	string	ID of the queue entry that should be ordered directly behind the entry.
<code>QueueEntryID</code>	string	ID of a queue entry.
<code>PrevQueueEntryID ?</code>	string	ID of the queue entry that should be ordered directly in front of the entry.
<code>Position ?</code>	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 <code>Position</code> .

### 5.6.3.9 SetQueueEntryPriority

[Modified in JDF 1.2](#)

Table 5-86: Contents of the SetQueueEntryPriority message

Object Type	Element name	Description
CommandTypeObj <a href="#">Modified in JDF 1.2</a>	QueueEntryPriParams	Defines the queue entry.
	QueueFilter ? <a href="#">New in JDF 1.2</a>	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 5.6.5, Queue-Handling Elements.		

The priority of the queue entry is modified. The QueueEntryPriParams element provides the required parameters. For details, see “Status transitions for QueueEntry handling messages” on page 174.

#### Structure of the QueueEntryPriParams Element

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

Table 5-87: Contents of the QueueEntryPriParams element

Name	Data Type	Description
Priority	integer	Number from 0 to 100, where “0” = lowest priority and “100” = maximum priority.
QueueEntryID	string	ID of a queue entry.

### 5.6.3.10 SubmitQueueEntry

[Modified in JDF 1.2](#)

Table 5-88: Contents of the SubmitQueueEntry message

Object Type	Element name	Description
CommandTypeObj <a href="#">Modified in JDF 1.2</a>	QueueSubmissionParams	Defines the job submission.
	QueueFilter ? <a href="#">New in JDF 1.2</a>	Defines a filter for the returned Queue element in the SubmitQueueEntry message.
ResponseTypeObj	QueueEntry ? <a href="#">Modified in JDF 1.2</a>	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 5.6.5, Queue-Handling Elements.		

The SubmitQueueEntry message submits a job to a queue. The QueueSubmissionParams element provides the required parameters.

#### Structure of the QueueSubmissionParams Element

The job submission may 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. The optional ReturnURL attribute specifies the location where the modified JDF should be sent after the job is completed or aborted.



Table 5-89: Contents of the QueueSubmissionParams element

Name	Data Type	Description
<i>Hold</i> = "false"	boolean	If <i>true</i> , the entry is submitted as held. If a JDF node is not executable due to resources being unavailable, it must be submitted with <i>Hold</i> = "true".
<i>NextQueueEntryID</i> ?	string	ID of the queue entry that should be ordered directly behind the entry.
<i>PrevQueueEntryID</i> ?	string	ID of the queue entry that should be ordered directly in front of the entry.
<i>Priority</i> = "1"	integer	Number from 0 to 100, where "0" = lowest priority and "100" = maximum priority. Note that QueueSubmissionParams/@Priority is not the same as JDF/NodeInfo/@Priority. QueueSubmissionParams/@Priority specifies the priority in the context of the device queue whereas JDF/NodeInfo/@Priority specifies the priority of the task in general. QueueSubmissionParams/@Priority may be modified due to additional scheduling information, (e.g., JDF/NodeInfo/@First-Start).
<i>refID</i> ? <a href="#">New in JDF 1.2</a>	NMTOKEN	Copy of the <i>ID</i> attribute of the initiating RequestQueueEntry message.
<i>ReturnJMF</i> ? <a href="#">New in JDF 1.2</a>	URL	Address of a JMF queue where a ReturnQueueEntry message must be sent when the QueueEntry is completed or aborted. Note that the <i>ReturnJMF</i> queue should be queried with a SubmissionMethods query to determine whether MIME is supported by the return queue.
<i>ReturnURL</i> ? <a href="#">Modified in JDF 1.2</a>	URL	URL where the JDF file should be written when the QueueEntry is completed or aborted. Only the JDF and not a MIME multipart package must be returned to the URL specified by <i>ReturnURL</i> . <i>ReturnURL</i> takes precedence when NodeInfo/@TargetRoute is specified in the submitted JDF. Note: It is not valid to return the JDF file by performing a SubmitQueueEntry or ReturnQueueEntry to the <i>ReturnURL</i> address, nor to write a multipart/related file to <i>ReturnURL</i> . The controller specified by <i>ReturnURL</i> must not accept JMF messages. See instead <i>ReturnJMF</i> . Only one of <i>ReturnURL</i> or <i>ReturnJMF</i> must be specified.
<i>URL</i> <a href="#">Modified in JDF 1.2</a>	URL	Location of the JDF to be submitted. In the case of MIME/Multipart/Related, the location may be either a URL or a CID.
<i>WatchURL</i> ? <a href="#">Modified in JDF 1.2</a>	URL	URL of the controller that should be notified when the status of the QueueEntry or the underlying job changes. Specifying <i>WatchURL</i> is equivalent to sending a subscription for an Events message with <i>SignalTypes</i> ="All".
<i>Disposition</i> ? <a href="#">New in JDF 1.2</a>	element	Definition how long the QueueEntry must be retained in the queue. If not specified, the QueueEntry must be removed from the queue immediately after process completion of the QueueEntry.

### File Submission

If the URL defines a file, the Device may retrieve the file at the location specified in the *URL* attribute. The following example declares a file on the network:

```
<Command ID="M1" Type="SubmitQueueEntry">
  <QueueSubmissionParams URL="File://MyNetWorkShare/AnyDirectory/job1.jdf"/>
</Command>
```

## HTTP External JDF Submission

The following example declares an intranet or Internet location. In this example, the queue controller can retrieve the file with a standard HTTP `get` command. Note that the job itself may be a MIME/Multipart entity. It may also be dynamically generated by a CGI script or another such tool.

```
<Command ID="M2" Type="SubmitQueueEntry">
  <QueueSubmissionParams URL="http://JobServer.JDF.COM?job1"/>
</Command>
```

## JDF Package Submission

If a controller is capable of decoding mime, it is legal to submit a MIME/Multipart/Related message. See “JDF Packaging” on page 560 for details of MIME/Multipart/Related packaging.

### 5.6.3.11 SuspendQueueEntry

[New in JDF 1.2](#)

Table 5-90: Contents of the 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 “Queue-Handling Elements” on page 187 for the definition of the elements listed above. The entry specified by QueueEntryDef remains in the queue but moved into the <i>Suspended</i> state.
For the definition of the elements listed above, see Section 5.6.5, Queue-Handling Elements.		

The entry specified by QueueEntryDef is suspended if its *Status* is “Running”. Its *Status* is set to “Suspended”. Whether other queue entries can be run while the queue entry remains suspended depends on implementation. The SuspendQueueEntry command has no effect on jobs with a *Status* other than “Running”. For details, see “Status transitions for QueueEntry handling messages” on page 174.

## 5.6.4 Global Queue Handling

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/queue in the *DeviceID* attribute of the JMF root. If no *DeviceID* is specified, the commands or queries are applied to all devices/queues that are controlled by the controller that received the message. The following individual messages are defined:

Table 5-91: Global queue-handling commands

Message type	Family	Description
CloseQueue	CR	The queue is closed. No jobs may 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 may be executed.
OpenQueue	CR	The queue is opened. Jobs may be accepted.
QueueEntryStatus <a href="#">Deprecated in JDF 1.2</a>	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 may 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 5-92: 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>

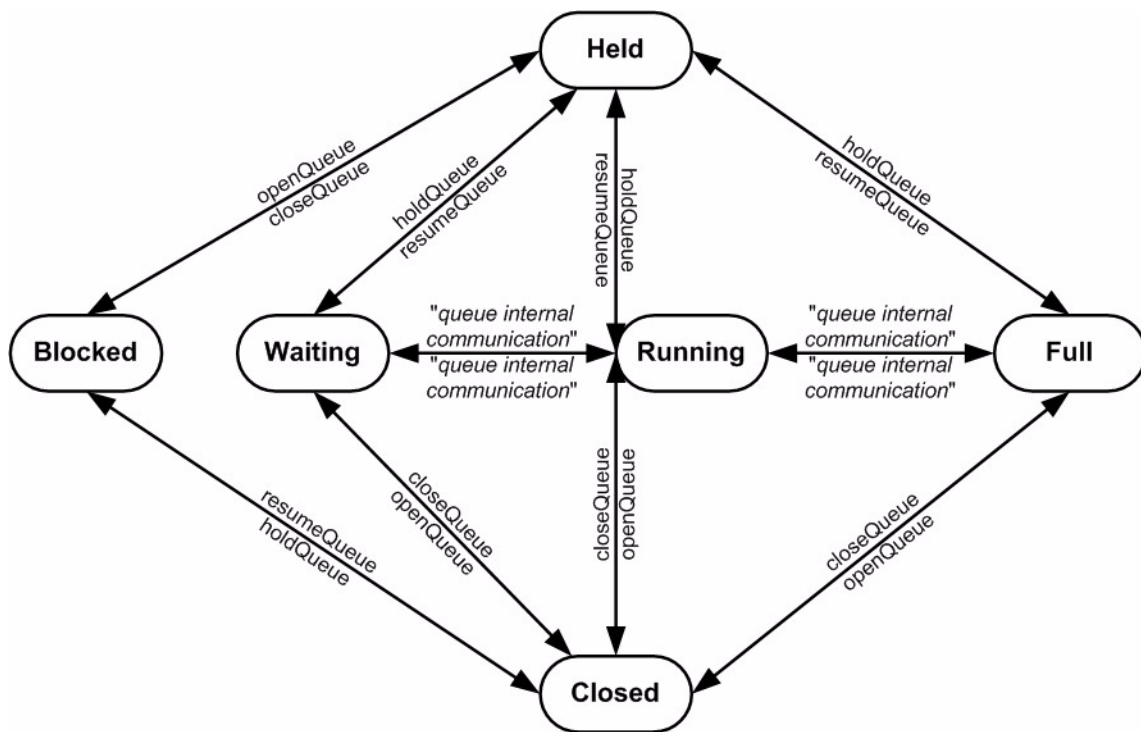


Figure 5.7: Effects of the global queue messages on the queue Status

### 5.6.4.1 CloseQueue

[Modified in JDF 1.2](#)

Table 5-93: Contents of the CloseQueue message

Object Type	Element name	Description
CommandTypeObj <a href="#">Modified in JDF 1.2</a>	QueueFilter ? <a href="#">New in JDF 1.2</a>	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 5.6.5, Queue-Handling Elements.		

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 may be executed.

### 5.6.4.2 FlushQueue

[Modified in JDF 1.2](#)

#### Structure of the FlushQueue Command Message

Table 5-94: Contents of the FlushQueue Command message

Object Type	Element name	Description
CommandTypeObj <a href="#">Modified in JDF 1.2</a>	QueueFilter ? <a href="#">New in JDF 1.2</a>	Defines a filter for the returned Queue element in the FlushQueue message.
	FlushQueueParams ? <a href="#">New in JDF 1.2</a>	Defines the QueueEntries to be removed. If not specified then only pending ( <i>Status</i> = "Waiting" and <i>Status</i> = "Held" queue entries are removed.)
ResponseTypeObj <a href="#">Modified in JDF 1.2</a>	Queue	Describes the state of the queue after the command has been executed.
	FlushQueueInfo ? <a href="#">New in JDF 1.2</a>	Defines the QueueEntries that were removed.
For the definition of the Queue element, see Section 5.6.5, Queue-Handling Elements.		

FlushQueue is used to remove QueueEntries 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 QueueEntries to remove. The QueueFilter in the FlushQueue message is applied to the Queue returned after the command is executed. The QueueFilter contained within the FlushQueueParams is used to specify which QueueEntries to remove.

#### Structure of the FlushQueue Query Message

Table 5-95: Contents of the 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 ? <a href="#">New in JDF 1.2</a>	Defines the QueueEntries that were removed.
For the definition of the Queue element, see Section 5.6.5, Queue-Handling Elements.		

When used as a Signal or Query, FlushQueue allows a controller to monitor queue flushing that is initiated by the device, (e.g. due to resource constraints.) The QueueFilter in the FlushQueue message is applied to the Queue returned after the command is executed. The QueueFilter contained within the FlushQueueInfo is used to specify which QueueEntries were removed.

#### Structure of the FlushQueueInfo Element

[New in JDF 1.2](#)

Table 5.1: Contents of the FlushQueueParams element

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

The `QueueFilter` in `FlushQueueParams` defines the `QueueEntry`s to be removed by `FlushQueue`. Those `QueueEntry`s meeting the criteria set in the `QueueFilter` will be removed.

### Structure of the `FlushQueueParams` Element

[New in JDF 1.2](#)

Table 5.2: Contents of the `FlushQueueParams` element

Name	Data Type	Description
<code>QueueFilter ?</code>	element	Defines a <code>QueueFilter</code> that specifies the <code>QueueEntry</code> s to be removed. If not specified, the <code>Queue</code> is completely flushed.

#### 5.6.4.3 HoldQueue

[Modified in JDF 1.2](#)

Table 5-96: Contents of the `HoldQueue` message

Object Type	Element name	Description
<code>CommandTypeObj</code> <a href="#">Modified in JDF 1.2</a>	<code>QueueFilter ?</code> <a href="#">New in JDF 1.2</a>	Defines a filter for the returned <code>Queue</code> element in the <code>HoldQueue</code> message.
<code>ResponseTypeObj</code>	<code>Queue</code>	Describes the state of the queue after the command has been executed.
For the definition of the <code>Queue</code> element, see Section 5.6.5, <code>Queue-Handling Elements</code> .		

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 should remain held after a `ResumeQueue`. New entries may 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.

#### 5.6.4.4 OpenQueue

[Modified in JDF 1.2](#)

Table 5-97: Contents of the `OpenQueue` message

Object Type	Element name	Description
<code>CommandTypeObj</code> <a href="#">Modified in JDF 1.2</a>	<code>QueueFilter ?</code> <a href="#">New in JDF 1.2</a>	Defines a filter for the returned <code>Queue</code> element in the <code>OpenQueue</code> message.
<code>ResponseTypeObj</code>	<code>Queue</code>	Describes the state of the queue after the command has been executed.
For the definition of the <code>Queue</code> element, see Section 5.6.5, <code>Queue-Handling Elements</code> .		

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

#### 5.6.4.5 QueueEntryStatus

[Deprecated in JDF 1.2](#)

In JDF 1.2 and beyond, use `QueueStatus` with an appropriate `QueueFilter` instead of `QueueEntryStatus`. See “`QueueEntryStatus`” on page 773 for details of this deprecated JMF element.

#### 5.6.4.6 QueueStatus

[Modified in JDF 1.2](#)

Table 5-98: Contents of the `QueueStatus` message

Object Type	Element name	Description
<code>QueryTypeObj</code> <a href="#">Modified in JDF 1.2</a>	<code>QueueFilter ?</code> <a href="#">New in JDF 1.2</a>	Defines a filter for the <code>QueueStatus</code> message.
<code>ResponseTypeObj</code>	<code>Queue</code>	Describes the status of the queue.
For the definition of the <code>Queue</code> element, see Section 5.6.5, <code>Queue-Handling Elements</code> .		

Returns a queue description.

### 5.6.4.7 ResumeQueue

[Modified in JDF 1.2](#)

Table 5-99: Contents of the ResumeQueue message

Object Type	Element name	Description
CommandTypeObj <a href="#">Modified in JDF 1.2</a>	QueueFilter ? <a href="#">New in JDF 1.2</a>	Defines a filter for the ResumeQueue message.
ResponseTypeObj	Queue	Describes the state of the queue after the command has been executed.
For the definition of the Queue element, see Section 5.6.5, Queue-Handling Elements.		

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

### 5.6.4.8 SubmissionMethods

Table 5-100: Contents of the SubmissionMethods message

Object Type	Element name	Description
QueryTypeObj	—	—
ResponseTypeObj	SubmissionMethods ?	Describes the submission methods supported by the queue.

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

#### Structure of the SubmissionMethods Element

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

Table 5-101: Contents of the SubmissionMethods element

Name	Data Type	Description
<a href="#">File ?</a> <a href="#">Deprecated in JDF 1.2</a>	boolean	Can retrieve a JDF from a File specified in the URL. In JDF 1.2 and beyond, include "file" in URLSchemes.
<a href="#">HotFolder ?</a>	URL	URL specification of a hot folder location.
<a href="#">HttpGet ?</a> <a href="#">Deprecated in JDF 1.2</a>	boolean	Can retrieve a JDF via HTTP get commands. In JDF 1.2 and beyond, include "http" in URLSchemes.
<a href="#">Packaging ?</a> <a href="#">New in JDF 1.2</a>	enumerations	List of packaging methods supported. If not specified the controller does not support receiving packaged messages and must retrieve JDF files using a URLScheme. Allowed values are: <i>MIME</i> – Accepts MIME/Multipart/Related packaging of JMF, JDF, and digital assets. This packaging is intended for bidirectional JMF messaging. When used for unidirectional JMF messaging, the package must be saved as a single file with a ".mjm" extension.
<a href="#">MIME ?</a> <a href="#">Deprecated in JDF 1.2</a>	boolean	Accepts MIME/Multipart/Related submission messages via a message post. In JDF 1.2 and beyond, use <i>Packaging</i> ="MIME".
<a href="#">URLSchemes ?</a> <a href="#">New in JDF 1.2</a>	NMTOKENS	List of schemes supported in for retrieving JDF files. If not specified, the controller does not support retrieving JDF files from remote URLs. Values include: <i>file</i> – The file scheme according to [RFC 1738]. <i>ftp</i> – FTP (File Transfer Protocol) <i>http</i> – HTTP (Hypertext Transport Protocol) <i>https</i> – HTTPS (Hypertext Transport Protocol — Secure)

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

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

## 5.6.5 Queue-Handling Elements

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

### Structure of the Queue Element

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 5-102: Contents of the Queue element

Name	Data Type	Description
<i>Status</i>	enumeration	Status of the queue. Possible values are: <i>Blocked</i> – Queue is completely inactive. No entries may be added and no entries are executed. The queue is closed and held. The queue requires an interaction like <code>OpenQueue</code> or <code>ResumeQueue</code> to reactivate it. <i>Closed</i> – Queue entries that are in the queue are executed, but no new entries may be submitted. The lock must be removed explicitly by the <code>OpenQueue</code> command. <i>Full</i> – Queue entries that are in the queue are executed but no new entries may 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 may 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 may be submitted but will not be executed until the queue is resumed by the <code>ResumeQueue</code> command.
<i>QueueSize</i> ? <a href="#">New in JDF 1.2</a>	integer	The maximum number of <code>QueueEntry</code> s that can be in the <code>Queue</code> . Note: <code>QueueEntry</code> s with <i>Status</i> = "Completed" or <i>Status</i> = "Aborted" must not count towards determining <code>Queue/@Status</code> based on the number of <code>QueueEntry</code> s versus the <i>QueueSize</i> .
<i>DeviceID</i>	string	Identifies the device that is represented by the queue.
<i>Device</i> *	element	The devices that execute entries in this queue. Only <code>Device/@DeviceID</code> should be specified in these elements.
<i>QueueEntry</i> * <a href="#">Modified in JDF 1.2</a>	element	<code>QueueEntry</code> elements (see Table 5-103, "Contents of the QueueEntry element," on page 188, 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 <code>QueueEntry/@Priority</code> first, which are then followed by the completed entries, sorted beginning with the youngest <code>QueueEntry/@EndTime</code> . The <code>Queue</code> will display a list of all <code>QueueEntry</code> s that are still accessible on the device using the queue entry handling messages that are defined in Table 5-103, "Contents of the QueueEntry element," on page 188. A <code>QueueEntry</code> is not automatically deleted when executed or aborted, but rather it remains in the <code>Queue</code> and its status is changed to <i>Completed</i> or <i>Aborted</i> accordingly. <code>QueueEntry</code> s with <i>Status</i> = "Completed" or <i>Status</i> = "Aborted" must not count towards determining <code>Queue/@Status</code> based on the number of <code>QueueEntry</code> s versus the <i>QueueSize</i> .

Example of a **Queue** message element:

```
<Queue DeviceID="Q12345" Status="Running">
  <QueueEntry JobID="111" JobPartID="0" Priority="1" QueueEntryID="111-0"
Status="Completed"/>
  <QueueEntry JobID="111" JobPartID="1" Priority="1" QueueEntryID="111-1"
Status="Running"/>
  <QueueEntry JobID="111" JobPartID="2" Priority="1" QueueEntryID="111-2"
Status="Waiting"/>
  <QueueEntry JobID="112" JobPartID="1" Priority="55" QueueEntryID="112-1"
Status="Held"/>
</Queue>
```

## Structure of the QueueEntry Element

[Modified in JDF 1.2](#)

Table 5-103: Contents of the QueueEntry element

Name	Data Type	Description
<b>DeviceID</b> ? <a href="#">New in JDF 1.2</a>	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.
<b>EndTime</b> ? <a href="#">New in JDF 1.2</a>	dateTime	Time when the job has been ended.
<b>JobID</b> ? <a href="#">Modified in JDF 1.1</a>	string	The Job ID of the JDF process.
<b>JobPartID</b> ?	string	The JobPartID of the JDF process.
<b>Priority</b> = "1"	integer	Priority of the QueueEntry. Values are 0-100. "0" is the lowest priority, while "100" is the highest priority.
<b>QueueEntryID</b>	string	ID of a QueueEntry. This ID is generated by the queue owner.
<b>StartTime</b> ? <a href="#">New in JDF 1.1</a>	dateTime	Time when the job has been started.
<b>Status</b> <a href="#">Modified in JDF 1.2</a>	enumeration	Status of the individual entry. Possible values are: <i>Running</i> – The queue entry is running on the device. <i>Waiting</i> – The queue entry is waiting and will be executed when resources are available. <i>Held</i> – The queue entry is held and will not execute until resumed. <i>Removed</i> – The queue entry has been removed. This status can only be sent when a persistent channel watches a queue and the queue entry is removed. <i>Suspended</i> – The queue entry was running and has been held. It will not continue to execute until resumed. <a href="#">New in JDF 1.2</a> <i>Completed</i> – Indicates that the queue entry has been executed correctly, and is finished. <a href="#">New in JDF 1.2</a> <i>Aborted</i> – Indicates that the process executing the node has been aborted, which means that execution will not be resumed again. <a href="#">New in JDF 1.2</a>
<b>SubmissionTime</b> ?	dateTime	Time when the entry was submitted to the queue.
<b>JobPhase</b> ? <a href="#">New in JDF 1.2</a>	element	Description of the current status of the Job that is associated with the QueueEntry.
<b>Part</b> * <a href="#">New in JDF 1.2</a>	element	Describes which parts of a job were submitted to the queue.
<b>Preview</b> * <a href="#">New in JDF 1.2</a>	element	Any number of previews may be associated with a QueueEntry and used for display purposes. <b>Preview/@PreviewUsage</b> should be <i>ThumbNail</i> or <i>Viewable</i> .



## Structure of the QueueEntryDef Element

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

Table 5-104: Contents of the QueueEntryDef element

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

## Structure of the QueueFilter Element

[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 QueueEntries to be removed by the FlushQueue message.

Table 5.105: Contents of the QueueFilter Element

Name	Data Type	Description
<i>MaxEntries</i> ?	integer	Maximum number of QueueEntries to provide in the Queue element. If not specified, fill in all matching QueueEntries.
<i>OlderThan</i> ?	dateTime	Only QueueEntries with a <i>SubmissionTime</i> older than or equal to this dateTime are provided in the Queue element or removed by the FlushQueue message. If not specified, there is no dateTime lower bound on candidates.
<i>NewerThan</i> ?	dateTime	Only QueueEntries with a <i>SubmissionTime</i> newer than or equal to this dateTime are provided in the Queue element or removed by the FlushQueue message. If not specified, there is no dateTime upper bound on candidates.
<i>QueueEntryDetails</i> = “ <i>Brief</i> ”	enumeration	Refines the level of provided information about the Queue. Possible 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 element <i>JDF</i> – Provide all available QueueEntry information including the associated JobPhase element and the associated JDF element in the JobPhase element. <i>Full</i> – Provide maximum available information including device capability descriptions. Includes Device elements which represent details of the device.
<i>StatusList</i> ?	enumerations	Only QueueEntry elements with a <i>Status</i> matching one of the entries in <i>StatusList</i> are considered. For a list of allowed values, see Table 5-103, “Contents of the QueueEntry element,” on page 188.
QueueEntryDef *	element	Defines an explicit list of queue entries. If not specified, all entries in the Queue are considered.
Device *	element	Filter of the Devices that queue entries returned should be targeted for. QueueEntry/@DeviceID must match QueueFilter/Device/@DeviceID for the QueueEntry to be returned in the queue. If not specified, all entries in the Queue are considered.

## 5.7 Extending Messages

This specification defines a set of predefined messages for general usage. Extensions to existing messages and additional message types may be defined using the standard extension rules described in Section 3.10, JDF Extensibility. Note, the generic content of Section 3.1.1, Generic Contents of JDF Elements is also valid for JMF elements. It is not allowed to define message extensions which duplicate the functionality of messaging types, messaging elements, or message attributes that are already defined in this specification.

For example the content of the *Type* attribute may be specified with a prefix that identifies the organization that defined the extension. The prefix and name should be separated by a single colon (':'). Any additional attributes and elements are allowed, and internal elements may be declared with explicit namespaces. The official namespace of JMF elements is xmlns="http://www.CIP4.org/JDFSchema\_1\_1". This namespace is identical to that defined for JDF in Section 3.10, JDF Extensibility. An example is provided:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Circus" TimeStamp="2003-07-25T12:32:48+02:00" Version="1.2" xmlns:Circus="Circus Schema URI">
  <Query ID="Q1" Type="Circus:IsClownHappy">
    <Circus:ClownParams Gender="male"/>
  </Query>
</JMF>
```

The response will also have the "Circus:" namespace identifier. All Circus elements are explicitly declared.

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Circus 2" TimeStamp="2003-07-25T12:32:48+02:00" Version="1.2" xmlns:Circus="Circus Schema URI">
  <Response ID="M1" Type="Circus:IsClownHappy" refID="Q1">
    <Circus:Clown happy="true" name="Joe"/>
    <Circus:Clown happy="false" name="John"/>
  </Response>
</JMF>
```

### 5.7.1 IfraTrack Support

The extending mechanism can be used to implement compatibility with other XML-based messaging standards, for example version 3.0 of IfraTrack. The *Type* attribute is set to the appropriate namespace, and the foreign message is included, as demonstrated in the following example:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="IFRA" TimeStamp="2003-07-25T12:32:48+02:00" Version="1.2" xmlns:IFRA="IfraTrack URI">
  <Query ID="Q1" Type="IFRA:IMF">
    <imf:IMF xmlns:imf="IfraTrack URI">
      Whatever you want (may be multiple top level elements)
    </imf:IMF>
  </Query>
</JMF>
```

The legal response would be:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="IFRA" TimeStamp="2003-07-25T12:32:48+02:00" Version="1.2" xmlns:IFRA="IfraTrack URI">
  <Response ID="M1" Type="IFRA:IMF" refID="Q1">
    <imf:IMF xmlns:imf="IfraTrack URI">
      The appropriate IFRA response(s)
    </imf:IMF>
  </Response>
</JMF>
```

Note that the application is free to select the appropriate response types in order to fulfill its local (IfraTrack) protocol requirements if it uses its own namespace. In the examples above the default namespace associated with the IMF query and response elements has been overwritten by the Ifra-namespace. Additional information on using IfraTrack and JDF is in "Modeling IfraTrack in JDF" on page 611.



#### More on IfraTrack


IfraTrack is a specification for the interchange of status and management information between local and global production management systems in newspaper production. For more information on IfraTrack, including a case study paper, please see [http://www.ifra.com/WebSite/news.nsf/\(StructuredSearchAll\)?OpenAgent&IFRATRACK](http://www.ifra.com/WebSite/news.nsf/(StructuredSearchAll)?OpenAgent&IFRATRACK)

# Chapter 6 Processes

The following chapter describes the processes that are defined in detail for JDF.

## 6.1 Process Template

Processes are defined by their input and output resources, therefore, all relevant resource information is provided in tables for each process. Furthermore, although they are not listed for each process, additional, optional input resources as defined in the following table as well as any implementation resources are implied for all processes defined in this chapter.

**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.

### Input Resources

Name	Description
<b>Resource</b>	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.
<b>Res1</b> ( <i>usage1</i> )	A resource of type <b>Res1</b> with the <i>ProcessUsage</i> = " <i>usage1</i> "
<b>Res1</b> ( <i>usage2</i> )	A resource of type <b>Res1</b> with the <i>ProcessUsage</i> = " <i>usage2</i> "
<b>ApprovalSuccess</b> *	Any number of <b>ApprovalSuccess</b> 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 <b>Approval</b> process, see Section 6.2.1, Approval.
<b>Implementation</b> *	Abstract resource that is a placeholder for any implementation resource (examples are <b>Employee</b> or <b>Device</b> ) that is associated with processing this node.
<b>PreflightReport</b> * <a href="#">New in JDF1.2</a>	Any number of <b>PreflightReport</b> 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 <b>Preflight</b> process, See "Preflight" on page 204.
<b>Preview</b> * <a href="#">New in JDF1.1A</a>	Any number of previews may be associated with a process and used for display purposes. <b>Preview/@PreviewUsage</b> should be <i>ThumbNail</i> or <i>Viewable</i> .

### Output Resources

Name	Description
<b>Resource</b>	Represents any output resource.
<b>Res1</b> ( <i>usage1</i> )	A resource of type <b>Res1</b> with the <i>ProcessUsage</i> = " <i>usage1</i> "
<b>Res1</b> ( <i>usage2</i> )	A resource of type <b>Res1</b> with the <i>ProcessUsage</i> = " <i>usage2</i> "

## 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 may either have a *Status* = "Draft" before the **Approval** and a *Status* = "Available" after a successful **Approval**. They may also have a *Status* = "Available" before the **Approval** and a *Status* = "Rejected" after an unsuccessful **Approval**.

### Input Resources

Name	Description
<b>ApprovalParams</b>	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 <b>Approval</b> process is a <b>ByteMap</b> , it is assumed that it will be displayed on a viewing device

### Output Resources

Name	Description
<b>ApprovalSuccess</b>	Result of any proofing process given, for example, by a customer or foreman. Note that <b>ApprovalSuccess</b> resources are only available on success.
Resource * (Accepted)	Represents the input resources that have been accepted for further processing by the approval process as output resources. This is typically used to transfer the resource <i>Status</i> of <i>Draft</i> to <i>Available</i> (see also Section 4.3.5.2, Formal Iterative Processing).
Resource * (Rejected)	Represents the input resources that have been rejected for further processing by the approval process as output resources. This may be used to define additional processing for rejected resources. <b>Resource/@Status</b> 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 quantity of the input and output of resources should be equal. Waiting for printed material to dry before finishing is an example of the **Buffer** process.

### Input Resources

Name	Description
<b>BufferParams</b>	The parameters, (e.g. times and locations of the <b>Buffer</b> 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> .

### Output Resources

Name	Description
Resource	The same resource after buffering. The resource must have a class 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., **RunLists** of the same content to form one resource). The quantity of the input and output of resources should be equal. The ordering of the input ResourceLinks must be honored.

### Input Resources

Name	Description
Resource +	The resources to be combined.

### Output Resources

Name	Description
Resource	Result of combining. The resource formed as a result of the <b>Combine</b> 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 may 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.

### Input Resources

Name	Description
<b>DeliveryParams</b>	Necessary information about the physical item or items to be delivered is stored here.
Resource ? <a href="#">Deprecated in JDF 1.2</a>	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 <b>DeliveryParams</b> /Drop/DropItem.

### Output Resources

Name	Description
Resource + <a href="#">Modified in JDF 1.2</a>	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.

### Input Resources

Name	Description
Resource *	Resources that are required to create the output Resource.
<b>ManualLaborParams</b>	Details on the <b>ManualLabor</b> process.

### Output Resources

Name	Description
Resource	The resource that was created by manual work. In general these will be <b>Components</b> , but handling resources may also be processed manually.

## 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).

### Input Resources

Name	Description
<b>OrderingParams</b>	Necessary information about the items to be ordered, (e.g., the supplier address, item quantity, or unit type).

### Output Resources

Name	Description
Resource + <a href="#">Modified in JDF 1.1</a>	All kinds of physical resources can be ordered.

## 6.2.7 Packing

[Deprecated in JDF 1.1](#)

See “Packing” on page 743 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 **Components** produced as intermediate or final output of a process. The **QualityControlResult** element should additionally be referenced from the output resource of the **QualityControl** process.

### Input Resources

Name	Description
Resource	The resource to be quality controlled. In general this will be a <b>Component</b> resource.
<b>QualityControlParams</b>	Detailed definition of the <b>QualityControl</b> process.

### Output Resources

Name	Description
<b>QualityControlResult</b>	Results of the process, (e.g. measurement statistics).
Resource	The resource after <b>QualityControl</b> is applied. Note that this resource will generally be partitioned by <i>Condition</i> to track the amount of accepted and rejected resources.

## 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.

### Input Resources

Name	Description
Resource * <a href="#">Modified in JDF 1.1</a>	Any type of resource. Generally these will be templates.
<b>ResourceDefinitionParams</b> ?	Details on how to handle defaults.

### Output Resources

Name	Description
Resource + <a href="#">Modified in JDF 1.1</a>	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 quantity of the input and output of resources should be equal.

### Input Resources

Name	Description
Resource	The resource to be 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 must be validated individually, database access is required. Verification in this situation may involve scanning the physical sheet and interpreting a bar code or alphanumeric characters. The decoded data may 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.

#### Input Resources

Name	Description
<b>DBSchema</b> ?	Schema description of the cross-reference database.
<b>DBSelection</b> ?	Database link that defines the database that contains cross-reference data.
<b>IdentificationField</b> *	Identifies the position and type of data for an automated, OCR-based verification process.
Resource ? <a href="#">New in JDF 1.2</a>	The resource to be verified. The input will most often be a resource with <i>Class="Quantity"</i> , e.g. <b>Component</b> or <i>"Parameter"</i> , e.g. <b>RunList</b> .
<b>VerificationParams</b>	Controls the verification requirements.

#### Output Resources

Name	Description
<b>ApprovalSuccess</b> ?	Signature file that defines verification success.
<b>DBSelection</b> ?	Database link where the verification data should be recorded.
Resource ? <a href="#">New in JDF 1.2</a>	The resource after verification. Most often the <b>Resource</b> will not be modified by <b>Verification</b> . It has been added here to allow modeling of <b>Verification</b> 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.

#### Input Resources

Name	Description
<b>Component</b> *	Components that are partial products of the product described by this node. If input <b>Components</b> are specified, a <b>BindingIntent</b> is required.
<b>ArtDeliveryIntent</b> ?	This resource specifies the prepress art delivery intent for a JDF job.
<b>BindingIntent</b> ?	This resource specifies the binding intent for a JDF job.
<b>ColorIntent</b> ?	This resource specifies the type of ink to be used for a JDF job.
<b>DeliveryIntent</b> ?	Summarizes the options that describe pickup or delivery time and location of the physical resources of a job.
<b>EmbossingIntent</b> ?	This resource specifies the embossing and/or foil stamping intent for a JDF job.
<b>FoldingIntent</b> ?	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.
<b>HoleMakingIntent</b> ?	This resource specifies the holemaking intent for a JDF job.
<b>InsertingIntent</b> ?	This resource specifies the placing or inserting of one component within another, using information that identifies page location, position and attachment method.

Name	Description
<b>LaminatingIntent</b> ?	This resource specifies the laminating intent for a JDF job using information that identifies whether or not the product is laminated.
<b>LayoutIntent</b> ?	This resource records the size of the finished pages for the product component.
<b>MediaIntent</b> ?	This resource describes the media to be used for the product component.
<b>NumberingIntent</b> ?	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.
<b>PackingIntent</b> ?	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.
<b>ProductionIntent</b> ?	This resource specifies the manufacturing intent and considerations for a JDF job using information that identifies the desired result or specified manufacturing path.
<b>ProofingIntent</b> ?	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.
<b>ScreeningIntent</b> ?	This resource specifies the screening intent parameters desired for a JDF job.
<b>ShapeCuttingIntent</b> ?	This resource specifies form and line cutting for a JDF job.
<b>SizeIntent</b> ? <a href="#">Deprecated in JDF 1.2</a>	This resource records the size of the finished pages for the product component. <b>SizeIntent</b> has been deprecated in JDF 1.1. All contents have been moved to <b>LayoutIntent</b> .

### Output Resources

Name	Description
<b>Component</b> +	Resource representation of the output this Product Intent Node. Multiple <b>Components</b> must be specified in a root node that contains a <b>DeliveryIntent</b> that references multiple <b>Components</b> 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 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.

#### Input Resources

Name	Description
<b>RunList</b>	List of assets used to create a listing of dependent assets.
<b>AssetListCreationParams</b>	Parameters of the <b>AssetListCreation</b> process

#### Output Resources

Name	Description
<b>RunList</b>	A listing of all assets that the assets listed in the input <b>RunList</b> are dependent on including the input assets. The dependent assets should be inserted into the output <b>RunList</b> as <b>RunList/LayoutElement/Dependencies/LayoutElement</b> .



## 6.4.2 ColorCorrection

[Modified in JDF 1.2](#)

**ColorCorrection** is the process of modifying the specification of colors in documents to achieve some desired visual result. The process may be performed to ensure consistent colors across multiple files of a job or to achieve a specific design intent, (e.g., “brighten the image up a little”).

**ColorCorrection** is distinct from **ColorSpaceConversion**, which is the process of changing how the colors specified in the job will be produced on paper. Rather, **ColorCorrection** is the process of modifying the desired result, whatever the specified colorspace might be.

The **ColorCorrection** process may be combined with the **ColorSpaceConversion** process, in which case the source and destination profiles used by the **ColorSpaceConversion** process would be supplied from **ColorSpaceConversionParams**. Either the direct *Adjustment* attribute or the ICC profile attribute *ColorCorrectionOp/FileSpec* with *ResourceUsage* = “*AbstractProfile*” can be used in this scenario to apply color corrections in the device independent ICC Profile Connection Space interpreted from the ICC source profile before the ICC destination profile is applied.

Alternatively, a **ColorCorrection** process may occur after a **ColorSpaceConversion** process. In this scenario only the *ColorCorrectionOp/FileSpec* with *ResourceUsage* = “*DeviceLinkProfile*” supplied in *ColorCorrectionOp* is used.

### Input Resources

Name	Description
<b>ColorantControl</b> ? <a href="#">Modified in JDF 1.1A</a>	Identifies the assumed color model for the job.
<b>ColorCorrectionParams</b> <a href="#">New in JDF 1.1</a>	Parameters of the <b>ColorCorrection</b> process
<b>RunList</b>	List of content elements that are to be operated on.

### Output Resources

Name	Description
<b>RunList</b>	List of color-corrected pages.

## 6.4.3 ColorSpaceConversion

**ColorSpaceConversion**, as the name implies, is the process of converting all colors used in the job to a known colorspace. There are two ways in which a controller can use this process to accomplish the color conversion. It can simply order the colors to be converted by the device assigned to the task, or it can request that the process simply tag the input data for eventual conversion. Additionally, the process may 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 DeviceLink<sup>1</sup> 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.

1. DeviceLink profiles are ICC profiles that map directly from one device color space to another device color space. Therefore, it represents a one-way link or connection between devices. Examples for Device-Link profiles are CMYK to CMYK print process conversions or RGB to CMYK color separations.

**Input Resources**

Name	Description
<b>ColorantControl</b> ? <a href="#">Modified in JDF1.1A</a>	Identifies the assumed color model for the job.
<b>ColorSpaceConversionParams</b>	Parameters that define how colorspaces will be converted in the file.
<b>RunList</b>	List of pages, sheets, or <b>ByteMaps</b> on which to perform the selected operation.

**Output Resources**

Name	Description
<b>ColorantControl</b> ?	Identifies the assumed color model for the job. The <b>ColorantControl</b> resource may be modified by a <b>ColorSpaceConversion</b> Process.
<b>RunList</b>	List of pages, sheets, or <b>ByteMaps</b> on which the selected operation has been performed.

**6.4.4 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.

**Input Resources**

Name	Description
<b>ContactCopyParams</b>	The settings of the contact copying task.
<b>DevelopingParams</b> ?	Controls the physical and chemical specifics of the media development process.
<b>ExposedMedia</b> +	The film or films to be copied onto the film or plate.
<b>Media</b>	The unexposed film or plate.
<b>TransferCurvePool</b> ?	Area coverage correction and coordinate transformations of the device.

**Output Resources**

Name	Description
<b>ExposedMedia</b>	The resulting exposed contact copy.

**6.4.5 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.

**Input Resources**

Name	Description
<b>RunList</b>	Ordered list of rasterized <b>ByteMaps</b> representing pages or surfaces.
<b>ScreeningParams</b> ? <a href="#">Modified in JDF 1.1</a>	Parameters specifying which halftoning mechanism is to be applied and with what specific controls.
<b>TransferFunctionControl</b> ? <a href="#">Modified in JDF 1.1</a>	Specifies which calibration to apply.

**Output Resources**

Name	Description
<b>RunList</b>	Ordered list of rasterized <b>ByteMaps</b> representing pages or surfaces.

### 6.4.6 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"*.

#### Input Resources

Name	Description
<b>LayoutElement *</b>	Page elements without links to a database.
<b>DBRules</b>	Description of the rules that should be applied to database records in order to generate graphic output.
<b>DBSchema</b>	Database schema that describe the structure of data in the database.

#### Output Resources

Name	Description
<b>LayoutElement *</b>	The document template is a <b>LayoutElement</b> 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> .

### 6.4.7 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.

#### Input Resources

Name	Description
<b>DBMergeParams</b>	Parameters of the merge process.
<b>DBSelection</b>	Instance database records to be merged into the document.
<b>LayoutElement *</b>	Document template page element with internal links to a database.

#### Output Resources

Name	Description
<b>RunList</b>	Page element without links to a database. This element usually contains a printable <b>LayoutElement</b> resource such as PPML, PDF or even plain ASCII.

### 6.4.8 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*"<sup>1</sup> the corresponding process will be **DigitalDelivery**.

It is not necessary to use the **DigitalDelivery** process to describe informal delivery of files during the workflow although **DigitalDelivery** may 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 669.

1. When **ArtDeliveryIntent/ArtDelivery/@ArtDeliveryType = "DigitalFile"**, the process may also be **Delivery**, in case the file is delivered on digital media.

**Input Resources**

Name	Description
<b>DigitalDeliveryParams</b>	Parameter specifying the artwork files delivery characteristics.
<b>RunList</b>	The list of digital files to be delivered.

**Output Resources**

Name	Description
<b>RunList</b>	The list of digital files which were actually delivered to the destination.

**6.4.9 FilmToPlateCopying**

[Deprecated in JDF 1.1](#)

**FilmToPlateCopying** has been replaced by the more generic **ContactCopying**. See “FilmToPlateCopying” on page 743 for details of this deprecated process.

**6.4.10 FormatConversion**

[New in JDF 1.1](#)

The **FormatConversion** process controls the conversion from one document type to another, for instance, TIFF to BMP.

**Input Resources**

Name	Description
<b>FormatConversionParams</b>	Set of parameters required to control the <b>FormatConversion</b> process.
<b>RunList</b>	List of documents and/or pages to be converted.

**Output Resources**

Name	Description
<b>RunList</b>	List of documents and pages that have been converted.

**6.4.11 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.

**Input Resources**

Name	Description
<b>ImageCompressionParams</b> ? <a href="#">New in JDF 1.1</a>	This resource provides a set of controls that determines how images will be compressed in the resulting “fat” PDL pages.
<b>ImageReplacementParams</b>	Describes the controls selected for the manipulation of images.
<b>RunList</b>	List of page contents on which to perform the selected operation.

**Output Resources**

Name	Description
<b>RunList</b>	List of page contents with images that have been manipulated as indicated by the <b>ImageReplacementParams</b> resource.

## 6.4.12 ImageSetting

The **ImageSetting** process is executed by an imagesetter or platesetter that images a bitmap onto the film or plate media. The **ImageSetting** process may also be used to describe hard copy proofing, (see “Approval” on page 191.)

### Input Resources

Name	Description
<b>ColorantControl</b> ? <a href="#">New in JDF 1.2</a>	The <b>ColorantControl</b> resources that define the ordering and usage of inks during marking on the imagesetter.
<b>DevelopingParams</b> ? <a href="#">New in JDF 1.1</a>	Controls the physical and chemical specifics of the media development process.
<b>ImageSetterParams</b> ? <a href="#">Modified in JDF 1.1</a>	Controls the device specific features of the imagesetter.
<b>Media</b>	The unexposed media.
<b>RunList</b>	Identifies the set of bitmaps to image. May contain bytemaps or images.
<b>TransferCurvePool</b> ? <a href="#">New in JDF 1.1</a>	Area coverage correction and coordinate transformations of the device.

### Output Resources

Name	Description
<b>ExposedMedia</b>	The exposed media resource.

## 6.4.13 Imposition

The **Imposition** process is responsible for combining several pages of input graphical content on to a single surface whose dimensions are reflective of the physical output media. Printer’s marks can be added to the surface in order to facilitate various aspects of the production process. Among other things, these marks are used for press alignment, color calibration, job identification, and as guides for cutting and folding.

Note that the **Imposition** process specifies the task of combining pages and marks on sheets. The task of setting up the parameters needed for **Imposition** (e.g., **Layout**) is defined either by **LayoutPreparation**, **Stripping** or by the generic **ResourceDefinition** process.

There are two mechanisms provided for controlling the flow of page images onto **Media**. The default mechanism, which provides the functionality of **Layout** in PJTF, explicitly identifies all page content for each **Sheet** imaged and references these pages by means of the **Documents** and/or **MarkDocuments** array. Setting the **Automated** attribute of the **Layout** resource to *true* activates a template approach to printing and relies upon the full **Documents** hierarchy to specify the page content to image. Automated impositioning is equivalent to the Print-Layout functionality in PJTF.

In JDF, there is a single **Layout** resource definition. Its structure is broad enough to encompass the needs of both fully specified and template-driven imposition. When described fully, the **Layout** resources include an array of **Signatures**. Each **Signature** in turn specifies an array of **Sheets**, and each **Sheet** can have up to two **Surfaces** (*Front* and *Back*), on which the page images and any marks are to be placed using **PlacedObjects**. A **Sheet** that specifies no **Surface** content will be blank. Pages that are to be printed must be placed onto **Surfaces** using **ContentObject** subelements which explicitly identify the page (via the *Ord* attribute which specifies an index into the document **RunList**). Thus, the **Layout** hierarchy specifies explicitly which pages will be imaged.

When describing automated imposition, **Layout** resources specify a single **Signature** of **Sheet(s)** where page contents are imaged. The (virtual) sequence of pages which is to be imaged via automated layout is defined by the **Document RunList**. Pages are drawn in order from this sequence to satisfy the **ContentObjects** in the **Surfaces** for the **Signature** in the **Layout**, and the **Signature** is repeated until all pages of the sequence are consumed. Each time the **Signature** is repeated, pages are consumed in “chunks”. The total of number of pages per iteration of the **Layout** is determined by the value of *MaxOrd* + 1 (if present in the **Layout**), or by the largest *Ord* value or calculated *OrdExpression* value for any **ContentObject** in the **Signature** (if *MaxOrd* is absent).

Attributes of the **Media** are given for each **Sheet** used in printing. Because the same **Signature** is repeated until all pages are consumed, the **Layout** hierarchy can provide hints or preferences about special needs for sets of page content via **InsertSheet** elements. Inserting media is a way to separate sections of the document content. Thus alternate content is printed only as necessary to fill areas which would normally have page content because new media has been added or to designate where a document section will begin as specified by the odd or even position of the **Signature**.

In a JDF model, imposition is defined separately from other processes, which may precede or follow it. A *Combined* node may combine **Imposition** with other processes (e.g., **Separation** or **Interpreting**) to describe a device that happens to perform both in a single execution module.

### Input Resources

Name	Description
<b>Layout</b>	A <b>Layout</b> resource that indicates how the content pages from the Document <b>RunList</b> and marks from the Marks <b>RunList</b> (see below) are combined onto imposed surfaces.
<b>RunList</b> (Document)	Structured list of incoming page contents which is transformed to produce the imposed surface images.
<b>RunList</b> ? (Marks)	Structured list of incoming marks. These are typically printer's marks such as fold marks, cut marks, punch marks, or color bars.

### Output Resources

Name	Description
<b>RunList</b>	Structured list of imposed surfaces. The <i>Type</i> of the <b>LayoutElements</b> must all be <i>Surface</i> . Typically the output <b>RunList</b> will be partitioned by <i>PartIDKeys</i> = " <i>SheetName Side Separation</i> ". If the <b>Imposition</b> process is executed before RIPing, this will generally be consumed by an <b>Interpreting</b> process. In the case of post-RIP <b>Imposition</b> , it will be consumed by <b>DigitalPrinting</b> or <b>ImageSetting</b> .

#### 6.4.14 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.

### Input Resources

Name	Description
<b>InkZoneCalculationParams</b>	Specific information about the printing press geometry (e.g., the number of zones) to calculate the <b>InkZoneProfile</b> .
<b>Layout</b> ? <a href="#">New in JDF 1.1</a>	Specific information about the <b>Media</b> (including type and color) and about the <b>Sheet</b> (placement coordinates on the printing cylinder).
<b>Preview</b>	A low resolution bitmap file representing the content to be printed.
<b>Sheet</b> ? <a href="#">Deprecated in JDF 1.1</a>	Specific information about the <b>Media</b> (including type and color) and about the <b>Sheet</b> (placement coordinates on the printing cylinder). Replaced by <b>Layout</b> in JDF 1.1.
<b>TransferCurvePool</b> ?	Function to apply <b>ContactCopying</b> , <b>DigitalPrinting</b> , and <b>ConventionalPrinting</b> process characteristics (e.g., press, climate, and substrate) under certain standardized circumstances. This function can be used to generate an accurate <b>InkZoneProfile</b> .

### Output Resources

Name	Description
<b>InkZoneProfile</b>	Contains information about ink coverage along and perpendicular to the ink zones for a specific press geometry.

### 6.4.15 Interpreting

The interpreting device consumes page descriptions and instructions for controlling the marking device, (e.g., image-setter, digital printers, CTP, combined digital printing processes, etc.). The parsing of graphical content in the page descriptions produces a canonical display list of the elements to be drawn on each page.

The interpreter may encounter, and must act upon, 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.

#### Input Resources

Name	Description
<b>ColorantControl</b> ? <a href="#">Modified in JDF 1.1</a>	Identifies the color model used by the job.
<b>FontPolicy</b> ?	Describes the behavior of the font machinery in absence of requested fonts.
<b>InterpretingParams</b>	Provides the parameters needed to interpret the PDL pages specified in the <b>RunList</b> resource.
<b>PDLResourceAlias</b> *	These resources allow a JDF to reference resources which are defined in a Page Description Language (PDL). For example, a <b>PDLResourceAlias</b> resource could refer to a font embedded in a PostScript file.
<b>RunList</b>	This resource identifies a set of PDL pages or surfaces which will be interpreted.

#### Output Resources

Name	Description
<b>RunList</b> ? <a href="#">New in JDF 1.2</a>	Pipe of streamed data which represents the results of <b>Interpreting</b> the pages in the <b>RunList</b> . The data is specified in <b>InterpretedPDLData</b> sub-elements. The format and detail of these is implementation specific. In general, it is assumed that the <b>Interpreting</b> and <b>Rendering</b> processes are tightly coupled and that there is no value in attempting to develop a general specification for the format of this data.
<b>InterpretedPDLData</b> ? <a href="#">Deprecated in JDF 1.2</a>	Pipe of streamed data which represents the results of <b>Interpreting</b> the pages in the <b>RunList</b> . In JDF 1.2 and beyond, a <b>RunList</b> with <b>InterpretedPDLData</b> subelements describes the output content data for <b>Interpreting</b> .

### 6.4.16 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.

#### Input Resources

Name	Description
<b>LayoutElement</b> *	Metadata about the PDL or application file, bitmap image file, text file, vector graphics file, etc.

#### Output Resources

Name	Description
<b>LayoutElement</b> ?	A URL of the PDL or application file is produced by this process. Only one of <b>LayoutElement</b> or <b>RunList</b> must be specified.
<b>RunList</b> ?	A <b>RunList</b> of <b>LayoutElement</b> resources of <i>ElementType Page</i> or <i>Document</i> is produced if this <b>LayoutElementProduction</b> task is the last process of type <b>LayoutElementProduction</b> .

### 6.4.17 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.

#### Input Resources

Name	Description
<b>LayoutPreparationParams</b>	Set of parameters required to control the <b>LayoutPreparation</b> process.
<b>RunList</b> ? (Document) <a href="#">Modified in JDF 1.2</a>	List of documents and/or pages that will be input into the layout. Note that this Runlist is for information only and not modified by the <b>LayoutPreparation</b> process.
<b>RunList</b> ? (Marks)	List of marks that will be input into the layout. These are typically printer's marks such as fold marks, cut marks, punch marks, or color bars.

#### Output Resources

Name	Description
<b>Layout</b>	The layout of the document to be imposed.
<b>RunList</b> (Marks) ?	List of marks that may be used as input of the following <b>Imposition</b> process.
<b>TransferCurvePool</b> ?	Definition of the transfer curves and coordinate systems of the devices.

### 6.4.18 PDFToPSConversion

The **PDFToPSConversion** process controls the generation of PostScript from a single PDF document. This process may be used at any time in a host-based PDF workflow to exit to PostScript for use of tools that consume such data. Additionally, it may be used to actively control the physical printing of data to a device that consumes PostScript data. The JDF model of this may include a **PDFToPSConversion** process in a *Combined* node with a **PSToPDFConversion** process.

#### Input Resources

Name	Description
<b>PDFToPSConversionParams</b>	Set of parameters required to control the generation of PostScript.
<b>RunList</b>	List of documents and pages to be converted to PostScript.

#### Output Resources

Name	Description
<b>RunList</b>	Stream or streams of resulting PostScript code. This PostScript code may end up physically stored in a file or be piped to another process. <b>PDFToPSConversionParams</b> / <i>@GeneratePageStreams</i> determines whether there is a single stream generated for all pages in the <b>RunList</b> or whether each page is generated in to a separate consecutive stream.

### 6.4.19 Preflight

[Modified in JDF 1.2](#)

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 may 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**.



## Input Resources

Name	Description
<b>PreflightParams</b>	A specified list of tests against which documents and/or pages should be tested.
<b>PreflightReportRulePool</b>	A list of rules used to build the <b>PreflightReport</b> . Those rules are attached to actions in the <b>ActionPool</b> .
<b>RunList</b>	The list of documents and/or pages to be preflighted.

## Output Resources

Name	Description
<b>PreflightReport</b>	<b>PreflightReport</b> is a container for logging information that is generated by the <b>Preflight</b> process.

### 6.4.20 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

**Compatibility warning.** Preflight has been modified substantially in JDF 1.2 and is no longer compatible with previous versions of JDF.

Chapter 4.8.2. of the *PostScript Language Reference Manual*).

### Rules for the Generation of the Preview Image

To be useful for the ink consumption calculation, the preview data must be generated with an appropriate resolution. This means not only spatial resolution, but also color or tonal resolution. Spatial resolution is important for thin lines, while tonal resolution becomes important with large areas filled with a certain tonal value. The maximum error caused by limited spatial and tonal resolution should be less than 1%.

### Spatial Resolution

Since some pixel of the preview image might fall on the border between two zones, their tonal values must be split up. In a worst case scenario, the pixels fall just in the middle between a totally white and a totally black zone. In this case, the tonal value is 50%, but only 25% contributes to the black zone. With the resolution of the preview image and the zone width as variables, the maximum error can be calculated using the following equation:

$$\text{error}[\%] = \frac{100}{4 * \text{resolution}[L/mm] * \text{zone\_width}[mm]}$$

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.

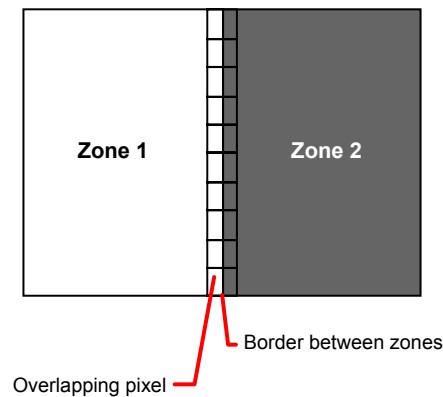


Figure 6.1: Worst case scenario for area coverage calculation

### Tonal Resolution

The kind of error caused by color quantization depends on the number of shades available. If the real tonal value is rounded to the closest (lower or higher) available shade, the error can be calculated using the following equation:

$$error[\%] = \frac{100}{2 * number\_of\_shades}$$

Therefore, at least 64 shades should be used.

### Line Art Resolution

When rasterizing line art elements, the minimal line width is 1 pixel, which means 1/resolution. Therefore, the relationship between the printing resolution and the (spatial) resolution of the preview image is important for these kind of elements. In addition, a specific characteristic of PostScript RIPs adds another error: within PostScript, each pixel that is touched by a line is set. Tests with different PostScript jobs have shown that a line art resolution of more than 300 dpi is normally sufficient for ink-consumption calculation.

### Conclusion

There are quite a few different ways to meet the requirements listed above. The following list includes several examples:

- The job can be 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.

## Input Resources

Name	Description
<b>ColorantControl</b> ? <a href="#">New in JDF 1.1</a>	The <b>ColorantControl</b> resources that define the ordering and usage of inks in print modules. Needed for generating thumbnails.
<b>ExposedMedia</b> ?	The <b>PreviewGeneration</b> process can use an exposed printing plate to produce a <b>Preview</b> resource. This task is performed using an analog plate-scanner. Only one of <b>ExposedMedia</b> , <b>Preview</b> , or <b>RunList</b> must be specified in any <b>PreviewGeneration</b> process.
<b>Preview</b> ? <a href="#">New in JDF 1.1</a>	Medium or low resolution bitmap file that can be used for calculation of overviews and thumbnails. Only one of <b>ExposedMedia</b> , <b>Preview</b> , or <b>RunList</b> must be specified in any <b>PreviewGeneration</b> process.
<b>PreviewGenerationParams</b>	Parameters specifying the size and the type of the preview.
<b>RunList</b> ?	High resolution bitmap data are consumed by the <b>PreviewGeneration</b> process. These data represent the content of a separation that is recorded on a printing plate or other such item. Only one of <b>ExposedMedia</b> , <b>Preview</b> , or <b>RunList</b> must be specified in any <b>PreviewGeneration</b> process.
<b>TransferCurvePool</b> ? <a href="#">New in JDF 1.1</a>	Area coverage correction and coordinate transformations of the device.

## Output Resources

Name	Description
<b>Preview</b>	The <b>Preview</b> 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. <b>Preview</b> may also be used to visualize resources as thumbnail images.

### 6.4.21 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 747 for details of this deprecated process. In JDF 1.2 and beyond, proofing is a combined process.

### 6.4.22 PStoPDFConversion

This section defines the controls required to invoke a device that accepts a PostScript stream and produces a set of PDF pages as output.

## Input Resources

Name	Description
<b>FontParams</b> ?	These parameters determine how the conversion process will handle font errors encountered in the PostScript stream.
<b>ImageCompressionParams</b> ?	This resource provides a set of controls that determines how images will be compressed in the resulting PDF pages.
<b>PStoPDFConversionParams</b> ?	These parameters control the operation of the process that interprets the PostScript stream and produces the resulting PDF pages.
<b>RunList</b>	This resource specifies where the PostScript stream is to be found.

## Output Resources

Name	Description
<b>RunList</b>	This resource identifies the location of the resulting PDF pages.

### 6.4.23 Rendering

The **Rendering** process consumes the display list of graphical elements generated by an interpreter. It color manages and scans/converts the graphical elements according to the geometric and graphic state information contained within the display list. The controls governing the external rendering processes provide overrides and additional parameters for controlling the behavior of the process.

#### Input Resources

Name	Description
<b>Media</b> ? <a href="#">Deprecated in JDF 1.1</a>	This resource provides a description of the physical media which will be marked. The physical characteristics of the media may affect decisions made during <b>Rendering</b> .
<b>RunList</b> ? <a href="#">New in JDF 1.2</a>	Pipe of streamed data that represents the results of <b>Interpreting</b> the pages in the <b>RunList</b> . The data is specified in <b>InterpretedPDLData</b> sub-elements. The format and detail of these is implementation specific. In general, it is assumed that the <b>Interpreting</b> and <b>Rendering</b> processes are tightly coupled and that there is no value in attempting to develop a general specification for the format of this data.
<b>InterpretedPDLData</b> ? <a href="#">Deprecated in JDF 1.2</a>	Pipe of streamed data that represents the results of <b>Interpreting</b> the pages in the <b>RunList</b> . In JDF 1.2 and beyond, a <b>RunList/InterpretedPDLData</b> sub-element describes the input content data for <b>Rendering</b> .
<b>RenderingParams</b> ?	This resource describes the format of the <b>ByteMaps</b> to be created and other specifics of the <b>Rendering</b> process.

#### Output Resources

Name	Description
<b>RunList</b>	Ordered list of rasterized <b>ByteMaps</b> representing pages

### 6.4.24 RIPing

RIPing is a *Combined* process 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 a typical RIP node is of *Type Combined*, as shown in the following example:

```
<JDF Type="Combined" Types="ColorSpaceConversion Interpreting Rendering Screening" ... />
```

It may also be specified as an abstract process group (See “Process Group Nodes” on page 43.) with **JDF/@Types** including the special token “*RIPing*”.

The **RIPing** process consumes page descriptions and instructions for producing the graphical output. It parses the graphical contents in the page descriptions, renders the contents, and produces a rasterized image of the page. This raster may contain contone data and be represented upon output as a **ByteMap**. Alternatively, the **RIPing** process may also perform halftone screening, in which case the output is in the form of a bitmap. It is also responsible for resolving all system resource references that include font handling and resource aliasing.

Instructions read by the RIP include information about the media, halftoning, color transformations, colorant controls and other items that affect that rasterized output. They do not, however, represent any specific controls for the physical output device, nor do they deal with any instructions intended for the finishing device.

The RIP must consider the color information parsed from the graphical content in the page descriptions, evaluate the instructions in the **ColorantControl** resource, and consider the **Media** resource attributes given in **InterpretingParams**. The RIP process may determine, and must perform if required, one or more **ColorSpaceConversion** process iteration(s) in conjunction with an **Interpreting** process.

In most cases, RIPing will be combined with a process that specifies physical marking, (e.g., **DigitalPrinting** or **ImageSetting**). In this case, the interpreter may encounter, and must 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 required before the output can be run through a **ConventionalPrinting** process. In theory, however, a workflow could include no intermediary steps between a **RIPing** process and a **DigitalPrinting** process. The following workflow scenarios represent possible process chains in each circumstance:

RIP→Screening→ImageSetting→ContactCopying→ConventionalPrinting

RIP→(Screening)→DigitalPrinting

Since RIPing is not a predefined JDF process, see the processes that contribute to the RIP for input and output resources.

### 6.4.25 Scanning

The **Scanning** process creates bitmaps from analog images using a scanner.

#### Input Resources

Name	Description
<b>ExposedMedia</b>	Description of the media to be scanned. The <b>ExposedMedia</b> should be partitioned by <i>RunIndex</i> , in order to provide unique mapping from <b>ExposedMedia</b> to the output <b>RunList</b> .
<b>ScanParams</b>	High level scanner settings. These settings are specifically not intended as a replacement for low-level device interfaces such as TWAIN.

#### Output Resources

Name	Description
<b>RunList</b>	List of <b>ByteMap</b> resources or <b>LayoutElement</b> resources of <i>Type = "Image"</i> .

### 6.4.26 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 required 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**.

#### Input Resources

Name	Description
<b>RunList</b>	Ordered list of rasterized <b>ByteMap</b> or <b>InterpretedPDLData</b> representing pages or surfaces.
<b>ScreeningParams</b>	Parameters specifying which halftone mechanism is to be applied and with what specific controls.

#### Output Resources

Name	Description
<b>RunList</b>	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.

### 6.4.27 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 **ByteMaps** is generated as the output. Yet another anticipated combination is with the process to deal with incoming device-dependent data. And finally, it may be combined with an **ImageReplacement** process in order to do image substitution for omitted or proxy images.

#### Input Resources

Name	Description
<b>ColorantControl</b> ? <a href="#">Modified in JDF 1.1A</a>	Identifies which colorants in the job are to be output.
<b>RunList</b>	List of pages that are to be operated on.
<b>SeparationControlParams</b>	Controls for the separation process.

#### Output Resources

Name	Description
<b>RunList</b>	List of separated pages or separated raster bytemaps.

### 6.4.28 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 748 for details of this deprecated process. In JDF 1.2 and beyond, soft proofing is a combined process.

### 6.4.29 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.)

#### Input Resources

Name	Description
<b>StrippingParams</b>	High level structured description of the imposition of one or multiple job parts or part versions.
<b>RunList</b> (Document) ?	List of documents. When available, this list can be used to generate a <b>Layout</b> and populated <b>RunList</b> (no <b>LayoutElements</b> of <i>ElementType</i> = “ <i>Reservation</i> ”) which can be fed into a subsequent <b>Imposition</b> process.
<b>TransferCurvePool</b> ?	Definition of the transfer curves and coordinate systems of the devices. The coordinate system of the <b>StrippingParams</b> coincides with the <b>Layout</b> coordinate system specified in the <b>TransferCurvePool</b> .
<b>Assembly</b> +	Describes how the sections of the different job parts imposed together are combined. If multiple <b>Assembly</b> resources are defined, mapping between <b>StrippingParams</b> and <b>Assembly</b> is achieved by matching the respective <i>JobID</i> and <i>JobPartID</i> attributes. <i>JobPartID</i> must be replaced by <i>AssemblyID</i> (similar to <b>StrippingParams/@SectionList</b> ).

## Output Resources

Name	Description
<b>Layout</b>	The layout of the document to be imposed.
<b>RunList</b> (Document) ?	List of documents that may be used as input of the following <b>Imposition</b> process.
<b>RunList</b> (Marks) ?	List of marks that may be used as input of the following <b>Imposition</b> process.

### Examples

The first example specifies three sheets based on folding catalog example F16-6. More examples can be found in Section V.8, Stripping

```
<StrippingParams ID="FoldCatalogSample" Class="Parameter" Status="Available"
WorkStyle="WorkAndBack" PartIDKeys="SheetName">
  <BinderySignature FoldCatalog="F16-6"/>
  <StrippingParams SheetName="Sheet1"/>
  <StrippingParams SheetName="Sheet2"/>
  <StrippingParams SheetName="Sheet3"/>
</StrippingParams>
```

The following example specifies three sheets: sheet1 and sheet2 are based on a B2x4 **BinderySignature** using the *WorkAndBack* workstyle, while sheet3 is based on **BinderySignature** B2x2 using the *WorkAndTurn* workstyle.

WorkAndBack B2x4				WorkAndTurn B2x2			
8	7	4	11	4	3	2	5
15	0	3	12	7	0	1	6

```
<BinderySignature ID="B2x4" Class="Parameter" Status="Available" NumberUp="4 2" >
  <SignatureCell FrontPages="15 0 3 12" BackPages="14 1 2 13" Orientation="Up"/>
  <SignatureCell FrontPages="8 7 4 11" BackPages="9 6 5 10" Orientation="Down"/>
</BinderySignature>
<BinderySignature ID="B2x2" Class="Parameter" Status="Available" NumberUp="2 2">
  <SignatureCell FrontPages="7 0" BackPages="6 1" Orientation="Up"/>
  <SignatureCell FrontPages="4 3" BackPages="5 2" Orientation="Down"/>
</BinderySignature>
<StrippingParams ID="L1" Class="Parameter" Status="Available" WorkStyle="WorkAndBack"
PartIDKeys="SheetName">
  <StrippingParams SheetName="Sheet1">
    <BinderySignatureRef rRef="B2x4"/>
  </StrippingParams>
  <StrippingParams SheetName="Sheet2">
    <BinderySignatureRef rRef="B2x4"/>
  </StrippingParams>
  <StrippingParams WorkStyle="WorkAndTurn" SheetName="Sheet3">
    <BinderySignatureRef rRef="B2x2"/>
    <Position RelativeBox="0 0 0.5 1"/>
    <Position RelativeBox="0.5 0 1 1" Orientation="Flip180"/>
  </StrippingParams>
</StrippingParams>
```

### 6.4.30 Tiling

The **Tiling** process allows the contents of **Surfaces** to be imaged onto separate pieces of media. Note that many different workflows are possible. **Tiling** must always follow **Imposition**, but it can operate on imposed PDL page contents or on contone or halftone data. **Tiling** will generally be combined with other processes. For example, **Tiling** might be combined with **ImageSetting**. In that case, the input would be a **RunList** that contains **ByteMaps** for each **Surface**.

#### Input Resources

Name	Description
<b>RunList</b> (Surface)	Structured list of imposed page contents or <b>ByteMaps</b> that are to be decomposed to produce the images for each tile. The <i>Type</i> value of <b>LayoutElement</b> resources must all be <i>Surface</i> .
<b>RunList</b> ? (Marks)	Structured list of incoming marks. These are typically printer's marks that provide the information needed to combine the tiles.
<b>Tile</b>	A partitioned <b>Tile</b> resource that describes how the <b>Surface</b> contents are to be decomposed.

#### Output Resources

Name	Description
<b>RunList</b>	Structured list of portions of the decomposed surfaces. The value of the <i>Type</i> attribute of the <b>LayoutElement</b> resources must be <i>Tile</i> .

### 6.4.31 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 may 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 should 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 specifies that a set of document pages should be modified to reduce or (ideally) eliminate visible misregistration errors in the final printed output. The process may be combined with **RIPing** or specified as a stand-alone process.

#### Input Resources

Name	Description
<b>ColorantControl</b> ? <a href="#">Modified in JDF 1.1A</a>	Identifies color model used by the job.
<b>FontPolicy</b> ? <a href="#">New in JDF 1.1</a>	Describes the behavior of the font machinery in absence of requested fonts.
<b>RunList</b>	Structured list of incoming page contents that are to be trapped.
<b>TrappingDetails</b>	Describes the general setting needed to perform trapping.

#### Output Resources

Name	Description
<b>RunList</b>	Structured list of the modified page contents after <b>Trapping</b> 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 may be used to fill other requirements related to printing technology.

### 6.5.1 ConventionalPrinting

[Clarified in JDF 1.2](#)

This process covers several conventional printing tasks, including sheetfed printing, web printing, web/ribbon coating, converting, and varnishing. Typically, each takes place after prepress and before postpress processes. Press machinery often includes postpress processes (e.g., **Folding**, **Numbering**, and **Cutting**) as in-line finishing operations. The **ConventionalPrinting** process itself does not cover these postpress tasks. Using a conventional printing press for producing a pressproof can be performed in the following two ways:

- A proof of type **Component** is produced with a **ConventionalPrinting** process. The result of this process is then sent to the **Approval** process, which in turn produces an **ApprovalSuccess** resource. That resource is then passed on to a second **ConventionalPrinting** process, which requires that the press be set up a second time.
- The *DirectProof* attribute of the **ConventionalPrintingParams** can be used to specify the proof if it is produced during the **ConventionalPrinting** process. In this case, the press need only be set up once.

Note that the definition and ordering of separations is specified by the *DeviceColorantOrder* attribute of the appropriate **ColorantControl** resource.

#### Input Resources

Name	Description
<b>ColorantControl</b> ?	The <b>ColorantControl</b> resources that define the ordering and usage of inks in print modules. The <b>ColorantControl</b> resource specifies the complete set of colors that will be printed on a sheet.
<b>Component</b> ? (Input)	Various components in the form of preprints can be used in <b>ConventionalPrinting</b> in lieu of <b>Media</b> . Examples include waste or a set of preprinted sheets.
<b>Component</b> ? (Proof) <a href="#">Clarified in JDF 1.2</a>	A Proof component is used if a proof was produced during an earlier print run. Note that the proof may be a <b>Component</b> produced in a previous run and must not necessarily have been produced explicitly as a proof. In general, only one of <b>Component</b> (Proof) or <b>ExposedMedia</b> (Proof) should be specified.
<b>ConventionalPrintingParams</b>	Specific parameters to set up the press.
<b>ExposedMedia</b> ? (Proof)	A Proof is used to compare color and content during <b>ConventionalPrinting</b> . This Proof is produced by a prepress proofing device.
<b>ExposedMedia</b> (Plate)	The printing plates and information about them are used to set up the press. The <b>ExposedMedia</b> (Plate) resource defines the set of plates to be used in the press run that is described by this node.

Name	Description
<b>Ink</b> ? <a href="#">Modified in JDF 1.1</a>	Information about the ink (e.g. brand, color) is useful to set up the press.
<b>InkZoneProfile</b> ?	The <b>InkZoneProfile</b> 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.
<b>Layout</b> ? <a href="#">New in JDF 1.1</a>	Sheet and Surface elements from the <b>Layout</b> tree (e.g., <b>CIELABMeasuringField</b> , <b>DensityMeasuringField</b> , or <b>ColorControlStrip</b> ) 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. <b>RegisterMark</b> 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.
<b>Media</b> ?	The physical substrate (e.g., paper or foil) and information about the <b>Media</b> (e.g., thickness, type, and size) are useful in setting up paper travel in the press. This resource must be present if no preprinted <b>Component</b> (Input) resource is used.
<b>PrintCondition</b> ? <a href="#">New in JDF 1.2</a>	Used to control the use of colorants when printing pages on a specific media. The attributes and elements of the <b>PrintCondition</b> resource describe the aim values for a given printing process.
<b>Sheet</b> ? <a href="#">Deprecated in JDF 1.1</a>	Specific information about the <b>Media</b> (including type and color) and about the <b>Sheet</b> , (e.g., placement coordinates on the printing cylinder). Replaced by <b>Layout</b> in JDF 1.1.
<b>TransferCurvePool</b> ? <a href="#">New in JDF 1.1</a>	Area coverage correction and coordinate transformations of the device.

### Output Resources

Name	Description
<b>Component</b> <a href="#">Modified in JDF 1.2</a>	Describes the printed sheets or ribbons which may be used by another printing process or postpress processes. Note that the <i>Amount</i> attribute of the ResourceLink to this resource indicates the number of copies of the entire job which will be produced. Prior to JDF 1.2 this <b>Component</b> was marked with a <i>ProcessUsage</i> = "Good". This is supported but not required in JDF 1.2 and beyond.
<b>Component</b> ? (Waste) <a href="#">Deprecated in JDF 1.2</a>	Produced waste of printed sheets or ribbons. In JDF 1.2 and beyond, <b>ConventionalPrinting</b> produces one <b>Component</b> that is optionally partitioned by <i>Condition</i> .

### 6.5.2 DigitalPrinting

**DigitalPrinting** is a direct printing process that, like **ConventionalPrinting**, occurs after prepress processes but before postpress processes. In **DigitalPrinting**, the data to be printed are not stored on an extra medium (e.g., a printing plate or a printing foil), but instead are stored digitally. The printed image is generated for every output using the digital data. Electrophotography, inkjet, and other technologies are used for transferring ink (both liquid ink and dry toner) onto the substrate. Furthermore, both sheet and web presses can be used as machinery for **DigitalPrinting**.

**DigitalPrinting** is often used to image a small area on preprinted **Components** to perform actions such as addressing or numbering another **Component**. This kind of process can be executed by imaging with an inkjet printer during press, postpress, or packaging operations. Therefore, **DigitalPrinting** is not only a press or prepress operation but sometimes also a postpress process.

Digital printing devices which provide some degree of finishing capabilities (e.g., collating and stapling) as well as some automated layout capabilities (e.g., N-up and duplex printing) may be modeled as a combined process which includes **DigitalPrinting**. Such a combined process may also include other processes, (e.g., **Approval**, **ColorCorrection**, **ColorSpaceConversion**, **ContoneCalibration**, **Cutting**, **Folding**, **HoleMaking**, **ImageReplacement**, **Imposition**, **Interpreting**, **LayoutPreparation**, **Perforating**, **Rendering**, **Screening**, **Stacking**, **Stitching**, **Trapping**, or **Trimming**).

Controls for **DigitalPrinting** are provided in the **DigitalPrintingParams** resource. The set of input resources of a combined process which includes **DigitalPrinting** may be used to represent an Internet Printing Protocol (IPP) job or a PPML job. See Application Notes for IPP and Variable Data printing. Note that putting a label on a product or DropItem is not **DigitalPrinting** but **Inserting**.

## Input Resources

Name	Description
<b>ColorantControl</b> ?	The <b>ColorantControl</b> resources that define the ordering and usage of inks in print modules.
<b>Component</b> * (Input)	Various components can be used in <b>DigitalPrinting</b> instead of <b>Media</b> . Examples include preprinted covers, waste, precut <b>Media</b> , or a set of preprinted sheets or webs. If multiple <b>Component</b> * (Input) resources are linked to one process, the mapping of media to content is defined in the partitions of <b>DigitalPrintingParams</b> .
<b>Component</b> ? (Proof) <a href="#">Clarified in JDF 1.2</a>	A Proof component is used if a proof was produced during an earlier print run, (see description in Section 6.5.1, ConventionalPrinting). Note that the proof may be a <b>Component</b> produced in a previous run and must not necessarily have been produced explicitly as a proof. In general, only one of <b>Component</b> (Proof) or <b>ExposedMedia</b> should be specified.
<b>DigitalPrintingParams</b>	Specific parameters to set up the machinery.
<b>ExposedMedia</b> ?	A <b>Proof</b> is useful for comparisons (completeness, color accuracy) with the print out of the <b>DigitalPrinting</b> process.
<b>Ink</b> ?	Ink or toner and information that is needed for <b>DigitalPrinting</b> .
<b>Layout</b> ? <a href="#">New in JDF 1.1</a>	Sheet and Surface elements from a <b>Layout</b> (e.g., the <b>CIELABMeasuringField</b> , <b>DensityMeasuringField</b> , or <b>ColorControlStrip</b> ) 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. <b>RegisterMarks</b> 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.
<b>Media</b> *	The physical <b>Media</b> and information about the <b>Media</b> (e.g., thickness, type, and size), is used to set up paper travel in the press. This has to be present if no preprinted <b>Component</b> (Input) resource is present. Unprinted <b>Media</b> used for covers are also defined as <b>Media</b> . Note that printing a job on more than one web or sheet at the same time is parallel processing.
<b>PrintCondition</b> ?	Used to control the use of colorants when printing pages on a specific media. The attributes and elements of the <b>PrintCondition</b> resource describe the aim values for a given printing process.
<b>RunList</b>	Rendered data in <b>ByteMaps</b> that will be printed on the digital press are needed for <b>DigitalPrinting</b> . The <b>RunList</b> contains only <b>ByteMaps</b> .
<b>Sheet</b> ? <a href="#">Deprecated in JDF 1.1</a>	Specific information about the <b>Media</b> (including type and color) and about the <b>Sheet</b> (placement coordinates on the printing cylinder). Replaced by <b>Layout</b> in JDF 1.1.
<b>TransferCurvePool</b> ? <a href="#">New in JDF 1.1</a>	Area coverage correction and coordinate transformations of the device.

## Output Resources

Name	Description
<b>Component</b> (Good) <a href="#">Modified in JDF 1.2</a>	Components are produced for other printing processes 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. Prior to JDF 1.2 this <b>Component</b> was marked with a <i>ProcessUsage</i> = "Good". This is supported but not required in JDF 1.2 and beyond.
<b>Component</b> ? (Waste) <a href="#">Deprecated in JDF 1.2</a>	Produced waste, may be used by other processes. In JDF 1.2 and beyond, waste is tracked by partitioning the output using the <i>Condition PartIDKey</i> .

### 6.5.3 IDPrinting

[Deprecated in JDF 1.1](#)

The IDPrinting process was deprecated in JDF/1.1. Instead, implementations should use the **DigitalPrinting** process combined with other processes, thus improving interoperability by reducing one of the combinations of processes. Also the **IDPrinting** process defined a number of resources and subelements which are deprecated since they duplicate other resources. See "IDPrinting" on page 749 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** 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 subelements. The generic **GlueApplication** ABOperation 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).

## Input Resources

Name	Description
<b>Component</b>	The <b>BlockPreparation</b> process consumes one <b>Component</b> and creates a book block.
<b>BlockPreparationParams</b>	Specific parameters to set up the machinery.

## Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the prepared book block. Its <i>ProductType</i> = "BookBlock"

### 6.6.3 BoxPacking

[New in JDF 1.1](#)

A pile, stack or bundle of products can be packed into a box or carton.

#### Input Resources

Name	Description
<b>Component</b>	The <b>BoxPacking</b> process puts a set of Components into the box <b>Component</b> .
<b>BoxPackingParams</b>	Specific parameters to set up the machinery.
<b>Component</b> (Box) ?	Details of the box or carton.

#### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the boxed <b>Component</b> .

### 6.6.4 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.



**Input Resources**

Name	Description
<b>Component</b>	Component to be bundled
<b>BundlingParams</b>	Bundling parameters.
<b>Media</b> ?	End boards to protect the bundle. For each bundle a pair of end boards is needed.

**Output Resources**

Name	Description
<b>Component</b>	The completed bundle.

Parameters like manufacturer and device type are defined in the **Device** element.

**6.6.5 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.

**Input Resources**

Name	Description
<b>Component</b> (CoverMaterial) ?	The cover material which may be either a preprinted and processed sheet of paper. If no <b>Component</b> is specified, a <b>Media</b> (CoverMaterial) must be specified.
<b>CaseMakingParams</b>	Specific parameters to set up the machinery.
<b>Media</b> (CoverMaterial) ?	The <b>CaseMaking</b> process may also consume unprocessed <b>Media</b> as cover material. Only one of <b>Media</b> (CoverMaterial) or <b>Component</b> (CoverMaterial) must be specified.
<b>Media</b> (CoverBoard) <a href="#">Modified in JDF 1.1A</a>	The cardboard <b>Media</b> used for the cover board.
<b>Media</b> (SpineBoard)?	The cardboard <b>Media</b> used for the spine board. If not specified, the same media as used for <b>Media</b> (CoverBoard) is used.

**Output Resources**

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the produced book case. Its <i>ProductType</i> = "BookCase".

**6.6.6 CasingIn**

[New in JDF 1.1](#)

The hard cover book case and the book block are joined in the **CasingIn** process.

**Input Resources**

Name	Description
<b>Component</b>	The prepared book block.
<b>Component</b> (Case)	The hard cover book case.
<b>CasingInParams</b>	Specific parameters to set up the machinery.

**Output Resources**

Name	Description
<b>Component</b>	One <sup>a</sup> <b>Component</b> 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.

### 6.6.7 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.

#### Input Resources

Name	Description
<b>Component</b> (BookBlock)	The operation requires one component: the block of sheets to be bound.
<b>Component</b> ? (Cover)	The empty cover with the U-shaped clamp that might, for example, have been printed before it is used during the <b>ChannelBinding</b> process.
<b>ChannelBindingParams</b>	Specific parameters to set up the machinery.

#### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the channel-bound component forming an item such as a brochure.

### 6.6.8 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.

#### Input Resources

Name	Description
<b>Component</b>	The operation requires one component: the pile of prepunched sheets often including a top and button cover.
<b>CoilBindingParams</b>	Specific parameters to set up the machinery.

#### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the coil-bound component forming an item such as a calendar.

### 6.6.9 Collecting

This process collects folded sheets or partial products, some of which may 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).

**Input Resources**

Name	Description
<b>CollectingParams</b> ?	Specific parameters to set up the machinery.
<b>Component</b> +	Variable amount of sheets to be collected.
<b>DBRules</b> *	Database input that describes which sheets should 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.
<b>DBSelection</b> ?	Database input that describes which sheets should be collected for a particular instance component.
<b>IdentificationField</b> ? <a href="#">Deprecated in JDF 1.2</a>	Information about identification marks on the component. In JDF 1.2 and beyond, this information is defined in the <b>Component</b> itself.

**Output Resources**

Name	Description
<b>Component</b>	A block of collected sheets is produced. This <b>Component</b> can be joined in further postpress processes.

**6.6.10 CoverApplication**[New in JDF 1.1](#)**CoverApplication** describes the process of applying a soft cover to a book block.**Input Resources**

Name	Description
<b>CoverApplicationParams</b>	Specific parameters to set up the machinery.
<b>Component</b>	The book block on which the cover is applied
<b>Component</b> (Cover)	The soft cover that is applied.

**Output Resources**

Name	Description
<b>Component</b>	The book block with the applied soft cover.

**6.6.11 Creasing**[New in JDF 1.1](#)

Sheets are creased or grooved to enable folding or to create even, finished page delimiters.

**Input Resources**

Name	Description
<b>Component</b> <a href="#">Modified in JDF 1.2</a>	This process consumes one <b>Component</b> : the printed sheets. Note that prior to JDF 1.2 this <b>Component</b> was optional, which was clearly a typing mistake in the specification.
<b>CreasingParams</b>	Details of the <b>Creasing</b> process.

**Output Resources**

Name	Description
<b>Component</b>	One creased <b>Component</b> is produced.



### 6.6.12 Cutting

Sheets are cut using a guillotine **Cutting** machine. Before **Cutting**, the sheets might be jogged and buffered. **CutBlocks** and or **CutMarks** 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 **CutBlock** elements specified do not directly imply a certain cutting sequence. Therefore, a sequence must be determined by a specialized agent.

#### Input Resources

Name	Description
<b>Component</b> ?	This process consumes one <b>Component</b> : the printed sheets.
<b>CutBlock</b> * <a href="#">Deprecated in JDF 1.1</a>	One or several <b>CutBlocks</b> can be used to find the <b>Cutting</b> sequence. Only one of <b>CutBlock</b> or <b>Cut</b> may be specified.
<b>CutMark</b> * <a href="#">Deprecated in JDF 1.1</a>	<b>CutMark</b> resources can be used to adapt the theoretical cut positions to the real positions of the corresponding blocks on the <b>Component</b> to be cut.
<b>CuttingParams</b> <a href="#">New in JDF 1.1</a>	Details of the <b>Cutting</b> process.
<b>Media</b> ?	Cutting can be performed to unprinted <b>Media</b> in order to adjust size or shape.

#### Output Resources

Name	Description
<b>Component</b> +	One or several blocks of cut components are produced. When <b>Media</b> are cut, the output <b>Components</b> can be input resources for processes such as <b>ConventionalPrinting</b> .

### 6.6.13 Dividing

[Deprecated in JDF 1.1](#)

**Dividing** has been replaced by **Cutting**. See “Dividing” on page 750 for details of this deprecated process.

### 6.6.14 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.

#### Input Resources

Name	Description
<b>Component</b>	This process consumes one <b>Component</b> :
<b>EmbossingParams</b>	Parameters to setup the machinery.
<b>Media</b> ?	If foil stamping or foil embossing, the stamping foil material is required.
<b>Tool</b> ?	The embossing stamp or calendar.

#### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is created.

### 6.6.15 EndSheetGluing

**EndSheetGluing** finalizes the folded **Sheet** or book block in preparation for case binding. It requires three **Component**s – 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.

#### Input Resources

Name	Description
<b>Component</b> (BackEndSheet)	A back-end sheet to be mounted on the book block.
<b>Component</b> (BookBlock)	A back-end sheet and a front-end sheet are glued onto the book block.
<b>Component</b> (FrontEndSheet)	A front-end sheet to be mounted on the book block.
<b>EndSheetGluingParams</b>	Specific parameters to set up the machinery.

#### Output Resources

Name	Description
<b>Component</b>	A book block is produced that includes the end sheets.

### 6.6.16 Feeding

[New in JDF 1.2](#)

The **Feeding** process separates sheets or signatures from a stack or stream and feeds single **Component**(s) to processes such as **Folding**, **Gathering**, **Collecting**, **ConventionalPrinting**, etc. In general, the **Feeding** process will be combined with the processes that consume the feed of **Component**(s) or **Media**.

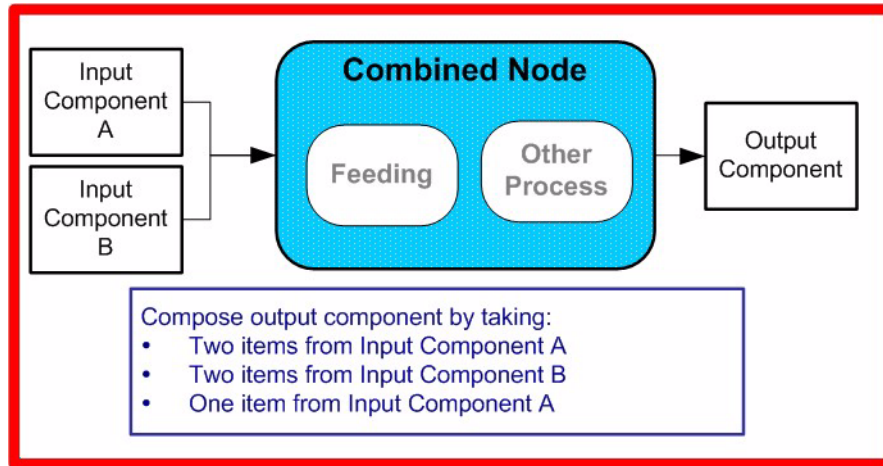
#### Input Resources

Name	Description
<b>Component</b> *	Sheets or signatures to be fed to the machinery. The <i>ProcessUsage</i> of the <b>Component</b> may be specified as any valid <i>ProcessUsage</i> of the a feed consuming process.
<b>FeedingParams</b>	Specific parameters to set up the Feeding Process
<b>Media</b> *	Media to be fed to the feeder machinery.

#### Output Resources

Name	Description
<b>Component</b> *	<b>Component</b> (s) fed to the consuming process.
<b>Media</b> *	<b>Media</b> fed to the consuming process.

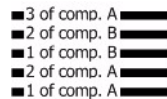
When used in a combined process with feed consuming process (e.g., **Gathering**), the **Feeding** process allows an arbitrary complex selection of input **Component** elements in any number, and in any order, as long as elements are consumed consecutively, (i.e., no random access within a single input component).



In our example above, one input component is a bundle component (*BundleType = Stack*) consisting of a collated set of three sheets, the other one is a collated set consisting of two sheets per set. Both sets are oriented face-up:



The output would then look like this in case of **Gathering**.



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. If required, sheets need to be flipped via orientation settings set in **FeedingParams/CollatingItem/@Orientation**.

### 6.6.17 Folding

Buckle folders or knife folders are used for **Folding** sheets. One or more sheets can be folded at the same time. Web presses often provide in-line **Folding** equipment. Longitudinal **Folding** is often performed using a former, a plow folder, or a belt, while jaw folding, chopper folding, or drum folding equipment is used for folding the sheets that have been divided.

The JDF **Folding** process covers both operations done in stand-alone **Folding** machinery – typically found for processing sheet fed printed materials – and in-line equipment of web printing presses. Creasing and/or slot perforating are sometimes necessary parts of the **Folding** operation that guarantee exact process execution. They depend on the folder used, the **Media**, and the folding layout. These operations are specified in the **Creasing** and **Perforating** processes respectively.

#### Input Resources

Name	Description
<b>Component</b>	<b>Components</b> , including a printed sheet or a pile of sheets, are used in the <b>Folding</b> process.
<b>FoldingParams</b>	Specific parameters to set up the machinery.

#### Output Resources

Name	Description
<b>Component</b> <a href="#">Modified in JDF 1.1</a>	The process produces a <b>Component</b> , which in most cases is a folded <b>Sheet</b> .

### 6.6.18 Gathering

In the **Gathering** process, ribbons, sheets, or other **Components** are accumulated on a pile that will eventually be stitched or glued in some way to create an individual **Component**. The input **Components** 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.



#### Input Resources

Name	Description
<b>Component +</b>	Variable amount of components including single sheets or folded sheets are used in the <b>Gathering</b> process. The first <b>Component</b> in the list lies at the bottom of the gathered pile.
<b>GatheringParams</b>	Specific parameters to set up the machinery.
<b>DBRules *</b>	Database input that describes which sheets should 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.
<b>DBSelection ?</b>	Database input that describes which sheets should be gathered for a particular instance component.
<b>IdentificationField ?</b> <a href="#">Deprecated in JDF 1.2</a>	Information about identification marks on the component. In JDF 1.2 and beyond, this information is defined in the <b>Component</b> itself.

#### Output Resources

Name	Description
<b>Component</b>	Components gathered together, (e.g., a pile of folded sheets).

### 6.6.19 Gluing

[New in JDF 1.1](#)

**Gluing** describes arbitrary methods of applying glue to a **Component**.

#### Input Resources

Name	Description
<b>Component</b>	This process consumes one <b>Component</b> : the printed sheets.
<b>GluingParams</b>	Details of the <b>Gluing</b> process.

#### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced, the input <b>Component</b> with glue applied to it.

### 6.6.20 HeadBandApplication

[New in JDF 1.1](#)

Head bands are applied to the hard cover book block.

#### Input Resources

Name	Description
<b>Component</b>	The prepared book block.
<b>HeadBandApplicationParams</b>	Specific parameters to set up the machinery.

## Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the hard cover block with head bands.

### 6.6.21 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.

#### Input Resources

Name	Description
<b>Component</b>	One <b>Component</b> (e.g., a printed sheet or a pile of sheets) are modified in the <b>HoleMaking</b> process.
<b>HoleMakingParams</b>	Specific parameters, including hole diameter and positions, used to set up the machinery.

#### Output Resources

Name	Description
<b>Component</b>	A <b>Component</b> with holes (e.g., a book block or a single sheet) is produced for further postpress processes.

### 6.6.22 Inserting

This process can be performed at several stages in postpress. The process can be used to describe the labeling of products, of packages, or the gluing-in of a **Component**, (e.g., a card, sheet, or CD-ROM). Two **Components** 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*” **Components**. 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.

#### Input Resources

Name	Description
<b>Component</b> (Mother)	Designates where to insert the child <b>Component</b> .
<b>Component</b> (Child)	The <b>Component</b> to be inserted in the mother <b>Component</b> .
<b>InsertingParams</b>	Specific parameters (e.g., placement) to set up the machinery.
<b>DBRules</b> ?	Database input that describes whether the child should be inserted for a particular instance <b>Component</b> . In this version the schema is only human readable text.
<b>DBSelection</b> ?	Database input that describes whether the child should be inserted for a particular instance <b>Component</b> .
<b>IdentificationField</b> ? <a href="#">Deprecated in JDF 1.2</a>	Information about identification marks on the <b>Component</b> . In JDF 1.2 and beyond, this information is defined in the <b>Component</b> itself.

#### Output Resources

Name	Description
<b>Component</b>	A mother <b>Component</b> is produced containing the inserted child <b>Component</b> .

### 6.6.23 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**.

#### Input Resources

Name	Description
<b>JacketingParams</b>	Specific parameters to set up the machinery.
<b>Component</b> (Book)	The book that the jacket is wrapped around.
<b>Component</b> (Jacket)	The description of the jacket.

#### Output Resources

Name	Description
<b>Component</b>	The jacketed book.

### 6.6.24 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.

#### Input Resources

Name	Description
<b>Component</b>	The <b>Labeling</b> process labels one <b>Component</b> with a set of labels.
<b>Component</b> (Label) ?	The label to be attached to the <b>Component</b> .
<b>LabelingParams</b>	Specific parameters to set up the machinery.

#### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the labeled <b>Component</b> .

### 6.6.25 Laminating

In the **Laminating** process, a plastic film is bonded to one or both sides of a **Component**'s media, and adhered under pressure with either a thermal setting or pressure sensitive adhesive.

#### Input Resources

Name	Description
<b>Component</b>	A Component is required for <b>Laminating</b> .
<b>LaminatingParams</b>	Specific parameters to set up the machinery.
<b>Media</b> ?	The laminating foil material.

#### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the laminated component.

## 6.6.26 LongitudinalRibbonOperations

[Deprecated in JDF 1.1](#)

In version 1.1 of JDF and beyond, in-line finishing is described using the “standard” finishing processes (e.g., **Creasing**, **Cutting**, **Folding**) or in a combined node with **ConventionalPrinting**. See “LongitudinalRibbonOperations” on page 751 for details of this deprecated process.

## 6.6.27 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 required, and the counters automatically increase incrementally. **Numbering** is also used for alphanumeric, automatic, and unique marking.

### Input Resources

Name	Description
<b>Component</b>	One <b>Component</b> (e.g., a printed sheet or a pile of sheets) are modified in the <b>Numbering</b> process.
<b>NumberingParams</b>	Specific parameters to set up the machinery.

### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the numbered sheet.

## 6.6.28 Palletizing

[New in JDF 1.1](#)

Bundles, stacks, piles or boxes can be loaded onto a palette.

### Input Resources

Name	Description
<b>Component</b>	The <b>Palletizing</b> process describes placing the bundle that is represented by the <b>Component</b> onto a palette.
<b>PalletizingParams</b>	Specific parameters to set up the machinery.
<b>Pallet</b>	The palette.

### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the loaded palette.

## 6.6.29 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**.

### Input Resources

Name	Description
<b>Component</b>	This process consumes one <b>Component</b> : the printed sheets.
<b>PerforatingParams</b>	Details of the <b>Perforating</b> process.

### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced.

### 6.6.30 PlasticCombBinding

In the **PlasticCombBinding** process, a plastic insert wraps through prepunched holes in the substrate. Most often, these holes are rectangular and elongated. After the plastic comb is opened with a special tool, the prepunched block of sheets – often together with a top and button cover – is inserted onto the “teeth” of the plastic comb. When released from the machine, the teeth return to their original cylindrical positions with the points tucked into the backside of the spine area. Special machinery can be used to reopen the plastic comb binding.

#### Input Resources

Name	Description
<b>Component</b>	The operation requires one component: the pile of sheets often including a top and button cover.
<b>PlasticCombBindingParams</b>	Specific parameters to set up the machinery.

#### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the plastic-comb-bound component forming an item such as a calendar.

### 6.6.31 PrintRolling

[New in JDF 1.2](#)

The single products like sheets, signatures, or partial products are rolled. The roll is the output component of the process and is used to store the products. As input component of a consuming process (e.g., **Gathering**, **Collecting**, and **Inserting**), the single components of a roll are used.



#### Input Resources

Name	Description
<b>Component</b>	<b>Component</b> to be rolled.
<b>PrintRollingParams ?</b>	Print rolling parameters.
<b>RollStand ?</b>	Rollstand to store the <b>Component(s)</b> as rolls.

#### Output Resources

Name	Description
<b>Component</b>	The print roll.

### 6.6.32 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.



**Input Resources**

Name	Description
<b>Component</b> (BookBlock)	The operation requires one component: the pile of prepunched sheets to be inserted into the ring binder.
<b>Component</b> ? (RingBinder)	The empty ring binder that might have been printed, for example, before it is used during the <b>RingBinding</b> process.
<b>RingBindingParams</b>	Specific parameters to set up the process/machinery.

**Output Resources**

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the ring-bound component forming an item such as a calendar.

**6.6.33 SaddleStitching**

[Deprecated in JDF 1.1](#)

**SaddleStitching** has been replaced by **Stitching** in JDF 1.1. See “SaddleStitching” on page 751 for details of this deprecated process.

**6.6.34 ShapeCutting**

[New in JDF 1.1](#)

The **ShapeCutting** process can be performed using tools such as hollow form punching, perforating, or die-cutting equipment.

**Input Resources**

Name	Description
<b>Component</b>	This process consumes one <b>Component</b> : the printed sheets.
<b>ShapeCuttingParams</b>	Details of the <b>ShapeCutting</b> process.
<b>Tool</b> ?	The die-cut die.

**Output Resources**

Name	Description
<b>Component</b>	One <b>Component</b> is produced.

**6.6.35 Shrinking**

[New in JDF 1.1](#)

Shrink-wrap foil must be treated in order to shrink.

**Input Resources**

Name	Description
<b>Component</b>	The <b>Shrinking</b> process shrinks the shrink-wrap that is wrapped around a bundle. The bundle including the shrink-wrap media is represented by this <b>Component</b> .
<b>ShrinkingParams</b>	Specific parameters to set up the machinery.

**Output Resources**

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the bundle including bundle including the shrunk shrink-wrap media.

### 6.6.36 SideSewing

[Deprecated in JDF 1.1](#)

Replaced by *ThreadSewing*. See “SideSewing” on page 751 for details of this deprecated process.

### 6.6.37 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).

#### Input Resources

Name	Description
<b>Component</b>	The raw book block.
<b>SpinePreparationParams</b>	Specific parameters to set up the machinery.

#### Output Resources

Name	Description
<b>Component</b>	The book block with a processed spine.

### 6.6.38 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.

#### Input Resources

Name	Description
<b>Component</b>	The book block that the spine is taped to.
<b>SpineTapingParams</b>	Specific parameters to set up the machinery.

#### Output Resources

Name	Description
<b>Component</b>	The book block with the spine.

### 6.6.39 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.

#### Input Resources

Name	Description
<b>Component</b>	The <i>Stacking</i> process consumes one <b>Component</b> and stacks it onto a stack.
<b>StackingParams</b>	Specific parameters to set up the machinery.

#### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the stack of input Components.

## 6.6.40 Stitching

[Clarified in JDF 1.2](#)

Gathered or collected sheets or signatures are stitched together with a cover.

### Input Resources

Name	Description
<b>Component</b>	The only required <b>Component</b> is the pile of gathered sheets, including the cover.
<b>StitchingParams</b>	Specific parameters to set up the machinery.

### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the gathered or collected sheets including the cover stitched together.

Components containing staples of different characteristics like shape, width, etc. are defined by a combined process. For example:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="CombinedStitch" JobID="Stitching
special" Type="Combined" Types="Stitching Stitching" Version="1.2">
  <ResourcePool>
    <StitchingParams Class="Parameter" ID="Stitch1" NumberOfStitches="2"
StapleShape="Butted" Status="Available" StitchPositions="100 700" StitchWidth="28.3"
WireBrand="Steel" WireGauge="2.3"/>
    <StitchingParams Class="Parameter" ID="Stitch2" NumberOfStitches="2"
StapleShape="Eyelet" Status="Available" StitchPositions="300 500" StitchWidth="42.5"
WireBrand="Steel" WireGauge="2.3"/>
  </ResourcePool>
  <ResourceLinkPool>
    <StitchingParamsLink CombinedProcessIndex="0" Usage="Input" rRef="Stitch1"/>
    <StitchingParamsLink CombinedProcessIndex="1" Usage="Input" rRef="Stitch2"/>
  </ResourceLinkPool>
</JDF>
```

## 6.6.41 Strapping

[New in JDF 1.1](#)

A bundle can 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)).

### Input Resources

Name	Description
<b>Component</b>	The <b>Strapping</b> process puts straps around a bundle that is represented by a <b>Component</b> .
<b>Strap ?</b>	The straps used.
<b>StrappingParams</b>	Specific parameters to set up the machinery.

### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the strapped <b>Component</b> .

## 6.6.42 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.

**Input Resources**

Name	Description
<b>Component</b>	The operation requires one component: the block of sheets to be bound.
<b>StripBindingParams</b>	Specific parameters to set up the machinery.

**Output Resources**

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the strip-bound component forming an item such as a book.

**6.6.43 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.

**Input Resources**

Name	Description
<b>Component</b>	This process consumes one <b>Component</b> : the printed sheets.
<b>ThreadSealingParams</b>	Details of the <b>ThreadSealing</b> process.

**Output Resources**

Name	Description
<b>Component</b>	One <b>Component</b> is produced.

**6.6.44 ThreadSewing**

This process may 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 may also be necessary if different types of paper are used.

**Input Resources**

Name	Description
<b>Component</b>	The operation requires one component: the gathered sheets.
<b>ThreadSewingParams</b>	Specific parameters to set up the machinery.

**Output Resources**

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the thread-sewn components forming an item such as a raw book block.

**6.6.45 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 may deliver the book blocks to the trimmer. A *Combined* operation in the trimming machinery would then execute a cut at the front, head, and tail in a cycle of two operations. Closed edges of folded signatures would then be opened while the book block is trimmed to its predetermined dimensions.

Some trimming machines such as magazine production systems can produce N-ups. In every case, however, the additional trimming cuts that divide the N-ups result in separated book blocks. Sometimes a stripe is trimmed out between the book blocks. To describe these operations, multiple **Trimming** processes must be defined in JDF.

**Input Resources**

Name	Description
<b>Component</b> <a href="#">Modified in JDF 1.2</a>	The bound book block or sheet that will be trimmed.
<b>TrimmingParams</b>	Specific parameters (e.g., trim size) to set up the machinery.

**Output Resources**

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the trimmed component.

**6.6.46 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.

**Input Resources**

Name	Description
<b>Component</b>	The operation requires one component: the pile of preprinted sheets often including a front and back cover.
<b>WireCombBindingParams</b>	Specific parameters to set up the machinery.

**Output Resources**

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the wire-comb bound component forming an item such as a calendar.

**6.6.47 Wrapping**

[New in JDF 1.1](#)

Single products, bundles, or pallets can be wrapped by film or paper.

**Input Resources**

Name	Description
<b>Component</b>	The <b>Wrapping</b> process wraps a bundle that is represented by a <b>Component</b> .
<b>WrappingParams</b>	Specific parameters to set up the machinery.
<b>Media ?</b>	The wrapping material.

**Output Resources**

Name	Description
<b>Component</b>	One <b>Component</b> is produced: the wrapped <b>Component</b> .

**6.6.48 Postpress Processes Structure**

[Modified in JDF 1.2](#)

**6.6.48.1 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.

### 6.6.48.1.1 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**

### 6.6.48.1.2 Block Joining

The block joining processes can be grouped into two major subcategories: conventional binding methods, which includes the processes of **Stitching**, **CoverApplication**, **SpinePreparation**, **SpineTaping**, **ThreadSealing**, and **ThreadSewing**; and single-leaf binding methods, which are listed in Section 6.6.48.1.2.1. 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**

#### 6.6.48.1.2.1 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.

#### 6.6.48.1.2.2 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**

#### 6.6.48.1.2.2.1 Mechanical Binding Methods

Single leafs are fastened into what is essentially a permanent system that is not meant to be reopened. However, special machinery can be used to reopen some of the mechanical binding systems described below.

In mechanical binding, printing and folding can be done in a conventional manner. The gathered sheets, however, often require the back to be trimmed, as well as the other three sides. Mechanical bindings are often used for short-run jobs such as ones that have been printed digitally. The most prominent mechanical binding processes are described in the sections that follow. Mechanical binding methods include:

- **ChannelBinding**
- **CoilBinding**
- **PlasticCombBinding**
- **RingBinding**
- **StripBinding**
- **WireCombBinding**

### 6.6.48.2 HoleMaking

- **HoleMaking**

### 6.6.48.3 Laminating

- *Laminating.*

### 6.6.48.4 Numbering

- *Numbering.*

### 6.6.48.5 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:

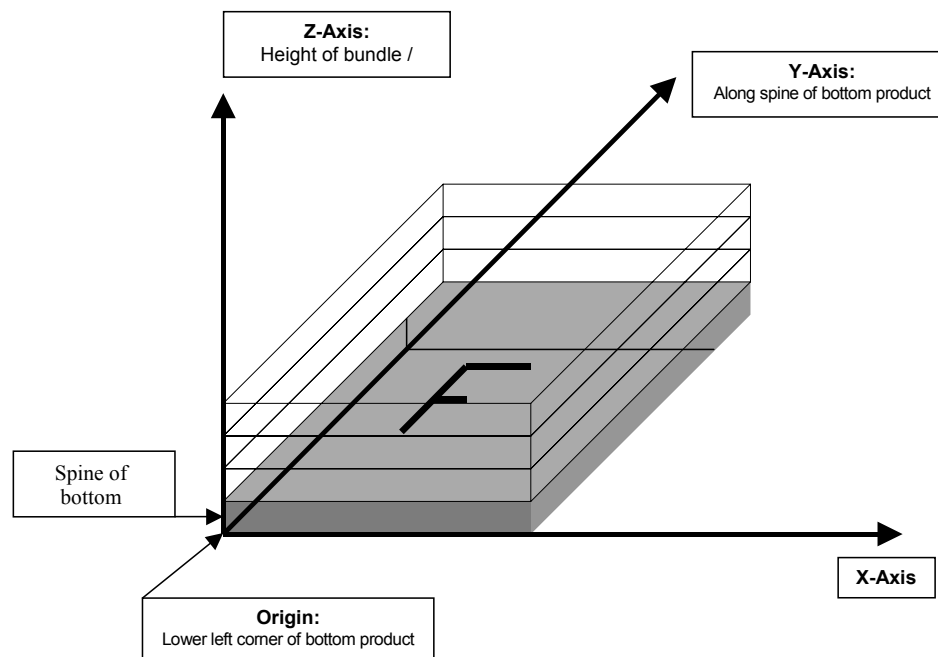


Figure 6.2: Packaging Process Coordinate System

### 6.6.48.6 Processes in Hardcover Book Production

The following processes refer to the production of hard cover books. As there are several processes which are needed to produce a hardcover, some of them are optional, others are essential. The processes described are in detail:

**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 here. Application of kraft paper to the book block is described in the *SpineTaping* process.

**CasingIn:** In this process, the case and the prepared book block are brought together.

**Jacketing:** In the jacketing process, the jacket is wrapped around the hardcover book.

Processes in hardcover book production include:

- *BlockPreparation*
- *CaseMaking*
- *CasingIn*
- *Collecting*

- **Gluing**
- **HeadBandApplication**
- **Jacketing**
- **SpinePreparation**
- **SpineTaping**
- **ThreadSealing**
- **ThreadSewing**

#### 6.6.48.7 Sheet Processes

Many printing processes produce sheets that must be 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**

#### 6.6.48.8 Tip-on/in

The following processes, **EndSheetGluing**, **Inserting**, are part of the postpress operations. They can be grouped together as the tip-on/in processes. Both processes can be performed by hand, tip-on/in machine, or by a press. Tip-on/in includes:

- **EndSheetGluing**
- **Inserting**

#### 6.6.48.9 Trimming

- **Trimming.**

#### 6.6.48.10 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 3.6, *Resources*. 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”. The characteristics of the product that are not specified through the use of intent resources will be selected by the system that processes the intent resources. The system that processes the intent data in a JDF job ticket may insert the details of its selection into the JDF data for the job. See “Conformance Requirements for Support of Attributes and Attribute Values” on page 7 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. Section 7.1.1, *Intent Resource Span Subelements* below, 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 3.6.1, *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. In some cases, a resource will contain an element with more than one value associated with it. If this is the case, the element name is listed as often as it appears, and a term in parentheses that identifies the kind of element is included in the column. A template of these tables is also provided below.

### Resource Properties Template

<b>Resource class:</b>	Defines the resource class.
<b>Resource referenced by:</b>	List of parent resources that contain elements of this type. Only valid for elements.
<b>Process Pairing:</b>	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 printed product. This is a list of the primary resources and not a complete list.
<b>Example Partition:</b>	List of recommended partitioning keys: For a complete list of partition keys, see the description of <i>PartIDKeys</i> in Table 3-27, “Contents of the Partitionable Resource Element,” on page 78. Note that resources may be partitioned by keys that are not specified in this list.
<b>Input of processes:</b>	List of node types that create the resource as an input resource.
<b>Output of processes:</b>	List of node types that create the resource as an output resource.

### Resource Structure Template

Name	Data Type	Description
Name of attribute	data type of attribute	Usage of the attribute.
Name of element	element	Subelements that must be defined locally within the resource.
Name of element	refelement	Elements that are based on other atomic resources or resource elements. These may either be in-line elements or instances of ResourceRef elements (see Section 3.7.6, <i>Inter-Resource Linking Using ResourceRef</i> ). In case of ResourceRef elements, a “Ref” must be appended to the name specified in the table column entitled “Name”.
<b>FileSpec</b> ( <i>ResourceUsage</i> )	refelement	<b>FileSpec</b> resources may have a <b>FileSpec/@ResourceUsage</b> attribute that specifies the context in which to use the <b>FileSpec</b> . <b>FileSpec/@ResourceUsage</b> must match the ( <i>ResourceUsage</i> ) value specified in the parentheses.

### 7.1.1 Intent Resource Span Subelements

Intent resources contain subelements that allow spans of values to be specified. These subelements also provide mechanisms to select a set of values from the provided range and map them to a set of quotes. These subelements are called span elements. The abstract span element to be used is determined by the data type of the values to be recorded. Span elements are defined to facilitate negotiation between buyer and provider as defined in Section 4.1.2, Defining Business Objects Using Intent Resources.

**Note:** The tables used below to define the structure of intent resources may contain attributes with both JDF data types (as defined in the "Data Structures" on page 9 and "Defined JDF enumeration Data Types" on page 575) as well as the following span elements (used in the data type column of the intent resource's structure table). All possible span elements are listed in the following table. The formatting of *XXXSpan* elements was chosen to be the same as *attribute* formatting although the Span elements are technically XML elements because the semantic usage of the Span elements is equivalent to the usage of attributes in process resources.

Each span element contains further attributes or subelements. The contents shared by all span elements are listed in Section 7.1.1.1, Structure of Abstract Span Subelement, below, and the contents particular to each span element type are described in the sections that follow.

Span Element Types	Data Type	Description
DurationSpan <a href="#">New in JDF 1.1</a>	element	Describes a set of duration values.
EnumerationSpan	element	Describes a set of enumeration values.
IntegerSpan	element	Describes a numerical range of integer values.
NameSpan	element	Describes a set of NMTOKEN values.
NumberSpan	element	Describes a numerical range of values.
OptionSpan	element	Describes an intent in which the principal information is that a specific option is requested.
ShapeSpan <a href="#">New in JDF 1.1</a>	element	Describes a set of shape values.
StringSpan	element	Describes a set of string values.
TimeSpan	element	Describes a set of dateTime values.
XYPairSpan	element	Describes a set of XYPair values.

#### 7.1.1.1 Structure of Abstract Span Subelement

Abstract span elements of intent resources have a common set of attributes and elements that define the priority, data type, and requested identity of the element. These attributes are described in the following table. In addition, abstract Span elements have three attributes that define the 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.

*Preferred* – A preferred value defined by the recipient of the product.

*Range* – A proposed range of values defined by the recipient of the product.

Name	Data Type	Description								
DataType	enumeration	Describes the data type of the span element within an intent resource. This attribute is provided for applications that do not have access to schema validation. Possible values are:								
		<table border="0"> <tr> <td><i>DurationSpan</i></td> <td><i>OptionSpan</i></td> </tr> <tr> <td><i>EnumerationSpan</i></td> <td><i>ShapeSpan</i></td> </tr> <tr> <td><i>IntegerSpan</i></td> <td><i>StringSpan</i></td> </tr> <tr> <td><i>NameSpan</i></td> <td><i>TimeSpan</i></td> </tr> <tr> <td><i>NumberSpan</i></td> <td><i>XYPairSpan</i></td> </tr> </table>	<i>DurationSpan</i>	<i>OptionSpan</i>	<i>EnumerationSpan</i>	<i>ShapeSpan</i>	<i>IntegerSpan</i>	<i>StringSpan</i>	<i>NameSpan</i>	<i>TimeSpan</i>
<i>DurationSpan</i>	<i>OptionSpan</i>									
<i>EnumerationSpan</i>	<i>ShapeSpan</i>									
<i>IntegerSpan</i>	<i>StringSpan</i>									
<i>NameSpan</i>	<i>TimeSpan</i>									
<i>NumberSpan</i>	<i>XYPairSpan</i>									
<u><a href="#">Priority?</a></u> <u><a href="#">Deprecated in JDF 1.2</a></u>	enumeration	<p>Indicates the importance of the specific intent. The following values have prescribed meanings:</p> <p><i>None</i></p> <p><i>Suggested</i> – The customer will accept a value of <i>Actual</i> that is different than the value of <i>Preferred</i> or outside of <i>Range</i>.</p> <p><i>Required</i> – <i>Actual</i> must be equal to <i>Preferred</i> or within <i>Range</i>. Note that the attribute <i>Preferred</i> is available in the data types which inherit from this abstract type.</p> <p><b>Note:</b> Replaced by <i>SettingsPolicy</i> in JDF 1.2 and beyond.</p>								

The following table describes the allowed values for filling the *Actual* attribute in a Span element defined by the combination of *Range*, *Preferred*, and *SettingsPolicy*.

SettingsPolicy	Preferred Exists	Range Exists	Suggested Value defined by:	Required Value defined by:
<i>BestEffort</i>	yes	no	<i>Preferred</i>	—
<i>BestEffort</i>	yes	yes	<i>Preferred</i>	—
<i>BestEffort</i>	no	yes	<i>Range</i>	—
<i>MustHonor</i>	yes	no	—	<i>Preferred</i>
<i>MustHonor</i>	yes	yes	<i>Preferred</i>	<i>Range</i>
<i>MustHonor</i>	no	yes	—	<i>Range</i>

### 7.1.1.2 Structure of the DurationSpan Subelement

[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, Structure of Abstract Span Subelement.

Name	Data Type	Description
<i>Actual?</i>	duration	The actual value selected for the quote.
<i>Preferred?</i>	duration	Provides a value specified by the person submitting the request, indicating what that person prefers. <i>Preferred</i> must fall within the range of values specified in <i>Range</i> .
<i>Range?</i>	DurationRange	Provides a valid range of time durations. If not specified, it defaults to the value of <i>Preferred</i> .

### 7.1.1.3 Structure of the EnumerationSpan Subelement

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, Structure of Abstract Span Subelement. It is identical to the NameSpan element except for the fact that it describes a closed list of enumeration values.

Name	Data Type	Description
<i>Actual</i> ?	enumeration	The actual value selected for the quote.
<i>Preferred</i> ?	enumeration	Provides a value specified by the person submitting the request, indicating what that person prefers. <i>Preferred</i> must fall within the range of values specified in <i>Range</i> .
<i>Range</i> ?	enumerations	Provides a set of discreet enumeration values. If not specified, it defaults to the value of <i>Preferred</i> .

### 7.1.1.4 Structure of the IntegerSpan Subelement

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, Structure of Abstract Span Subelement.

Name	Data Type	Description
<i>Actual</i> ?	integer	The actual value selected for the quote.
<i>Preferred</i> ?	integer	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of <i>Preferred</i> must fall within the range of values specified in <i>Range</i> .
<i>Range</i> ?	IntegerRangeList	Provides either a set of discreet values, a range of values, or a combination of the two that comprise all allowed values for the Span. If not specified, it defaults to the value of <i>Preferred</i> .

### 7.1.1.5 Structure of the NameSpan Subelement

This Span subelement is used to describe name ranges. It inherits from the abstract Span element described in Section 7.1.1.1, Structure of Abstract Span Subelement. It is identical to the EnumerationSpan element except for the fact that it describes an extensible list of NMTOKEN values.

Name	Data Type	Description
<i>Actual</i> ?	NMTOKEN	The actual value selected for the quote.
<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 either a set of discreet values, a range of values, or a combination of the two that comprise all allowed values for the Span. If not specified, it defaults to the value of <i>Preferred</i> .

#### 7.1.1.5.1 Specifying New Values in a NameSpan Subelement

NameSpan elements generally define an open list of predefined values. If a value that is not included in the list must be specified, a comment that defines that value can be included in the NameSpan using the new name as a *Name* attribute of the comment, as demonstrated in the following example:

```
<HoleType DataType="NameSpan" Range="36Hole 42Hole">
<Comment Name="36Hole">6 equidistant holes on each side of a hexagonal piece of paper </
Comment>
<Comment Name="42Hole">7 equidistant holes on each side of a hexagonal piece of paper </
Comment>
</HoleType>
```

### 7.1.1.6 Structure of the NumberSpan Subelement

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, Structure of Abstract Span Subelement.

Name	Data Type	Description
<i>Actual</i> ?	double	The actual value selected for the quote.
<i>Preferred</i> ?	double	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of <i>Preferred</i> must fall within the range of values specified in <i>Range</i> .
<i>Range</i> ?	DoubleRangeList	Provides either a set of discreet values, a range of values, or a combination of the two that comprise all allowed values for the Span. When not known, the value of <i>Range</i> shall default to the value of <i>Preferred</i> .

### 7.1.1.7 Structure of the OptionSpan Subelement

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, Structure of Abstract Span Subelement.

Name	Data Type	Description
<i>Actual</i> ?	boolean	The actual value selected for the quote. If the option is included = "true".
<i>Preferred</i> ?	boolean	Provides a value specified by the person submitting the request, indicating what that person prefers.
<i>Range</i> ?	string	<i>Detail</i> provides information about the option.

### 7.1.1.8 Structure of the ShapeSpan Subelement

[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, Structure of Abstract Span Subelement.

Name	Data Type	Description
<i>Actual</i> ?	shape	The actual value selected for the quote.
<i>Preferred</i> ?	shape	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of <i>Preferred</i> must fall within the range of values specified in <i>Range</i> .
<i>Range</i> ?	ShapeRangeList	Provides either a set of discreet values, a range of values, or a combination of the two that comprise all allowed values for the Span. If not specified, it defaults to the value of <i>Preferred</i> .

### 7.1.1.9 Structure of the StringSpan Subelement

This Span subelement is used to describe string ranges. It inherits from the abstract Span element described in Section 7.1.1.1, Structure of Abstract Span Subelement.

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>Range</i> *	telem	Provides either a set of discreet values, a range of values, or a combination of the two that comprise all allowed values for the Span. If not specified, it defaults to the value of <i>Preferred</i> .

### 7.1.1.10 Structure of the TimeSpan Subelement

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, Structure of Abstract Span Subelement.

Name	Data Type	Description
<i>Actual</i> ?	dateTime	The actual value selected for the quote.
<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 either a set of discreet values, a range of values, or a combination of the two that comprise all allowed values for the Span. If not specified, it defaults to the value of <i>Preferred</i> .

### 7.1.1.11 Structure of the XYPairSpan Subelement

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, Structure of Abstract Span Subelement.

Name	Data Type	Description
<i>Actual</i> ?	XYPair	The actual value selected for the quote.
<i>Preferred</i> ?	XYPair	Provides a value specified by the person submitting the request, indicating what that person prefers. The value of <i>Preferred</i> must fall within the range of values specified in <i>Range</i> .
<i>Range</i> ?	XYPairRangeList	Provides either a set of discreet values, a range of values, or a combination of the two that comprise all allowed values for the Span. If not specified, it defaults to the value of <i>Preferred</i> .

## 7.1.2 ArtDeliveryIntent

This resource specifies the prepress art delivery intent for a JDF job and maps the items to the appropriate reader pages and separations. Art delivery refers to any physical or electronic asset that is required for processing the job.

### Resource Properties

<b>Resource class:</b>	Intent
<b>Resource referenced by:</b>	—
<b>Process Resource Pairing:</b>	<b>DeliveryParams, DigitalDeliveryParams</b>
<b>Example Partition:</b>	<i>Option</i>
<b>Input of processes:</b>	Any product node
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
<i>ArtDeliveryDate</i> ? <a href="#">New in JDF 1.1</a>	TimeSpan	Specifies the latest time by which the transfer of the artwork will be made.
<i>ArtDeliveryDuration</i> ? <a href="#">New in JDF 1.1</a>	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, only one of either <i>ArtDeliveryDate</i> or <i>ArtDeliveryDuration</i> may be specified. Within a purchase order, only <i>ArtDeliveryDate</i> is allowed.

Name	Data Type	Description
<a href="#">ArtHandling ?</a> <a href="#">New in JDF 1.1</a>	EnumerationSpan	<p>Describes what should happen to the artwork after usage. The return or pickup address must be specified by a <b>Contact</b> with <b>ContactTypes</b> including "ArtReturn". Possible values are:</p> <p><i>ReturnWithProof</i> – The artwork is delivered back to the customer together with the proof, if there is any.</p> <p><i>ReturnWithProduct</i> – The artwork is delivered back to the customer together with the final product.</p> <p><i>Return</i> – The artwork is delivered back independently directly after usage.</p> <p><i>Pickup</i> – The customer picks up the artwork.</p> <p><i>Destroy</i> – The printer must destroy 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>
<a href="#">DeliveryCharge ?</a> <a href="#">New in JDF 1.1</a>	EnumerationSpan	<p>Specifies who pays for a delivery being made by a third party. Possible values are:</p> <p><i>Printer</i></p> <p><i>Buyer</i></p>
<a href="#">Method ?</a> <a href="#">Modified in JDF 1.2</a>	NameSpan	<p>Identifies a required delivery method, may be a generic item. Possible values are:</p> <p><i>EMail</i></p> <p><i>ExpressMail</i></p> <p><i>InterofficeMail</i></p> <p><i>OvernightService</i></p> <p><i>Courier</i></p> <p><i>CompanyTruck</i></p> <p><i>ISDNSoftware</i></p> <p><i>NetworkCopy</i> – This includes LAN and VPN.</p> <p><i>WebServer</i> – Upload / download from HTTP / FTP server.</p> <p><i>InstantMessaging</i></p> <p>May also be a delivery service brand, such as:</p> <p><i>UPS</i></p> <p><i>DHL</i></p> <p><i>FedEx</i></p> <p>Or any digital delivery service brand.</p>
<a href="#">PreflightStatus = "NotPerformed"</a> <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	enumeration	<p>Information about a preflight process probably applied to the artworks before being submitted. Possible values are:</p> <p><i>NotPerformed</i> – No preflighting was applied.</p> <p><i>WithErrors</i> – Preflighting resulted in error messages and optionally warning messages.</p> <p><i>WithWarnings</i> – Preflighting resulted in warning messages and no errors.</p> <p><i>WithoutErrors</i> – Preflighting was successful. No errors and no warnings occurred.</p>

Name	Data Type	Description
<i>ReturnList</i> = "None" <a href="#">New in JDF 1.1</a>	NMTOKENS	Type of printer created intermediate materials that should be sent to the customer after usage. Possible 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 required for processing the job, (e.g., a die for die cutting or embossing stamp). <i>None</i> – No intermediate materials should be returned to the customer.
<i>ReturnMethod</i> ? <a href="#">New in JDF 1.1</a>	NameSpan	Identifies a required delivery method for returning the artwork, if <i>ArtHandling</i> = "Return" and for the printer created materials listed in <i>ReturnList</i> . The predefined values are the same as the list specified in <i>Method</i> .
<i>ServiceLevel</i> ? <a href="#">New in JDF 1.2</a>	StringSpan	The service level of the specific carrier. Contain values "Next Day", "2nd Day Air", "Ground", etc.
<i>Transfer</i> ? <a href="#">New in JDF 1.1</a>	EnumerationSpan	Describes the responsibility of the transfer. Possible values are: <i>BuyerToPrinterDeliver</i> – The buyer delivers the artwork to the printer. The printer may specify in the quote a special <b>Contact</b> with <i>ContactTypes</i> including <i>Delivery</i> , where the buyer should send the artwork. <i>BuyerToPrinterPickup</i> – The printer picks up the artwork. The <b>Contact</b> with <i>ContactTypes</i> including <i>pickup</i> describes, where the printer has to pick up the artwork.
<i>ArtDelivery</i> + <a href="#">Modified in JDF 1.1</a>	element	Individual delivery.
<b>Company</b> ? <a href="#">Deprecated in JDF 1.1</a>	refelement	Address and further information of the art delivery. This must only be specified if the printer is expected to pick up the art delivery at this address. In JDF 1.1 and beyond, <b>Company</b> is a subelement of <b>Contact</b> .
<b>Contact</b> * <a href="#">New in JDF 1.1</a>	refelement	Address and further information about the transfer of the artwork. The actual delivery address is specified as the <b>Address</b> of the <b>Contact/@ContactTypes</b> including <i>Delivery</i> . Only one <b>Contact/@ContactTypes</b> , including <i>Delivery</i> , may be specified. The actual pickup address is specified as the <b>Address</b> of the <b>Contact/@ContactTypes</b> including <i>Pickup</i> . Only one <b>Contact/@ContactTypes</b> , including <i>Pickup</i> , may be specified.

### Structure of ArtDelivery Elements

Each **ArtDelivery** element defines a set of existing products that are required 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**.



Name	Data Type	Description
<i>Amount</i> ? <a href="#">Modified in JDF 1.2</a>	integer	Number of physical objects to be delivered. Only valid if no detailed resource description (e.g., <i>ExposedMedia</i> , <i>RunList</i> , <i>ScanParams</i> , <i>DigitalMedia</i> , or <i>Tool</i> ) is specified.
<i>ArtDeliveryDate</i> ? <a href="#">New in JDF 1.1</a>	TimeSpan	Specifies the latest time by which the transfer of the artwork will be made.
<i>ArtDeliveryDuration</i> ? <a href="#">New in JDF 1.1</a>	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, only one of either <i>ArtDeliveryDate</i> or <i>ArtDeliveryDuration</i> may be specified. Within a purchase order, only the <i>ArtDeliveryDate</i> is allowed.
<i>ArtDeliveryType</i> <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	NMTOKEN	Type of artwork supplied. Possible values include: <i>DigitalFile</i> – Digital data irrespective of the delivery mechanism. The union of <i>DigitalMedia</i> and <i>DigitalNetwork</i> . <a href="#">New in JDF 1.2</a> <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 required for processing the job, (e.g., a die for die cutting or embossing stamp). None – No artwork exists, and it must be created.
<i>ArtHandling</i> ? <a href="#">New in JDF 1.1</a>	Enumeration-Span	Describes what should happen to the artwork after usage. The return or pickup address must be specified by a <i>Contact</i> with <i>ContactTypes</i> including "ArtReturn". Possible values are: <i>ReturnWithProof</i> – The artwork is delivered back to the customer together with the proof, if there is any. <i>ReturnWithProduct</i> – The artwork is delivered back to the customer together with the final product. <i>Return</i> – The artwork is delivered back independently directly after usage. <i>Pickup</i> – The customer picks up the artwork. <i>Destroy</i> – The printer must destroy the artwork. <i>PrinterOwns</i> – The artwork belongs to the printer. <i>Store</i> – The printer has to store the artwork for future purposes. Defaults to the value of <b>ArtDeliveryIntent/@ArtHandling</b> .
<i>DeliveryCharge</i> ? <a href="#">New in JDF 1.1</a>	Enumeration-Span	Specifies who pays for a delivery being made by a third party. Possible values are: <i>Printer</i> <i>Buyer</i> Defaults to the value of <b>ArtDeliveryIntent/@DeliveryCharge</b> .
<i>HasBleeds</i> = "false"	boolean	If "true", the file has bleeds.
<i>IsTrapped</i> = "false"	boolean	If "true", the file has been trapped.

Name	Data Type	Description
<a href="#">Method ?</a> <a href="#">Modified in JDF 1.2</a>	NameSpan	Identifies a required delivery method. It may be a generic item from the list defined in <i>Method</i> in <b>ArtDeliveryIntent</b> . Defaults to the value of <b>ArtDeliveryIntent/@Method</b> .
<a href="#">PageList ?</a>	Integer-RangeList	Set of pages of the output <b>Component</b> that are filled by this <b>ArtDelivery</b> . This maps the pages in the <b>ArtDelivery</b> to the Pages in the product that is produced. For example if <i>PageList</i> = "3~5", page 0 of the <b>ArtDelivery</b> (e.g., <i>RunList</i> ) is page 3 in the product, page 1 is page 4, etc. If not specified, the <i>PageList</i> must include all pages in reader order. The indices specified in <i>PageList</i> reference the <i>PageData</i> elements defined in <b>PageList</b> .
<a href="#">PreflightOutput ?</a> <a href="#">New in JDF 1.1</a>	URL	Pointer to the output information created by the preflight tool, if <i>PreflightStatus</i> is either <i>WithoutErrors</i> or <i>WithErrors</i> .
<a href="#">PreflightStatus ?</a> <a href="#">New in JDF 1.1</a>	enumeration	Information about a preflight process. The values are identical to those of <i>PreflightStatus</i> in <b>ArtDeliveryIntent</b> . Defaults to the value of <b>ArtDeliveryIntent/@PreflightStatus</b> .
<a href="#">ReturnMethod ?</a> <a href="#">New in JDF 1.1</a>	NameSpan	Identifies a required delivery method for returning the artwork, if <i>ArtHandling</i> = "Return". Defaults to the value of <b>ArtDeliveryIntent/@ReturnMethod</b> .
<a href="#">ServiceLevel ?</a> <a href="#">New in JDF 1.2</a>	StringSpan	The service level of the specific carrier. Contain values "Next Day", "2nd Day Air", "Ground", etc. Defaults to the value of <b>ArtDeliveryIntent/@ServiceLevel</b> .
<a href="#">Transfer ?</a> <a href="#">New in JDF 1.1</a>	Enumeration-Span	Describes the responsibility of the transfer. The values are identical to those of <i>Transfer</i> in <b>ArtDeliveryIntent</b> . Defaults to the value of <b>ArtDeliveryIntent/@Transfer</b> .
<a href="#">Company ?</a> <a href="#">Deprecated in JDF 1.1</a>	refelement	Address and further information about the art delivery. This must only be specified if the printer is expected to pick up the art delivery at this address. In JDF 1.1 and beyond, <b>Company</b> is a subelement of <b>Contact</b> .
<a href="#">Component ?</a> <a href="#">Deprecated in JDF 1.1</a>	refelement	Description of a physical component, (e.g., physical artwork). If neither <b>Component</b> , <b>ExposedMedia</b> , nor <b>RunList</b> are specified, no details of the <b>ArtDelivery</b> except the <i>ArtDeliveryType</i> and <i>Amount</i> are known.
<a href="#">Contact *</a> <a href="#">New in JDF 1.1</a>	refelement	Address and further information about the art transfer. Defaults to the value of <b>ArtDeliveryIntent/Contact</b> .
<a href="#">DigitalMedia ?</a> <a href="#">New in JDF 1.2</a>	refelement	Description of any digital media, (e.g., CD or tape with artwork that will be delivered). If neither <b>ExposedMedia</b> , <b>RunList</b> , <b>DigitalMedia</b> , nor <b>Tool</b> are specified, no details of the <b>ArtDelivery</b> except the <i>ArtDeliveryType</i> and <i>Amount</i> are known.
<a href="#">ExposedMedia ?</a> <a href="#">Modified in JDF 1.2</a>	refelement	Description of exposed media, (e.g., film, plate, or proof). If neither <b>ExposedMedia</b> , <b>RunList</b> , <b>DigitalMedia</b> , nor <b>Tool</b> are specified, no details of the <b>ArtDelivery</b> , except the <i>ArtDeliveryType</i> and <i>Amount</i> , are known.
<a href="#">RunList ?</a> <a href="#">Modified in JDF 1.2</a>	refelement	Link to digital artwork that is accessible via a set of URLs that are defined in the <b>RunList/LayoutElement/FileSpec/@URL</b> . If neither <b>DigitalMedia</b> , <b>ExposedMedia</b> , <b>RunList</b> , <b>DigitalMedia</b> , nor <b>Tool</b> are specified, no details of the <b>ArtDelivery</b> except the <i>ArtDeliveryType</i> and <i>Amount</i> are known.
<a href="#">ScanParams ?</a>	refelement	Description of a <b>ScanParams</b> that defines scanning details for the exposed media defined by <b>ExposedMedia</b> .
<a href="#">Tool ?</a> <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	refelement	Details of the <b>Tool</b> if <i>ArtDeliveryType</i> = "Tool". If <b>ExposedMedia</b> , <b>RunList</b> , <b>DigitalMedia</b> , nor <b>Tool</b> are specified, no details of the <b>ArtDelivery</b> 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 type of binding required and which side is to be bound. The input components that are used as a cover should have a *ProcessUsage* of *Cover*. The input components that are used as a hard cover jacket should have a *ProcessUsage* of *Jacket*. 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:	<b>BlockPreparationParams, CaseMakingParams, CasingInParams, ChannelBindingParams, CoilBindingParams, CoverApplicationParams, EndSheetGluingParams, GlueApplication, GluingParams, GlueLine, InsertingParams, JacketingParams, PlasticCombBindingParams, RingBindingParams, SaddleStitchingParams, SpinePreparationParams, SpineTapingParams, StitchingParams, StripBindingParams, ThreadSealingParams, ThreadSewingParams, WireCombBindingParams</b>

Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<a href="#">BackCoverColor?</a> <a href="#">New in JDF 1.1</a>	Enumeration-Span	Defines the color of the back cover material of the binding. Allowed values are defined in Section A.3.3.2, NamedColor. If not specified, it defaults to the value of <i>CoverColor</i> .
<a href="#">BindingColor?</a>	Enumeration-Span	Defines the color of the spine material of the binding. Allowed values are defined in Section A.3.3.2, NamedColor.
<a href="#">BindingLength?</a>	Enumeration-Span	Indicates which side should be bound when no content. Thus, no orientation is available, but a quote for binding is required. Possible values are: <i>Long</i> <i>Short</i>
<a href="#">BindingOrder="Gathering"</a> <a href="#">New in JDF 1.1</a>	enumeration	Specifies whether the child <b>Components</b> should be collected or gathered if multiple child <b>Components</b> are combined. One of: <i>Collecting</i> —The child <b>Components</b> are collected on a spine and placed within one another. The first <b>Component</b> is on the outside. <i>Gathering</i> —The child <b>Components</b> are gathered on a pile and placed on top of one another. The first <b>Component</b> is on the top. <i>List</i> —More complex ordering of child <b>Components</b> is specified using the <i>BindList</i> in this intent resource for this product.
<a href="#">BindingSide?</a>	Enumeration-Span	Indicates which side should be bound. Possible values are: <i>Top</i> <i>Bottom</i> <i>Right</i> <i>Left</i> Each of these values is intended to identify an edge of the job. These edges are defined relative to the orientation of the first page in the job with content on it. Default = <i>BindingLength</i> value, unless a non-empty <i>BindList</i> was specified. If both <i>BindingSide</i> and <i>BindingLength</i> are specified, <i>BindingSide</i> has precedence.

Name	Data Type	Description
<a href="#">BindingType</a> <a href="#">Modified in JDF 1.2</a>	Enumeration-Span	Describes the desired binding for the job. Possible values are: <i>Adhesive</i> – This type of binding can be handled with the <b>AdhesiveBinding</b> process. It includes perfect binding. Deprecated in JDF1.1 and replaced with <i>SoftCover</i> or <i>HardCover</i> . <i>ChannelBinding</i> – This type of binding can be handled with the <b>ChannelBinding</b> process. <i>CoilBinding</i> – This type of binding can be handled with the <b>CoilBinding</b> process. <i>CornerStitch</i> – Stitch in the corner that is at the clockwise end binding edge. For example, to stitch in the top left corner, set <i>BindingSide</i> = "Left". This type of binding can be handled with the <b>Stitching</b> process. <a href="#">New in JDF 1.2</a> <i>EdgeGluing</i> – Gluing gathered sheets at one edge of the pile. This Type of Binding can be handled with the <b>Gluing</b> process. <i>HardCover</i> – This type of binding defines a hard-cover bound book. <i>LooseBinding</i> – This type of binding defines a stack of pages with no additional binding. <i>PlasticComb</i> – This type of binding can be handled with the <b>PlasticCombBinding</b> process. <i>Ring</i> – This type of binding can be handled with the <b>RingBinding</b> process. <i>SaddleStitch</i> – This type of binding can be handled with the <b>Stitching</b> process. <i>Sewn</i> – This type of binding can be handled with the <b>ThreadSewing</b> process. <i>SideSewn</i> – This type of binding can be handled with the <b>ThreadSewing</b> process. <i>SideStitch</i> – This type of binding can be handled with the <b>Stitching</b> process.
		<i>SoftCover</i> – This type of binding defines a soft cover bound book. It includes perfect binding. <i>StripBind</i> – This type of binding can be handled with the <b>StripBinding</b> process. <i>Tape</i> – This type of binding is an inexpensive version of the <i>SoftCover</i> . <i>ThreadSealing</i> – This type of binding can be handled with the <b>ThreadSealing</b> process. <i>WireComb</i> – This type of binding can be handled with the <b>WireCombBinding</b> process.
<i>CoverColor</i> ?	Enumeration-Span	Defines the color of the cover material of the binding. Allowed values are defined in Section A.3.3.2, <i>NamedColor</i> .
<a href="#">AdhesiveBinding</a> ? <a href="#">Deprecated in JDF 1.1</a>	element	Details of <b>AdhesiveBinding</b> . Replaced with <b>SoftCoverBinding</b> in JDF 1.1.
<a href="#">BindList</a> ? <a href="#">New in JDF 1.1</a>	element	Details of binding of individual child Components.
<a href="#">BookCase</a> ? <a href="#">Deprecated in JDF 1.1</a>	element	Details of the Book Case. Used in Combination with <b>AdhesiveBinding</b> , <b>ThreadSewing</b> or <b>ThreadSealing</b> . Replaced with <b>HardCoverBinding</b> in JDF 1.1.
<a href="#">ChannelBinding</a> ?	element	Details of <b>ChannelBinding</b> .
<a href="#">CoilBinding</a> ?	element	Details of <b>CoilBinding</b> .
<a href="#">EdgeGluing</a> ? <a href="#">New in JDF 1.1</a>	element	Details of <b>EdgeGluing</b> .

Name	Data Type	Description
HardCoverBinding ? <a href="#">New in JDF 1.1</a>	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 ? <a href="#">New in JDF 1.1</a>	element	Details of SoftCoverBinding.
Tape ? <a href="#">New in JDF 1.1</a>	element	Details of Tape binding.
Tabs ?	element	Details of Tabs.
ThreadSealing ?	element	Details of ThreadSealing.
ThreadSewing ?	element	Details of ThreadSewing.
StripBinding ? <a href="#">New in JDF 1.1</a>	element	Details of StripBinding.
VeloBinding ? <a href="#">Removed in JDF 1.1</a>	element	Details of VeloBinding. Renamed to StripBinding in JDF 1.1.
WireCombBinding ?	element	Details of WireCombBinding.

### Structure of BindList Subelement

[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.

Name	Data Type	Description
BindItem *	element	Individual bind item description. Defaults to BindingIntent/BindingSide value if empty, (i.e., as if the BindList element weren't there).

### Structure of BindItem Subelement

[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*

Name	Data Type	Description
<i>BindingType</i> ?	Enumeration-Span	Describes the desired binding for the individual BindItem. The list of possible values is defined in <b>BindingIntent/@BindingType</b> . Defaults to the value of <b>BindingIntent/@BindingType</b> .
<i>ChildFolio</i> ?	XYPair	Definition of the fold between two pages in the BindItem component that is bound to the cover. The two numbers (as integers) in the <i>ChildFolio</i> attribute are the page numbers of the two outer pages of the child <b>Component</b> which touch the cover or another child <b>Component</b> . The pages are counted in the order as described in <i>FolioCount</i> of the child product. Defaults to the spine of the child.

Name	Data Type	Description
<i>ParentFolio</i>	XYPair	Definition of the fold between two pages in the Cover <b>Component</b> that receive the <b>BindItem</b> . The two numbers (as integers) in the <i>ParentFolio</i> attribute are the page numbers in the Cover <b>Component</b> which touch the child <b>Component</b> . The pages are counted in the order as described in <i>FolioCount</i> of the cover product.
<i>Transformation ?</i>	matrix	Rotation and offset between the <b>Component</b> to be inserted and the parent <b>Component</b> . For details on transformations, see Section 2.5.2, How and Where Coordinates and Transformations Are Used/Defined in JDF
<i>WrapPages ?</i>	IntegerRangeList	List of pages of the Cover that wrap around a <b>BindItem</b> 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 only be specified if the folding is ambiguous.
<i>BookCase ?</i> <a href="#">Deprecated in JDF 1.1</a>	element	Details of the hard cover Book Case. Used in Combination with <b>HardCoverBinding</b> .
<i>ChannelBinding ?</i>	element	Details of <b>ChannelBinding</b> .
<i>CoilBinding ?</i>	element	Details of <b>CoilBinding</b> .
<i>EdgeGluing ?</i>	element	Details of <b>EdgeGluing</b> .
<i>HardCoverBinding ?</i>	element	Details of <b>HardCoverBinding</b> .
<i>PlasticCombBinding ?</i>	element	Details of <b>PlasticCombBinding</b> .
<i>RingBinding ?</i>	element	Details of <b>RingBinding</b> .
<i>SaddleStitching ?</i>	element	Details of <b>SaddleStitching</b> .
<i>SideSewing ?</i>	element	Details of <b>SideSewing</b> .
<i>SideStitching ?</i>	element	Details of <b>SideStitching</b> .
<i>SoftCoverBinding ?</i>	element	Details of <b>SoftCoverBinding</b> .
<i>StripBinding ?</i>	element	Details of <b>StripBinding</b> .
<i>Tape ?</i>	element	Details of <b>Tape</b> binding.
<i>Tabs ?</i>	element	Details of <b>Tabs</b> .
<i>ThreadSealing ?</i>	element	Details of <b>ThreadSealing</b> .
<i>ThreadSewing ?</i>	element	Details of <b>ThreadSewing</b> .
<i>WireCombBinding ?</i>	element	Details of <b>WireCombBinding</b> .

### Structure of the AdhesiveBinding Subelement.

[Deprecated in JDF 1.1](#)

The table defining the deprecated AdhesiveBinding subelement has been moved to "BindingIntent Deprecated Subelements" on page 752.

### Structure of the BookCase Subelement.

[Deprecated in JDF 1.1](#)

The table defining the deprecated BookCase subelement has been moved to "BindingIntent Deprecated Subelements" on page 752.

### Structure of the ChannelBinding Subelement.

Name	Data Type	Description
<i>Cover ?</i>	OptionSpan	If " <i>true</i> ", the clamp used in <b>ChannelBinding</b> includes a preassembled cover.
<i>Thickness ?</i>	NumberSpan	Specifies thickness of board which is wrapped as front and back covers of a case bound book, in points.

**Structure of the CoilBinding Subelement.**

Name	Data Type	Description
<i>CoilMaterial</i> ?	EnumerationSpan	The coil materials available for <b>CoilBinding</b> . Possible values are: <i>Steel</i> – Plain steel. <i>ColorCoatedSteel</i> – Coated steel. <i>Plastic</i> – Plastic.
<b>HoleList</b> ? <a href="#">New in JDF 1.2</a>	refelement	Details of the holes for coil binding.

**Structure of the EdgeGluing Subelement.**[New in JDF 1.1](#)

Name	Data Type	Description
<i>EdgeGlue</i> ?	EnumerationSpan	Glue type used to glue the edge of the gathered sheets. Possible values are: <i>ColdGlue</i> <i>Hotmelt</i> <i>PUR</i> – Polyurethane rubber.

**Structure of the HardCoverBinding Subelement.**[New in JDF 1.1](#)

Name	Data Type	Description
<i>BlockThreadSewing</i> ?	OptionSpan	Option if the block is also thread sewn.
<i>EndSheets</i> ?	OptionSpan	Option if end sheets are applied.
<i>StripMaterial</i> ?	EnumerationSpan	Spine taping strip material. Possible values are: <i>Calico</i> <i>Cardboard</i> <i>CrepePaper</i> <i>Gauze</i> <i>Paper</i> <i>PaperlinedMules</i> <i>Tape</i>
<i>HeadBands</i> ?	OptionSpan	The following <b>CaseBinding</b> choice specifies the use of headbands on a case bound book. If "true", headbands are inserted both top and bottom.
<i>HeadBandColor</i> ?	EnumerationSpan	Defines the color of the headband. Allowed values are defined in Section A.3.3.2, <b>NamedColor</b> .
<i>Jacket</i> ?	EnumerationSpan	Specifies whether a hard cover jacket is needed and how it is attached. If specified, details of the jacket are described in the <b>Component</b> with <b>ProcessUsage</b> of <i>Jacket</i> . Possible values: <i>None</i> – No jacket is required. <i>Loose</i> – The jacket is loosely wrapped. <i>Glue</i> – Jacket is glued to the spine
<i>JapanBind</i> ?	OptionSpan	Bind the book block at the open edge, so that the folds are visible on the outside. If not specified, explicitly, this option is never selected.
<i>SpineBrushing</i> ?	OptionSpan	Brushing option for <b>SpinePreparation</b> .
<i>SpineFiberRoughing</i> ?	OptionSpan	Fiber roughing option for <b>SpinePreparation</b> .

Name	Data Type	Description
<i>SpineGlue</i> ?	EnumerationSpan	Glue type used to glue the book block to the cover. Possible values are: <i>ColdGlue</i> <i>Hotmelt</i> <i>PUR</i> – Polyurethane rubber.
<i>SpineLevelling</i> ?	OptionSpan	Leveling option for <b>SpinePreparation</b> .
<i>SpineMilling</i> ?	OptionSpan	Milling option for <b>SpinePreparation</b> .
<i>SpineNotching</i> ?	OptionSpan	Notching option for <b>SpinePreparation</b> .
<i>SpineSanding</i> ?	OptionSpan	Sanding option for <b>SpinePreparation</b> .
<i>SpineShredding</i> ?	OptionSpan	Shredding option for <b>SpinePreparation</b> .
<i>Thickness</i> ?	NumberSpan	Specifies thickness of board which is wrapped as front and back covers of a case bound book, in points.
<i>TightBacking</i> ?	EnumerationSpan	Definition of the geometry of the back of the book block. This can be one of: <i>Flat</i> – Flat backing. <i>Round</i> – Rounding way. <i>FlatBacked</i> – Backing way. <i>RoundBacked</i> – Rounding way, backing way.
<b>RegisterRibbon</b> *	refelement	Number, materials, colors and details of register ribbons.

### Structure of the PlasticCombBinding Subelement.

Name	Data Type	Description
<i>PlasticCombType</i> ? <a href="#">Modified in JDF 1.1</a>	NameSpan	The distance between the “teeth” in <b>PlasticCombBinding</b> 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 663 exist: <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 The following values are <a href="#">deprecated in JDF 1.1</a> . <i>Euro</i> – Distance = 12 mm; Holes = 7 mm x 3 mm <i>USA1</i> – Distance = 14.28 mm; Holes = 8 mm x 3 mm
<b>HoleList</b> ? <a href="#">New in JDF 1.2</a>	element	Details of the holes for the plastic comb. Note that <i>Shape</i> is always rectangular by design of the plastic combs.



**Structure of the RingBinding Subelement.**

Name	Data Type	Description
<i>BinderMaterial</i> ? <a href="#">New in JDF 1.1</a>	NameSpan	The following describe <b>RingBinding</b> binder materials used. Values include: <i>Cardboard</i> – Cardboard with no covering. <i>ClothCovered</i> – Cardboard with cloth covering. <i>Plastic</i> – Binder cover fabricated from solid plastic sheet material, (e.g., PVC sheet). <i>VinylCovered</i> – Cardboard with colored vinyl covering.
<i>HoleType</i> ? <a href="#">New in JDF 1.1</a>	EnumerationSpan	Predefined hole pattern for the ring system. Multiple hole patterns are not allowed, (e.g., 3-hole ring binding and 4-hole ring binding holes on one piece of media). For details of the hole types and a list of allowed values, refer to "JDF/CIP4 Hole Pattern Catalog" on page 663.
<i>RingDiameter</i> ?	NumberSpan	Size of the rings in points. The value used in production must be suitable for specified <i>HoleType</i> (s). Note that in ring shapes other than round, this size is specified by industry-standard method.
<i>RingMechanic</i> ?	OptionSpan	The ring binder used includes a lever for opening and closing.
<i>RingShape</i> ?	NameSpan	The following <b>RingBinding</b> shapes are used: <i>Round</i> <i>Oval</i> <i>D-shape</i> <i>SlantD</i>
<i>RingSystem</i> ? <a href="#">Deprecated in JDF 1.1</a>	NameSpan	<i>2HoleEuro</i> <i>3HoleUS</i> <i>4HoleEuro</i> <i>RingSystem</i> have been replaced by <i>HoleType</i> .
<i>RivetsExposed</i> ?	OptionSpan	The following <b>RingBinding</b> choice describes mounting of the ring mechanism in binder case. If " <i>true</i> ", the heads of the rivets are visible on the exterior of the binder. If " <i>false</i> ", the binder covering material covers the rivet heads.
<i>ViewBinder</i>	NameSpan	The following <b>RingBinding</b> clear vinyl outer wrap types are used on top of a colored base wrap: <i>Embedded</i> – Printed material is embedded by sealing between the colored and clear vinyl layers during binder manufacturing. <i>Pocket</i> – Binder is designed so that printed material may be inserted between the color and clear vinyl layers after binder manufacturing.

**Structure of the SaddleStitching Subelement.**

Name	Data Type	Description
<i>StitchNumber</i> ? <a href="#">New in JDF 1.1</a>	IntegerSpan	Number of stitches used for saddle stitching.

**Structure of the SideSewing Subelement.**

This is a placeholder that may be filled with private or future data.

Name	Data Type	Description
------	-----------	-------------

**Structure of the SideStitching Subelement.**

Name	Data Type	Description
<a href="#">StitchNumber ?</a> <a href="#">New in JDF 1.2</a>	IntegerSpan	Number of stitches used for side stitching.

**Structure of the SoftCoverBinding Subelement.**

[New in JDF 1.1](#)

Name	Data Type	Description
<a href="#">BlockThreadSewing ?</a>	OptionSpan	Specifies whether the block is also thread sewn.
<a href="#">GlueProcedure ?</a>	Enumeration-Span	Glue procedure used to glue the book block to the cover. Possible values are: <i>Spine</i> <i>SideOnly</i> – Glued at the side/endsheets but not the spine. <i>SingleSide</i> – Swiss Brochure. <i>SideSpine</i> – Both side gluing and SpineGluing.
<a href="#">Scoring ?</a>	Enumeration-Span	Scoring option for <b>SoftCoverBinding</b> . Possible values are: <i>TwiceScored</i> <i>QuadScored</i> <i>None</i> Values are based on viewing the cover in its flat, prebound state.
<a href="#">SpineBrushing ?</a>	OptionSpan	Brushing option for <b>SpinePreparation</b> .
<a href="#">SpineFiberRoughing ?</a>	OptionSpan	FiberRoughing option for <b>SpinePreparation</b> .
<a href="#">SpineGlue ?</a>	Enumeration-Span	Glue type used to glue the book block to the cover. Possible values are: <i>ColdGlue</i> <i>Hotmelt</i> <i>PUR</i> – Polyurethane rubber.
<a href="#">SpineLevelling ?</a>	OptionSpan	Leveling option for <b>SpinePreparation</b> .
<a href="#">SpineMilling ?</a>	OptionSpan	Milling option for <b>SpinePreparation</b> .
<a href="#">SpineNotching ?</a>	OptionSpan	Notching option for <b>SpinePreparation</b> .
<a href="#">SpineSanding ?</a>	OptionSpan	Sanding option for <b>SpinePreparation</b> .
<a href="#">SpineShredding ?</a>	OptionSpan	Shredding option for <b>SpinePreparation</b> .

**Structure of the StripBinding Subelement.**

[New in JDF 1.1](#)

Name	Data Type	Description
<a href="#">HoleList ?</a> <a href="#">New in JDF 1.2</a>	refelement	Note that <i>Shape</i> is always round by design of the strip poles.

**Structure of the Tape Subelement.**

[New in JDF 1.1](#)

Name	Data Type	Description
<a href="#">TapeColor ?</a>	Enumeration-Span	Defines the color of the tape material of the binding. Allowed values are defined in Section A.3.3.2, NamedColor.

## Structure of the Tabs Subelement.

Specifies tabs.

Name	Data Type	Description
<i>TabBanks</i> = "1"	Integer	Number of rows of tabs on the face of the book.
<i>TabsPerBank</i> ?	Integer	Number of equal-sized tabs in a single bank, if all positions were filled. Note that banks may have tabs only in some of the possible positions
<i>TabExtensionDistance</i> ?	NumberSpan	Distance tab extends beyond the body of the book block, in points.
<i>TabExtensionMylar</i> ?	OptionSpan	If "true", the tab extension will be mylar reinforced.
<i>TabBindMylar</i> ?	OptionSpan	If "true", the tab bind edge will be mylar reinforced.
<i>TabBodyCopy</i> ?	OptionSpan	If "true", Color will be applied not only on tab extension, but also on tab body. Note that lack of body copy allows all tabs within a bank to be printed on a single sheet.
<i>TabMylarColor</i> ?	EnumerationSpan	Specifies the color of the mylar used to reinforce the tab extension. This is conditional on <i>TabExtensionMylar</i> being "true". Allowed values are defined in Section A.3.3.2, NamedColor.

## Structure of the ThreadSealing Subelement.

This is a placeholder that may be filled with private or future data.

Name	Data Type	Description
------	-----------	-------------

## Structure of the ThreadSewing Subelement.

Name	Data Type	Description
<i>Sealing</i> ?	OptionSpan	If "true", thermo-sealing is required in <b>ThreadSewing</b> .

## Structure of the WireCombBinding Subelement.

Name	Data Type	Description
<i>WireCombMaterial</i> ?	EnumerationSpan	The material used for forming the <b>WireCombBinding</b> . Possible values are: <i>Steel-Silver</i> <i>ColorCoatedSteel</i>
<i>WireCombShape</i> ?	EnumerationSpan	The shape of the <b>WireCombBinding</b> . Possible values are: <i>Single</i> – Each "tooth" is made with one wire. <i>Twin</i> – The shape of each "tooth" is made with a double wire, (e.g., Wire-O).
<b>HoleList</b> ? <a href="#">New in JDF 1.2</a>	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:	<b>Color, ColorantControl, ColorCorrectionParams, ColorPool, ColorSpaceConversionParams</b>
Example Partition:	<i>Option, PageNumber, Side</i>
Input of processes:	Any product node
Output of processes:	—

## Resource Structure

Name	Data Type	Description
<p><i>Coatings</i> ?</p> <p><a href="#">Modified in JDF 1.1</a></p>	StringSpan	<p>Material usually applied to a full surface on press as a protective or gloss-enhancing layer over ink. Possible values include:</p> <p><i>DullVarnish</i></p> <p><i>GlossVarnish</i></p> <p><i>UV</i></p> <p><i>Aqueous</i></p> <p><i>Silicone</i></p> <p>The individual strings within <i>Coatings</i> are of type NMTOKENS and may contain multiple entries from the above list.</p>
<p><i>ColorStandard</i> ?</p> <p><a href="#">Modified in JDF 1.2</a></p>	NameSpan	<p>The color process (i.e., printing condition) requested for the job. If both of <i>ColorStandard</i> or <i>ColorsUsed</i> are specified, the union of the two is specified. For example, if <i>ColorStandard</i> specifies CMYK and <i>ColorsUsed</i> contains one Spot color, then CMYK + Spot is specified. Possible values include:</p> <p><i>CMYK</i> – Generic four color process.</p> <p><i>FIRST</i> – Flexographic Image Reproduction Specifications &amp; Tolerances.</p> <p><i>GRACOL</i> – General Requirements for Applications in Commercial Offset Lithography</p> <p><i>Hexachrome</i> – 6 Colors CMYK+Orange and Green.</p> <p><i>HIFI</i> – 7 Colors CMYK+Red, Green and Blue.</p> <p><i>ISO12647</i> – ISO offset standard. <a href="#">Deprecated in JDF 1.2</a></p> <p><i>JapanColor2001</i> – Japan Color 2001 standard [japancolor].</p> <p><i>Monochrome</i> – Generic single color printing condition, (e.g., black and white or one single spot color).</p> <p><i>None</i> – No marks. Used to define one-sided printing. <a href="#">Deprecated in JDF 1.2</a> (Use <b>LayoutIntent/@Sides</b> instead.)</p> <p><i>SNAP</i> – Specifications for Newsprint Advertising Production</p> <p><i>SWOP</i> – Specifications for Web Offset Publications. Registered by ANSI with the ICC as <i>ICC:CGATSTR001</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.</p> <p>If both of <i>ColorICCStandard</i> and <i>ColorStandard</i> are specified, then <i>ColorICCStandard</i> defines the ICC specific details, whereas <i>ColorStandard</i> defines the generic color standard.</p>

Name	Data Type	Description
<a href="#">ColorICCStandard ?</a> <a href="#">New in JDF 1.2</a>	StringSpan	<p><i>ColorICCStandard</i> may be used to identify a specific standard printing condition, by reference to Characterization Data registered with the ICC (<a href="http://www.color.org/drsection1.html">http://www.color.org/drsection1.html</a>). This printing condition reference corresponds to the OutputIntent characterization referencing capability in PDF/X. The syntax will be ReferenceName as shown in the examples below. ReferenceName is the standard reference string name used in both JDF and PDF/X, defined for each printing condition in the characterization registry on the ICC website. Values include:</p> <p><i>FOGRA11</i> – Registered by FOGRA pertaining to offset commercial and specialty printing according to ISO 12647-2, positive plates, paper type 1 (gloss-coated, above 70 g/m<sup>2</sup>), and paper type 2 (matte-coated, above 70 g/m<sup>2</sup>), 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 ISO 12647-2, positive plates, paper type 1 (gloss-coated, above 70 g/m<sup>2</sup>), and paper type 2 (matte-coated, above 70 g/m<sup>2</sup>), 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><b>Note:</b> If both of <i>ColorICCStandard</i> or <i>ColorsUsed</i> are specified, the union of the two is specified. If both of <i>ColorICCStandard</i> and <i>ColorStandard</i> are specified, then <i>ColorICCStandard</i> defines the ICC specific details, whereas <i>ColorStandard</i> defines the generic color standard.</p>
<a href="#">Coverage ?</a>	NumberSpan	<p>Cumulative colorant coverage percentage. For example, a full sheet of 100% deep black in CMYK has <i>Coverage</i> = "400". Typical coverages based on one color plane are:</p> <p><i>Light</i> = 1-9%</p> <p><i>Medium</i> = 10-35%</p> <p><i>Heavy</i> = 36+%</p>
<a href="#">InkManufacturer ?</a> <a href="#">Deprecated in JDF 1.2</a>	NameSpan	Name of the manufacturer of the ink requested, (e.g., "CIP4_Ink_Company", "ACMEInk", etc.).
<a href="#">ColorPool ?</a> <a href="#">New in JDF 1.1</a> <a href="#">Clarified in JDF 1.2</a>	refelement	Additional details about the colors used. The <b>ColorPool</b> resource may include some or all details about both <b>ColorsUsed</b> separation spot colors, spot colors contained in job files that will be printed using process color equivalents, and the <i>ColorStandard</i> process colors.
<a href="#">ColorsUsed ?</a>	element	Array of colorant separation names that are requested. If not specified, the values are implied from <i>ColorStandard</i> . If specified, <b>ColorsUsed</b> must contain a list of all separation names used by the job. <b>Note:</b> If additional information about the colors and colorants is required, it can be specified in the referenced <b>ColorPool</b> resource.

## Structure of the ColorsUsed Subelement

Name	Data Type	Description
SeparationSpec * <a href="#">Modified in JDF 1.2</a>	refelement	<p>These can be process colors, generic spot colors or named spot colors.</p> <p>In addition, partial coating is specified by adding a SeparationSpec with anything from <i>Coatings</i> as <i>Name</i>:</p> <p><i>Aqueous</i>  <i>Bronzing</i>  <i>DullVarnish</i>  <i>GlossVarnish</i>  <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. <a href="#">New in JDF 1.2</a></p> <p><i>UV</i></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:	<b>Address, DeliveryParams</b>
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>Accepted</i> = "false"	boolean	The quote that is specified by this <b>DeliveryIntent</b> has been accepted.
<i>AdditionalAmount</i> = "1" <a href="#">New in JDF 1.2</a>	integer	Number of components used to calculate the value of the <i>AdditionalPrice</i> attribute in the Pricing. This value applies to the number of additional items in one DropIntent/DropItemIntent and not to the total additional number of items.
<i>BuyerAccount</i> ?	string	Account ID of the buyer with the delivery service.
<i>DeliveryCharge</i> ? <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	Enumeration-Span	<p>Specifies who pays for a delivery being made by a third party. Possible values are:</p> <p><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, pre-press suppliers, etc.).</p> <p><i>Buyer</i> – The customer specified in CustomerInfo.</p> <p><i>Other</i> – The contact with <i>ContactType</i> = "DeliveryCharge". <a href="#">New in JDF 1.2</a></p>
<i>Earliest</i> ?	TimeSpan	Specifies the earliest time after which the transfer may be made.
<i>EarliestDuration</i> ?	DurationSpan	Specifies the earliest time by which the transfer must be made relative to the date of the purchase order. Within an RFQ or a Quote, only one of either <i>Earliest</i> or <i>EarliestDuration</i> may be specified. Within a purchase order only the <i>Earliest</i> is allowed.

Name	Data Type	Description
<i>Method</i> ?	NameSpan	Identifies a required delivery method, may be a generic item from the following list: <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 must be stored by the supplier. <i>OvernightService</i> May also be a delivery service brand, for example: <i>UPS</i> <i>DHL</i> <i>FedEx</i>
<i>Ownership</i> = <i>Origin</i>	enumeration	Point of transfer of ownership: <i>Origin</i> – Ownership of goods is transferred upon leaving point of origin. <i>Destination</i> – Ownership is transferred upon receipt at destination.
<i>Overage</i> ?	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> ? <a href="#">Deprecated in JDF 1.1</a>	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 <b>Company</b> . If <i>Pickup</i> = " <i>true</i> ", the <b>DeliveryIntent</b> describes an input to the job, (e.g., a CD for inserting, a preprinted cover, etc.). In this case <b>Company</b> describes the location where the merchandise is picked up.
<i>Required</i> ?	TimeSpan	Specifies the time by which the transfer must be made.
<i>RequiredDuration</i> ?	DurationSpan	Specifies the time by which the transfer must be made relative to the date of the purchase order. Within an RFQ or a Quote, only one of either <i>Required</i> or <i>RequiredDuration</i> must be specified. Within a purchase order, only <i>Required</i> is allowed.
<i>ReturnMethod</i> ? <a href="#">New in JDF 1.1</a>	NameSpan	Identifies a required delivery method for returning the surplus material if <i>SurplusHandling</i> = " <i>Return</i> ". The values may be of the same list as specified in <i>Method</i> .
<i>ServiceLevel</i> ? <a href="#">New in JDF 1.2</a>	StringSpan	The service level of the specific carrier. Contain values " <i>Next Day</i> ", " <i>2nd Day Air</i> ", " <i>Ground</i> ", etc.
<i>SurplusHandling</i> ? <a href="#">New in JDF 1.1</a>	Enumeration-Span	Describes what should happen with unused or redundant parts of the transfer specified with <i>Transfer</i> = " <i>BuyerToPrinterDeliver</i> " or " <i>BuyerToPrinterPickup</i> " after the job. The return delivery or pickup address is specified in the <i>Contact</i> with <i>ContactTypes</i> including <i>SurplusReturn</i> . Possible values are: <i>ReturnWithProduct</i> – The surplus material is delivered back to the customer together with the final product. <i>Return</i> – The surplus material is delivered back independently directly after usage. <i>Pickup</i> – The customer picks up the surplus material. <i>Destroy</i> – The printer must destroy 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.

Name	Data Type	Description
<a href="#">Transfer ?</a> <a href="#">New in JDF 1.1</a>	Enumeration-Span	Describes the direction and responsibility of the transfer. Possible values are: <i>BuyerToPrinterDeliver</i> – The <b>DeliveryIntent</b> 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 may specify in the quote a special <b>Contact</b> with <i>ContactTypes</i> including "Delivery", where the buyer should send the merchandise. <i>BuyerToPrinterPickup</i> – The <b>DeliveryIntent</b> 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 <b>Contact</b> with <i>ContactTypes</i> including "Pickup", where the printer has to pick up the merchandise. <i>PrinterToBuyerDeliver</i> – The <b>DeliveryIntent</b> describes an output of the job. In this case, the printer delivers the merchandise to the buyer. The <b>Contact</b> that has <i>ContactTypes</i> including "Delivery", where the printer should send the merchandise. <i>PrinterToBuyerPickup</i> – The <b>DeliveryIntent</b> describes an output of the job. In this case, the buyer picks up the merchandise. The printer may specify in the quote a special <b>Contact</b> that has <i>ContactTypes</i> including "Pickup", where the buyer should pick up the merchandise.
<a href="#">Underage ?</a>	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.
<a href="#">Company ?</a> <a href="#">Deprecated in JDF 1.1</a>	refelement	Address and further information of the addressee. In JDF 1.1 and beyond, <b>Company</b> is referenced from <b>Contact</b> .
<a href="#">Contact *</a> <a href="#">New in JDF 1.1</a>	refelement	Address and further information of the <b>Contact</b> responsible for the transfer. The actual delivery address is specified as the <b>Address</b> of the <b>Contact</b> with <i>ContactTypes</i> that includes "Delivery". The actual pickup address is specified as the <b>Address</b> of the <b>Contact</b> with <i>ContactTypes</i> that includes "Pickup". For each of the values "Delivery", "Pickup", and "Billing", only one <b>Contact</b> with <i>ContactTypes</i> including these values may be specified.
<a href="#">DropIntent +</a>	element	Includes all locations where the product will be delivered. Note that multiple <b>DropIntents</b> specify multiple deliveries and not options for delivery.
<a href="#">Pricing ?</a>	element	Pricing elements that define the pricing of the complete <b>DeliveryIntent</b> including any <b>DropIntents</b> or <b>DropItemIntents</b> that may contain further Pricing elements.

## Structure of DeliveryIntent Elements

### 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**.

Name	Data Type	Description
<a href="#">AdditionalAmount ?</a> <a href="#">New in JDF 1.2</a>	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 <b>DropIntent</b> / <b>DropItemIntent</b> and not to the total additional number of items. If not specified, defaults to the value of <b>DeliveryIntent</b> / <i>@AdditionalAmount</i> .
<a href="#">BuyerAccount ?</a> <a href="#">New in JDF 1.2</a>	string	Account ID of the buyer with the delivery service. Defaults to the value of <b>DeliveryIntent</b> / <i>@BuyerAccount</i> .



Name	Data Type	Description
<i>Earliest ?</i>	TimeSpan	Specifies the earliest time after which the transfer may be made.
<i>EarliestDuration ?</i>	DurationSpan	Specifies the earliest time by which the transfer must be made relative to the date of the purchase order. Within an RFQ or a Quote, only one of either <i>Earliest</i> or <i>EarliestDuration</i> may be specified. Within a purchase order, only the <i>Earliest</i> is allowed.
<i>Method ?</i>	NameSpan	Identifies a required delivery method. The values are identical to those of <i>Method</i> in the <b>DeliveryIntent</b> root. Defaults to the value of <b>DeliveryIntent/@Method</b> .
<i>Pickup ?</i> <a href="#">Deprecated in JDF 1.1</a>	boolean	If " <i>true</i> ", the merchandise is picked up. If <i>Pickup</i> = " <i>false</i> ", the <b>DropIntent</b> is delivered to the address specified in <b>Company</b> . If <i>Pickup</i> = " <i>true</i> ", the <b>DropIntent</b> describes an input to the job, (e.g., a CD for inserting, a preprinted cover, etc.). In this case, <b>Company</b> describes the location where the merchandise is picked up.
<i>Required ?</i>	TimeSpan	Specifies the time by which the delivery must be made.
<i>RequiredDuration ?</i>	DurationSpan	Specifies the time by which the delivery must be made relative to the date of the purchase order. Within an RFQ or a Quote, only one of either <i>Required</i> or <i>RequiredDuration</i> must be specified. Within a purchase order, only <i>Required</i> is allowed.
<i>ReturnMethod ?</i> <a href="#">New in JDF 1.1</a>	NameSpan	Identifies a required delivery method for returning the surplus material, if <i>SurplusHandling</i> = " <i>Return</i> ". Defaults to the value of <b>DeliveryIntent/@ReturnMethod</b> .
<i>ServiceLevel ?</i> <a href="#">New in JDF 1.2</a>	StringSpan	The service level of the specific carrier. Contain values " <i>Next Day</i> ", " <i>2nd Day Air</i> ", " <i>Ground</i> ", etc. Defaults to the value of <b>DeliveryIntent/@ServiceLevel</b> .
<i>SurplusHandling ?</i> <a href="#">New in JDF 1.1</a>	Enumeration-Span	Describes what should happen with unused or redundant parts of the transfer. The values are identical to those of <i>SurplusHandling</i> in <b>DeliveryIntent</b> . Defaults to the value of <b>DeliveryIntent/@SurplusHandling</b> .
<i>Transfer ?</i> <a href="#">New in JDF 1.1</a>	Enumeration-Span	Describes the direction and responsibility of the transfer. The values are identical to those of <i>Transfer</i> in <b>DeliveryIntent</b> . Defaults to the value of <b>DeliveryIntent/@Transfer</b> .
<b>Company ?</b> <a href="#">Deprecated in JDF 1.1</a>	refelement	Address and further information of the addressee. In JDF 1.1 and beyond <b>Company</b> is a subelement of <b>Contact</b> .
<b>Contact *</b> <a href="#">New in JDF 1.1</a>	refelement	Address and further information of the <b>Contact</b> responsible for the transfer. The actual delivery address is specified as the <b>Address</b> of the <b>Contact</b> with <i>ContactTypes</i> that includes " <i>Delivery</i> ". The actual pickup address is specified as the <b>Address</b> of the <b>Contact</b> with <i>ContactTypes</i> that includes " <i>Pickup</i> ". For each of the values " <i>Delivery</i> ", " <i>Pickup</i> ", and " <i>Billing</i> ", only one <b>Contact</b> with <i>ContactTypes</i> including these values may be specified. Defaults to the <b>DeliveryIntent/Contact</b>
DropItemIntent +	element	A <b>DropIntent</b> may consist of multiple products, which are represented by their respective <b>PhysicalResource</b> resources. Each <b>DropItemIntent</b> element describes a number of individual resources that is part of this <b>DropIntent</b> .
Pricing ?	element	Pricing element that defines the pricing of the <b>DropIntent</b> .

### Structure of the DropItemIntent Subelement

Name	Data Type	Description
<i>AdditionalAmount</i> ? <a href="#">Modified in JDF 1.2</a>	integer	Number of components used to calculate the value of the <i>AdditionalPrice</i> attribute in the Pricing. If not specified, defaults to the value of <i>DropIntent/@AdditionalAmount</i> .
<i>Amount</i> ?	integer	Specifies the final number of resources delivered. If not specified, defaults to the total amount of the resource that is specified by <i>PhysicalResource</i> or 1 if this <i>DropItemIntent</i> specifies a proof.
<i>OrderedAmount</i> ?	integer	Specifies the original number of resources ordered. If not specified, defaults to the value of <i>Amount</i> .
<i>Proof</i> ? <a href="#">New in JDF 1.1</a>	string	This <i>DropItem</i> refers to a proof that is specified in a <i>ProofItem</i> of the <b>ProofingIntent</b> of this product node. <b>ProofingIntent/ProofItem/@ProofName</b> must match <i>Proof</i> . One of either <i>PhysicalResource</i> or <i>Proof</i> must be specified.
<i>Unit</i> ?	string	Unit of measurement for the <i>Amount</i> specified in the <i>PhysicalResource</i> . Defaults to the value of <i>Unit</i> defined in the resource described by the <i>PhysicalResource</i> .
<i>PhysicalResource</i> ? <a href="#">Modified in JDF 1.1</a>	refelement	Description of the individual item that is delivered. One of either <i>PhysicalResource</i> or <i>Proof</i> must be specified. Note that <i>PhysicalResource</i> is an abstract resource and that the element must be an instance of <i>PhysicalResource</i> , (e.g., <b>Component</b> ).
<i>Pricing</i> ?	element	Pricing element that defines the pricing of the <i>DropItemIntent</i> .

### Contents of the Pricing Subelement

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 ISO 4217. It defaults to the currency defined in the parent quote.
<i>HasPrice</i> = "true"	boolean	Specifies whether the line item defined by this quote has a price. If "false", the line item is not included in the parent quote, and the price is unknown and must be added. If "true", the line item is included in the parent quote.
<i>Item</i> ?	string	Name of the item that this particular quote element describes. If not specified, Pricing applies to the entire <i>DropItemIntent</i> .
<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> ? <a href="#">New in JDF 1.1</a>	element	Details of the payment method.
<i>Pricing</i> *	element	Individual items of the quote. Note that a parent quote defines the complete quote, (i.e., including the values defined in the line items of any child quotes but excluding all line items with <i>HasPrice</i> = "false"). The sum of line items need not be identical to the parent quote.

### Contents of the Payment Subelement

[New in JDF 1.1](#)

Name	Data Type	Description
<i>PayTerm</i> ?	telem	Describes the payment terms & conditions.
<i>CreditCard</i> ?	element	Specifies credit card information

## Contents of the CreditCard Subelement

[New in JDF 1.1](#)

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.
<i>Type</i>	NMTOKEN	Credit card brand. Possible values include: <i>Amex</i> <i>DinersClub</i> <i>Discovery</i> <i>MasterCard</i> – This includes derived brands, (e.g., EuroCard). <i>Visa</i>

### 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 or not the product is embossed or stamped and, if desired, the complexity of the affected area.

#### Resource Properties

<b>Resource class:</b>	Intent
<b>Resource referenced by:</b>	—
<b>Process Resource Pairing:</b>	<b>EmbossingParams</b>
<b>Example Partition:</b>	<i>Option, PageNumber, Side</i>
<b>Input of processes:</b>	Any product node
<b>Output of processes:</b>	—

#### Resource Structure

Name	Data Type	Description
EmbossingItem +	element	Each embossed image is described by one EmbossingItem.

#### Structure of the EmbossingItem Subelement

Name	Data Type	Description
<i>Direction</i>	EnumerationSpan	The direction of the image. Possible values are: <i>Both</i> – Both debossing and embossing in one stamp. <i>Depressed</i> – Debossing. <i>Raised</i> – Embossing.
<i>EdgeAngle ?</i>	NumberSpan	The angle of a beveled edge in degrees. Typical values are an angle of: 30, 40, 45, 50, or 60 degrees. For <i>EdgeAngle</i> to exist, <i>EdgeShape</i> = "Beveled" must be specified.
<i>EdgeShape ?</i>	EnumerationSpan	The transition between the embossed surface and the surrounding media may be rounded or beveled (angled). Possible values are: <i>Rounded</i> <i>Beveled</i>

Name	Data Type	Description
<i>EmbossingType</i>	StringSpan	The strings defined in <i>EmbossingType</i> are whitespace separated combinations of the following tokens. Possible values for the tokens are: <i>BlindEmbossing</i> – Embossed forms that are not inked or foiled. The color of the image is the same as the paper. <i>FoilEmbossing</i> – Combines embossing with foil stamping in one single impression. <i>FoilStamping</i> – Using a heated die to place a metallic or pigmented image from a coated foil on the paper. <i>RegisteredEmbossing</i> – Creates an embossed image that exactly registers to a printed image.
<i>FoilColor</i> ?	EnumerationSpan	Defines the color of the foil material which is used within the <b>FoilStamp</b> process. Allowed values are defined in Section A.3.3.2, <i>NamedColor</i> .
<i>Height</i> ?	NumberSpan	The height of the levels. This value specifies the <i>vertical</i> distance between the highest and lowest point of the stamp, regardless of the value of <i>Direction</i> .
<i>ImageSize</i> ?	XYPairSpan	The size of the bounding box of one single image.
<i>Level</i> ?	EnumerationSpan	The level of embossing. Possible values are: <i>SingleLevel</i> <i>MultiLevel</i> <i>Sculpted</i>
<i>Position</i> ?	XYPairSpan	Position of the center of the bounding box of the embossed image in the coordinate system of the <b>Component</b> .

### 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 366.

#### Resource Properties

**Resource class:** Intent  
**Resource referenced by:** —  
**Process Resource Pairing:** **CreasingParams, CuttingParams, Fold, FoldingParams, PerforatingParams**

**Example Partition:** *Option*  
**Input of processes:** Any product node  
**Output of processes:** —

#### Resource Structure

Name	Data Type	Description
<i>FoldingCatalog</i>	NameSpan	Description of the folding scheme as specified in the folding catalog attribute in the format “Fn-i”. See JDF Folding Catalog descriptions in Figure 7.9 and Figure 7.10. <b>Note:</b> 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 <b>LayoutIntent/@Foliocount</b> for a discussion of pagination of folded end products.
<i>Folds</i> ? <a href="#">Deprecated in JDF 1.1</a>	XYPair	Number of folds in x and in y direction. This attribute specifies the number of folds seen in the sheet after folding not the number of fold operations needed to achieve that result. If not specified, it must be inferred from <i>FoldingCatalog</i> . The product $2*(X+1)*(Y+1)$ of <i>Folds</i> must always match the n of “Fn-i” of <i>FoldingCatalog</i> .
<b>Fold</b> * <a href="#">New in JDF 1.1</a>	element	This describes the details of folding operations in the sequence described by the value of <i>FoldingCatalog</i> . <b>Fold</b> must be specified if non-symmetrical folds are requested.

## 7.1.8 HoleMakingIntent

[Clarified in JDF 1.2](#)

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

<b>Resource class:</b>	Intent
<b>Resource referenced by:</b>	—
<b>Process Resource Pairing:</b>	<b>Hole, HoleLine, HoleMakingParams, Media</b>
<b>Example Partition:</b>	<i>Option</i>
<b>Input of processes:</b>	Any product node
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
<i>Extent</i> ? <a href="#">New in JDF 1.2</a>	XYPair	Size (bounding box) of the hole in points when specifying a standard hole pattern in <i>HoleType</i> . If not specified the implied default defined in "JDF/CIP4 Hole Pattern Catalog" on page 663 is assumed. Ignored when <i>HoleType</i> = " <i>Explicit</i> ".
<i>HoleReferenceEdge</i> = " <i>Left</i> " <a href="#">New in JDF 1.1</a>	enumeration	The edge of the media relative to where the holes should be punched. Use with <i>HoleType</i> . Possible values are: <i>Left</i> <i>Right</i> <i>Top</i> <i>Bottom</i> <i>Pattern</i> - Specifies that the reference edge implied by the value of <i>HoleType</i> in "JDF/CIP4 Hole Pattern Catalog" on page 663 is used.
<i>HoleType</i> <a href="#">Modified in JDF 1.1</a>	StringSpan	Predefined hole pattern. Multiple hole patterns are specified as one NMTOKENS string, (e.g., 3-hole ring binding and 4-hole ring binding holes on one piece of media). For details of hole types and a list of additional allowed values, refer to "JDF/CIP4 Hole Pattern Catalog" on page 663. Values are: <i>Explicit</i> - Holes are defined in an array of <i>Hole</i> elements. Additional values defined in "JDF/CIP4 Hole Pattern Catalog" on page 663 <a href="#">The following values are deprecated from JDF 1.0</a> <i>2HoleEuro</i> - Replace by either R2m-DIN or R2m-ISO. <i>3HoleUS</i> - Replace by R3I-US <i>4HoleEuro</i> - Replace by R4m-DIN-A4 or R4m-DIN-A5.
<i>HoleList</i> ?	element	Array of all <i>Hole</i> elements. Used only when <i>HoleType</i> = " <i>Explicit</i> ", otherwise this element is not used.

## 7.1.9 InsertingIntent

This resource specifies the placing or inserting of one component within another, using information that identifies page location, position, and attachment method. The receiving component is defined by a *ProcessUsage* attribute of "*Parent*". All other input components are mapped to the *Insert* elements by their ordering in the ResourceLinkPool.

## Resource Properties

Resource class:	Intent
Resource referenced by:	—
Process Resource Pairing:	<b>InsertingParams, InsertSheet</b>
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

## Resource Structure

Name	Data Type	Description
<i>GlueType</i> ?	EnumerationSpan	Glue used to fasten the insert. Possible values are: <i>Permanent</i> <i>Removable</i>
InsertList	Element	List of individual inserts.
<i>Method</i> ?	EnumerationSpan	Possible values are: <i>BindIn</i> – Apply glue to fasten the insert <i>BlowIn</i> – Loose insert.

## Structure of InsertList Subelement

Name	Data Type	Description
<i>Insert</i> *	element	Individual insert description.

## Structure of Insert Subelement

Name	Data Type	Description
<i>Folio</i>	IntegerRangeList	List of potential folios where the insert is to be placed. A <i>Folio</i> is defined by its first page in case <i>Method</i> = " <i>BlowIn</i> " and by the page that the glue is applied in case <i>Method</i> = " <i>BindIn</i> ". In general, a list of folios will only be supplied for <i>Method</i> = " <i>BlowIn</i> ". The pages are counted in the order, which is described in <i>FolioCount</i> of the parent <b>Component</b> .
<i>GlueType</i> ?	EnumerationSpan	Glue used to fasten the insert. Possible values are: <i>Removable</i> <i>Permanent</i> Defaults to the value of <b>InsertingIntent</b> / <i>@GlueType</i> .
<i>Method</i> ?	EnumerationSpan	Inserting method. Possible values are: <i>BindIn</i> – Apply glue to fasten the insert. <i>BlowIn</i> – Loose insert. Defaults to the value of <b>InsertingIntent</b> / <i>@Method</i> .
<i>SheetOffset</i> ? <a href="#">Deprecated in JDF 1.1</a>	XYPair	Offset between the <b>Component</b> to be inserted and finished page identified by folio in the parent <b>Component</b> . 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 <b>Component</b> to be inserted and the parent <b>Component</b> . If not specified, the identity matrix is applied.
<i>WrapPages</i> ? <a href="#">New in JDF 1.1</a>	IntegerRangeList	List of finished pages of the cover that wrap around an <b>Insert</b> 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 only be specified if the folding is ambiguous.
<b>GlueLine</b> * <a href="#">New in JDF 1.1</a>	element	Array of all <b>GlueLine</b> elements used to glue in the insert. Must not be specified in conjunction with <i>GlueType</i> .

### 7.1.10 LaminatingIntent

This resource specifies the laminating intent for a JDF job using information that identifies whether or not the product is laminated and, if desired, the temperature and thickness of the laminant.

#### Resource Properties

Resource class:	Intent
Resource referenced by:	—
Process Resource Pairing:	<b>LaminatingParams</b>
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>Laminated ?</i> <a href="#">Deprecated in JDF 1.1</a>	OptionSpan	If " <i>true</i> ", the product is laminated. If no <b>LaminatingIntent</b> is specified, the product must not be laminated.
<i>Temperature</i>	EnumerationSpan	Temperature used in the lamination process. Possible values are: <i>Hot</i> <i>Cold</i>
<i>Surface ?</i>	EnumerationSpan	The surface to be laminated. One of: <i>Front</i> <i>Back</i> <i>Both</i>
<i>Thickness ?</i>	NumberSpan	Thickness of the laminating material. Measured in microns [ $\mu\text{m}$ ].

### 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 should 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:	<b>Layout, LayoutPreparationParams, StrippingParams</b>
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>Dimensions ?</i> <a href="#">New in JDF 1.1</a> <a href="#">Clarified in JDF 1.2</a>	XYPair-Span	Specifies the width (X) and height (Y) in points, respectively, of the media or product <b>Component</b> unfolded. For example, <i>Dimensions</i> for a z-fold is the unfolded dimensions, while <i>FinishedDimensions</i> is the folded dimensions, if known. Use <i>Dimensions</i> if <i>FinishedDimensions</i> is not known. <i>Dimensions</i> 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 present, <i>FinishedDimensions</i> takes precedence

Name	Data Type	Description
<p><i>FinishedDimensions</i> ?</p> <p><a href="#">New in JDF 1.1</a></p> <p><a href="#">Clarified in JDF 1.2</a></p>	ShapeSpan	<p>Specifies the width (X), height (Y), and depth (Z) in points, respectively, of the finished product <b>Component</b> after all finishing operations, including folding, trimming, etc. If the Z coordinate is 0, it is ignored. Only <i>FinishedDimensions</i> should be specified if both <i>FinishedDimensions</i> and <i>Dimensions</i> are known.</p> <p><b>Compatibility Warning.</b> In JDF 1.1 height and width were erroneously switched in the description.</p>
<p><i>FinishedGrainDirection</i> ?</p> <p><a href="#">New in JDF 1.2</a></p>	enumerationSpan	<p>Specifies the media grain direction of the finished page with respect to the binding edge. Possible values are:</p> <p><i>ParallelToBind</i> – Grain direction is parallel to the binding edge.</p> <p><i>PerpendicularToBind</i> – Grain direction is perpendicular to the binding edge.</p>
<p><i>FinishedPageOrientation</i> ?</p> <p><a href="#">Deprecated in JDF 1.1</a></p>	enumeration	<p>Indicates the desired orientation of the finished media. Possible 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><b>In JDF 1.1</b>, the finished page orientation is implied by the value of <i>Dimensions</i> and <i>FinishedDimensions</i>. If height (X) &gt; width (Y), the product is portrait.</p>
<p><i>FolioCount</i> = "Booklet"</p> <p><a href="#">New in JDF 1.1</a></p>	enumeration	<p>Defines the method used when counting finished pages. One of:</p> <p><i>Booklet</i> – Each sample of the component consists of two finished pages, (e.g., a leaf—the front side and the back side of one sample of the component). Folds as specified by <b>FoldingIntent/ @FoldingCatalog</b> 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 <math>2*(X+1)*(Y+1)</math>, 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 top left of the front side of the top media to the bottom right of the back side of the bottom media. <i>Flat</i> should 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 <b>FoldingIntent/ @FoldingCatalog</b> (see Figure 7.9 and Figure 7.10) for the product. All sheets are counted, even if they are not included in the product, (e.g., due to a <b>ShapeCuttingIntent</b>).</p>
<p><i>NumberUp</i> = "1 1"</p> <p><a href="#">Modified in JDF 1.2</a></p>	XYPair	<p>Specifies a regular, multi-up grid of page cells into which content pages are mapped.</p> <p><b>Compatibility Warning.</b> 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>



Name	Data Type	Description
<p><i>Pages</i> ?</p> <p><a href="#">New in JDF 1.1</a></p> <p><a href="#">Modified in JDF 1.2</a></p>	IntegerSpan	<p>Specifies the number of finished pages (surfaces) of the product component, including blank pages.</p> <p><i>Pages</i> multiplied with <i>Dimensions</i> then divided by two (2) identifies the amount of paper that is used in the product. <i>Pages</i> describes the paper usage regardless of document layout. This value must be an even number. For example, the value for <i>Pages</i> for a two-sided booklet with seven reader pages would be "8", whether the booklet were saddle stitched or glued.</p> <p><b>Compatibility Warning.</b> 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> ?</p> <p><a href="#">New in JDF 1.1</a></p> <p><a href="#">Clarified in JDF 1.2</a></p>	IntegerSpan	<p>Specifies the number of non-identical finished pages of the product component, (i.e., the number of distinct master pages copied to produce the product). If not specified, the value of <i>Pages</i> is used as the default. For example, if there are ten finished pages, in which three are identical, <i>PageVariance</i> = "8" since it would take eight master copies to produce the product.</p>
<p><i>RotatePolicy</i> ?</p> <p><a href="#">New in JDF 1.2</a></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 <b>Media</b> defined in <i>Dimensions</i> or <i>FinishedDimensions</i>.</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> ?</p> <p><a href="#">Modified in JDF 1.2</a></p>	enumeration	<p>Indicates whether contents should be printed on one or both sides of the media. Possible 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. <a href="#">New in JDF 1.2</a></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>

Name	Data Type	Description
<a href="#">SizePolicy ?</a> <a href="#">New in JDF 1.2</a>	EnumerationSpan	<p>Allows printing even if the container size defined in <i>Dimensions</i> or <i>FinishedDimensions</i> does not match the requirements of the data. The page container is one cell on the NUp grid of the <b>Media</b> defined in <i>Dimensions</i> or <i>FinishedDimensions</i>.</p> <p><i>ClipToMaxPage</i> – The page contents should be clipped to the size of the container. The printed area is centered in the source image.</p> <p><i>FitToPage</i> – The page contents should be scaled up or down to fit the container. The aspect ratio is maintained.</p> <p><i>ReduceToFit</i> – The page contents should be scaled down but not scaled up to fit the container. The aspect ratio is maintained.</p> <p><i>Tile</i> – the page contents should be split into several tiles, each printed on its own container.</p>
<a href="#">Layout ?</a> <a href="#">New in JDF 1.1</a> <a href="#">Clarified in JDF 1.2</a>	refelement	Specifies the details of a more complex <b>Layout</b> . Must not be specified together with <i>NumberUp</i> . Note that the <b>Layout</b> specified in <b>LayoutIntent</b> specifies the layout definition of the finished product and not the layout of the production sheets.

### 7.1.12 MediaIntent

[Modified and Clarified 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:	<b>Media</b>
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	

#### Resource Structure

Name	Data Type	Description
<a href="#">BackCoatings ?</a>	EnumerationSpan	Identical to <i>FrontCoatings</i> , but applied to the back surface of the media. Default = value of <i>FrontCoatings</i> .
<a href="#">Brightness ?</a> <a href="#">Clarified in JDF 1.2</a>	NumberSpan	Reflectance percentage of diffuse blue reflectance as defined by ISO2470 – ISO 2470:1977 <i>Paper and board—Measurement of diffuse blue reflectance factor (ISO brightness)</i> . The reflectance is reported per ISO2470 as the diffuse blue reflectance factor of the paper or board in percent to the nearest 0.5% reflectance factor.
<a href="#">BuyerSupplied ?</a>	OptionSpan	Indicates whether the customer will supply the media. Note that the <b>Media</b> resource can be used to specify additional media requirements, particularly when the media is supplied by the customer.

Name	Data Type	Description
<p><i>Dimensions ?</i> <a href="#">Deprecated in JDF 1.2</a></p>	XYPairSpan	<p>Specifies the size of the supplied media in points when <i>BuyerSupplied</i> evaluates to "true". Must be ignored if <i>BuyerSupplied</i> evaluates to "false". Note that the size of the finished product is always specified in <b>LayoutIntent FinishedDimensions</b>.</p> <p>In JDF 1.2 and beyond the specifics of <i>BuyerSupplied</i> media should be specified using a <b>Media</b> resource. The dimensions of the finished product are specified with <b>LayoutIntent/@Dimensions</b> or <b>LayoutIntent/@FinishedDimensions</b>.</p>
<p><i>FrontCoatings ?</i> <a href="#">Modified in JDF 1.2</a></p>	EnumerationSpan	<p>What pre-process coating has been applied to the front surface of the media. Possible values are:</p> <p><i>None</i></p> <p><i>Coated</i> – A coating of a system specified type. <a href="#">New in JDF 1.2</a></p> <p><i>Glossy</i></p> <p><i>HighGloss</i></p> <p><i>InkJet</i> – A coating intended for use with inkjet technology. <a href="#">New in JDF 1.2</a></p> <p><i>Matte</i></p> <p><i>Satin</i></p> <p><i>Semigloss</i></p>
<p><i>Grade ?</i> <a href="#">Clarified in JDF 1.2</a></p>	IntegerSpan	<p>The intended grade of the media on a scale of 1 through 5. <i>Grade</i> is ignored if <i>MediaType</i> is not "Paper". <i>Grade</i> of paper material is defined in accordance with the paper "types" set forth in [iso12647-2]. Offset printing paper types are defined with the following integer values:</p> <ol style="list-style-type: none"> <li>1 "Gloss-coated paper"</li> <li>2 "Matt-coated paper"</li> <li>3 "Gloss-coated, web paper"</li> <li>4 "Uncoated, white paper"</li> <li>5 "Uncoated, yellowish paper"</li> </ol> <p><b>Note:</b> ISO 12647-2 paper type attribute values do not align with U.S. GRACOL paper grade attribute values, (i.e., ISO 12647-2 type 1 does not equal U.S. GRACOL grade 1.)</p>
<p><i>GrainDirection ?</i> <a href="#">New in JDF 1.2</a></p>	EnumerationSpan	<p>Direction of the grain in the coordinate system defined by <b>LayoutIntent/@Dimensions</b> or <b>LayoutIntent/@FinishedDimensions</b>. Possible 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>HoleCount ?</i> <a href="#">Deprecated in JDF 1.1</a></p>	IntegerSpan	<p>The intended number of holes that should 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>

Name	Data Type	Description
<i>HoleType</i> ? <a href="#">New in JDF 1.1</a>	StringSpan	Predefined hole pattern that specifies the pre-punched holes in the media. Multiple hole patterns are specified as one NMTOKENS string, (e.g, 3-hole ring binding and 4-hole ring binding holes on one piece of media.) For details of hole types and a list of additional allowed values, refer to "JDF/CIP4 Hole Pattern Catalog" on page 663. Values are: <i>None</i> – no holes Additional values are defined in "JDF/CIP4 Hole Pattern Catalog" on page 663
<i>MediaColor</i> ?	EnumerationSpan	Color of the media. Allowed values are defined in Section A.3.3.2, <i>NamedColor</i> . If more-specific, specialized, or site-specific media color names are needed, use <i>MediaColorDetails</i> .
<i>MediaColorDetails</i> ? <a href="#">New in JDF 1.2</a>	StringSpan	A more specific, specialized, or site-defined name for the media color. If <i>MediaColorDetails</i> is supplied, <i>MediaColor</i> must also be supplied. Note that there is a one-to-many relationship between entries in <i>MediaColor</i> and <i>MediaColorDetails</i> , (e.g., <i>MediaColorDetails</i> values of <i>Burgundy</i> and <i>Ruby</i> both correspond to a <i>MediaColor</i> of <i>DarkRed</i> ).
<i>MediaSetCount</i> ?	integer	When the input media is grouped in sets, identifies the number of pieces of media in each set. For example, if the <i>UserMediaType</i> is " <i>PreCutTabs</i> ", a <i>MediaSetCount</i> of 5 would indicate that each set includes 5 tab sheets.
<i>MediaType</i> ? <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	EnumerationSpan	Describes the medium being employed. Possible values are: <i>Disc</i> – CD or DVD disc to be printed on. <a href="#">New in JDF 1.2</a> <i>Other</i> – Any other media. <i>Paper</i> <i>Transparency</i>
<i>MediaUnit</i> ? <a href="#">Deprecated in JDF 1.2</a>	EnumerationSpan	Describes the format of the media as it is delivered to the device. Possible values are: <i>Roll</i> <i>Sheet</i> <b>Reason for deprecation:</b> Intent attributes pertain to finished product, not the raw media format. If <i>BuyerSupplied</i> = " <i>true</i> ", then the <b>Media</b> resource can be used to provide this attribute.
<i>Opacity</i> = " <i>Opaque</i> " <a href="#">Modified in JDF 1.2</a>	EnumerationSpan	The opacity of the media. See <i>OpacityLevel</i> to specify the degree of opacity for any of these values. Possible values are: <i>Opaque</i> – the media is opaque. With two-sided printing the printing on the other side does not show through under normal incident light. <i>Translucent</i> – The media is translucent to a system specified amount. For example, translucent media can be used for back lit viewing. <a href="#">New in JDF 1.2</a> <i>Transparent</i> – the media is transparent to a system specified amount.

Name	Data Type	Description
<i>OpacityLevel</i> ? <a href="#">New in JDF 1.2</a>	NumberSpan	Normalized TAPPI Opacity, (Cn), as defined and computed in ISO 2471:1998 “ <i>Paper and board—Determination of opacity (paper backing)—Diffuse reflectance method</i> ”. Refer also to TAPPI T 519 “ <i>Diffuse opacity of paper (d/0° paper backing)</i> ” for calculation examples.
<i>PrePrinted</i> = “ <i>false</i> ”	boolean	Indicates whether the media is preprinted.
<i>Recycled</i> ? <a href="#">Deprecated in JDF 1.2</a>	OptionSpan	If “ <i>true</i> ”, recycled media is requested. In JDF 1.2 and beyond, use <i>RecycledPercentage</i> .
<i>RecycledPercentage</i> ? <a href="#">New in JDF 1.2</a>	NumberSpan	The percentage, between 0 and 100, of recycled material that the media must contain.
<i>StockBrand</i> ?	StringSpan	Strings providing available brand names. The customer may know exactly what paper is to be used. Example is “Lustro” or “Warren Lustro” even though the manufacturer name is included.
<i>StockType</i> ?	NameSpan	Strings describing the available stock. Examples include: <i>Bristol</i> <i>Cover</i> <i>Bond</i> <i>Newsprint</i> <i>Index</i> <i>Offset</i> – This includes book stock. <i>Tag</i> <i>Text</i>
<i>Texture</i> ?	NameSpan	The intended texture of the media. Examples include: <i>Antique</i> – Rougher than vellum surface. <i>Calendared</i> – Extra-smooth or polished, uncoated paper. <i>Linen</i> – Texture of coarse woven cloth. <i>Smooth</i> <i>Stipple</i> – Fine pebble finish. <i>Vellum</i> – Slightly rough surface.
<i>Thickness</i> ? <a href="#">New in JDF 1.1</a>	NumberSpan	The thickness of the chosen medium. Measured in microns [μm].

Name	Data Type	Description
<i>UserMediaType</i> ?	NMTOKEN	<p>A human-readable description of the type of media. The value can be used by an operator to select the correct media to load. The semantics of the values will be site-specific. Possible values include:</p> <p><i>Continuous</i> – Continuously connected sheets of an opaque material. Which edge is connected is not specified.</p> <p><i>ContinuousLong</i> – Continuously connected sheets of an opaque material connected along the long edge.</p> <p><i>ContinuousShort</i> – Continuously connected sheets of an opaque material connected along the short edge.</p> <p><i>Envelope</i> – Envelopes that can be used for conventional mailing purposes.</p> <p><i>EnvelopePlain</i> – Envelopes that are not preprinted and have no windows.</p> <p><i>EnvelopeWindow</i> – Envelopes that have windows for addressing purposes.</p> <p><i>FullCutTabs</i> – Media with a tab that runs the full length of the medium so that only one tab is visible extending out beyond the edge of non-tabbed media.</p> <p><i>Labels</i> – Label stock, (e.g., a sheet of peel-off labels).</p> <p><i>Letterhead</i> – Separately cut sheets of an opaque material including a letterhead.</p> <p><i>MultiLayer</i> – Form medium composed of multiple layers which are preattached to one another, (e.g., for use with impact printers).</p> <p><i>MultiPartForm</i> – Form medium composed of multiple layers not preattached to one another; each sheet may be drawn separately from an input source.</p> <p><i>Photographic</i> – Separately cut sheets of an opaque material to produce photographic quality images.</p> <p><i>PreCutTabs</i> – Media with tabs that are cut so that more than one tab is visible extending out beyond the edge of non-tabbed media.</p> <p><i>Stationery</i> – Separately cut sheets of an opaque material.</p> <p><i>TabStock</i> – Media with tabs (either precut or full-cut).</p> <p><i>Transparency</i> – Separately cut sheets of a transparent material.</p>
<i>USWeight</i> ? <a href="#">Deprecated in JDF 1.2</a>	NumberSpan	The intended weight of the media, measured in pounds per ream of basis size. Only one of <i>Weight</i> and <i>USWeight</i> may be specified. If known, <i>Weight</i> should be specified in grammage (g/m <sup>2</sup> .) In JDF 1.2 and beyond, use <i>Weight</i> .
<i>Weight</i> ? <a href="#">Clarified in JDF 1.2</a>	NumberSpan	The intended weight of the media, measured in grammage (g/m <sup>2</sup> ) of the media. See "North American Media Weight Explained" on page 627 for an explanation of how to calculate the US weight from the grammage for different stock types.

### 7.1.13 NumberingIntent

This resource describes the parameters of stamping or applying variable marks in order to produce unique components, for items such as lottery notes or currency.

#### Resource Properties

Resource class:	Intent
Resource referenced by:	—
Process Resource Pairing:	<b>NumberingParams</b>
Example Partition:	—
Input of processes:	<b>Numbering</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>ColorName</i> ?	EnumerationSpan	Defines the color of the numbering. Allowed values are defined in Section A.3.3.2, <i>NamedColor</i> .
<b>ColorPool</b> ?	refelement	Additional details about the colors used.
NumberItem +	element	Individual position of the numbers on the finished page.

#### Structure of NumberItem Subelement

Name	Data Type	Description
<i>ColorName</i> ?	EnumerationSpan	Defines the color of the numbering. Allowed values are defined in Section A.3.3.2, <i>NamedColor</i> . If not specified, it defaults to the values defined in <b>NumberingIntent/@ColorName</b> .
<i>Orientation</i> ?	NumberSpan	Rotation of the numbering machine in degrees. If <i>Orientation</i> = 0, the top of the numbers is along the leading edge.
<i>StartValue</i> = "1"	string	First value of the numbering machine.
<i>Step</i> = "1"	integer	Number that specifies the difference between two subsequent numbers of the numbering machine.
<i>XPosition</i> ?	NumberSpan	Position of the number in the X direction of the product.
<i>YPosition</i> ?	NumberSpan	Position of the number in the Y direction of the product.
<b>SeparationSpec</b> ?	refelement	Specifies the name of the <b>Color</b> in the <b>ColorPool</b> 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:	<b>BoxPackingParams, Bundle, Component, PalletizingParams, Pallet, ShrinkingParams, StackingParams, Strap, StrappingParams, WrappingParams</b>
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

## Resource Structure

Name	Data Type	Description
<i>BoxedQuantity</i> ?	IntegerSpan	How many units of <i>product</i> in a box.
<i>BoxShape</i> ?	ShapeSpan	Describes the length, width, and height of the box, in points.
<i>CartonQuantity</i> ?	IntegerSpan	How many units of <i>product</i> in a carton.
<i>CartonShape</i> ?	ShapeSpan	Describes the length, width, and height of the carton, in points. For example, 288 544 1012
<i>CartonMaxWeight</i> ?	NumberSpan	Maximum weight of an individual carton, in kilograms.
<i>CartonStrength</i> ?	NumberSpan	Strength of the carton, in kilograms.
<i>FoldingCatalog</i> ?	NameSpan	Description of the folding scheme for folding the product for packaging as specified in the folding catalog attribute in the format “Fx-y”. See JDF Folding Catalog descriptions in Figure 7.9 and Figure 7.10. <b>Note:</b> The folding scheme in this context refers to the folding of the finished product for packaging only. The folding has no effect on the page/folio definition.
<i>PalletQuantity</i> ?	IntegerSpan	Number of <i>product</i> per pallet
<i>PalletSize</i> ?	XYPairSpan	Describes the length and width of the pallet, in points, (e.g., “3500 3500”).
<i>PalletMaxHeight</i> ?	NumberSpan	Maximum height of a loaded pallet, in points.
<i>PalletMaxWeight</i> ?	NumberSpan	Maximum weight of a loaded pallet, in kilograms.
<i>PalletType</i> ?	NameSpan	Type of pallet used. Examples include: <i>2Way</i> – Two-way entry <i>4Way</i> – Four-way entry <i>Euro</i> – Standard 1*1 m Euro pallet
<i>PalletWrapping</i> ?	NameSpan	Wrapping of the completed pallet. Examples include: <i>Banding</i> <i>None</i> – explicitly requests no wrapping. <i>StretchWrap</i>
<i>WrappedQuantity</i> ?	IntegerSpan	Number of units of product per wrapped package.
<i>WrappingMaterial</i> ?	NameSpan	Examples include: <i>None</i> – explicitly requests no wrapping. <i>PaperBand</i> <i>Polyethylene</i> <i>RubberBand</i> <i>ShrinkWrap</i>

### 7.1.15 ProductionIntent

This resource specifies the manufacturing intent and considerations for a JDF job using information that identifies the desired result or specified manufacturing path.

#### Resource Properties

<b>Resource class:</b>	Intent
<b>Resource referenced by:</b>	—
<b>Process Resource Pairing:</b>	All
<b>Example Partition:</b>	<i>Option</i>
<b>Input of processes:</b>	Any product node
<b>Output of processes:</b>	—



## Resource Structure

Name	Data Type	Description
<i>PrintPreference ?</i>	EnumerationSpan	Intended result or goal. Possible values are: <i>Balanced</i> – Request for a manufacturing process that balances the requirements for cost, speed, and quality. <i>CostEffective</i> – Request for the most cost effective manufacturing process. <i>Fastest</i> – Request for the most time effective manufacturing process. Cost and Quality may be sacrificed for a fast turnaround time. <i>HighestQuality</i> – Request for the manufacturing process which will result in the highest quality.
<i>PrintProcess ?</i>	EnumerationSpan	Print process requested. Allowed values are: <i>Electrophotography</i> <i>Flexography</i> <i>Gravure</i> <i>Inkjet</i> <i>Lithography</i> – Includes offset printing <i>Letterpress</i> <i>Screen</i> <i>Thermography</i>

### 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, LayoutPreparationParams, Media, RenderingParams, ScreeningParams, SeparationControlParams, StrippingParams**

**Example Partition:** *Option*  
**Input of processes:** Any product node  
**Output of processes:** —

#### Resource Structure

Name	Data Type	Description
<a href="#">ProofItem *</a> <a href="#">New in JDF 1.1</a>	element	Specifies the details of the proofs that are required. If no ProofItem exists in a ProofingIntent, it explicitly specifies that no proofs are desired.

#### Structure of the ProofItem Element

All parameters of **ProofingIntent** have been moved into ProofItem in JDF 1.1

Name	Data Type	Description
<a href="#">Amount ?</a> <a href="#">Modified in JDF 1.1</a>	IntegerSpan	Specifies the total number of copies of this proof that is required. If not specified, it defaults to an IntegerSpan with <i>Preferred</i> = "1".
<a href="#">BrandName ?</a> <a href="#">Modified in JDF 1.1</a>	StringSpan	Brand name of the proof, (e.g., Iris).

Name	Data Type	Description
<a href="#">ColorType ?</a> <a href="#">Modified in JDF 1.1</a>	EnumerationSpan	Color quality of the proof. Possible values are: <i>Monochrome</i> – Generic single color printing condition, (e.g., black and white or one single spot color). <i>BasicColor</i> – Color does not match precisely. This implies the absence of a color matching system. <i>MatchedColor</i> – Color is matched to the output of the press using a color matching system.
<a href="#">Contract = "false"</a> <a href="#">Modified in JDF 1.1</a>	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).
<a href="#">HalfTone ?</a> <a href="#">Modified in JDF 1.1</a>	OptionSpan	Specifies whether the proof should emulate halftone screens.
<a href="#">ImageStrategy ?</a> <a href="#">New in JDF 1.2</a>	EnumerationSpan	Identifies which images (OPI or other) will be printed on a proof or displayed as a soft proof. <i>NoImages</i> – No images are imaged on the proof. <i>LowResolution</i> – Low resolution images are imaged on the proof. <i>HighResolution</i> – High resolution production images are imaged on the proof, resulting in proofs that accurately represent the final product.
<a href="#">PageIndex ?</a> <a href="#">New in JDF 1.1</a>	IntegerRangeList	List of pages in the numbering scheme given by the <i>FolioCount</i> attribute of the component that should be proofed. Where no range is specified then all pages shall be proofed.
<a href="#">ProofName ?</a> <a href="#">New in JDF 1.1</a>	string	Name of the <b>ProofItem</b> . This field must exist if delivery of a proof is specified in <b>DeliveryIntent</b> .
<a href="#">ProofTarget ?</a> <a href="#">Modified in JDF 1.1</a>	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 should be sent to the URL specified in <i>ProofTarget</i> .
<a href="#">Technology ?</a> <a href="#">Modified in JDF 1.1</a>	NameSpan	Technology used for making the proof. Possible values are: <i>BlueLine</i> <i>DyeSub</i> <i>InkJet</i> <i>Laser</i> <i>PressProof</i> <i>SoftProof</i>
<a href="#">ProofType?</a> <a href="#">Modified in JDF 1.1</a>	EnumerationSpan	The kind of proof. Possible values are: <i>Page</i> – Page proof <i>Imposition</i> – Imposition proof <i>None</i> – No proof is required.
<a href="#">SeparationSpec *</a> <a href="#">New in JDF 1.1</a>	refelement	Separations that are to be proofed. If not specified, all separations are proofed.
<a href="#">ApprovalParams ?</a> <a href="#">New in JDF 1.2</a>	refelement	List of people (e.g., a customer, printer, or manager) who can sign the <b>ApprovalSuccess</b> .

### 7.1.17 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:	<b>ScreeningParams, SeparationControlParams</b>
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>DotSize</i> ?	NumberSpan	Specifies the dot size of the screen in microns [ $\mu\text{m}$ ] when FM screening is used, otherwise <i>DotSize</i> is ignored.
<i>Frequency</i> ?	NumberSpan	Specifies the line frequency of the screen in lines per inch (lpi) when AM screening is used, otherwise <i>Frequency</i> is ignored.
<i>FrequencySelection</i> ?	EnumerationSpan	Selects the AM or FM frequency range. Possible values are: <i>LowestFrequency</i> – Lowest AM or FM frequency supported. <i>MiddleFrequency</i> – Middle AM or FM frequency supported <i>HighestFrequency</i> – Highest AM or FM frequency supported
<i>ScreeningType</i> ?	EnumerationSpan	General type of screening. Possible values are: <i>AM</i> – May be line or dot. <i>FM</i>

### 7.1.18 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:	<b>CuttingParams, ShapeCuttingParams</b>
Example Partition:	<i>Option</i>
Input of processes:	Any product node
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
ShapeCut *	element	Array of all ShapeCut elements. Used when each shape is exactly specified.

#### Structure of ShapeCut Subelement

Name	Data Type	Description
<i>CutBox</i> ?	rectangle	Specification of a rectangular window. (See Section A.2.31, rectangle for a definition of the rectangle data type.)
<i>CutOut</i> = "false"	boolean	If "true", the inside of a specified shape must be removed. If "false", the outside of a specified shape must be removed. An example of an inside shape is a window, while an example of an outside shape is a shaped greeting card.

Name	Data Type	Description
<a href="#"><i>CutPath</i> ?</a> <a href="#">Modified in JDF 1.2</a>	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</i> = "true".
<a href="#"><i>CutType</i> ?</a> <a href="#">Modified in JDF 1.1</a>	EnumerationSpan	Type of cut or perforation used. Possible values are: <i>Cut</i> – Full cut. <i>Perforate</i> – Interrupted perforation that does not span the entire sheet.
<a href="#"><i>ShapeDepth</i> ?</a> <a href="#">New in JDF 1.1</a>	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 must be applied. Only the recto finished page of a leaf should be specified.
<i>ShapeType</i>	EnumerationSpan	Describes any precision cutting other than hole making. Possible values are: <i>Rectangular</i> <i>Round</i> <i>Path</i>
<i>TeethPerDimension</i> ?	NumberSpan	Number of teeth in a given perforation extent in teeth/point. MicroPerforation is defined by specifying a large number of teeth (n>1000).

### 7.1.19 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

[Clarified in JDF 1.2](#)

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 3.6.1, 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 (which may be attributes or 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 3-13, "Contents of the abstract Resource element," on page 53. 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 may be referenced by more than one resource. For an example, see the resource element **SeparationSpec**. If the resource is not referenced by multiple resources, it is described inside the resource section of the resource to which it belongs. For example, see the Structure of the BundleItem Element of the Bundle resource. If an element inside a resource section of the resource is needed to be referenced by multiple resources in a revision of JDF, then that element is promoted to its own section. For example, **ColorSpaceConversionOp** was a sub-element of **ColorSpaceConversionParams** in JDF/1.1. The resource class of an atomic resource also defines the superclasses from which the resource inherits additional contents. The **Consumable**, **Quantity**, and **Handling** resource elements inherit from the **PhysicalResource** element, which in turn inherits from the **Resource** element. **Parameter** and **Implementation** resource elements inherit from the **Resource** element directly. Non-atomic resources (i.e., resource subelements) do not inherit contents from resource superclasses.

Examples for resources that may be used as atomic resources or resource elements are: **Employee**, **InsertSheet**, **LayoutElement**, and **Media**. For example, if the **Media** is used as an atomic resource, it inherits all content from the resource class *Consumable*. If it is used as a resource element, then the **Media** may have only an ID as defined by Table 3-25, “Contents of the abstract ResourceElement,” on page 69.

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 an element with more than one value associated with it. If this is the case, the element name is listed as often as it appears, and a term in parentheses that identifies the kind of element is included in the column. For an example, see Section 7.2.62, *EndSheetGluingParams* or Section 7.2.147, *Sheet*. An example of the tables in this section is provided below.

### Resource Properties Template

<b>Resource class:</b>	Defines the resource class or specifies ResourceElement if the element does not inherit content from a resource class.
<b>Resource referenced by:</b>	List of parent resources that contain elements of this type. Only valid for elements.
<b>Example Partition:</b>	List of recommended partitioning keys: For a complete list of partition keys, see the description of <i>PartIDKeys</i> in Table 3-27, “Contents of the Partitionable Resource Element,” on page 78. Note that resources may be partitioned by keys that are not specified in this list.
<b>Input of processes:</b>	List of node types that use the resource as an input resource.
<b>Output of processes:</b>	List of node types that create the resource as an output resource

### Resource Structure Template

Name	Data Type	Description
Name of attribute	data type of attribute	Usage of the attribute.
Name of element	element	Subelements that must be defined locally within the resource.
Name of element	refelement	Elements that are based on other atomic resources or resource elements. These may either be in-line elements or instances of ResourceRef elements. (See Section 3.7.6, Inter-Resource Linking Using ResourceRef). In case of ResourceRef elements, a “Ref” must be appended to the name specified in the table column entitled “Name”.
<b>FileSpec</b> (ResourceUsage)	refelement	<b>FileSpec</b> resources may have a <b>FileSpec/@resourceUsage</b> attribute that specifies the context in which to use the <b>FileSpec</b> . <b>FileSpec/@resourceUsage</b> must match the (ResourceUsage) value specified in the parentheses.

## 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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	Contact, Location (see Table 3-16, “Contents of the Location element,” on page 58)
<b>Example:</b>	—
<b>Input of processes:</b>	—
<b>Output of processes:</b>	—

## Resource Structure

Name	Data Type	Description
<i>City</i> ?	string	City or locality of address (vCard: ADR:locality).
<i>Country</i> ?	string	Country of address (vCard: ADR:country).
<i>CountryCode</i> ?	string	Country of address. This value conforms to the ISO 3166 standard in which countries are represented as two-character codes.
<i>PostBox</i> ?	string	Post office address (vCard: ADR:pobox. For example: P.O. Box 101).
<i>PostalCode</i> ?	string	Zip code or postal code of address (vCard: ADR:pcode).
<i>Region</i> ?	string	State or province (vCard: ADR:region).
<i>Street</i> ?	string	Street address (vCard: ADR:street).
<i>ExtendedAddress</i> ?	telem	Extended address (vCard: ADR:extadd. For example: Suite 245).

### 7.2.3 AdhesiveBindingParams

[Deprecated in JDF 1.1](#) See "AdhesiveBindingParams" on page 753 for details of this deprecated resource.

### 7.2.4 ApprovalParams

This resource provides the details of an approval process.

#### Resource Properties

**Resource class:** Parameter

**Resource referenced by:** **ConventionalPrintingParams, DigitalPrintingParams, ProofingIntent**

**Example Partition:** —

**Input of processes:** **Approval**

**Output of processes:**

#### Resource Structure

Name	Data Type	Description
ApprovalPerson *	element	List of people (e.g., a customer, printer, or manager) who can sign the approval.
<i>MinApprovals</i> = "1" <a href="#">New in JDF 1.2</a>	integer	Minimum number of ApprovalPersons with <i>ApprovalRole</i> = "Group" that must sign the <b>ApprovalSuccess</b> for the <b>ApprovalSuccess</b> to be <i>Available</i> .

#### Structure of ApprovalPerson Subelement

Name	Data Type	Description
<i>Obligated</i> ? <a href="#">Deprecated in JDF 1.2</a>	boolean	If "true", the person is required to sign this approval. In JDF 1.2 and beyond, use <i>ApprovalRole</i> .
<i>ApprovalRole</i> = "Obligated" <a href="#">New in JDF 1.2</a>	enumeration	One of: <i>Group</i> – The approver belongs to a group of which <i>MinApprovals</i> members must sign. <i>Obligated</i> – The approver must sign the approval. <i>Informative</i> – The approver is informed of the approval process but not required for success. If he does not approve, the approval is still valid.
<b>Contact</b>	refelement	Contact (e.g., a customer, printer, or manager) who must sign the approval. One value included in the <i>ContactTypes</i> attribute of this <b>Contact</b> element should be <i>Administrator</i> .

## 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:	<i>DocIndex, DocRunIndex, RunIndex, RunPage, RunTags, SetIndex, SheetName, Side, SignatureName, TileID</i>
Input of processes:	Any process
Output of processes:	<b>Approval, Verification</b>

### Resource Structure

Name	Data Type	Description
<b>FileSpec</b> ?	refelement	The file that contains the approval signature. If <b>FileSpec</b> does not exist, <b>ApprovalSuccess</b> is a logical placeholder.
<b>Contact</b> * <a href="#">New in JDF 1.2</a>	refelement	List of contacts that have signed off on this approval.

## 7.2.6 Assembly

[New in JDF 1.2](#)

The Assembly describes how the sections of one or multiple jobs or job parts are bound together.

### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>Stripping</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<b>AssemblyID</b> ?	string	Identification of the <b>Assembly</b> if <b>Stripping</b> produces multiple Assembly elements.
<b>BindingSide</b> = "Left"	enumeration	Indicates which side should be bound. One of: <i>Left</i> <i>Right</i> <i>Top</i> <i>Bottom</i> <i>BindingSide</i> is ignored when <i>Order</i> = "None".
<b>JobID</b> ?	string	Identification of the original job the <b>Assembly</b> belongs to. If not specified, it defaults to the value specified or implied in the JDF node.
<b>Order</b> = "Gathering"	enumeration	One of: <i>Collecting</i> – The sections are placed within one another. The first section is on the outside. <i>Gathering</i> – The sections are placed on top of one another. The first section is on the top. <i>None</i> – The sections are not bound. Typically for flatwork jobs. <i>List</i> – More complex ordering of the sections
<b>AssemblySection</b> *	element	Individual <b>AssemblySection</b> which are gathered. <b>AssemblySections</b> must only be specified when <i>Order</i> = "List".

### Structure of the AssemblySection Subelement

Name	Data Type	Description
<i>AssemblyID</i> ?	string	Identification of the AssemblySection if <b>Stripping</b> produces a multi-section <b>Assembly</b> . If not specified, it defaults to the value specified or implied in the parent <b>Assembly</b> or AssemblySection.
<i>JobID</i> ?	string	Identification of the original job the AssemblySection belongs to. If not specified, it defaults to the value specified or implied in the parent <b>Assembly</b> or AssemblySection.
<i>Order</i> = "Gathering"	enumeration	One of: <i>Collecting</i> – The child AssemblySections are placed within one another. The first section is on the outside. <i>Gathering</i> – The child AssemblySections are placed on top of one another. The first section is on the top.
AssemblySection *	element	Additional AssemblySection elements which are collected or gathered to create this AssemblySection.

### 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:	<b>AssetListCreation</b>
Output of processes:	

#### Resource Structure

Name	Data Types	Description
<i>AssetTypes</i> ?	regExp	Specifies what type of assets should 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 <b>RunList</b> . 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.
<b>FileSpec</b> ( <i>SearchPath</i> ) *	relement	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 <b>FileSpec</b> is specified, the search path is the directory in which the input asset resides and must not be searched recursively.



## 7.2.8 AutomatedOverPrintParams

[Clarified in JDF 1.2](#)

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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>RenderingParams, SeparationControlParams</b>
<b>Example Partition:</b>	—
<b>Input of processes:</b>	—
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Types	Description
<i>OverPrintBlackLineArt</i> = "false" <a href="#">Clarified in JDF 1.2</a>	boolean	Indicates whether overprint should 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" <a href="#">Clarified in JDF 1.2</a>	boolean	Indicates whether overprint should be set to "true" for black text. If "true", overprint of black line art is applied regardless of any values in the PDL. If "false", <i>TextSizeThreshold</i> and <i>TextBlackLevel</i> are ignored and PDL line art overprint operators are processed.
<i>TextSizeThreshold</i> ?	integer	Indicates the point size for text below which black text will be set to overprint. For asymmetrically scaled text, the minimum point size between both axes will be used. If not specified, all text is set to overprint.
<i>TextBlackLevel</i> = "1"	double	A value between 0.0 and 1.0 which indicates the minimum black level for the text stroke or fill colors that cause the text to be set to overprint.
<i>LineArtBlackLevel</i> ?	double	A value between 0.0 and 1.0 which indicates the minimum black level for the stroke or fill colors that cause the line art to be set to overprint. Defaults to the value of <i>TextBlackLevel</i> .

## 7.2.9 BinderySignature

[New in JDF 1.2](#)

The **BinderySignature** is conceptually a folding dummy. It represents multiple pieces of paper, which are folded together in the folder. It is a reusable, size-independent object.

### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>StrippingParams</b>
<b>Example Partition:</b>	<i>WebName</i>
<b>Input of processes:</b>	—
<b>Output of processes:</b>	—

## Resource Structure

Name	Data Types	Description
<i>NumberUp</i> = "1 1"	XYPair	Specifies a regular, multi-up grid of <b>SignatureCells</b> into which content pages are mapped. The first value specifies the number of columns of <b>SignatureCells</b> , and the second value specifies the number of rows of <b>SignatureCells</b> in the multi-up grid (both numbers are integers). <i>NumberUp</i> must only be specified in the <b>BinderySignature</b> root.
<i>BindingEdge</i> = "Left"	enumeration	Specifies the binding edge of this <b>BinderySignature</b> . One of: <i>Left</i> <i>Right</i> <i>Top</i> <i>Bottom</i> <i>None</i>
<i>FoldCatalog</i> ?	string	Describes the type of fold according to the folding catalog in the format " <i>Fx-y</i> " as shown in the <b>Fold</b> resource (See "Fold" on page 366.) One of <b>SignatureCell</b> , <i>FoldCatalog</i> , or <b>Fold</b> must be specified.
<b>Fold</b> *	element	Describes the folding operations in the sequence in which they should be carried out. When both <b>Fold</b> and <b>FoldCatalog</b> are specified, <b>FoldCatalog</b> defines the topology of the folding scheme, and the specifics of each individual fold are described by the <b>Fold</b> elements. The <b>Fold</b> elements have precedence. Only one of <b>SignatureCell</b> and <b>Fold</b> must be specified.
<b>SignatureCell</b> *	element	Describes the <b>SignatureCells</b> used in this <b>BinderySignature</b> . <b>SignatureCell</b> elements are ordered in X-Y direction starting at the lower left-hand corner of the <b>BinderySignature</b> . When both <b>SignatureCell</b> and <b>FoldCatalog</b> are specified, <b>FoldCatalog</b> defines the topology of the folding scheme, and the specifics of each individual signature cell are described by the <b>SignatureCell</b> elements. The <b>SignatureCell</b> elements have precedence. Only one of <b>SignatureCell</b> or <b>Fold</b> must be specified.

## Structure of the SignatureCell Subelement

**SignatureCell** elements describe a set of individual page cells in a **BinderySignature**.

**Note:** "Page number" in the table below refers to finished pages numbered from 0-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-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 may contain final page numbers only when **Assembly/@Order="None"**.

Name	Data Types	Description
<i>BackFacePages</i> ?	IntegerList	Page numbers for the back finished pages forming a foldout.
<i>BackPages</i> ?	IntegerList	Page numbers of the back finished pages of a <b>SignatureCell</b> . The number of entries in <b>FrontPage</b> and <b>BackPage</b> must be identical. The entries with an identical index in <i>FrontPages</i> and <i>BackPages</i> are back-to-back in the <b>Layout</b> . If not specified, the layout is one-sided.
<i>BottleAngle</i> ?	double	Indicates the bottle angle, which is the slight rotation of the <b>SignatureCell</b> required to compensate for the rotation fault introduced when making cross-folds.

Name	Data Types	Description
<i>BottleAxis</i> ?	enumeration	Indicates the point around which the cell is bottled. One of: <i>FaceFoot</i> <i>FaceHead</i> <i>SpineFoot</i> <i>SpineHead</i>
<i>FrontFacePages</i> ?	IntegerList	Page numbers for the front finished pages forming a foldout.
<i>FrontPages</i> ?	IntegerList	Page numbers of the front finished pages of a <b>SignatureCell</b> . Multiple page cells with the same properties except for the finished pages to which they are assigned may be summarized as one <b>SignatureCell</b> with multiple entries in <i>FrontPages</i> .
<i>Orientation</i> = "Up"	enumeration	Indicates the orientation of the <b>SignatureCell</b> . One of: <i>Down</i> – 180° rotation. <i>Up</i> – 0° rotation.
<i>SectionIndex</i> = "0"	integer	Unique logical index of the page section that should fill this <b>SignatureCell</b> . This is an undirected logical index. The actual section index is defined in <b>StrippingParams</b> / <i>@SectionList</i> .

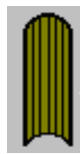
### 7.2.10 BlockPreparationParams

[New in JDF 1.1](#)

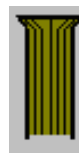
This resource describes the settings of a **BlockPreparation** process. For the tightbacking there are four different kinds of book forms:



flat  
*Flat*



round  
*Round*



flat and backed  
*FlatBacked*

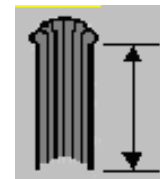
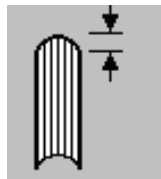


rounded and backed  
*RoundBacked*

For the rounding and for the backing there are two additional measurements:

Rounding: rounding way

Backing: backing way



#### Resource Properties

Resource class: Parameter  
 Resource referenced by: —  
 Example Partition: —  
 Input of processes: **BlockPreparation**

## Resource Structure

Name	Data Type	Description
<i>Backing ?</i>	double	Backing distance in points.
<i>Rounding ?</i>	double	Rounding distance in points.
<i>TightBacking ?</i>	enumeration	Definition of the geometry of the back of the book block. Allowed values are: <i>Flat</i> <i>FlatBacked</i> – Backing way <i>Round</i> – Rounding way <i>RoundBacked</i> – Rounding way, backing way
<b>RegisterRibbon *</b>	refelement	Description of the register ribbons that are included within the book block.

### 7.2.11 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:	<b>BoxPacking</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<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>FillMaterial ?</i>	NMTOKEN	Material to fill boxes that are not completely filled. 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>

### 7.2.12 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:	<b>Buffer</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>MinimumWait ?</i>	duration	Minimum amount of time that an individual resource must be buffered.

## 7.2.13 Bundle

[New in JDF 1.1](#)

Bundles are used to describe various kinds of sets of **Components**. Note that **Bundles** may be created by many press or postpress processes and not only **Bundling**.

### Resource Properties

Resource class:	Quantity
Resource referenced by:	<b>Component</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>BundleType</i> = "Stack"	enumeration	One of: <i>BoundSet</i> – Stack of components that are bound together. <i>Box</i> <i>Carton</i> <i>CollectedStack</i> - Components collected on a saddle, result of collecting process <i>CompensatedStack</i> - Loose stack of compensated components <i>Pallet</i> <i>Roll</i> - Rolled components on a print roll. <i>Sheet</i> – Multiple individual items printed onto one Sheet. <i>Stack</i> – Loose stack of equally stacked components. <i>StrappedStack</i> - Strapped stack of equally stacked components. <i>StrappedCompensatedStack</i> - Strapped stack of compensated components. <i>WrappedBundle</i>
<i>FolioCount</i> ?	integer	Total amount of individual finished pages that this bundle contains. If not specified, it must be calculated from the individual <b>BundleItems</b> .
<i>ReaderPageCount</i> ?	integer	Total amount of individual reader pages that this bundle contains. If not specified, it must be calculated from the individual <b>BundleItems</b> .
<i>TotalAmount</i> ?	integer	Total amount of individual products that this bundle contains. If not specified, it must be calculated from the individual <b>BundleItems</b> .
<b>BundleItem</b> *	element	References to the individual items that form this <b>Bundle</b> .

### Structure of the BundleItem Element

A **Bundle** is described as a set of **BundleItems**. Since **BundleItems** reference **Components** which themselves may reference further **Bundles**, the structure is recursive.

Name	Data Type	Description
<i>Amount</i>	integer	Number of this type of items.
<i>ItemName</i> ? <a href="#">New in JDF 1.2</a>	NMTOKEN	Name of the bundle item. Used for referencing individual <b>BundleItems</b> in a <b>Bundle</b> .
<i>Orientation</i> ?	Orientation	Named Orientation of the <b>Component</b> respective to the <b>Bundle</b> coordinate system. For details, see Table 2-3, "Matrices and Orientation values used to describe the orientation of a Component," on page 24. Only one of <i>Orientation</i> or <i>Transformation</i> must be specified.
<i>Transformation</i> ?	matrix	Orientation of the <b>Component</b> respective to the <b>Bundle</b> coordinate system.
<b>Component</b>	referencelement	Reference to a <b>Component</b> that is part of this <b>Bundle</b> .

The following example code shows a JDF that describes Boxing and Palletizing for 4200 books. The appropriate **Bundle** elements are **highlighted**. The resources have not yet been completely filled in.

```
<JDF xmlns="http://www.CIP4.org/JDFSchemas_1_1" ID="Bundle" Status="Waiting"
Type="ProcessGroup" Version="1.2">
  <!-- The BoxPacking process consumes the thing to pack and the boxes-->
  <!-- The BoxPacking process creates packed boxes -->
  <JDF ID="n0235" Status="Waiting" Type="BoxPacking">
    <ResourceLinkPool>
      <ComponentLink ProcessUsage="Box" Usage="Input" rRef="BoxID"/>
      <BoxPackingParamsLink Usage="Input" rRef="BoxParamsID"/>
      <ComponentLink Usage="Input" rRef="ComponentID"/>
      <ComponentLink Usage="Output" rRef="PackedBoxID"/>
    </ResourceLinkPool>
    <!-- The BoxPacking process has the following local resources -->
    <ResourcePool>
      <BoxPackingParams Class="Parameter" ID="BoxParamsID" Status="Available"/>
      <Component Amount="100" Class="Quantity" ID="BoxID" Status="Available"/>
    </ResourcePool>
  </JDF>
  <ResourcePool>
    <!-- This Component describes a Box with 42 Books -->
    <Component Amount="100" Class="Quantity" ID="PackedBoxID" Status="Unavailable">
      <Bundle BundleType="Box" TotalAmount="42">
        <BundleItem Amount="42">
          <ComponentRef rRef="ComponentID"/>
        </BundleItem>
      </Bundle>
    </Component>
    <Component Amount="4200" Class="Quantity" ID="ComponentID" Status="Available"/>
    <!-- This Component describes the contents of the pallet: 100 Boxes w. 42 Books -->
    <Component Amount="10" Class="Quantity" ID="palletContentsID" Status="Unavailable">
      <Bundle BundleType="Pallet" TotalAmount="420">
        <BundleItem Amount="10">
          <ComponentRef rRef="PackedBoxID"/>
        </BundleItem>
      </Bundle>
    </Component>
  </ResourcePool>
  <JDF ID="n0239" Status="Waiting" Type="Palletizing">
    <ResourceLinkPool>
      <ComponentLink Usage="Input" rRef="PackedBoxID"/>
      <PalletLink Usage="Input" rRef="palletID"/>
      <PalletizingParamsLink Usage="Input" rRef="palletParamsID"/>
      <ComponentLink Usage="Output" rRef="palletContentsID"/>
    </ResourceLinkPool>
    <ResourcePool>
      <Pallet Amount="10" Class="Consumable" ID="palletID" Status="Available"/>
      <PalletizingParams Class="Parameter" ID="palletParamsID" Status="Available"/>
    </ResourcePool>
  </JDF>
</JDF>
```

## 7.2.14 BundlingParams

[New in JDF 1.2](#)

**BundlingParams** describes the details of a **Bundling** process.

### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource references:</b>	—
<b>Example Partition:</b>	—
<b>Input of processes:</b>	<b><i>Bundling</i></b>
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
<i>Copies</i> ?	integer	Number of copies within a bundle. Only one of <i>Copies</i> and <i>Length</i> must be specified.
<i>Length</i> ?	double	Length of a bundle. Only one of <i>Copies</i> and <i>Length</i> must be specified.

## 7.2.15 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 or may not 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 may be replaced.

### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource references:</b>	<b>RunList</b>
<b>Example Partition:</b>	—
<b>Input of processes:</b>	
<b>Output of processes:</b>	

### Resource Structure

Name	Data Type	Description
<i>BandOrdering</i> ?	enumeration	Identifies the precedence given when ordering the produced bands. Possible values are: <i>BandMajor</i> – The position of the bands on the page is prioritized over the color. <i>ColorMajor</i> – All bands of a single color are played in order before progressing to the next plane. This is only possible with non-interleaved data. This field is required for non-interleaved data and is ignored for interleaved data.
<i>FrameHeight</i>	integer	Height of the overall image that may be broken into multiple bands
<i>FrameWidth</i>	integer	Width of overall image that may be broken into multiple columns
<i>Halftoned</i>	boolean	Indicates whether or not the data has been halftoned.
<i>Interleaved</i>	boolean	If "true", the data are interleaved, or chunky. Otherwise the data are non-interleaved, or planar.
<i>PixelSkip</i> ?	integer	Number of bits to skip between pixels of interleaved data.
<i>Resolution</i>	XYPair	Output resolution.
Band +	element	Array of bands containing raster data.
<b>ColorPool</b> ? <a href="#">New in JDF 1.2</a>	refelement	Details of the colors represented in this <b>Bytemap</b> .
<b>FileSpec</b> ( <i>RasterFileLocation</i> )?	refelement	A <b>FileSpec</b> resource pointing to a location where the raster should be (or already is) stored.
PixelColorant +	element	Ordered list containing information about which colorants are represented and how many bits per pixel are used.

### Structure of Band Subelement

Name	Data Type	Description
<i>Data</i>	URL	Actual bytes of data.
<i>Height</i>	integer	Height in pixels of the band.
<i>Mask</i> ?	URL	1-bit mask of raster data indicating which bits of the band data should actually be used. It is required that the mask dimensions and resolution be equivalent to the contents of the band itself.
<i>WasMarked</i>	boolean	Indicates whether any rendering marks were made in this band. This attribute allows a band to be skipped if no marks were made in the band.
<i>Width</i>	integer	Width in pixels of the band.

### Structure of PixelColorant Subelement

Name	Data Type	Description
<i>ColorantName</i>	string	Name of colorant.
<i>PixelDepth</i>	integer	Number of bits per pixel for each colorant.

### 7.2.16 CaseMakingParams

[New in JDF 1.1](#)

This resource describes the settings of a **CaseMaking** process.

#### Resource Properties

Resource class: Parameter  
 Resource referenced by: —  
 Example Partition: —  
 Input of processes: **CaseMaking**

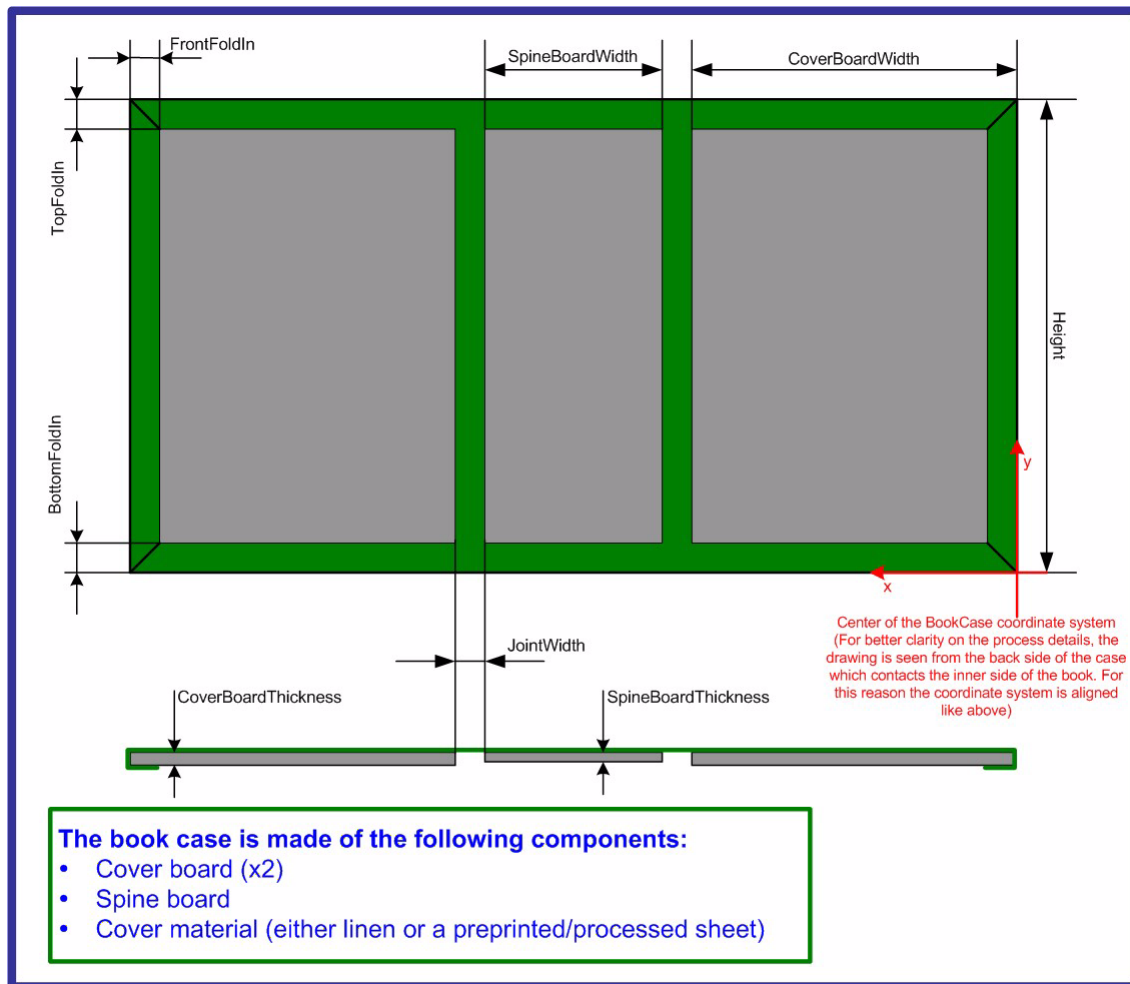


Figure 7.1: CaseMakingParams



## Resource Structure

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. Possible values include: <i>LibraryCorner</i> – The American Library Corner style.
<i>FrontFoldIn</i> ?	double	Defines the width of the part of the cover material on the front edges inside of the case.
<i>Height</i> ?	double	Height of the book case, in points.
<i>JointWidth</i> ?	double	Width of the joint as seen when laying the cardboard on the cover material, in points.
<i>SpineWidth</i> ?	double	Width of the spine cardboard, in points.
<i>TopFoldIn</i> ?	double	Defines the width of the cover material on the top edge inside of the case.
<b>GlueLine</b> ?	refelement	As the glue is applied to the whole back side of the cover material, <b>GlueLine/@AreaGluing</b> must be set to "true".

### 7.2.17 CasingInParams

[New in JDF 1.1](#)

This resource describes the settings of a **CasingIn** process. The geometry is always centered See Figure 7.2.

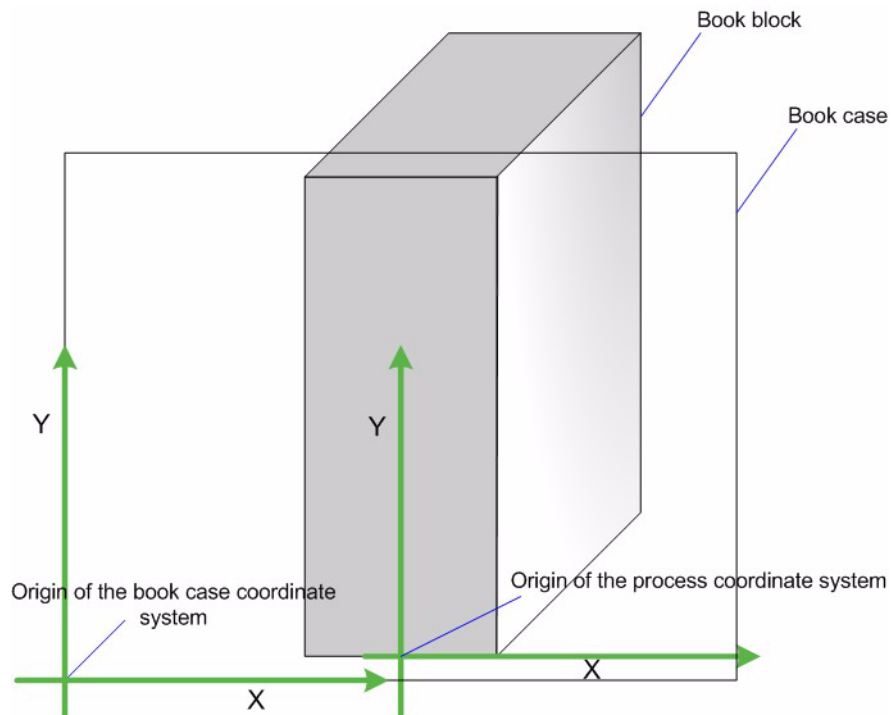


Figure 7.2: Parameters and Coordinate System for CasingIn

## Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>CasingIn</b>

## Resource Structure

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.
<b>GlueLine</b> +	refelement	Properties of the glue used.

### 7.2.18 ChannelBindingParams

This resource describes the details of the **ChannelBinding** process. Figure 7.3 depicts the **ChannelBinding** process.

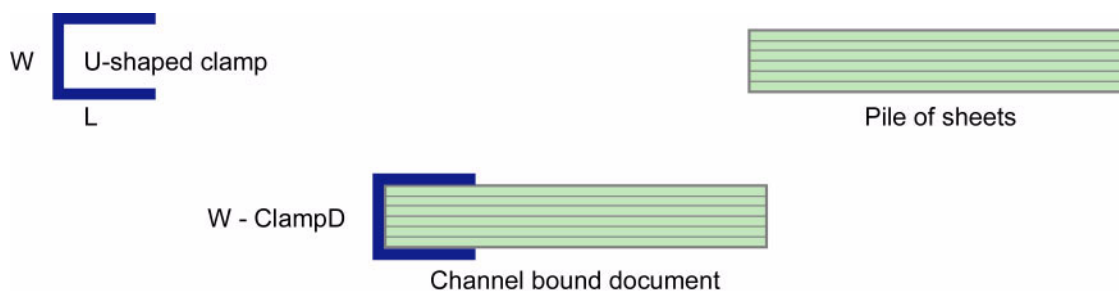


Figure 7.3: Parameters used for channel binding

The symbols W, L, and ClampD of Figure 7.3 are described by the attributes *ClampD* and *ClampSize* of the table below.

## Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>ChannelBinding</b>
Output of processes:	—

## Resource Structure

Name	Data Type	Description
<i>Brand</i> ?	string	The name of the clamp (or preassembled cover with clamp) manufacturer and the name of the specific item.
<i>ClampColor</i> ?	NamedColor	Determines the color of the clamp/cover. If <i>ClampSystem</i> = <i>true</i> , then the color of the cover is also meant.
<i>ClampD</i> ?	double	The distance of the clamp that was “pressed away” (see Figure 7.3 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 7.3) 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 7.3). The third corresponds to the spine length (not visible in Figure 7.3). The spine length is perpendicular on the paper plane.
<i>ClampSystem</i> = “false”	boolean	If “true” the clamp is inside of a preassembled cover.

## 7.2.19 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:	<b>ColorControlStrip, Surface</b>
Example Partition:	—
Input of processes:	Any printing process
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>Center</i>	XYPair	Position of the center of the color measuring field in the coordinates of the <b>MarkObject</b> that contains this mark. If the measuring field is defined within a <b>ColorControlStrip</b> , <i>Center</i> refers to the rectangle defined by <i>Center</i> and <i>Size</i> of the <b>ColorControlStrip</b> .
<i>CIELab</i>	LabColor	L*a*b* color specification.
<i>DensityStandard</i> ? <a href="#">Deprecated in JDF 1.1</a>	enumeration	Density filter standard used during density measurements. Possible values are: <i>ANSIA</i> – ANSI Status A <i>ANSIE</i> – ANSI Status E <i>ANSII</i> – ANSI Status I <i>ANSIT</i> – ANSI Status T. <i>DIN16536</i> <i>DIN16536NB</i> In JDF 1.1 and beyond, use <b>ColorMeasurementConditions/ @DensityStandard</b>
<i>Diameter</i> ? <a href="#">Modified in JDF 1.1</a>	double	Diameter of the measuring field.
<i>Light</i> <a href="#">Deprecated in JDF 1.1</a>	NMTOKEN	Type of light. Possible values include: <i>D50</i> <i>D65</i>
<i>Observer</i> ? <a href="#">Deprecated in JDF 1.1</a>	integer	Observer in degree (2 or 10). In JDF 1.1 and beyond, use <b>ColorMeasurementConditions/ @Observer</b>
<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> ? <a href="#">Deprecated in JDF 1.1</a>	string	Description of measurement setup. In JDF 1.1 and beyond, use details from <b>ColorMeasurementConditions</b>
<i>Tolerance</i> ? <a href="#">Modified in JDF 1.1</a>	double	Tolerance in ΔE.
<b>ColorMeasurementConditions</b> ? <a href="#">New in JDF 1.1</a>	refelement	Detailed description of the measurement conditions for color measurements.

### 7.2.20 CoilBindingParams

This resource describes the details of the **CoilBinding** process.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>CoilBinding</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>Brand ?</i>	string	The name of the coil manufacturer and the name of the specific item.
<i>Color ?</i>	NamedColor	Determines the color of the coil.
<i>Diameter ?</i>	double	The coil diameter to be produced is determined by the height of the block of sheets to be bound.
<i>Material ?</i>	enumeration	The material used for forming the coil binding: <i>LaqueredSteel</i> <i>NylonCoatedSteel</i> <i>PVC</i> <i>TinnedSteel</i> <i>ZincsSteel</i>
<i>Shift ?</i> <a href="#">Deprecated in JDF 1.2</a>	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 <b>HoleMakingParams</b> / <i>@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".
<b>HoleMakingParams ?</b> <a href="#">New in JDF 1.2</a>	refelement	Details of the holes in <b>CoilBinding</b> .

### 7.2.21 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:	—
Input of processes:	<b>Collecting</b>
Output of processes:	

#### Resource Structure

Name	Data Type	Description
------	-----------	-------------

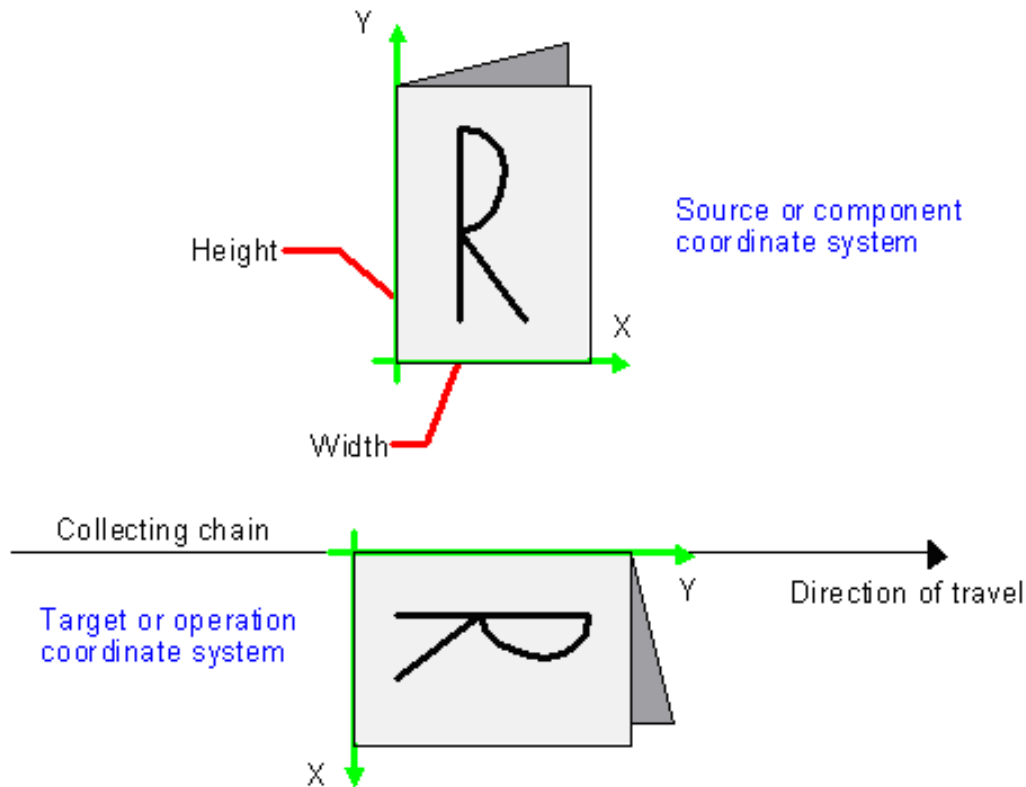


Figure 7.4: Coordinate systems used for collecting

## 7.2.22 Color

### [Clarified in JDF 1.2](#)

JDF describes spot color inks and, along that line, process color inks. Spot colors are named colors that may 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** = "Device*CMYK*"). 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 it 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 colorbooks such as Pantone.

**Resource Properties**

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>ColorPool</b>
<b>Example Partition:</b>	—
<b>Input of processes:</b>	—
<b>Output of processes:</b>	—

**Resource Structure**

Name	Data Type	Description
<i>CMYK</i> ? <a href="#">Modified in JDF 1.2</a>	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 <b>FileSpec</b> element with a <i>ResourceUsage</i> = "ColorProfile" when the target CMYK is different from the PDL CMYK.
<i>ColorBook</i> ? <a href="#">Modified in JDF 1.2</a>	string	Definition of the color identification book name that is used to represent this color. The colorbook name must match the name defined by the colorbook vendor. Examples include: <i>PANTONE C</i> <i>CIP4 ColorBook Uncoated Grade 5</i> <b>Note:</b> The data type was modified from NMTOKEN to string in JDF 1.2.
<i>ColorBookEntry</i> ? <a href="#">Modified in JDF 1.2</a>	string	Definition of the <b>Color</b> within the <i>ColorBook</i> standard. This entry must exactly match the <i>Colorbook</i> entry as defined by the <i>ColorBook</i> vendor including capitalization and media type extension. When using ICC Profiles, this maps to the NCL2 value of a named-ColorType tag of an ICC color profile. This entry is used to map from the JDF <b>Color</b> to an ICC namedColorType tag.
<i>ColorBookPrefix</i> ?	string	Definition of the name prefix of the colorbook entry within a named ICC profile. This entry is used to map from the JDF <b>Color</b> 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 <b>Color</b> to an ICC namedColorType tag.
<i>ColorName</i> ? <a href="#">New in JDF 1.1</a>	NamedColor	Mapping to a color name. Allowed values are defined in Section A.3.3.2, NamedColor.

Name	Data Type	Description
<p><i>ColorType</i> ?  <a href="#">Modified in JDF 1.2</a></p>	enumeration	<p>A name that characterizes the colorant. Possible values are:</p> <p><i>DieLine</i> – Marks made with colorants of this type are ignored for trapping. Trapping processes need not generate a color plane for this colorant. <i>DieLine</i> can be used for auxiliary process separations. <i>DieLine</i> marks will generally appear on proof output but will not be marked on final output, (e.g., plates). Note that the <b>ColorantControl</b> resource must be correctly set up for the RIP and that <i>ColorType</i> = "<i>DieLine</i>" does not implicitly remove the <i>DieLine</i> separation from final output.</p> <p><i>Normal</i> – Marks made with colorants of this type, marks covered by colorants of this type, and marks on top of colorants of this type are trapped.</p> <p><i>Transparent</i> – Marks made with colorants of this type should be ignored for trapping. Trapping processes should not generate a color plane for this colorant. <i>ColorType</i> = "<i>Transparent</i>" should be used for varnish.</p> <p><i>Opaque</i> – Marks covered by colorants of this type are ignored for trapping. <i>Opaque</i> can be used for metallic inks.</p> <p><i>OpaqueIgnore</i> – Marks made with colorants of this type and marks covered by colorants of this type are ignored for trapping. <i>OpaqueIgnore</i> can be used for metallic inks.</p>
<p><i>Density</i> ?  <a href="#">New in JDF 1.2</a></p>	double	<p>Density value of colorant (100% tint). Whereas <i>NeutralDensity</i> describes measurements of inks on substrate with wide-band filter functions, <i>Density</i> is derived from measurements of inks on substrate with special small-band filter functions according to ANSI and DIN.</p>
<p><i>Lab</i> ?</p>	LabColor	<p>L, a, b value of the 100% tint value of the colorant.</p>
<p><i>MappingSelection</i> =  "<i>UsePDLValues</i>"  <a href="#">New in JDF 1.2</a></p>	enumeration	<p>This value specifies the mapping method to be used for this Color. Possible values are:</p> <p><i>UsePDLValues</i> – Use color values specified in the PDL for this color. See [ColorPS].</p> <p><i>UseLocalPrinterValues</i> – Use the Printer’s best local mapping for this <b>Color</b>.</p> <p><i>UseProcessColorValues</i> – Use the values defined in this <b>Color</b>.</p> <p><b>Color/@MappingSelection</b> can be specifically used to indicate how a combination of process colorant values will be obtained for any spot color when the separation spot colorant itself is not to be used.</p>

Name	Data Type	Description
<i>MediaType</i> ? <a href="#">Modified in JDF 1.2</a>	string	Specifies the media type. Possible values include: <i>Coated</i> – Pertains to gloss coated. <i>Matte</i> – Pertains to matte or dull coated. <i>Uncoated</i>
<i>Name</i> <a href="#">Clarified in JDF 1.2</a>	string	Name of the colorant. This is the value that must match the <i>Name</i> attribute of a <b>SeparationSpec</b> that references this color, (e.g., in <b>ColorantControl/DeviceNSpace/SeparationSpec/@Name</b> or <b>ColorantControl/ColorantParams/SeparationSpec/@Name</b> ). This <i>Name</i> attribute may also be referenced from the <i>Name</i> attribute in the <b>Ink</b> resource. Name may also be referenced from <b>ColorantAlias/@ReplacementColorantName</b> . Only one <b>Colorant</b> with any given <i>Name</i> must be specified in a <b>ColorPool</b> .
<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> ? <a href="#">New in JDF 1.2</a>	hexBinary	Representation of the original 8-bit byte stream of the <b>Color Name</b> . Used to transport the original byte representation of a <b>Color Name</b> when moving JDF tickets between computers with different locales. Only one <b>Colorant</b> with any given <i>RawName</i> must be specified in a <b>ColorPool</b> .
<i>sRGB</i> ?	sRGBColor	sRGB value of the 100% tint value of the colorant.
<i>UsePDALternateCS</i> ? <a href="#">Deprecated in JDF 1.2</a>	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 <b>DeviceNColor</b> of this <b>Color</b> must be used depending on the value of <b>ProcessColorModel</b> in <b>ColorantControl</b> . In JDF 1.2 and beyond, use <i>MappingSelection</i> .
<b>ColorMeasurementConditions</b> ? <a href="#">New in JDF 1.1</a>	refelement	Detailed description of the measurement conditions for color measurements.
<b>DeviceNColor</b> * <a href="#">Clarified in JDF 1.2</a>	element	Elements that define the colorant in a non-standard device-dependent process color space. <i>DeviceNColor</i> can be specified when <i>Name</i> is a spot colorant (not one of the <i>DeviceNSpace</i> colorants) and <i>ProcessColorModel</i> = "DeviceN" in <b>ColorantControl</b> .
<b>FileSpec</b> ( <i>ColorProfile</i> )? <a href="#">Clarified in JDF 1.2</a>	refelement	A <b>FileSpec</b> 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.



Name	Data Type	Description
<b>FileSpec</b> ( <i>TargetProfile</i> )?	refelement	A <b>FileSpec</b> resource pointing to an ICC profile that defines the target output device in case the object that uses the <b>Color</b> has been color space converted to a device color space. TargetProfile applies to the alternate color defined by the value of <i>UsePDLAlternateCS</i> .
<b>PrintConditionColor</b> * <a href="#">New in JDF 1.2</a>	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.).
<b>TransferCurve</b> * <a href="#">Modified in JDF 1.1</a>	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”

### Structure of DeviceNColor Subelement

Name	Data Type	Description
<i>ColorList</i>	DoubleList	Value of the 100% tint value of the colorant in the ordered DeviceN space. The list must have <i>N</i> elements. A value of 0 specifies no ink and a value of 1 specifies full ink. The mapping of indices to colors is specified in the DeviceNSpace element of the <b>ColorantControl</b> 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 <b>ColorantControl/DeviceNSpace/@Name</b> .

### Structure of PrintConditionColor Subelement

#### [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., half-tone) 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.

Name	Data Type	Description
<i>CMYK</i> ?	CMYK-Color	<i>CMYK</i> of the PrintConditionColor. If not specified, defaults to the parent <b>Color/@CMYK</b> .
<i>ColorBook</i> ?	string	<i>ColorBook</i> of the PrintConditionColor. If not specified, defaults to the parent <b>Color/@ColorBook</b> .
<i>ColorBookEntry</i> ?	string	<i>ColorBookEntry</i> of the PrintConditionColor. If not specified, defaults to the parent <b>Color/@ColorBookEntry</b> .
<i>ColorBookPrefix</i> ?	string	<i>ColorBookPrefix</i> of the PrintConditionColor. If not specified, defaults to the parent <b>Color/@ColorBookPrefix</b> .
<i>ColorBookSuffix</i> ?	string	<i>ColorBookSuffix</i> of the PrintConditionColor. If not specified, defaults to the parent <b>Color/@ColorBookSuffix</b> .
<i>Density</i> ?	double	<i>Density</i> of the PrintConditionColor. If not specified, defaults to the parent <b>Color/@Density</b> .
<i>Lab</i> ?	LabColor	<i>Lab</i> of the PrintConditionColor. If not specified, defaults to the parent <b>Color/@Lab</b> .

<b>MappingSelection</b> ? <a href="#">New in JDF 1.2</a>	enumeration	This value specified the mapping method to be used for this Color. Possible values are: <i>UsePDLValues</i> – Use color values specified in the PDL for this color. See [ColorPS]. <i>UseLocalPrinterValues</i> – Use the Printer's best local mapping for this <b>Color</b> . <i>UseProcessColorValues</i> – Use the values defined in this <b>Color</b> . If not specified, defaults to the parent <b>Color/@MappingSelection</b> .
<b>MediaSide</b> = "Both"	enumeration	<b>Media</b> front and back surfaces can be different, affecting color results. If the <b>Media/@FrontCoatings</b> , <b>Media/@BackCoatings</b> , or <b>Media/@Gloss</b> attributes indicate differences in surface then Media-Side can be used to specify the side of the media to which the <b>PrintConditionColor</b> attributes pertain. Values are: <i>Front</i> <i>Back</i> <i>Both</i>
<b>NeutralDensity</b> ?	double	<b>NeutralDensity</b> of the <b>PrintConditionColor</b> . If not specified, defaults to the parent <b>Color/@NeutralDensity</b>
<b>PrintConditionName</b> ?	NMTOKEN	<b>PrintConditionName</b> specifies a particular screening condition and printing condition that this <b>PrintConditionColor</b> element applies to. In order to map a <b>PrintCondition</b> with a <b>PrintConditionColor</b> , <b>PrintConditionName</b> must match <b>PrintCondition/@Name</b> . If not specified, this <b>PrintConditionColor</b> matches all <b>PrintCondition</b> but may still be dependent on <b>Media</b> .
<b>sRGB</b> ?	sRGBColor	<b>sRGB</b> of the <b>PrintConditionColor</b> . If not specified, defaults to the parent <b>Color/@sRGB</b> .
<b>DeviceNColor</b> *	element	<b>DeviceNColor</b> of the <b>PrintConditionColor</b> . If not specified, defaults to the parent <b>Color/@DeviceNColor</b> .
<b>FileSpec</b> ( <i>TargetProfile</i> )	refelement	<b>FileSpec</b> ( <i>TargetProfile</i> ) of the <b>PrintConditionColor</b> . If not specified, defaults to the parent <b>Color/FileSpec</b> ( <i>TargetProfile</i> )
<b>Media</b> *	refelement	Specifies one or more <b>Media</b> that this <b>PrintConditionColor</b> applies to. When <b>PrintConditionColor</b> is present, the parent attribute, <b>Color/@MediaType</b> , is ignored. If <b>Media</b> is not specified, <b>PrintConditionColor</b> applies to print processes with a matching <b>PrintConditionName</b> .
<b>TransferCurve</b> *	refelement	<b>TransferCurve</b> of the <b>PrintConditionColor</b> . If not specified, defaults to the parent <b>Color/TransferCurve</b> .

### Color Example

This is an example of the structure for **Color**. The transfer curves in this example are defined for process CMYK and sRGB, independently.

```
<Color CMYK="0.2 0.3 0.4 0.5" Density="3.14" Lab="20. 30. 40." MediaType="Coated"
Name="PANTONE Deep Blue" sRGB="0.6 0.7 0.9">
  <TransferCurve Curve="0 0 .5 .4 1 1" Separation="Cyan"/>
  <TransferCurve Curve="0 0 .5 .6 1 1" Separation="Magenta"/>
  <TransferCurve Curve="0 0 1 1" Separation="Yellow"/>
  <TransferCurve Curve="0 0 1 1" Separation="Black"/>
  <TransferCurve Curve="0 0 1 1" Separation="sRed"/>
  <TransferCurve Curve="0 0 1 1" Separation="sGreen"/>
  <TransferCurve Curve="0 0 1 1" Separation="sBlue"/>
</Color>
```

## 7.2.23 ColorantAlias

[Modified in JDF 1.2](#)

**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* = "Pantone 135 CV", "PANTONE 135", and *ReplacementColorantName* = "PANTONE 135 C" maps string values: "Pantone 135 CV" and "PANTONE 135" to the string value: "PANTONE 135 C". Note that **ColorantAlias** was elevated from a sub-element of **ColorantControl** to a top level resource in JDF 1.2.

### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<b>ColorIntent, ColorantControl</b>
Input of Processes:	—
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>ReplacementColorantName</i> <a href="#">Clarified in JDF 1.2</a>	string	The value of the colorant name string to be substituted for the colorant name strings in the SeparationSpec element list.
<i>SeparationSpec</i> + <a href="#">Modified in JDF 1.2</a>	refelement	The names of the colorants to be replaced in PDL files.

## 7.2.24 ColorantControl

[Modified in JDF 1.2](#)

**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 must be translated into device colorant information.

Colorants are referenced in **ColorantControl** by name only. Additional details about individual colorants can be found in the **COLOR** element of the **ColorPool** resource. **ColorantControl** 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 should 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/@ColorantNameSet** 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/@ColorantNameSet** 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**

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>DocIndex, RunIndex, RunTags, SheetName, Side, SignatureName</i>
<b>Input of Processes:</b>	<b><i>ConventionalPrinting, ColorSpaceConversion, DigitalPrinting, ImageSetting, Interpreting, PreviewGeneration, Separation, Trapping</i></b>
<b>Output of processes:</b>	—

**Resource Structure**

Name	Data Type	Description
<i>ForceSeparations</i> = "false"	boolean	If "true", forces all colorants to be output as individual separations, regardless of any values defined in <b>ColorantControl</b> , (i.e., all separations in a document are assumed to be valid and are output individually). A value of "false" specifies to respect the parameters specified in <b>ColorantControl</b> and elsewhere in the JDF.
<i>ProcessColorModel</i> ? <a href="#">Clarified in JDF 1.2</a>	NMTOKEN	Specifies the model to be used for rendering the colorants defined in color spaces into process colorants. Possible values include: <i>DeviceCMY</i> <i>DeviceCMYK</i> <i>DeviceGray</i> <i>DeviceN</i> — The specific DeviceN color space to operate on is defined in the <b>DeviceNSpace</b> resource. If this value is specified then the <b>DeviceNSpace</b> and <b>ColorPool</b> relements must also be present. <i>DeviceRGB</i>
<b>ColorantAlias</b> * <a href="#">Modified in JDF 1.2</a>	relement	Identify one or more named colorants that should be replaced with a specified named colorant. The identified colorant remappings in this <b>ColorantAlias</b> may be consolidated for processing from the information received in the <b>LayoutElement/ElementColorParams/ColorantAlias</b> resources with the job content.
<i>ColorantOrder</i> ? <a href="#">Clarified in JDF 1.2</a>	element	The ordering of named colorants to be processed, for example in the RIP. All of the colorants named must either occur in the <b>ColorantParams</b> 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 <b>ColorantParams</b> list, or implied by the <i>ProcessColorModel</i> , but not listed in <i>ColorantOrder</i> , must not be output. They must still be processed for side effects in the colorants that are listed such as knockouts or trapping. If not present, then all colorants specified in <b>ColorantParams</b> and implied by <i>ProcessColorModel</i> are output. The explicit or implied value of <i>ColorantOrder</i> may be modified by an implied partition of the <b>ColorantControlLink</b> . If one or more <b>ColorantControlLink/Part/@Separation</b> are specified, <i>ColorantOrder</i> is reduced to the list. It is an error to specify values of <b>ColorantControlLink/Part/@Separation</b> that are not explicitly stated or implied by <i>ColorantOrder</i> .

Name	Data Type	Description
<b>ColorantParams</b> ? <a href="#">Clarified in JDF 1.2</a>	element	A set of named colorants. This list defines all the colorants that are expected to be available on the device where the process will be executed. Named colors found in the PDL that are not listed in <b>ColorantParams</b> will be implemented through their <i>ProcessColorModel</i> equivalents. (See <b>ElementColorParams</b> 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 <b>ColorIntent/ColorsUsed</b> will in general be mapped to <b>ColorantParams</b> for each spot color to be used as part of any intent to process conversion.
<b>ColorPool</b> ? <a href="#">Clarified in JDF 1.2</a>	refelement	Pool of <b>Color</b> 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 <b>ColorPool</b> . The subset that <i>ColorantControl</i> uses from <b>ColorPool</b> is the subset of or all <i>ProcessColorModel</i> colors, and the subset of or all spot colors designated to be processed in this instance using specific separation colorants. <b>ColorPool</b> 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 <b>ColorPool/Color/@MappingSelection</b> = "UseProcessColorValues". In that case, the process color equivalent for the spot color is taken from the available information in the <b>Color</b> resource for that spot color.
<b>ColorSpaceSubstitute</b> *	element	These subelements identify a colorant that should be replaced by another colorant.
<b>DeviceColorantOrder</b> ? <a href="#">Clarified in JDF 1.2</a>	element	The ordering of named colorants (e.g., laydown order) to be output on the device <sup>a</sup> , such as press modules. All of the colorants named must occur in <b>ColorantOrder</b> if it is present. If <b>ColorantOrder</b> is not present, then all of the colorants named must occur in the <b>ColorantParams</b> list, or be implied by the <i>ProcessColorModel</i> . If the <b>DeviceColorantOrder</b> element is not specified, the colorant laydown order defaults to <b>ColorantOrder</b> .
<b>DeviceNSpace</b> * <a href="#">Modified in JDF 1.2</a>	refelement	Defines the colorants that make up a DeviceN color space. The <b>DeviceNSpace</b> attribute is required when the <i>ProcessColorModel</i> value is <i>DeviceN</i> .

- a. Note that this must be synchronized with the device output ICC profile.

### Structure of ColorantOrder, ColorantParams, and DeviceColorantOrder Elements

Name	Data Type	Description
<b>SeparationSpec</b> * <a href="#">Modified in JDF 1.2</a>	refelement	The names of the colorants that define the respective lists.

### Structure of ColorSpaceSubstitute Subelement

Name	Data Type	Description
<b>PDLResourceAlias</b> <a href="#">Clarified in JDF 1.2</a>	refelement	A reference to a color space description that replaces the color space defined by the colorants described by the <b>SeparationSpec</b> element(s).
<b>SeparationSpec</b> + <a href="#">Modified in JDF 1.2</a>	refelement	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 7.1: Example output for different values of *ProcessColorModel*, *ColorantOrder*, *ColorantControlLink*, *ColorantParams*, 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 DeviceColorantOrder is not present then lay down order will be Cyan first, Magenta last.)
DeviceCMYK	Spot1 Spot2	Cyan Magenta Yellow Black Spot2	—	Spot1	Cyan Magenta Yellow Black Spot2
DeviceCMYK	Spot1 Spot2	Cyan Magenta Yellow Black Spot2	Cyan Magenta	Spot1 Spot2 Yellow Black	Cyan Magenta
DeviceN (with example N=2 colorants as identified in DeviceNSpace)	Spot1 Spot2	Spot2 DeviceN 1 DeviceN 2	—	Spot1	DeviceN 1 DeviceN 2 Spot2 The reordering is accomplished using DeviceColorantOrder.

### 7.2.25 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

$\varphi$  = Value of the *Rotation* attribute

## Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>Surface</b>
<b>Example Partition:</b>	—
<b>Input of processes:</b>	Any printing process.
<b>Output of processes:</b>	—

## Resource Structure

Name	Data Type	Description
<i>Center</i>	XYPair	Position of the center of the color control strip in the coordinates of the <b>MarkObject</b> that contains this mark.
<i>Rotation ?</i>	double	Rotation in degrees. Positive graduation figures indicate counter-clockwise rotation; negative figures indicate clockwise rotation.
<i>Size</i>	XYPair	Size of the color control strip.
<i>StripType ?</i>	NMTOKEN	Type of color control strip. This attribute can be used for specifying a predefined, company-specific color control strip.
<b>CIELABMeasuringField</b> * <a href="#">New in JDF 1.1</a>	refelement	Details of a CIELab measuring field that is part of this <b>ColorControlStrip</b> .
<b>DensityMeasuringField</b> * <a href="#">New in JDF 1.1</a>	refelement	Details of a density measuring field that is part of this <b>ColorControlStrip</b> .

## 7.2.26 ColorCorrectionParams

[Modified in JDF 1.2](#)

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 required). 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 625 attribute description.

**Note:** These color adjustments are not available in any product intent resource, (e.g., **ColorIntent**). In order to request such adjustment in a product intent job ticket supplied to a print provider, attach to a product intent node an incomplete **ColorCorrection** process with a **ColorCorrectionParams** resource specifying the requested “AdjustXXX” attributes.

## Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>DocIndex, RunIndex, RunTags, SheetName, Side, SignatureName</i>
<b>Input of processes:</b>	<b>ColorCorrection</b>
<b>Output of processes:</b>	—

## Resource Structure

Name	Data Type	Description
<i>ColorManagementSystem</i> ? <a href="#">Clarified in JDF 1.2</a>	string	Identifies the preferred ICC color-management system to use when performing color transformations. When specified, this attribute overrides any default selection of a color management system by an application and overrides the “CMM Type” value (bytes 4-7 of an ICC Profile Header) in any of the job related ICC profiles. This string attribute value identifies the manufacturer of the preferred CMM and must match one of the registered four-character ICC CMM Type values. See the <i>ICC Manufacturer's Signature Registry</i> at <a href="http://www.color.org">http://www.color.org</a> . Example values: “ADBE” for the Adobe CMM, and “KODA” for the Kodak CMM.
<b>FileSpec</b> ( <i>FinalTargetDevice</i> )?	refelement	A <b>FileSpec</b> resource pointing to an ICC profile that describes the characterization of the final output target device.
<b>FileSpec</b> ( <i>WorkingColorSpace</i> )? <a href="#">Deprecated in JDF 1.1</a>	refelement	A <b>FileSpec</b> 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.

## Structure of ColorCorrectionOp Subelement

Name	Data Type	Description
<i>SourceObjects</i> = “All”	enumerations	Identifies which class(es) of incoming graphical objects will be operated on. Possible values are: <i>All</i> <i>ImagePhotographic</i> – Contone images. <i>ImageScreenShot</i> – Images largely comprised of rasterized vector art. <i>Text</i> <i>LineArt</i> – Vector objects other than text. <i>SmoothShades</i> – Gradients and blends.
<i>AdjustCyanRed</i> ? <a href="#">New in JDF 1.2</a>	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> ? <a href="#">New in JDF 1.2</a>	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> ? <a href="#">New in JDF 1.2</a>	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> ? <a href="#">New in JDF 1.2</a>	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> ? <a href="#">New in JDF 1.2</a>	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.)



Name	Data Type	Description
<a href="#"><i>AdjustLightness</i> ?</a> <a href="#">New in JDF 1.2</a>	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.)
<a href="#"><i>AdjustSaturation</i> ?</a> <a href="#">New in JDF 1.2</a>	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.)
<b>FileSpec</b> ( <i>AbstractProfile</i> )? <a href="#">New in JDF 1.2</a>	reference	A <b>FileSpec</b> 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.)
<b>FileSpec</b> ( <i>DeviceLinkProfile</i> )? <a href="#">New in JDF 1.2</a>	reference	A <b>FileSpec</b> 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.)

## 7.2.27 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 Publication 15.2 – 1986 “Colorimetry, Second Edition” and ISO 13655:1996 “Graphic technology—Spectral measurement and colorimetric computation for graphic arts images.” The default measurement conditions for spectral measurements are illuminant D50 and 2 degree observer.

Density measurements refer to ISO 5-3:1995 “Photography—Density measurements—Part 3: Spectral conditions” and ISO 5-4:1995 “Photography—Density measurements—Part 4: Geometric conditions for reflection density.” 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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>CIELABMeasuringField, Color, DensityMeasuringField, Media, PrintCondition</b>
<b>Example Partition:</b>	—
<b>Input of processes:</b>	—
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
<i>DensityStandard</i> = "ANSIT"	enumeration	Density filter standard used during density measurements. Possible values are: <i>ANSIA</i> – ANSI Status A <i>ANSIE</i> – ANSI Status E <i>ANSII</i> – ANSI Status I <i>ANSIT</i> – ANSI Status T <i>DIN16536</i> <i>DIN16536NB</i>
<i>Illumination</i> = "D50"	enumeration	Illumination used during spectral measurements. Possible values are: <i>D50</i> <i>D65</i> <i>Unknown</i>

Name	Data Type	Description
<i>InkState</i> ?	enumeration	State of the ink during color measurements. Possible values are: <i>Dry</i> <i>Wet</i> <i>NA</i> <a href="#">Deprecated in JDF 1.2</a>
<i>Instrumentation</i> ?	string	Specific instrumentation used for color measurements, (e.g., manufacturer, model number, and serial number).
<i>MeasurementFilter</i> ?	enumeration	Optical Filter used during color measurements. Possible values are: <i>None</i> – No filter used. <i>Pol</i> – Polarization filter used <i>UV</i> – Ultraviolet cut filter used
<i>Observer</i> = "2"	integer	CIE standard observer function (2 degree and 10 degree) used during spectral measurements. Values are in degree (2 or 10).
<i>SampleBacking</i> ?	enumeration	Backing material used behind the sample during color measurements. Possible values are: <i>Black</i> <i>White</i> <i>NA</i> <a href="#">Deprecated in JDF 1.2</a>
<i>WhiteBase</i> ?	enumeration	Reference for white calibration used for density measurements. Possible values are: <i>Absolute</i> – Means the instrument is calibrated to a device-specific calibration target (absolute white) for absolute density measurements. <i>Paper</i> – Means the instrument is calibrated relative to paper white

### 7.2.28 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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>ColorantControl</b>
<b>Example Partition:</b>	—
<b>Input of processes:</b>	—
<b>Output of processes:</b>	—

#### Resource Structure

Name	Data Type	Description
<b>Color</b> *	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). <b>Note:</b> This value will typically be identical to <b>ColorIntent/@ICCCColorStandard</b> or <b>ColorIntent/@ColorStandard</b> .

## 7.2.29 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

<b>Resource class:</b>	ResourceElement
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>DocIndex, RunIndex, RunTags, SheetName, Side, SignatureName</i>
<b>Input of processes:</b>	<b>ColorSpaceConversion</b>
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
<i>ColorManagementSystem?</i> <a href="#">Clarified in JDF 1.2</a>	string	Identifies the preferred ICC color management system to use when performing color transformations. When specified, this attribute overrides any default selection of a color management system by an application and overrides the “CMM Type” value (bytes 4-7 of an ICC Profile Header) in any of the job related ICC profiles. This string attribute value identifies the manufacturer of the preferred CMM and must match one of the registered four-character ICC CMM Type values. See the <i>ICC Manufacturer's Signature Registry</i> at <a href="http://www.color.org">http://www.color.org</a> . Example values: “ADBE” for the Adobe CMM, and “KODA” for the Kodak CMM.
<i>ConvertDevIndepColors?</i> <a href="#">Deprecated in JDF 1.1</a>	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 <b>ColorSpaceConversionOp</b> with <i>SourceCS</i> = “DevIndep” in JDF 1.1.

Name	Data Type	Description
<p><i>ICCProfileUsage</i> = "UsePDL"  <a href="#">New in JDF 1.2</a></p>	enumeration	<p>This attribute specifies where to obtain the destination profile for the current iteration of the <b>ColorSpaceConversion</b> process, (i.e., either from the PDL (PDF/X job content) or supplied in the JDF <b>ColorSpaceConversionParams</b> resource). <i>ICCProfileUsage</i> provides an order precedence. Possible <i>ICCProfileUsage</i> values are:</p> <p><i>UsePDL</i>:</p> <ol style="list-style-type: none"> <li>1 Use the embedded profile.</li> <li>2 Use the profile specified in the <b>LayoutElement/ElementColorParams/FileSpec</b>(Reference-OutputProfile).</li> <li>3 Use the profile specified in the <b>LayoutElement/ElementColorParams/FileSpec</b>(ActualOutputProfile).</li> <li>4 Use the profile specified in <b>ColorSpaceConversionParams/FileSpec</b>(FinalTargetDevice).</li> <li>5 Use the system specified profile.</li> </ol> <p><i>UseSupplied</i>:</p> <ol style="list-style-type: none"> <li>1 Use the profile specified in the <b>LayoutElement/ElementColorParams/FileSpec</b>(Reference-OutputProfile).</li> <li>2 Use the profile specified in the <b>LayoutElement/ElementColorParams/FileSpec</b>(ActualOutputProfile).</li> <li>3 Use the profile specified in <b>ColorSpaceConversionParams/FileSpec</b>(FinalTargetDevice).</li> <li>4 Use the system specified profile.</li> </ol>
<p><b>FileSpec</b>  <i>(FinalTargetDevice)?</i></p>	refelement	<p>A <b>FileSpec</b> resource pointing to an ICC profile that describes the characterization of the final output target device. This item is required when converting, but optional for tagging or untagging.</p>
<p><b>FileSpec</b>  <i>(WorkingColorSpace)?</i>  <a href="#">Deprecated in JDF 1.1</a></p>	refelement	<p>A <b>FileSpec</b> resource pointing to an ICC profile that describes the assumed characterization of <i>CMYK</i>, <i>RGB</i>, and <i>Gray</i> color spaces.</p>
<p><b>ColorSpaceConversionOp</b> *  <a href="#">Clarified in JDF 1.2</a></p>	refelement	<p>List of <b>ColorSpaceConversionOp</b> elements, each of which identifies a type of object, defines the source color space for that type of object, and specifies the behavior of the conversion operation for that type of object.</p> <p><b>Note:</b> <b>ColorSpaceConversionOp</b> was elevated from a sub-element of <b>ColorSpaceConversionParams</b> in JDF 1.2.</p>

### 7.2.30 ColorSpaceConversionOp

[Clarified in JDF 1.2](#)

**Note:** **ColorSpaceConversionOp** was elevated from a sub-element of **ColorSpaceConversionParams** in JDF 1.2. The **ColorSpaceConversionOp** resource identifies a type of object, defines the source color space for that type of object, and specifies the behavior of the conversion operation for that type of object. Many of these attribute descriptions refer to ICC Color Profiles[ICC.1]. See also the International Color Consortium (ICC) web site at <http://www.color.org>.

#### Resource Properties

Resource class:	ResourceElement
Resource referenced by:	<b>ColorSpaceConversionParams, LayoutElement</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

Name	Data Type	Description
<i>IgnoreEmbeddedICC</i> = "false"	boolean	If "true", specifies that embedded source ICC profiles must be ignored and that the ICC profile [ICC.1] defined by <i>SourceProfile</i> must be used instead.
<i>Operation</i> ? <a href="#">Modified in JDF 1.2</a>	enumeration	Controls which of five functions the color space conversion utility performs. Possible values are: <i>Convert</i> – Transforms graphical elements to final target color space. <i>Tag</i> – Associates appropriate working space profile with uncharacterized graphical element. <i>Untag</i> – Removes all profiles and color characterizations from graphical elements. <i>Retag</i> – Equivalent to a sequence of <i>Untag</i> → <i>Tag</i> . <i>ConvertIgnore</i> – Equivalent to a sequence of <i>Untag</i> → <i>Convert</i> . <i>Operation</i> must be specified in the context of <b>ColorspaceConversionParams</b> and must not be specified in the context of <b>ElementColorParams</b> . <b>Note:</b> The table below describes the effect of this attribute in combination with the <i>SourceCS</i> and <i>IgnoreEmbeddedICC</i> attributes.
<i>PreserveBlack</i> = "false" <a href="#">New in JDF 1.1</a>	boolean	Controls how the tints of black (K in CMYK) should 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 should be converted into other shades of black. The algorithm is implementation-specific.

Name	Data Type	Description
<p><i>RenderingIntent</i> = "ColorSpaceDependent" " <a href="#">Modified in JDF 1.2</a></p>	enumeration	<p>Identifies the rendering intents associated with <i>SourceObjects</i> elements. Possible ICC-defined rendering intent values are:</p> <p><i>Saturation</i> <i>Perceptual</i> <i>RelativeColorimetric</i> <i>AbsoluteColorimetric</i> <i>ColorSpaceDependent</i> – Perceptual rendering intent is used when the source color encoding is an RGB encoding. On the other hand, <i>RelativeColorimetric</i> rendering intent is used when the source and destination color encodings are identical.</p> <p><b>Compatibility Warning.</b> The default has changed in JDF 1.2. f <i>Perceptual</i>.</p>
<p><i>RGBGray2Black</i> = "false" <a href="#">Modified in JDF 1.2</a></p>	boolean	<p>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 <b>ColorSpaceConversion</b> process in determining how to allocate RGB values to the black (K) channel. After the <b>ColorSpaceConversion</b> process is completed, the <b>Rendering</b> process uses <b>AutomatedOverprintParams</b> to determine overprint behavior for the previously determined black (K) channel.</p>
<p><i>RGBGray2BlackThreshold</i> = "1" <a href="#">New in JDF 1.2</a></p>	double	<p>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".</p>

Name	Data Type	Description
<p><i>SourceCS</i>  <a href="#">Modified in JDF 1.2</a></p>	enumeration	<p>Identifies which of the incoming color spaces will be operated on. Possible values are:</p> <p><i>Calibrated</i> – Operates on <i>CalGray</i> and <i>CalRGB</i> color spaces. <a href="#">New in JDF 1.2</a></p> <p><i>CIEBased</i> – Operates on CIE-based color spaces (<i>CIEBasedA</i>, <i>CIEBasedABC</i>, <i>CIEBasedDEF</i>, and <i>CIEBasedDEFG</i>). <a href="#">New in JDF 1.2</a></p> <p><i>CMYK</i> – Operates on deviceCMYK.</p> <p><i>DeviceN</i> – Identifies the source color encoding as a <i>DeviceN</i> color space. The specific <i>DeviceN</i> color space to operate on is defined in the <b>DeviceNSpace</b> resource. If <i>DeviceN</i> is specified, then the <b>DeviceNSpace</b> and <b>ColorantControl/ColorPool</b> relements must also be present. <a href="#">New in JDF 1.2</a></p> <p><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>). <a href="#">Clarified in JDF 1.2</a></p> <p><i>Gray</i> – Operates on <i>deviceGray</i>.</p> <p><i>ICCBased</i> – Operates on color spaces defined using ICC profiles. <i>ICCBased</i> 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 should be treated as being encoded in the Alternate or underlying color space, and a <b>ColorSpaceConversionOp</b> where <i>SourceCS</i> = “DevIndep” will not be applied, unless that color space is also device independent. <a href="#">New in JDF 1.2</a></p> <p><i>Lab</i> – Operates on <i>Lab</i>. <a href="#">New in JDF 1.2</a></p> <p><i>RGB</i> – Operates on <i>deviceRGB</i>. <a href="#">Modified in JDF 1.2</a></p> <p><i>Separation</i> – Operates on <i>Separation</i> color spaces (spot colors). The specific separation(s) to operate on are defined in the <b>SeparationSpec</b> element(s). If no <b>SeparationSpec</b> is defined, the operation will operate on all the separation color spaces in the input <b>RunList</b>. <a href="#">New in JDF 1.2</a></p> <p><i>YUV</i> – Operates on <i>YUV</i> (Also known as YCbCr). See [CCIR601-2] <a href="#">New in JDF 1.2</a></p> <p><b>Note:</b> 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.</p> <p><b>Note:</b> See table below for a description on how the <i>SourceCS</i> values map into the most relevant file.</p>
<p><i>SourceObjects</i> = “All”</p>	enumerations	<p>List of object classes that identifies which incoming graphical objects will be operated on. Possible values are:</p> <p><i>All</i></p> <p><i>ImagePhotographic</i> – Contone images.</p> <p><i>ImageScreenShot</i> – Images largely comprised of rasterized vector art.</p> <p><i>Text</i></p> <p><i>LineArt</i> – Vector objects other than text.</p> <p><i>SmoothShades</i> – Gradients and blends.</p>

Name	Data Type	Description
<a href="#"><i>SourceRenderingIntent</i> ?</a> <a href="#">New in JDF 1.2</a>	enumeration	Identifies the rendering intent transform elements to be selected from the source profile that will be used to interpret objects of type identified by the <i>SourceObjects</i> and <i>SourceCS</i> attributes. Possible ICC-defined [ICC.1] rendering intent values are: <i>Saturation</i> <i>Perceptual</i> <i>RelativeColorimetric</i> <i>AbsoluteColorimetric</i> <i>ColorSpaceDependent</i> – Perceptual rendering intent is used when the source color encoding is an RGB encoding. Relative colorimetric rendering intent is used when the source and destination color encodings are identical. If not specified, <i>SourceRenderingIntent</i> inherits the value of <i>RenderingIntent</i> . <b>Note:</b> The <i>SourceRenderingIntent</i> will pertain to the source profile used in a particular <b>ColorSpaceConversion</b> process, (e.g., sources may be the native original color space, an intermediate working color space, or an reference output simulation color space).
<a href="#"><b>DeviceNSpace</b> ?</a> <a href="#">New in JDF 1.2</a>	refelement	<b>DeviceNSpace</b> 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 <b>DeviceNSpace</b> are provided in the <b>ColorantControl/ColorPool</b> resource, which must also be present
<b>FileSpec</b> ( <i>AbstractProfile</i> )? <a href="#">New in JDF 1.2</a>	refelement	A <b>FileSpec</b> resource pointing to an ICC profile [ICC.1] that describes the characterization of an Abstract Profile for specifying a preference adjustment.
<b>FileSpec</b> ? ( <i>SourceProfile</i> ) <a href="#">Clarified in JDF 1.2</a>	refelement	A <b>FileSpec</b> resource pointing to an ICC profile [ICC.1] that describes the assumed source color space. See <i>IgnoreEmbeddedICC</i> for policies on using external profiles.
<b>SeparationSpec</b> * <a href="#">New in JDF 1.2</a>	refelement	<b>SeparationSpec</b> resource(s) defining on which separation(s) to operate when <i>SourceCS</i> = " <i>Separation</i> ".

**Notes:**

*DevIndep* has been retained for backwards compatibility with JDF 1.1 and because there will probably be cases where the same processing *should* be applied to all device independent spaces. An equivalent “DevDep” has not been added because it's less likely that all device-dependent spaces should be treated in the same way. The following table summarizes how the *SourceCS* attribute is mapped to/from different file formats.

Table 7.2: Mapping of *SourceCS* enumeration values to color spaces in the most common input file formats.

SourceCS	File Format	Color Space
<i>RGB</i>	PDF (2)	DeviceRGB (1)
	PostScript	DeviceRGB
	TIFF	PhotometricInterp = 2
<i>CMYK</i>	PDF (2)	DeviceCMYK (1)
	PostScript (2)	DeviceCMYK
	TIFF	PhotometricInterp = 5 Samples per pixel = 4



Table 7.2: Mapping of SourceCS enumeration values to color spaces in the most common input file formats.

SourceCS	File Format	Color Space
<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 v4 encoding).
<i>ICCBased</i>	PDF (2)	ICCBased
	PostScript (2)	n/a
	PostScript/EPS	The EPS file has an embedded ICC profile.
	TIFF	The TIFF file has an embedded ICC profile.
<i>Separation</i>	PDF (2)	Separation
	PostScript (2)	Separation
	TIFF	PhotometricInterp = 5 (Applies only to one of the planes in the separated image.)
<i>DeviceN</i>	PDF (2)	DeviceN
	PostScript (2)	DeviceN
	TIFF	PhotometricInterp = 5, Samples per pixel = 4

(1) DeviceCMYK, DeviceRGB, and DeviceGray in PDF files should be mapped through DefaultCMYK, DefaultRGB, or DefaultGray color spaces, if present, before determining whether this operation should be applied.

- (2) Where a *Pattern* or *Indexed* color space has been used, the base color space is used to determine if this operation should be applied.

Table 7.3: Effect of color space conversion operations on color spaces.

Source CS	Operation	Ignore Embedded ICC	FileSpec (Source-Profile)	Description
CMYK	Tag	false	CMYK ICC profile	Changes the CMYK color spaces (i.e., those without ICC profiles) in the <b>RunList</b> to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile.
		true	CMYK ICC profile	Changes the CMYK color spaces and all <i>ICCBased</i> color spaces with four components (CMYK) in the <b>RunList</b> to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile.
	Untag	n/a	n/a	n/a
	Convert	false	CMYK ICC profile	Converts the objects and/or images in CMYK color spaces (i.e., those without ICC profiles) using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile
		true	CMYK ICC profile	Converts the objects and/or images in CMYK color spaces and in four components (CMYK) <i>ICCBased</i> color spaces, using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.
RGB	Tag	false	RGB ICC profile	Changes the RGB color spaces (i.e., those without ICC profiles) in the <b>RunList</b> to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile.
		true	RGB ICC profile	Changes the RGB color spaces and all <i>ICCBased</i> color spaces with three components (RGB) in the <b>RunList</b> to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile.
	Untag	n/a	n/a	n/a
	Convert	false	RGB ICC profile	Converts the objects and/or images in RGB color spaces (i.e., those without ICC profiles) using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.
		true	RGB ICC profile	Converts the objects and/or images in RGB color spaces and in three components (RGB) <i>ICCBased</i> color spaces, using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.

Table 7.3: Effect of color space conversion operations on color spaces.

Source CS	Operation	Ignore Embedded ICC	FileSpec (Source-Profile)	Description
Gray	Tag	false	Mono-chrome ICC profile	Changes the <i>Gray</i> color spaces (i.e., those without ICC profiles) in the <b>RunList</b> to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile
		true	Mono-chrome ICC profile	Changes the <i>Gray</i> color spaces and all <i>ICCBased</i> color spaces with one component (Gray) in the <b>RunList</b> to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile.
	Untag	n/a	n/a	n/a
	Convert	false	Mono-chrome ICC profile	Converts the objects and/or images in Gray color spaces (i.e., those without ICC profiles) using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.
		true	Mono-chrome ICC profile	Converts the objects and/or images in <i>Gray</i> color spaces and in one component (Gray) <i>ICCBased</i> color spaces, using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.
YUV, Lab	Tag	n/a	Lab or YUV ICC profile	Changes the <i>YUV</i> or <i>Lab</i> color spaces in the <b>RunList</b> to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile. If <i>SourceProfile</i> is a <i>YUV</i> profile only <i>YUV</i> color spaces are affected; if <i>SourceProfile</i> is an <i>Lab</i> profile only <i>Lab</i> color spaces are affected.
	Untag	n/a	n/a	This operation does not have any effect.
	Convert	n/a	n/a	Converts the objects and/or images in the specified color spaces using the source definition embedded in the file and the <i>FinalTargetDevice</i> ICC profile as output profile.
Calibrated	Tag	n/a	RGB or Mono-chrome ICC profile	Changes the <i>Calibrated</i> color spaces in the <b>RunList</b> to an <i>ICCBased</i> color space using the <i>SourceProfile</i> ICC profile. If <i>SourceProfile</i> is an <i>RGB</i> profile only <i>CalRGB</i> color spaces are affected; if <i>SourceProfile</i> is a monochrome profile only <i>CalGray</i> color spaces are affected.
	Untag	n/a	n/a	Changes <i>CalRGB</i> color spaces to RGB color space and <i>CalGray</i> color spaces to <i>Gray</i> color space.
	Convert	n/a	n/a	The corresponding objects in the specified color space(s) are converted using the source definition embedded in the file and the <i>FinalTargetDevice</i> ICC profile as output profile.
CIE-Based	Tag	n/a	n/a	This operation does not have any effect.
	Untag	n/a	n/a	This operation does not have any effect.
	Convert	n/a	n/a	The corresponding objects in the specified color space(s) are converted using the source definition embedded in the file and the <i>FinalTargetDevice</i> ICC profile as output profile.

Table 7.3: Effect of color space conversion operations on color spaces.

Source CS	Operation	Ignore Embedded ICC	FileSpec (Source-Profile)	Description
ICC-Based	Tag	n/a	n/a	n/a <b>Note:</b> In order to change the profile associated to an <i>ICCBased</i> , an <i>Untag</i> operation (see below) should be performed before tagging. These two operations can be combined in a <i>Retag</i> operation.
	Untag	n/a	n/a	The ICC profiles in the input <b>RunList</b> are removed. The resulting color spaces depend on the input file format: <ul style="list-style-type: none"> <li>• PDF – Use the corresponding alternate color space.</li> <li>• EPS – Use the PostScript file color spaces; the ICC profile comment in the EPS header is removed</li> <li>• TIFF – Use the color space defined by the photometric interpretation tag.</li> </ul>
	Convert	false	n/a	The <i>ICCBased</i> color spaces are converted using the corresponding embedded ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.
		true	n/a	This operation does not have any effect (To ignore embedded ICC profiles when converting, the <i>CMYK</i> , <i>RGB</i> or <i>Gray</i> SourceCS enumeration values must be used with the <i>IgnoreEmbeddedICC</i> flag set to "true". Each <i>SourceCS</i> value will require a different <i>ColorSpaceConversionOp</i> instance, with the corresponding ICC profile.)
Dev-Indep	Tag	n/a	n/a	This operation does not have any effect. The specific <i>SourceCS</i> enumeration values have to be used to select the color spaces to tag.
	Untag	n/a	n/a	Untags <i>ICCBased</i> and <i>Calibrated</i> color spaces in the <b>RunList</b> . It does not have any effect on the other device independent color space.
	Convert	false	n/a	Converts all the device independent color spaces ( <i>CIEBased</i> , <i>Lab</i> , <i>YUV</i> , <i>Calibrated</i> and <i>ICCBased</i> ) using the corresponding characterizations embedded in the file and the <i>FinalTargetDevice</i> ICC profile as output profile.
		true	n/a	This operation does not have any effect. The specific <i>SourceCS</i> enumeration values have to be used to select the color spaces to convert.

Table 7.3: Effect of color space conversion operations on color spaces.

Source CS	Operation	Ignore Embedded ICC	FileSpec (Source-Profile)	Description
Separation	Tag	n/a	Named color ICC profile	In PostScript or PDF, it sets the alternate color space to an <i>ICCBased</i> color space with the given ICC profile.
			No profile specified	In PostScript or PDF, it sets the alternate color space to the color definition in the <b>ColorPool</b> , if present. If there is no color definition in the <b>ColorPool</b> , this operation does not have any effect.
	Untag	n/a	n/a	This operation does not have any effect.
	Convert	false	n/a	The specified separation(s) are converted using the alternate color space definitions in the <b>RunList</b> .
		true	Named color ICC profile	Converts the specified separation(s) using the <i>SourceProfile</i> profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile.
	No profile specified	No profile specified	Converts the specified separation(s) using the color definition in the <b>ColorPool</b> and the <i>FinalTargetDevice</i> ICC profile if needed	
DeviceN	Tag	false	N component ICC profile	Changes the <i>DeviceN</i> color spaces in the <b>RunList</b> to <i>ICCBased</i> color spaces using the <i>SourceProfile</i> ICC profile. This operation only affects the selected <i>DeviceN</i> color spaces that have exactly the same number of components than the <i>SourceProfile</i> .
	Untag	n/a	n/a	This operation does not have any effect.
	Convert	false	n/a	In PostScript or PDF, the specified <i>DeviceN</i> color spaces are converted using the alternate color space.
		true	N component ICC profile	Converts the specified <i>DeviceN</i> color spaces using the <i>SourceProfile</i> ICC profile as input profile and the <i>FinalTargetDevice</i> ICC profile as output profile. This operation only affects the selected <i>DeviceN</i> color spaces that have exactly the same number of components than the <i>SourceProfile</i> .

**Note:** If the correct ICC profile is not specified for an operation that requires it, the operation does not have any effect.

### 7.2.31 ComChannel

A communication channel to a person or company such as an E-mail address, phone number, or fax number.

#### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>Contact, Person, CustomerMessage *</b>
<b>Example Partition:</b>	—
<b>Input of processes:</b>	—
<b>Output of processes:</b>	—

## Resource Structure

Name	Data Type	Description
<a href="#">ChannelType</a> <a href="#">Modified in JDF 1.2</a>	enumeration	Type of the communication channel. Possible values are: <i>Phone</i> – Telephone number. <i>Email</i> – E-mail address. <i>Fax</i> – Fax machine. <i>WWW</i> – WWW home page or form. <i>JMF</i> – JMF messaging channel. <i>PrivateDirectory</i> – Account of a registered customer of a certain service. (The list of the registered accounts is maintained by the service vendor). The private directory service vendor name should appear in <i>ChannelTypeDetails</i> attribute. <i>InstantMessaging</i> – IM service address. The IM service vendor name should appear in <i>ChannelTypeDetails</i> attribute.
<a href="#">ChannelTypeDetails?</a> <a href="#">New in JDF 1.2</a>	NMTOKEN	Description of the value of the <i>ChannelType</i> attribute. See Table 7-1 on page 322 for examples. Should be the service vendor name in case <i>ChannelType</i> is <i>PrivateDirectory</i> or <i>InstantMessaging</i> .
<a href="#">ChannelUsage?</a> <a href="#">New in JDF 1.2</a>	NMTOKENS	Communication channel usage. Possible values include: <i>Business</i> – Business purpose usage, (e.g. office phone number, fax). <i>Private</i> – Private purpose usage, (e.g., private phone number, fax, E-mail). <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.
<a href="#">Locator</a> <a href="#">Modified in JDF 1.2</a>	string	Locator of this type of channel in a form such as a phone number, a URL, or an E-mail address. In case a URL is defined for the <i>ChannelType</i> in [RFC2396] or [RFC2806] it is recommended to use the URL syntax, as follows: "mailto:a@b.com" instead of "a@b.com" if <i>ChannelType</i> = "Email", "tel:+49-69-92058800" if <i>ChannelType</i> = "Phone", and "fax:+49.6151.155.299" if <i>ChannelType</i> = "Fax".

Table 7-1: ChannelTypeDetails predefined values for different ChannelType values

ChannelType	ChannelTypeDetails predefined value	Description
<i>Phone</i>	<i>LandLine</i>	Land line telephone number.
	<i>Mobile</i>	Mobile/Cellular telephone number.
	<i>Secure</i>	Secure phone line.
	<i>ISDN</i>	ISDN line telephone number.
<i>WWW</i>	<i>Form</i>	Upload form.
	<i>Target</i>	Upload target URL.

## Examples

```
<ComChannel ChannelType="Phone" ChannelTypeDetails="Mobile" ChannelUsage="Business"
Locator="44 07808 907 919"/>
```

```
<ComChannel ChannelType="InstantMessaging" ChannelTypeDetails="MyIMService"
ChannelUsage="Private" Locator="123456789"/>
```

### 7.2.32 Company

Specifies contacts to a company including detailed information about contact persons and addresses. This structure can be used in many situations where addresses or contact persons are needed. Examples of contacts are customer, supplier, company, and addressees. The structure is derived from the vCard format. It comprises the organization name and organizational units (ORG) of the organizational properties defined in the vCard format. The corresponding XML types of the vCard are quoted in the table.

#### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	Contact
<b>Example Partition:</b>	—
<b>Input of processes:</b>	—
<b>Output of processes:</b>	—

#### Resource Structure

Name	Data Type	Description
<i>OrganizationName</i>	string	Name of the organization or company (vCard: ORG:orgnam. For example: ABC, Inc.).
<b>Contact *</b> <a href="#">Deprecated in JDF 1.1</a>	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").

### 7.2.33 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 must then be joined and trimmed. Nearly every postpress process has a **Component** resource as an input as well as an output. Typically the first components in the process chain are some printed sheets or ribbons, while the last component is a book or a brochure. **Component** resources are grouped by kind in much the same way that nodes are classified as Combined, Process, or Product. The five categories of **Component** resources are: *Ribbon*, *Sheet*, *Block*, *PartialProduct*, and *FinalProduct*. These categories are defined in greater detail below:

*Ribbon* — Part of the web that enters the folder, divider etc. In case the web is not slit, the web and the ribbon are identical.

*Sheet* — This source type is appropriate if a flat sheet, (e.g., a postcard to be glued in is used as an input component). "Flat" in this case means that the sheet has not been folded or cut before the operation.

*Block* — This source type is appropriate if a folded sheet, a cut portion of the sheet, or a cut and folded portion of a sheet is used as an input component.

*PartialProduct* — This source type is appropriate if a partial product should be used as an input component.

*FinalProduct* — This source type is appropriate if this **Component** is the final product.

#### Terms and Definitions for Components

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 may have a different meaning in other cultures, (i.e., those writing from right to left).

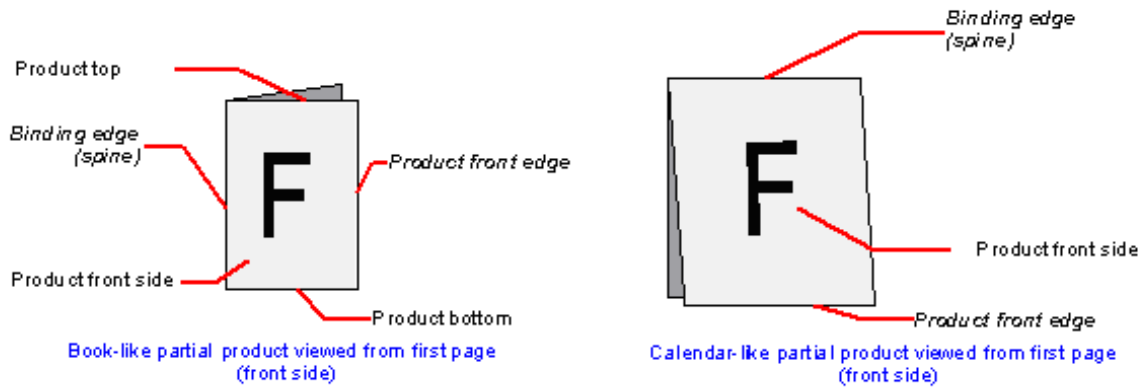


Figure 7.5: Terms and definitions for components

The table below describes the terms used to define the components.

Table 7-2: Terms and definitions for components

Edge	Description
Binding edge	The edge on which the (partial) product is glued or stitched. This edge is also often called <i>working edge</i> or <i>spine</i> .
Product front edge	The side, where you open the (partial) product. This edge is opposite to the binding edge.
Registered edge	A side on which a collection of sheets or partial products is aligned during a production step. All production steps require two registered edges, which must not be opposite to each other. The two registered edges define the coordinate system used within the production step. When there is a binding edge, this is one of the registered edges.

### Resource Properties

<b>Resource class:</b>	Quantity
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>Condition, RibbonName, SheetName, SignatureName, WebName</i>
<b>Input of processes:</b>	Many
<b>Output of processes:</b>	Many



## Resource Structure

Name	Data Type	Description
<a href="#">ComponentType</a> <a href="#">Modified in JDF 1.2</a>	enumerations	Specifies the category of the component. Possible values are: <i>Ribbon</i> – The <b>Component</b> is a ribbon on a web press. <i>Sheet</i> – Single layer (sheet) of paper. <i>Block</i> – Folded or stacked product, (e.g., book block). <i>Proof</i> – The <b>Component</b> is a proof. <i>Web</i> – The <b>Component</b> is a web on a web press. Further details of the component are specified in <i>ProductType</i> . Only one of <i>FinalProduct</i> or <i>PartialProduct</i> may be specified in addition to one of the five enumerations specified above. <i>FinalProduct</i> – The Component is the final product that was ordered by the customer. <i>PartialProduct</i> – The Component has been partially processed.
<a href="#">Dimensions ?</a>	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 not be equal to the dimensions of the sheet before it was folded. The dimension is always the bounding box around the <b>Component</b> . If not specified, a portrait orientation (Y>X) is assumed <b>Note:</b> It is crucial for enabling postpress to specify <i>Dimensions</i> unless they really are unknown.
<a href="#">IsWaste = "false"</a>	boolean	If "true", the <b>Component</b> is waste from a previous process that may be used to set up a machine.
<a href="#">MaxHeat ?</a>	double	Maximum temperature the <b>Component</b> 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).
<a href="#">Overfold ?</a> <a href="#">New in JDF 1.1</a>	double	Expansion of the overfold of a <b>Component</b> . This attribute may be needed for the <b>Inserting</b> or other postpress processes.
<a href="#">OverfoldSide ?</a> <a href="#">New in JDF 1.1</a>	enumeration	Specifies the longer side of a folded component. Values are: <i>Front</i> <i>Back</i>
<a href="#">ProductType ?</a> <a href="#">Modified in JDF 1.2</a>	NMTOKEN	Type of product that this component specifies. Possible values include: <i>BackCover</i> <i>Body</i> – Generic content inside of a cover. <a href="#">New in JDF 1.2</a> <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>FrontCover</i> <i>Insert</i> <a href="#">New in JDF 1.2</a> <i>Jacket</i> – Hard cover case jacket. <i>Label</i> <i>Poster</i> <i>Unknown</i> – <a href="#">Deprecated in JDF 1.2</a>
<a href="#">ReaderPageCount ?</a> <a href="#">New in JDF 1.1</a>	integer	Total amount of individual reader pages that this <b>Component</b> contains. Count of -1 means "unknown." If not specified, the value is unknown.

Name	Data Type	Description
<i>SheetPart</i> ?	rectangle	Only useful when <i>ComponentType</i> = <i>Block</i> and when <i>SourceSheet</i> is present. Position of the block on the <i>Sheet</i> in <i>SurfaceContentsBox</i> coordinates used in this <b>Component</b> .
<i>SourceRibbon</i> ?	string	Only required when <i>ComponentType</i> = <i>Ribbon</i> . <i>RibbonName</i> of the ribbon used in this <b>Component</b> .
<i>SourceSheet</i> ?	string	Only required when <i>ComponentType</i> = <i>Sheet</i> or <i>Block</i> . Matches the <b>Layout/Signature/Sheet/@Name</b> used in this <b>Component</b> .
<i>SourceWeb</i> ?	string	Only required when <i>ComponentType</i> = <i>Ribbon</i> . <i>WebName</i> of the ribbon used in this <b>Component</b> .
<i>SurfaceCount</i> ? <a href="#">New in JDF 1.1</a>	integer	Total amount of individual surfaces that this <b>Component</b> contains.
<i>Transformation</i> ? <a href="#">Deprecated in JDF 1.1</a>	matrix	Matrix describing the transformation of the orientation of a <b>Component</b> for the process using this resource as input. This is needed to convert the coordinate system of the <b>Component</b> to the coordinate system of the process. When this attribute is not present, the identity matrix (1 0 0 1 0 0) is assumed. In version 1.1 and beyond, use <i>ResourceLink/@Transformation</i> or <i>ResourceLink/@Orientation</i> .
<b>Bundle</b> ? <a href="#">New in JDF 1.1</a>	refelement	Description of a <b>Bundle</b> of <b>Components</b> if the <b>Component</b> represents multiple individual items. If no <b>Bundle</b> is present, the <b>Component</b> represents an individual item. Note that it is essential to keep a reference of the child <b>Components</b> that comprise a <b>Component</b> , as this information is useful to postpress processes.
<b>Disjointing</b> ?	refelement	A stack of components can be processed using physical separators. This is useful in operations such as feeding.
<b>Sheet</b> ? <a href="#">Deprecated in JDF 1.2</a>	refelement	The <b>Sheet</b> resource that describes the details of this <b>Component</b> if <i>ComponentType</i> = <i>Sheet</i> or <i>Block</i> . Replaced with <b>Layout</b> in JDF 1.2 and beyond. The <b>Sheet</b> in the referenced <b>Layout</b> is accessed by matching <i>SourceSheet</i> with <b>Layout/Signature/Sheet/@Name</b>
<b>Layout</b> ? <a href="#">New in JDF 1.2</a>	refelement	Specifies the original <b>Layout</b> of the source sheet of the <b>Component</b> when <i>ComponentType</i> = "Sheet" or "Block". The original <b>Sheet</b> is the <b>Sheet</b> element where <i>SourceSheet</i> matches the <b>Layout/Signature/Sheet/@Name</b> used in this <b>Component</b>

### Example Use of Condition Partition Key Attribute

[New in JDF 1.2](#)

The *Condition* partition key describes whether a **Component** partition is waste and what type of waste it is. The amount of each *Condition*, in the standard manner for partitionable resources. See Section 3.8.2, Description of Partitionable Resources.)

```

<ResourcePool>
  <Component Class="Quantity" ID="Res6" PartIDKeys="Condition" PartUsage="Implicit"
Status="Available">
    <Component Condition="Good" IsWaste="false"/>
    <Component Condition="Waste" IsWaste="true"/>
  </Component>
</ResourcePool>
<ResourceLinkPool>
  <ComponentLink Amount="3057" Usage="Output" rRef="Res6">
    <AmountPool>
      <PartAmount Amount="2970">
        <Part Condition="Good"/>
      </PartAmount>
    </AmountPool>
  </ComponentLink>
</ResourceLinkPool>

```

```

</PartAmount>
<PartAmount Amount="87">
  <Part Condition="Waste"/>
</PartAmount>
</AmountPool>
</ComponentLink>
</ResourceLinkPool>

```

### 7.2.34 Contact

Element describing a contact to a person or address.

#### Resource Properties

**Resource class:** Parameter

**Resource referenced by:** **ApprovalParams, ArtDeliveryIntent, CustomerInfo, DeliveryIntent, DeliveryParams, DropIntent**

**Example Partition:** —

**Input of processes:** —

**Output of processes:** —

#### Resource Structure

Name	Data Type	Description
<b>ContactTypes</b> <a href="#">Modified in JDF 1.2</a>	NMTOKENS	Classification of the contact. Possible values include: <i>Accounting</i> – Address of where to send to the bill. <i>Administrator</i> – Person to contact for queries concerning the execution of the job. <i>Approver</i> – Person who approves this job. <a href="#">New in JDF 1.2</a> <i>ArtReturn</i> – Return delivery or pickup address for artwork of this job. <i>Billing</i> – Contact information that refers to a payment method, (e.g., credit card). <i>Customer</i> – The end customer. <i>Delivery</i> – Delivery address for all products of this job. <i>DeliveryCharge</i> – The <b>Contact</b> is charged for delivery of this job. <i>Owner</i> – The owner of a resource. <i>Pickup</i> – The pickup address for all products of this job. <i>Sender</i> – The source address of the delivery. <a href="#">New in JDF 1.2</a> <i>Supplier</i> – Address of a supplier of needed goods. <i>SurplusReturn</i> – Return delivery or pickup address for surplus products of this job.
<b>ContactTypeDetails ?</b>	string	Details of the Contact's role or roles. For instance if <i>ContactTypes</i> includes " <i>Delivery</i> " this could be a description which delivery location this <b>Contact</b> is responsible for.
<b>Address ?</b>	refelement	Element describing the address.
<b>ComChannel *</b> <a href="#">Modified in JDF 1.2</a>	refelement	Communication channels to the contact. These elements define communication channels that may be assigned to multiple <b>Persons</b> , for instance the communication channel of a reception area.
<b>Company ?</b> <a href="#">New in JDF 1.1</a>	refelement	Company that this <b>Contact</b> is associated with.
<b>Person ?</b>	refelement	Name of the contact person.

### 7.2.35 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:	<i>ContactCopying</i>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>ContactScreen</i> = "false"	boolean	<i>ContactScreen</i> = "true", if a halftone screen on film should be used to produce halftones.
<i>Cycle</i> ?	integer	Number of exposure light units to be used. The amount depends on the subject to be exposed.
<i>Diffusion</i> ?	enumeration	The diffusion foil setting. Possible values are: <i>On</i> <i>Off</i>
<i>PolarityChange</i> = "true"	boolean	<i>PolarityChange</i> = "true", if the copy should change polarity w.r.t. the original image.
<i>RepeatStep</i> = "1 1"	XYPair	Number (as integers) of copies in each direction for a Step/Repeat camera.
<i>Vacuum</i> ?	double	Amount of vacuum pressure to be used, measured in bars.
<b>ScreeningParams</b> ?	refelement	Properties of the halftone screen on film. Ignored if <i>ContactScreen</i> = "false".

### 7.2.36 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 3.8.2, Description of Partitionable Resources.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>BlockName, FountainNumber, RibbonName, Separation, SheetName, Side, SignatureName, WebName, PartVersion.</i>
Input of processes:	<i>ConventionalPrinting</i>
Output of processes:	—

## Resource Structure

Name	Data Type	Description
<i>DirectProof</i> = "false"	boolean	If "true", the proof is directly produced and subsequently an approval may 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 not required for setup, but it is required for the actual print run. If the <b>ConventionalPrinting</b> process is waiting for a <i>DirectProof</i> , the <i>Status</i> of the JDF node is switched to <i>Stopped</i> with the <i>StatusDetails</i> = "WaitForApproval".
<i>Drying</i> ?	enumeration	The way in which ink is dried after a print run. Possible values are: <i>UV</i> – Ultraviolet dryer <i>Heatset</i> – Heatset dryer <i>IR</i> – Infrared dryer <i>On</i> – Use the device default drying unit. <i>Off</i>
<i>FirstSurface</i> ? <a href="#">Modified in JDF 1.2</a>	enumeration	Printing order of the surfaces on the sheet. Possible values are: <i>Either</i> - <a href="#">Deprecated in JDF 1.2</a> Omit <i>FirstSurface</i> to specify either. <i>Front</i> <i>Back</i>
<i>FountainSolution</i> ?	enumeration	State of the fountain solution module in the printing units. Possible values are: <i>On</i> <i>Off</i>
<i>MediaLocation</i> ?	string	Identifies the location of the <b>Media</b> . 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 3.8.2.6, Locations of Physical Resources) there should be a match between one <i>Location</i> partition key and this <i>MediaLocation</i> value.
<i>ModuleAvailableIndex</i> ? <a href="#">New in JDF 1.1</a>	IntegerRange-List	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> .
<i>ModuleDrying</i> ?	enumeration	The way in which ink is dried in individual modules. Possible values are: <i>UV</i> – Ultraviolet dryer <i>Heatset</i> – Heatset dryer <i>IR</i> – Infrared dryer <i>On</i> – Use the device default drying unit. <i>Off</i>
<i>ModuleIndex</i> ?	IntegerRange-List	Zero-based, ordered list of print modules that should 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> .
<i>PerfectingModule</i> ? <a href="#">New in JDF 1.1</a>	integer	Index of the perfecting module if <i>WorkStyle</i> = <i>Perfecting</i> and multiple perfecting modules are installed.

Name	Data Type	Description
<i>Powder?</i>	double	Quantity of powder in %.
<i>PrintingType</i> <a href="#">Modified in JDF 1.2</a>	enumeration	Type of printing machine. Possible values are: <i>ContinuousFed</i> – connected sheets including fan fold. <a href="#">New in JDF 1.2</a> <i>SheetFed</i> – separate cut sheets. <i>WebFed</i> – paper supplied to press on rolls.
<i>SheetLay?</i>	enumeration	Lay of input media. Reference edge of where paper is placed in feeder. Possible values are: <i>Left</i> <i>Right</i> <i>Center</i>
<i>Speed?</i>	double	Maximum print speed in sheets/hour (sheet fed) or meters/hour (web fed). Defaults to device specific full speed.
<i>WorkStyle?</i>	enumeration	The direction in which to turn the press sheet. Possible values are: <i>Simplex</i> – No turning <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>Perfecting</i> – Many sheetfed printing presses have perfecting cylinder(s) built in. The leading edge of the print sheet changes as the sheet is turned by the perfecting cylinder, but the side lays remain unaltered. In this regard, this <i>WorkStyle</i> is similar to <i>WorkAndTumble</i> , but <i>Perfecting</i> is an in-line operation during the press run. Therefore, an additional plate (set) is required during this press run. <i>WorkAndTurn</i> – Refers to the turning of the first-run sheet for subsequent perfecting. The front lays remain unchanged but the side lays must be altered. The alteration can be made by hand or using a pile turner. Turning happens after the first press run and the plate (set) is used again in the second press run, imaging the other sheet surface. <i>WorkAndTumble</i> – The <i>WorkAndTumble</i> method is also used for perfecting. The leading edge of the print sheet changes as the sheet is turned, but the side lays remain unaltered. Tumbling happens after the first press run and the plate (set) is used again in the second press run, imaging the other sheet surface. <i>WorkAndTwist</i> – Done between two press runs. The pallet is twisted 180 degrees before the second run is performed so that the front lay and the side lay both change. The surface to be imaged is the same at both runs. Each run prints only part of the surface. The plate (set) stays in the machine. This <i>WorkStyle</i> is used for saving plate or film material. It is no longer a common <i>WorkStyle</i> .

Name	Data Type	Description
<b>ApprovalParams</b> ? <a href="#">New in JDF 1.2</a>	refelement	Details of the direct approval process, when <i>DirectProof</i> = "true".
<b>Ink</b> * <a href="#">Modified in JDF 1.2</a>	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 <b>Ink/@Family</b> = "Varnish". If both <b>Ink</b> and <b>ExposedMedia (Plate)</b> are specified for a given separation, spot varnishing is specified. If only <b>Ink</b> and not <b>ExposedMedia (Plate)</b> is specified, overall varnishing is specified. In JDF 1.2 and beyond, <b>Ink</b> may occur in multiples in order to specify multiple layers of varnish. <b>Note:</b> The color inks are direct input resources of the process and must not be specified here.

### 7.2.37 CostCenter

This resource describes an individual area of a company that has separated accounting.

#### Resource Properties

Resource class:	ResourceElement
Resource referenced by:	<b>Device, Employee</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>CostCenterID</i>	string	Identification of the cost center
<i>Name</i> ?	string	Name of the cost center.

### 7.2.38 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:	<b>CoverApplication</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>CoverOffset</i> ? <a href="#">Deprecated in JDF 1.2</a>	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 in <b>ResourceLink/@Transformation</b> of the <b>ComponentLink</b> of the cover.
<b>GlueApplication</b> *	refelement	Describes where and how to apply glue to the book block.
<b>Score</b> *	element	Describes where and how to score the cover.

### Structure of Score Subelement

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. Possible values are: <i>FromInside</i> <i>FromOutside</i>

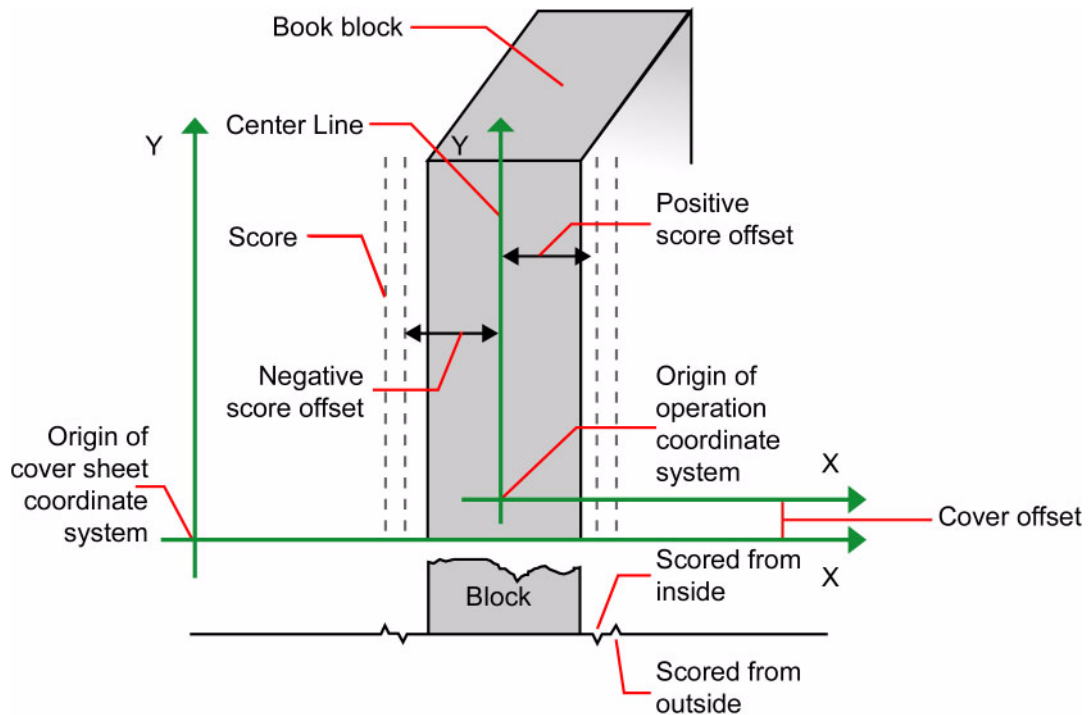


Figure 7.6: Parameters and coordinate system for cover application

### 7.2.39 CreasingParams

[New in JDF 1.1](#)

**CreasingParams** define the parameters for creasing or grooving a sheet.

#### Resource Properties

Resource class: Parameter

Resource referenced by: —

Example Partition: *BlockName, RibbonName, SheetName, SignatureName, WebName*

Input of processes: **Creasing**

Output of processes: —

#### Resource Structure

Name	Data Type	Description
Crease *	element	Defines one or more Crease lines.



## Crease

Crease defines an individual crease line on a **Component**.

Name	Data Type	Description
<i>Depth</i> ? <a href="#">New in JDF 1.2</a>	double	Depth of the <b>Crease</b> , measured in microns [ $\mu\text{m}$ ].
<i>RelativeTravel</i> ? <a href="#">New in JDF 1.2</a>	double	Relative distance of the reference edge relative to <i>From</i> in the coordinates of the incoming <b>Component</b> . <i>RelativeTravel</i> is always based on the complete size of the input <b>Component</b> 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> ? <a href="#">New in JDF 1.2</a>	XYPair	Relative starting position of the tool. <i>RelativeStartPosition</i> is always based on the complete size of the input <b>Component</b> 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 <b>Component</b> .
<i>RelativeWorkingPath</i> ? <a href="#">New in JDF 1.2</a>	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 <b>Component</b> 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 <b>Component</b> .
<i>StartPosition</i> ? <a href="#">Modified in JDF 1.2</a>	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> ? <a href="#">New in JDF 1.2</a>	double	Distance of the reference edge relative to <i>From</i> . If both <i>Travel</i> and <i>RelativeTravel</i> are specified, <i>RelativeTravel</i> is ignored. At least one of <i>Travel</i> or <i>RelativeTravel</i> must be specified.
<i>WorkingPath</i> ?	XYPair	Working path of the tool beginning at <i>StartPosition</i> . Since the tools can only work parallel to the edges, one coordinate must be zero. If both <i>WorkingPath</i> and <i>RelativeWorkingPath</i> are specified, <i>RelativeWorkingPath</i> is ignored. At least one of <i>WorkingPath</i> or <i>RelativeWorkingPath</i> must be specified.
<i>WorkingDirection</i>	enumeration	Direction from which the tool is working. Possible values are: <i>Top</i> – From above. <i>Bottom</i> – From below.

### 7.2.40 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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>CuttingParams</b>
<b>Example Partition:</b>	—
<b>Input of processes:</b>	—
<b>Output of processes:</b>	—

## Resource Structure

Name	Data Type	Description
<i>BlockElementSize</i> ?	XYPair	Element dimension in X and Y direction. The default value should be equivalent to the XYPair value in <i>BlockSize</i> .
<i>BlockElementType</i> ?	enumeration	Element type. Possible values are: <i>CutElement</i> – Cutting element. <i>PunchElement</i> – Punching element.
<i>BlockName</i>	NMTOKEN	Name of the block. Used for reference by the <b>CutMark</b> resource. Note that <b>CutBlock</b> resources are not partitioned although they are nested. The semantics of nested <b>CutBlocks</b> 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 <b>Component</b> coordinate system.
<i>BlockType</i>	enumeration	Block type. Possible values are: <i>CutBlock</i> – Block to be cut. <i>SaveBlock</i> – Protected block, cut only via outer contour. <i>TempBlock</i> – Auxiliary block that is not taken into account during cutting. <i>MarkBlock</i> – Contains no elements, only marks.

### 7.2.41 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:	<b>CuttingParams, Surface</b>
Example Partition:	–
Input of processes:	–
Output of processes:	–

#### Resource Structure

Name	Data Type	Description
<i>Blocks</i> ? <a href="#">Modified in JDF 1.1</a>	NMTOKENS	Values of the <i>BlockName</i> partition attributes of the blocks defined by the <b>CutMark</b> resource.
<i>MarkType</i>	enumeration	Cut mark type. Possible values are: <i>CrossCutMark</i> <i>TopVerticalCutMark</i> <i>BottomVerticalCutMark</i> <i>LeftHorizontalCutMark</i> <i>RightHorizontalCutMark</i> <i>LowerLeftCutMark</i> <i>UpperLeftCutMark</i> <i>LowerRightCutMark</i> <i>UpperRightCutMark</i>
<i>Position</i>	XYPair	Position of the logical center of the cut mark in the coordinates of the <b>MarkObject</b> 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 7-3: Cut Mark Types

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

## 7.2.42 CuttingParams

[New in JDF 1.1](#)

This resource describes the parameters of a **Cutting** process that uses nested **CutBlocks** as input.

### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>BlockName, RibbonName, SheetName, SignatureName, WebName</i>
<b>Input of processes:</b>	<b>Cutting</b>
<b>Output of processes:</b>	—

## Resource Structure

Name	Data Type	Description
<b>CutBlock</b> *	refelement	One or several <b>CutBlocks</b> can be used to find the <b>Cutting</b> sequence. Only one of <b>CutBlock</b> or <b>Cut</b> may be specified.
<b>CutMark</b> *	refelement	<b>CutMark</b> resources can be used to adapt the theoretical cut positions to the real positions of the corresponding blocks on the <b>Component</b> to be cut.
<b>Cut</b> *	element	<b>Cut</b> elements describe an individual cut. Only one of <b>CutBlock</b> or <b>Cut</b> may be specified.

## Structure of the Cut Subelement

Cut describes one straight cut with an arbitrary tool.

Name	Data Type	Description
<i>RelativeStartPosition</i> ? <a href="#">New in JDF 1.2</a>	XYPair	Relative starting position of the tool. <i>RelativeStartPosition</i> is always based on the complete size of the input <b>Component</b> 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 <b>Component</b> .
<i>RelativeWorkingPath</i> ? <a href="#">New in JDF 1.2</a>	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 <b>Component</b> 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 <b>Component</b> .
<i>StartPosition</i> ? <a href="#">Modified in JDF 1.2</a>	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> ? <a href="#">Modified in JDF 1.2</a>	XYPair	Working path of the tool beginning at <i>StartPosition</i> . Since the tools can only work parallel to the edges, one coordinate must be zero. If both <i>WorkingPath</i> and <i>RelativeWorkingPath</i> are specified, <i>RelativeWorkingPath</i> is ignored. At least one of <i>WorkingPath</i> or <i>RelativeWorkingPath</i> must be specified.
<i>WorkingDirection</i>	enumeration	Direction from which the tool is working. Possible values are: <i>Top</i> – From above. <i>Bottom</i> – From below.

### 7.2.43 DBMergeParams

This resource specifies the parameters of the **DBTemplateMerging** process.

#### Resource Properties

Resource class:	Parameter
Resource references:	—
Resource inheritance:	—
Example Partition:	—
Input of processes:	<b>DBTemplateMerging</b>
Output of processes:	—

## Resource Structure

Name	Data Type	Description
<i>SplitDocuments?</i>	integer	Indicates how often to split documents to create a new file.
FileSpec?	reference	URL of the generated destination file. This is most often a printable file type, (e.g., PDF or PPML). If FileSpec is not specified, <b>DBMergeParams</b> must be a <b>Pipe</b> resource.

### 7.2.44 DBRules

This resource specifies the rules that should 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:	<b><i>DBDocTemplateLayout, Inserting, Collecting, Gathering</i></b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
Comment +	text	Human-readable description of the database rules that map database fields to image or text content.

### 7.2.45 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:	<b><i>DBDocTemplateLayout, Verification</i></b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>DBSchemaType</i>	enumeration	Database type. Possible values are: <i>CommaDelimited</i> <i>SQL</i> <i>XML</i>
Comment +	text	Human-readable description of the database schema.

### 7.2.46 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:	<b><i>DBTemplateMerging, Inserting, Collecting, Gathering, Verification</i></b>
Output of processes:	<b><i>Verification</i></b>

#### Resource Structure

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).

### 7.2.47 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 required date and time.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b><i>Delivery</i></b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>Earliest ?</i>	dateTime	Specifies the earliest time after which the delivery may be made.
<i>Method ?</i>	string	Identifies a required 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. For a list of pre-defined <i>Method</i> values, see <b><i>DeliveryIntent/@Method</i></b> .
<i>Pickup ?</i> <a href="#">Deprecated in JDF 1.2</a>	boolean	If " <i>true</i> ", the merchandise is picked up. If " <i>false</i> ", the merchandise is delivered. Replaced with <i>Transfer</i> in JDF 1.2.
<i>Required ?</i>	dateTime	Specifies the time by which the delivery must be made.
<i>ServiceLevel ?</i> <a href="#">New in JDF 1.2</a>	string	The service level of the specific carrier. Contain values " <i>Next Day</i> ", " <i>2nd Day Air</i> ", " <i>Ground</i> ", etc.

<a href="#">Transfer ?</a> <a href="#">New in JDF 1.2</a>	enumeration	Describes the direction and responsibility of the transfer. Possible values are: <i>BuyerToPrinterDeliver</i> – The <b>DeliveryIntent</b> 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 may specify in the quote a special <b>Contact</b> with <i>ContactTypes</i> , including "Delivery", where the buyer should send the merchandise to. <i>BuyerToPrinterPickup</i> – The <b>DeliveryIntent</b> 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 <b>Contact</b> with <i>ContactTypes</i> , including "Pickup", where the printer has to pick up the merchandise. <i>PrinterToBuyerDeliver</i> – The <b>DeliveryIntent</b> describes an output of the job. In this, case the printer delivers the merchandise to the buyer. The <b>Contact</b> that has <i>ContactTypes</i> , including "Delivery", specifies where the printer should send the merchandise. <i>PrinterToBuyerPickup</i> – the <b>DeliveryIntent</b> describes an output of the job. In this case, the buyer picks up the merchandise. The printer may specify in the quote a special <b>Contact</b> that has <i>ContactTypes</i> including "Pickup", where the buyer should pick up the merchandise.
<a href="#">Company ?</a> <a href="#">Deprecated in JDF 1.1</a>	refelement	Address and further information of the addressee. In JDF 1.1 and beyond, use <b>Contact/Company</b>
<a href="#">Contact *</a> <a href="#">New in JDF 1.1</a>	refelement	Address and further information of the <b>Contact</b> responsible for this delivery.
Drop +	element	All locations where the product will be delivered.

### Structure of the Drop Subelement

Name	Data Type	Description
<a href="#">Earliest ?</a>	dateTime	Specified the earliest time after which the delivery may be made. Default = <b>DeliveryParams/@Earliest</b> .
<a href="#">Method ?</a>	string	Identifies a required delivery method, (e.g., <i>ExpressMail</i> or <i>InterofficeMail</i> ). Default = <b>DeliveryParams/@Method</b> . Note that it is strongly recommended to use an NMTOKEN compatible string without blank spaces in this attribute. For a list of predefined <i>Method</i> values, see <b>DeliveryIntent/@Method</b> .
<a href="#">Pickup ?</a> <a href="#">Deprecated in JDF 1.2</a>	boolean	If "true", the merchandise is picked up. If "false", the merchandise is delivered. Default = <b>DeliveryParams/@Pickup</b> . Replaced with <i>Transfer</i> in JDF 1.2.
<a href="#">Required ?</a>	dateTime	Specifies the time by which the delivery must be made. Default = <b>DeliveryParams/@Required</b> .
<a href="#">ServiceLevel ?</a> <a href="#">New in JDF 1.2</a>	string	The service level of the specific carrier. Contain values "Next Day", "2nd Day Air", "Ground", etc. Default = <b>DeliveryParams/@ServiceLevel</b> .
<a href="#">TrackingID ?</a> <a href="#">New in JDF 1.2</a>	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.
<a href="#">Transfer ?</a> <a href="#">New in JDF 1.2</a>	enumeration	Describes the direction and responsibility of the transfer. The default is and possible values are defined in <b>DeliveryParams/@Transfer</b> .
<a href="#">Company ?</a> <a href="#">Deprecated in JDF 1.1</a>	refelement	Address and further information of the addressee. Defaults to the value of <b>Company</b> specified in the root <b>DeliveryParams</b> resource.

Name	Data Type	Description
<b>Contact *</b> <a href="#">New in JDF 1.1</a>	refelement	Address and further information of the <b>Contact</b> responsible for this delivery. Default = <b>DeliveryParams/Contact</b> .
DropItem +	element	A <b>Drop</b> may consist of multiple products, which are represented by their respective <b>PhysicalResource</b> resources. Each <b>DropItem</b> describes an individual resource that is part of this <b>Drop</b> .

### Structure of the DropItem Subelement

Name	Data Type	Description
<i>Amount</i> ?	integer	Specifies the number of <b>PhysicalResource</b> ordered. If <i>Amount</i> is not specified, defaults to the total amount of the resource that is referenced by <b>PhysicalResource</b> .
<i>Unit</i> ?	string	Unit of measurement for the <i>Amount</i> of the resource that is referenced by <b>PhysicalResource</b> . Defaults to the value of <i>Unit</i> defined in <b>PhysicalResource</b> .
<i>TrackingID</i> ? <a href="#">New in JDF 1.2</a>	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> .
<b>PhysicalResource</b> ? <a href="#">Modified in JDF 1.2</a>	refelement	Description of the individual item to be delivered. It can be any kind of physical resource. This was <b>Component</b> prior to JDF 1.2.

### 7.2.48 DensityMeasuringField

This resource contains information about a density measuring field.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>ColorControlStrip, Surface/PlacedObject</b>
Example Partition:	—
Input of processes:	Any printing process
Output of processes:	

#### Resource Structure

Name	Data Type	Description
<i>Center</i>	XYPair	Position of the center of the density measuring field in the coordinates of the <b>MarkObject</b> that contains this mark. If the measuring field is defined within a <b>ColorControlStrip</b> , <i>Center</i> refers to the rectangle defined by <i>Center</i> and <i>Size</i> of the <b>ColorControlStrip</b> .
<i>Density</i> <a href="#">Modified in JDF 1.1A</a>	DoubleList	Density value for each process color measured with filter. The data type was modified to NumberList in JDF 1.1A in order to accommodate density values >1.0.
<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 use 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.
<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 %).
<b>ColorMeasurementConditions ?</b> <a href="#">New in JDF 1.1</a>	refelement	Detailed description of the measurement conditions for color measurements.

## 7.2.49 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:** *ContactCopying, FilmToPlateCopying, ImageSetting*

**Output of processes:** —

### Resource Structure

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>PostExposeTime ?</i>	duration	Duration of the postexposing process. Only one of <i>PostBakeTime</i> and <i>PostExposeTime</i> must be specified.

## 7.2.50 Device

Information about a specific device. This optionally includes information about the devices capabilities. For more information, see Section 3.6.1.3, Implementation Resources and Section 4.8, Capability and Constraint Definitions.

### Resource Properties

Resource class:	Implementation
Resource referenced by:	—
Example Partition:	—
Input of processes:	Any process
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>DeviceFamily</i> ? <a href="#">Deprecated in JDF 1.1</a>	string	Manufacturer family type ID. <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. Must be the same over time for a specific device instance, (i.e., must survive reboots). For UPNP devices, this matches UPNP:UDN. See [UPNP]
<i>DeviceType</i> ?	string	Manufacturer type ID, including a revision stamp.
<i>Directory</i> ? <a href="#">New in JDF 1.1</a> <a href="#">Clarified in JDF 1.2</a>	URL	Defines a directory where the URLs that are associated with this <b>Device</b> can be located. If <i>Directory</i> is specified, it must be an Absolute URI [RFC2396] that specifies a Base URI to resolve each of the various URL attributes of <b>Device</b> . See "Resolving RunList/@Directory and FileSpec/@URL URI references" on page 649 and [FileURL].
<i>FriendlyName</i> ? <a href="#">New in JDF 1.1</a>	string	Short user-friendly title.
<i>JDFErrorURL</i> ? <a href="#">New in JDF 1.2</a>	URL	URL where, by default, the device may copy JDF output job tickets that are aborted or in error.. If this is a directory, it specifies the device default error output folder. If not specified, it defaults to the value of <i>JDFOutputURL</i> .
<i>JDFInputURL</i> ? <a href="#">New in JDF 1.2</a>	URL	URL where, by default, the device may accept JDF input job tickets.. If this is a directory, it specifies the device default directory. The persistence of JDF tickets in this location is implementation dependent. If not specified, the Device does not accept JDF without a JMF SubmitQueueEntry.
<i>JDFOutputURL</i> ? <a href="#">New in JDF 1.2</a>	URL	URL where, by default, the device may copy successfully completed JDF output job tickets. If this is a directory, it specifies the device default output folder.
<i>JDFVersions</i> ? <a href="#">New in JDF 1.1</a>	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).
<i>JMFSenderID</i> ? <a href="#">New in JDF 1.1</a>	string	ID of the controller will process JMF messages for the device. This corresponds to the <i>SenderID</i> attribute that must be specified for the device in JMF messages.
<i>JMFURL</i> ? <a href="#">New in JDF 1.1</a>	URL	URL of the device port that will accept JMF messages.
<i>KnownLocalizations</i> ? <a href="#">New in JDF 1.2</a>	languages	A list of all language codes supported by the device for localization. If not specified, then the device supports no localizations.
<i>Manufacturer</i> ? <a href="#">New in JDF 1.1</a>	string	Manufacturer name.
<i>ManufacturerURL</i> ? <a href="#">New in JDF 1.1</a>	string	Web site for manufacturer.

Name	Data Type	Description
<i>ModelDescription</i> ? <a href="#">New in JDF 1.1</a>	string	Long description for end user.
<i>ModelName</i> ? <a href="#">New in JDF 1.1</a>	string	Model name.
<i>ModelNumber</i> ? <a href="#">New in JDF 1.1</a>	string	Model number.
<i>ModelURL</i> ? <a href="#">New in JDF 1.1</a>	string	Web site for model.
<i>SerialNumber</i> ? <a href="#">New in JDF 1.1</a>	string	Serial number of the device.
<i>PresentationURL</i> ? <a href="#">New in JDF 1.1</a>	string	URL to presentation for device It is a URL to a device-provided UI for user configuration, status, etc. Thus, if the device has an embedded Web server, this is a URL to the configuration page hosted on that Web server.
<i>UPC</i> ? <a href="#">New in JDF 1.1</a>	string	Universal Product Code for the device. A 12-digit, all-numeric code that identifies the consumer package. Managed by the Uniform Code.
<i>CostCenter</i> ?	element	MIS cost center ID.
<i>DeviceCap</i> * <a href="#">New in JDF 1.1</a>	element	Description of the capabilities of the device. The <b>DeviceCap</b> elements are combined with a logical OR, (i.e., if a JDF resides within any parameter space defined by a <b>DeviceCap</b> , the device can process the job). For details see <a href="#">Section 7.3, Device Capability Definitions</a> .
<i>IconList</i> ? <a href="#">New in JDF 1.1</a>	element	List of locations of icons that can be used to represent the device.

### Structure of the IconList Subelement

[New in JDF 1.1](#)

The **IconList** is a list of individual icon descriptions.

Name	Data Type	Description
Icon +	refelement	Individual icon description.

### Structure of the Icon Subelement

[New in JDF 1.1](#)

An **ICON** represents a device in the user interface.

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 <b>ICON</b> represents. Any combination of: <i>Unknown</i> – No link to the device exists <i>Idle</i> <i>Down</i> <i>Setup</i> <i>Running</i> <i>Cleanup</i> <i>Stopped</i> If not specified, the <b>ICON</b> is used for all of the above. The meaning of the individual enumerations is described in the <b>DeviceInfo</b> message element. See <a href="#">Section , KnownDevices</a> .
<i>FileSpec</i>	element	Details of the file containing the icon data.

### 7.2.51 DeviceMark

[New in JDF 1.1](#) Promoted from subelement status in the **Surface** resource with new attributes defined below.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>Surface</b>
Example Partition:	<i>Side</i>
Input of processes:	—
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>Font ?</i>	NMTOKEN	The name of the font that should be used for the <b>DeviceMark</b> . Values include: <i>Courier</i> <i>Helvetica</i> <i>Helvetica-Condensed</i> <i>Times-Roman</i>
<i>FontSize ?</i>	integer	The size of the font that should be used for the <b>DeviceMark</b> , in points $\geq 0$ .
<i>MarkJustification ?</i>	enumeration	Description of the preferred <b>DeviceMark</b> justification. Interpreted in context of the <i>MarkOrientation</i> . One of: <i>Center</i> <i>Left</i> <i>Right</i>
<i>MarkOffset ?</i>	XYPair	Description of the preferred <b>DeviceMark</b> offset. Interpreted in context of the device dependent default position in the coordinate system defined by <i>MarkOrientation</i> .
<i>MarkOrientation ?</i>	enumeration	Description of the preferred <b>DeviceMark</b> orientation One of: <i>Horizontal</i> <i>Vertical</i>
<i>MarkPosition ?</i>	enumeration	Description of the preferred <b>DeviceMark</b> position. One of: <i>Top</i> <i>Bottom</i> <i>Left</i> <i>Right</i>

### 7.2.52 DeviceNSpace

[Modified in JDF 1.2](#)

[**Note:** **DeviceNSpace** was elevated to a resource in JDF 1.2.] **DeviceNSpace** may be used in several ways. For example, defining the specific colorants of a **DeviceNSpace**:

- **ColorantControl/ColorPool/@ColorantNameSet** 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/@ColorantNameSet** matches, and then the corresponding
  - **ColorantControl/ColorPool/Color/DeviceNColor/@ColorList** attribute gives the set of **DeviceNSpace** colorant percent values necessary to construct the,
  - **ColorantControl/@ColorantParams** colorant (also named **ColorantControl/ColorPool/Color/@Name**) in using **DeviceNSpace** colorants.

### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>ColorantControl, ColorSpaceConversionParams</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

### Resource Structure

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.
<b>SeparationSpec</b> * <a href="#">Modified in JDF 1.2</a>	refelement	Ordered list of colorant names that define the DeviceN color space. Note that these colorants must be specified in a corresponding <b>ColorantParams</b> element of the <b>ColorantControl</b> or be implied by <i>ProcessColorModel</i> . In other words, they must be real, physical colorants.

### 7.2.53 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:	<b>DigitalDelivery</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<b>Contact</b> *	refelement	Source and destination address for the transfer of the artwork. The destination delivery address is specified as the <b>ComChannel</b> of the <b>Contact</b> with <i>ContactTypes</i> including <i>Delivery</i> . Only one <b>Contact</b> with <i>ContactTypes</i> including <i>Delivery</i> may be specified per destination. If multiple delivery destinations are specified within one <b>DigitalDelivery</b> process, the <b>Contact</b> with <i>ContactTypes</i> including <i>Delivery</i> must be partitioned by the partition key " <i>Location</i> ". In case the output <b>RunList</b> completely specifies the destination, the <b>Contact</b> with <i>ContactTypes</i> including <i>Delivery</i> should be omitted. This may generally be the case when <i>Method</i> = " <i>NetworkCopy</i> " or " <i>WebServer</i> ". <b>Contact</b> with <i>ContactTypes</i> including " <i>Sender</i> " will specify the source address.
<i>DigitalDeliveryDirection ?</i>	enumeration	Describes which side activates the delivery. <i>Push</i> – The artwork will be sent (the source end is active). <i>Pull</i> – The artwork will be retrieved (the destination end is active).

Name	Data Type	Description
<i>DigitalDeliveryProtocol?</i>	NMTO-KEN	Identifies the delivery network protocol. Values include: <i>FTP</i> <i>HTTP</i> <i>HTTPS</i> <i>SMTP</i>
<i>Method?</i>	NMTO-KEN	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> May also be a digital delivery service brand, includes: <i>Vio</i> <i>WAMNET</i>

### Compression & Encoding of the transferred files:

In order to instruct a digital delivery device to compress or encode the files one may use the input and output **RunList** with **FileSpec/@Compression** attribute, even if no URL is specified. See “DigitalDelivery Examples” on page 699. for a set of examples.

## 7.2.54 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:	<b>ArtDeliveryIntent</b> , <b>DeliveryParams</b>
Example Partition:	—
Input of processes:	—
Output of processes:	<b>Delivery</b>

### Resource Structure

Name	Data Type	Description
<i>Capacity?</i>	integer	Size of the digital media in megabytes.
<i>MediaLabel?</i>	string	Electronic label of the media.
<i>MediaType</i>	NMTO-KEN	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.
<b>RunList?</b>	refelement	Link to the relevant files on the media. The URL specified in this <b>RunList</b> should be a relative path to the media's mount point.

## 7.2.55 DigitalPrintingParams

This resource contains attributes and elements used in executing the **DigitalPrinting** process. The *PrintingType* attribute in this resource defines two types of printing: *SheetFed* and *WebFed*. The principal difference between them is the shape of the paper each is equipped to accept. Presses that execute *WebFed* processes use substrates that are continuous and cut after printing is accomplished. Most newspapers are printed on web-fed presses. *SheetFed* printing, on the other hand, accepts precut substrates.

### 7.2.55.1 Coordinate systems in DigitalPrinting

[New in JDF 1.2](#)

Figure 2.10 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:	<i>BlockName, DocRunIndex, DocSheetIndex, PartVersion, Run, RunIndex, RunTags, SheetIndex, Separation, SheetName, Side, SignatureName, DocIndex</i>
Input of processes:	<b>DigitalPrinting</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<a href="#">Collate ?</a> <a href="#">New in JDF 1.1</a>	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 <b>RunList/@DocCopies</b> and job copies can be requested by specifying the output <b>Component Amount</b>.</p> <p><i>None</i> – Do not collate sheets in the document or document(s) in the job.</p> <p><i>Sheet</i> – Collate the sheets in each document; do not collate the documents in the job. The result of <i>Sheet</i> and <i>SheetAndSet</i> is the same when there is one document in the set. The result of <i>Sheet</i> and <i>SheetSetAndJob</i> is the same when there is one document in the set and one set in the job.</p> <p><i>SheetAndSet</i> – Collate the sheets in the document and collate the documents in the set. Do not collate the sets in the job. The result of <i>SheetAndSet</i> and <i>SheetSetAndJob</i> is the same when there is one set in the job.</p> <p><i>SheetSetAndJob</i> – Collate the sheets in the document and collate the documents in the set and collate the sets in the job.</p> <p>The following example consists of two documents, A and B, each having two sheets, A1, A2 and B1, B2. The number of document copies requested is one for both documents and the number of job copies requested is three (<b>Component Amount</b> = 3). The job contains no document set boundaries.</p> <p>If <b>Collate</b> = "<i>None</i>", the sheet order will be: A1A1A1 A2A2A2 B1B1B1 B2B2B2</p> <p>If <b>Collate</b> = "<i>Sheet</i>", the sheet order will be: A1A2 A1A2 A1A2 B1B2 B1B2 B1B2</p> <p>If <b>Collate</b> = "<i>SheetAndSet</i>" or "<i>SheetSetAndJob</i>", the sheet order will be: A1A2 B1B2 A1A2 B1B2 A1A2 B1B2</p>

Name	Data Type	Description
<i>DirectProofAmount</i> = "0" <a href="#">New in JDF 1.2</a>	integer	If greater than zero (>0), a set of proofs is directly produced and subsequently an approval may 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 not required for setup, but it is required for the actual print run. If the <b>DigitalPrinting</b> process is waiting for a <i>DirectProofAmount</i> , the JDF node's <i>Status</i> is switched to "Stopped" with the <i>StatusDetails</i> = "WaitForApproval".
<i>ManualFeed</i> = "false" <a href="#">New in JDF 1.1</a>	boolean	Indicates whether the media will be fed manually.
<i>NonPrintableMarginBottom</i> ? <a href="#">New in JDF 1.2</a>	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 <b>DigitalPrinting</b> process. The <b>DigitalPrinting</b> process must put marks up to, but not in, the non-printable margin area. The <b>Media</b> 's origin is unaffected by <i>NonPrintableMarginBottom</i> . These margins are independent of the PDL content.
<i>NonPrintableMarginLeft</i> ? <a href="#">New in JDF 1.2</a>	double	Same as <i>NonPrintableMarginBottom</i> except for the left margin.
<i>NonPrintableMarginRight</i> ? <a href="#">New in JDF 1.2</a>	double	Same as <i>NonPrintableMarginBottom</i> except for the right margin.
<i>NonPrintableMarginTop</i> ? <a href="#">New in JDF 1.2</a>	double	Same as <i>NonPrintableMarginBottom</i> except for the top margin.
<i>OutputBin</i> ? <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	NMTOKEN	Specifies the bin to which the finished document should be output. Suggested values can be found in "Input Tray and Output Bin Names" on page 633.
<i>PageDelivery</i> ? <a href="#">New in JDF 1.1</a>	enumeration	Indicates how pages are to be delivered to the output bin or finisher. Possible values are: <i>FanFold</i> – The output is alternating face-up, face down. <i>SameOrderFaceUp</i> – Order as defined by the RunList, with the "front" sides of the media up. <i>SameOrderFaceDown</i> – Order as defined by the RunList, with the "front" sides of the media down. <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>PrintingType</i> ? <a href="#">Modified in JDF 1.2</a>	enumeration	Type of printing machine. Possible values are: <i>ContinuousFed</i> – connected sheets including fan fold. <a href="#">New in JDF 1.2</a> <i>SheetFed</i> <i>WebFed</i>
<i>PrintQuality</i> ? <a href="#">Deprecated in JDF 1.1</a>	enumeration	Indicates how pages are to be delivered to the output bin or finisher. Possible values are: <i>High</i> – Highest quality available on the printer. <i>Normal</i> – The default quality provided by the printer. <i>Draft</i> – Lowest quality available on the printer Replaced by <b>InterpretingParams/@PrintQuality</b>



Name	Data Type	Description
<i>SheetLay</i> ?	enumeration	Lay of input media. Reference edge of where paper is placed in feeder. Possible values are: <i>Left</i> <i>Right</i> <i>Center</i>
<b>ApprovalParams</b> ? <a href="#">New in JDF 1.2</a>	refelement	Details of the direct approval process, when <i>DirectProofAmount</i> > 0.
<b>Component</b> ? <a href="#">New in JDF 1.1</a>	refelement	Describes the preprocessed media to be used. Different <b>Media</b> and/or <b>Components</b> may be specified in different partition leaves to enable content-driven input <b>Media</b> selection. For any given partition, only one of <b>Media</b> or <b>Component</b> must be specified per partition.
<b>Disjointing</b> ? <a href="#">New in JDF 1.1</a>	refelement	Describes how individual components are separated from one another in the output bin.
<b>Media</b> ? <a href="#">New in JDF 1.1</a>	refelement	Describes the media to be used. Different <b>Media</b> and/or <b>Components</b> may be specified in different partition leaves to enable content driven input <b>Media</b> selection. For any given partition, only one of <b>Media</b> or <b>Component</b> must be specified per partition.
<b>MediaSource</b> ? <a href="#">Deprecated in JDF 1.1</a>	refelement	Describes the media to be used. For any given partition, only one of <b>MediaSource</b> or <b>Component</b> may be specified. Replaced with <b>Media</b> in JDF 1.1.

### 7.2.56 Disjointing

The **Disjointing** resource describes how individual components are separated from one another on a stack.

#### Resource Properties

Resource class:	ResourceElement
Resource referenced by:	<b>Component, DigitalPrintingParams, GatheringParams</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>Number</i> ?	integer	Number of sheets that make up one component.
<i>Offset</i> ?	XYPair	Offset dimension in X- and Y-dimensions that separates the components.
<i>OffsetAmount</i> ?	integer	The number of components that are shifted in <i>OffsetDirection</i> simultaneously.
<i>OffsetDirection</i> ? <a href="#">Clarified in JDF 1.2</a>	enumeration	Offset-shift action for the first component. A component can be offset to one of two positions—left or right. Possible values are: <i>Alternate</i> – The position of the first component is opposite to the position of the previous component and subsequent components are each offset to alternating positions. For example, if the last item in the stack was positioned to the right then the subsequent items will be positioned to the left, right, left, right, and so on. <i>Left</i> – Offset consecutive components sideways to the left, next to the right. <i>None</i> – Do not offset consecutive components. The position of all components is the same as the position of the previous component. <i>Right</i> – Offset consecutive components sideways to the right, next to the left. <i>Straight</i> – Same as <i>None</i> . <a href="#">Deprecated in JDF 1.2</a>

Name	Data Type	Description
<a href="#">Overfold ?</a> <a href="#">Deprecated in JDF 1.1</a>	double	Expansion of the overfold of a sheet. This attribute may be needed for the <b>Inserting</b> or other postpress processes. Moved to <b>Component</b> .
<a href="#">IdentificationField *</a> <a href="#">Modified in JDF 1.1</a>	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 <b>IdentificationField</b> element.
<a href="#">InsertSheet ?</a>	refelement	Some kind of physical marker (e.g., a paper strip or a yellow paper sheet) that separates the components.

## 7.2.57 Disposition

[New in JDF 1.2](#)

This element describes how long an asset must be maintained by a device. The device will perform an action defined by **Disposition/@DispositionAction** when a "disposition time" occurs. Disposition time is defined either as:

$$Until \leq "Disposition\ time" \leq Until + ExtraDuration$$

$$ProcessCompleteTime + MinDuration \leq "Disposition\ time" \leq$$

$$ProcessCompleteTime + MinDuration + ExtraDuration$$

### Resource Properties

**Resource class:** ResourceElement

**Resource referenced by:** **FileSpec, SubmitQueueEntry/QueueSubmissionParams, RunList**

**Example Partition:** —

**Input of processes:** —

**Output of processes:** —

### Resource Structure

Name	Data Type	Description
<a href="#">DispositionAction</a> = "Delete"	enumeration	<i>Delete</i> – The asset must be deleted when disposition time occurs. <i>Archive</i> – The asset must be archived when disposition time occurs.
<a href="#">DispositionUsage ?</a>	enumeration	Specifies the usage of the asset by the process. If not specified, Disposition applies to all processes that link to the resource. <i>Input</i> – Disposition applies only to processes that use the asset as an input resource. <i>Output</i> – Disposition applies only to processes that use the asset as an output resource.
<a href="#">ExtraDuration ?</a>	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 neither <i>ExtraDuration</i> , <i>MinDuration</i> , nor <i>Until</i> are specified, the asset is retained for a system specified time.
<a href="#">MinDuration ?</a>	duration	Indicates the minimum duration that the device should retain the asset after the process that uses the asset completes.
<a href="#">Priority = "0"</a>	integer	Value between 0 and 100 that specifies the order in which assets will be deleted or archived when the values of <i>Duration</i> , <i>MinDuration</i> , nor <i>Until</i> cannot be honored, (e.g. when local storage runs low). Assets with <i>Priority</i> = "0" will be deleted first.
<a href="#">Until ?</a>	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.

## 7.2.58 DividingParams

[Deprecated in JDF 1.1.](#)

Since the **Dividing** process has been replaced by **Cutting**, this resource is no longer required. See "DividingParams" on page 755 for details of this deprecated resource.

## 7.2.59 ElementColorParams

[New in JDF 1.2](#)

This resource provides a container for color metadata applicable to a **LayoutElement**.

### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>LayoutElement</b> , <b>PageList</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>ColorManagementSystem</i> ?	NMTOKEN	Identifies the preferred ICC color management system to use when performing color transformations on the particular <b>LayoutElement</b> . When specified, this attribute overrides any default selection of a color management system by an application and overrides the "CMM Type" value (bytes 4-7 of an ICC Profile Header) in any of the job related ICC profiles. This string attribute value identifies the manufacturer of the preferred CMM and must match one of the registered four-character ICC CMM Type values. See the ICC Manufacturer's Signature Registry at <a href="http://www.color.org">http://www.color.org</a> . 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. Possible values are: <i>PDLActual</i> – The embedded PDL output printing condition defines the actual output intent profile, (e.g., the final press output). <i>PDLReference</i> – The embedded PDL output printing condition defines the reference output intent profile, (e.g., the press profile for proofing). <i>IgnorePDL</i> – The embedded ICC output profile is incorrect and should be ignored.
<b>AutomatedOverPrintParams</b> ?	refelement	A resource that provides controls for the automated selection of overprinting of black text or graphics.
<b>ColorantAlias</b> *	refelement	Each resource instance specifies a replacement colorant name string to be used instead of one or more named colorant strings found in the layout element.
<b>ColorSpaceConversionOp</b> ?	refelement	List of <b>ColorSpaceConversionOp</b> subelements, each of which identifies a type of object, defines the source color space for that type of object, and specifies the behavior of the conversion operation for that type of object. If not present, the default conversion behavior is derived from <i>ColorStandard</i> . <b>ColorSpaceConversionOp/@Operation</b> is ignored in the context of <b>ElementColorParams</b> .
<b>FileSpec</b> ? ( <i>ActualOutputProfile</i> )	refelement	A <b>FileSpec</b> resource pointing to an ICC profile that describes the characterization of an actual output target device.
<b>FileSpec</b> ? ( <i>ReferenceOutputProfile</i> )	refelement	A <b>FileSpec</b> resource pointing to an ICC profile that describes a reference output print condition behavior that should 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 <b>ColorIntent/@ColorStandard</b> .

## 7.2.60 EmbossingParams

[New in JDF 1.1](#)

This resource contains attributes and elements used in executing the **Embossing** process.

### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>BlockName, RibbonName, SheetName, SignatureName, WebName</i>
Input of processes:	<b>Embossing</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>Emboss</i> *	element	One <i>Emboss</i> element is specified for each impression.

### Structure of the Emboss Subelement

Name	Data Type	Description
<i>Direction</i>	enumeration	The direction of the image. Possible values are: <i>Both</i> – Both debossing and embossing in one stamp. <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. For <i>EdgeAngle</i> to exist, <i>EdgeShape</i> = <i>Beveled</i> must be specified.
<i>EdgeShape</i> = "Rounded"	enumeration	The transition between the embossed surface and the surrounding media may be rounded or beveled (angled). Possible values are: <i>Rounded</i> <i>Beveled</i>
<i>EmbossingType</i>	enumeration	Possible values include <i>BlindEmbossing</i> – Embossed forms that are not inked or foiled. The color of the image is the same as the paper. <i>EmbossedFinish</i> – The overall design or pattern impressed in laminated paper when passed between metal rolls engraved with the desired pattern. Produced on a special embossing to create finishes such as linen. <i>FoilEmbossing</i> – Combines embossing with foil stamping in one single impression. <i>FoilStamping</i> – Using a heated die to place a metallic or pigmented image from a coated foil on the paper. <i>RegisteredEmbossing</i> – Creates an embossed image that exactly registers to a printed image.
<i>Height</i> ?	double	The height of the levels. This value specifies the <i>vertical</i> distance between the highest and lowest point of the stamp, regardless of the value of <i>Direction</i> .
<i>ImageSize</i> ?	XYPair	The size of the bounding box of one single image.
<i>Level</i> ?	enumeration	The level of embossing. Possible values are: <i>SingleLevel</i> <i>MultiLevel</i> <i>Sculpted</i>
<i>Position</i> ?	XYPair	Position of the lower left corner of the bounding box of the embossed image in the coordinate system of the <b>Component</b> .

## 7.2.61 Employee

Information about a specific device or machine operator (see Section 3.6.1.3, Implementation Resources). **Employee** is also used to describe the contact person who is responsible for executing a node, as defined in the **NodeInfo** field of a JDF node.

### Resource Properties

<b>Resource class:</b>	Implementation
<b>Resource referenced by:</b>	NodeInfo
<b>Example Partition:</b>	—
<b>Input of processes:</b>	Any process
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
<i>PersonalID</i> ?	string	ID of the relevant MIS employee.
<i>Roles</i> ? <a href="#">New in JDF 1.2</a>	NMTO- KENS	Defines the list of roles that the employee fills. Values include: <i>Apprentice</i> – Employee that is in training, (“Auszubildender” / “Auszubildende” in German). <i>Assistant</i> – Assistant operator. <i>Craftsman</i> – Trained employee, (“Geselle” / “Facharbeiter” in German). <i>CSR</i> – Customer Service Representative <i>Manager</i> – Manager. <i>Master</i> – Highly trained employee, (“Meister” in German). <i>Operator</i> – Operator. <i>ShiftLeader</i> – The leader of the shift.
<i>Shift</i> ?	string	Defines the shift to which the employee belongs.
<b>CostCenter</b> ?	element	MIS cost center ID.
<b>Person</b> ?	refelement	Describes the employee. If no <b>PERSON</b> element is specified, the <b>Employee</b> resource represents any employee who fulfills the selection criteria.

## 7.2.62 EndSheetGluingParams

This resource describes the attributes and elements used in executing the **EndSheetGluing** process.

### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	—
<b>Input of processes:</b>	<b>EndSheetGluing</b>
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
EndSheet (Front)	element	Information about the front-end sheet. The <i>Side</i> attribute of this element must be “Front”.
EndSheet (Back)	element	Information about the back-end sheet. The <i>Side</i> attribute of this element must be “Back”.

## Structure of EndSheetGluingParams Elements

### EndSheet

Name	Data Type	Description
<i>Offset ?</i> <a href="#">Deprecated in JDF 1.2</a>	XYPair	Offset of end sheet in X and Y direction. In JDF 1.2 and beyond, <i>Offset</i> is implied by the Transformation matrix in ResourceLink/ <i>@Transformation</i> of the EndSheet's ComponentLink.
<i>Side</i>	enumeration	Location of the end sheet. Possible values are: <i>Front</i> <i>Back</i>
<b>GlueLine</b>	element	Description of the glue line.

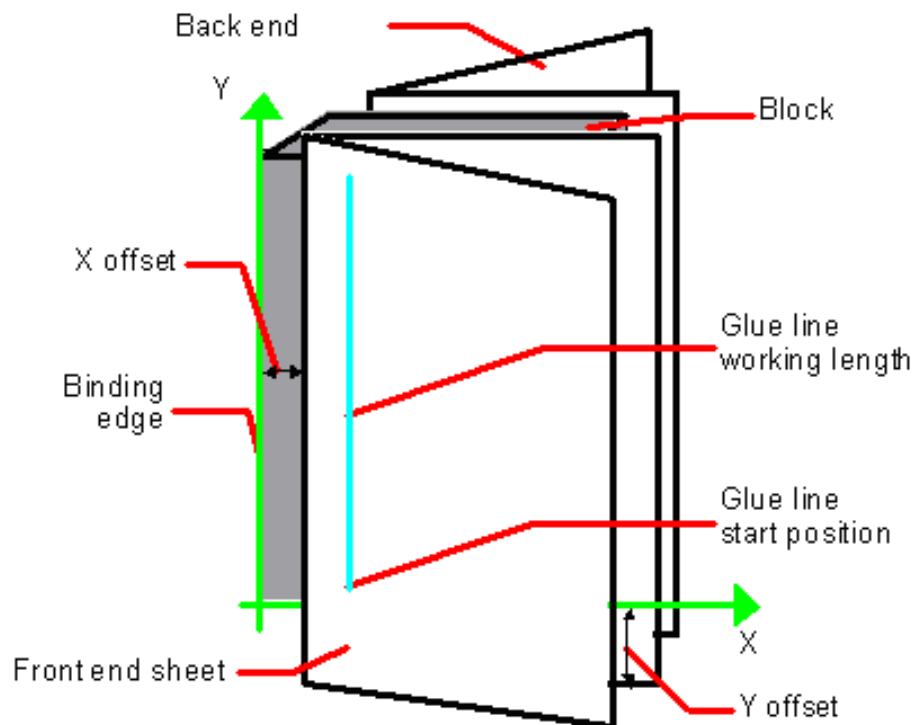


Figure 7.7: Parameters and coordinate system used for end-sheet gluing

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).

### 7.2.63 ExposedMedia

This resource represents a processed **Media**-based handling resource such as film, plate, or paper proof. It is also used as an input resource for the **Scanning** process.

#### Resource Properties

Resource class:	Handling
Resource referenced by:	—
Example Partition:	<i>DocIndex, RunIndex, RunTags, Separation, SheetName, Side, SignatureName, TileID, WebName</i>
Input of processes:	<b>ContactCopying, ConventionalPrinting, PreviewGeneration, DigitalPrinting, Scanning</b>
Output of processes:	<b>ContactCopying, ImageSetting, FilmToPlateCopying, Proofing</b>

#### Resource Structure

Name	Data Type	Description
<i>ColorType</i> ?	enumeration	Possible values are: <i>Color</i> <i>GrayScale</i> <i>Monochrome</i> – Black and white.
<i>Polarity</i> = "true"	boolean	<i>false</i> if the media contains a negative image.
<i>ProofName</i> ? <a href="#">New in JDF 1.2</a>	string	When this <b>ExposedMedia</b> specifies a proof, <i>ProofName</i> is the name of the <b>ProofingIntent</b> /ProofItem that specified this proof in the product intent section.
<i>ProofQuality</i> ? <a href="#">Modified in JDF 1.2</a>	enumeration	This attribute is present if the <b>ExposedMedia</b> resource describes a proof. Possible values are: <i>None</i> – Not a proof or the quality is unknown. <a href="#">Deprecated in JDF 1.2</a> <i>Halftone</i> – The halftones are emulated. <i>Contone</i> – No halftones, but exact color. <i>Conceptual</i> – Color does not match precisely.
<i>ProofType</i> ? <a href="#">Modified in JDF 1.2</a>	enumeration	<i>None</i> – Not a proof or the type is unknown. <a href="#">Deprecated in JDF 1.2</a> <i>Page</i> – Page proof. <i>Imposition</i> – Imposition proof.
<i>PunchType</i> ?	string	Name of the registration punch scheme. Possible values include, but are not limited to: <i>Bacher</i> <i>Stoesser</i> If not specified, no holes are punched.
<i>Resolution</i> ?	XYPair	Resolution of the output.
<b>FileSpec</b> ( <i>OutputProfile</i> )?	refelement	A <b>FileSpec</b> resource pointing to an ICC profile that describes the output process for which this media was exposed.
<b>Media</b>	refelement	Describes media specifics such as size and type.
<b>ScreeningParams</b> ?	refelement	Used to describe the screening in case of rasterized media.

## 7.2.64 FeedingParams

[New in JDF 1.2](#)

The parameters for any JDF Feeder processing device.

### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>DocIndex, RunIndex, RunTags, Separation, SheetName, Side, SignatureName, TileID, WebName</i>
<b>Input of processes:</b>	<b>Feeding</b>
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
Feeder *	element	Defines the specifics of an individual Feeder. If a <b>Component</b> or <b>Media</b> 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 <b>Component(s)</b> . If a CollatingItem is not defined, then one <b>Component</b> in the order of input resource link list is consumed.

### Structure of the Feeder Element

Name	Data Type	Description
<i>AlternatePositions ?</i>	IntegerList	Positions of alternate feeders including the feeder specified in <i>Position</i> on a feeding chain. Alternate feeders share the load according to the policy defined in <i>FeederSynchronization</i> . If not specified, it defaults to the value of <i>Position</i> . <i>AlternatePositions</i> must be non-negative.
<i>Position ?</i>	integer	<i>Position</i> of feeder on a collecting and gathering chain in chain movement direction. <i>Position</i> = "0" is first feeder feeding to the collecting and gathering chain. Only one Feeder may 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.).
<i>FeederSynchronization</i> = "Primary"	enumeration	Specifies the synchronization of multiple Feeder(s) with identical <i>Component(s)</i> : Values include: <i>Alternate</i> – The feeders specified in <i>Position</i> alternate. <i>Backup</i> – This Feeder is the backup feeder for the <b>Component</b> 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 <b>Component</b> .
<i>FeederType ?</i>	NMTOKEN	Specifies the feeder type. Values include: <i>Sheet</i> – Single sheet feeder. <i>Signature</i> – Single signature feeder. <i>Folding</i> – Folding feeder that folds the input Component or Media. <i>Gluing</i> – Gluing feeder <i>AddOn</i> – Add on feeder, (e.g., CDs).



Name	Data Type	Description
<i>Loading ?</i>	NMTOKEN	Specifies the feeder loading. Values include: <i>Bundle</i> – Stream feeder, using the output of the <b>Bundling</b> process. <i>FanFold</i> – Automatic loading of <b>FanFold Media</b> . <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 <b>PrintRolling</b> process.
<i>Opening = "None"</i>	enumeration	Specifies the opening of signatures: <i>Back</i> – Overfold on back. <i>Front</i> – Overfold on front. <i>None</i> – Signatures are not opened. <i>Sucker</i> – Sucker opening, no overfold.
<b>Component ?</b>	refelement	Specifies the <b>Component</b> that is to be loaded into this Feeder. Only one of <b>Component</b> or <b>Media</b> must be specified.
FeederQualityParams ?	element	Definition of the setup and policy for feeding quality.
<b>Media ?</b>	refelement	Specifies the <b>Media</b> that is to be loaded into this Feeder. Only one of <b>Component</b> or <b>Media</b> must be specified.

### Structure of the FeederQualityParams Subelement

The FeederQualityParams element defines the setup and policy for feeding quality control. It may be specified individually for each Feeder.

Name	Data Type	Description
<i>IncorrectComponentQuality ?</i>	enumeration	Defines the operation of the incorrect components quality control: Supported values are: <i>NotActive</i> – Quality control is not active. <i>Check</i> – Check the quality and register. <i>Waste</i> – Check the quality and register. A component failing the test is waste. <i>StopNoWaste</i> – Check the quality and register. Device will stop after the defined number of consecutive errors. The error will be corrected, (e.g., manually). <i>StopWaste</i> – Check the quality and register. A component failing the test is waste, and the device will stop after the defined number of consecutive errors.
<i>IncorrectComponents ?</i>	integer	Number of consecutive incorrect components until the device stops.
<i>DoubleFeedQuality ?</i>	enumeration	Defines the operation of the double feed quality control. For a list of supported values see: <i>IncorrectComponentQuality</i> .
<i>DoubleFeeds ?</i>	integer	Number of consecutive double feeds until the device stops.
<i>BadFeedQuality ?</i>	enumeration	Defines the operation of the bad feed quality control. For a list of supported values see: <i>IncorrectComponentQuality</i> .
<i>BadFeeds ?</i>	integer	Number of consecutive bad feeds until the device stops.

## CollatingItem Element Structure

Name	Data Type	Description
<i>Amount</i> = "1"	integer	Determines, how many consecutive items shall be consumed.
<i>BundleDepth</i> ?	integer	In case of nested bundles with <i>BundleType</i> = "Stack", this parameter addresses the element to be consumed within the "tree" of such bundles. If the real bundle depth level ( <i>BundleType</i> = "Stack") is smaller than the value of <i>BundleDepth</i> , individual stack items (i.e., the smallest available level) shall be consumed. If the input component referenced does not contain bundles, then this parameter is ignored.
<i>Orientation</i> ?	Orientation	Named <i>Orientation</i> of the <i>CollatingItem</i> relative to the input coordinate system. For details see Table 2-3 on page 24. Only one of <i>Orientation</i> or <i>Transformation</i> must be specified. This transformation specified here is applied in addition to orientation/transformation specified in the respective resource link.
<i>Transformation</i> ?	matrix	Orientation of the <b>Component</b> respective to the input coordinate system. This <i>Transformation</i> specified here is applied in addition to orientation/transformation specified in the respective resource link. Only one of <i>Orientation</i> and <i>Transformation</i> must be specified. In neither are 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. <i>StackItem</i> – Apply individually to the smallest element on the stack which can be manipulated individually, (e.g., to a single sheet in the case of a stack of sheets). <i>Component</i> – Apply to each single element of a <i>CollateItem</i> individually. <i>CollateItem</i> – apply to an <i>CollateItem</i> as a whole. <b>Note:</b> If <i>Amount</i> = "1", <b>Component</b> and <i>CollateItem</i> are referring to the same object and, therefore, result in the same output.
<b>Component</b> ?	reference	References one of the input components to the process to be (partially) consumed by the <i>CollatingItem</i> element. It is an error to reference components which are not input components of the process.
<b>Media</b> ?	reference	References one of the input media to the process to be consumed by the <i>CollatingItem</i> element. It is an error to reference media, which are not input media of the process.

## Examples:

**Example 1:** For all *CollationItems*, *Orientation* = "Flip0", *TransformationContext* = "StackItem" or "Component". The output would exactly look like before, except that each sheet is now face-down.

*TransformationContext* = "Component" is equivalent in this case, since "StackItems" and "Component" are referring to the same thing — a single sheet.

**Example 2:** For all *CollationItems*, *Orientation* = Flip0, *TransformationContext* = *CollateItem*:

```

■ 3 of comp. A ■■■■■
■ 2 of comp. B ■■■■■
■ 1 of comp. B ■■■■■
■ 2 of comp. A ■■■■■
■ 1 of comp. A ■■■■■

```

(All face down)

**Note:** Most real world devices are processing stack items one by one and hence will hardly support *TransformationContext* = "CollateItem". This requires some kind of buffer for the stack items belonging to a single collating item plus a flipping mechanism for **PrintRoll** process.

## 7.2.65 FileSpec

[Modified in JDF 1.2](#)

Specification of a file or a set of files. If a single **FileSpec** instance specifies a set of files, it must do so using the *FileFormat* and *FileTemplate* attributes to specify a sequence of URLs. Otherwise, each **FileSpec** instance specifies a single file. If that single file is inside a container file (e.g., a Zip file or is compressed or encoded as indicated by *Compression*), the **FileSpec** instance must define a **Container** subelement which defines another **FileSpec** instance that specifies the container file. In such a case, the attributes of each **FileSpec** instance must apply only to the properties of the file at that level.

### Resource Properties

**Resource class:** Parameter  
**Resource referenced by:** **DBMergeParams, LayoutElement, PDLResourceAlias, ScanParams**

**Example Partition:** *Separation*

**Input of processes:** —

**Output of processes:** —

### Resource Structure

Name	Data Type	Description
<i>Application</i> ?	string	Creator application. See <i>AppVersion</i> for the application version number.
<i>AppOS</i> ? <a href="#">Modified in JDF 1.2</a>	string	Operating system of the application that created the file. Possible values include: <i>DG_UX</i> <i>HP_UX</i> <i>IRIX</i> <i>Linux</i> <i>Mac</i> <i>Solaris</i> <i>Windows</i> Additional values may 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 651 for combinations of <i>AppOS</i> and <i>OSVersion</i> values.
<i>AppVersion</i> ? <a href="#">Clarified in JDF 1.2</a>	string	Version of the value of the <i>Application</i> attribute. The following are some examples: "8.1" "8.1 (4331)" "9.0.3 SR3437"
<i>Checksum</i> ? <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.1A</a>	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.

Name	Data Type	Description
<i>Compression</i> = "None" <a href="#">Modified in JDF 1.2</a>	NMTOKEN	Indicates the compression or encoding for the entire file. This is not compression used internally within the file. Possible values include: <i>Base64</i> – A format for encoding arbitrary binary information for transmission by electronic mail. [RFC3548] <i>BinHex</i> – BinHex encoding converts an 8-bit file into a 7-bit format, similar to UUencoding [RFC1741]. <i>Compress</i> – UNIX compression [RFC1977]. <i>Deflate</i> – The file is compressed using Zip public domain compression format [RFC1951]. <i>Gzip</i> – GNU Zip compression technology [RFC1952]. <i>MacBinary</i> – A format that combines the two forks of a Mac file, together with the file information into a single binary data stream, suitable for storage or transferring through non-Mac systems. [macbinary] <i>None</i> – The file is neither compressed nor encoded. <i>UUencode</i> – A set of algorithms for converting files into a series of 7-bit ASCII characters that can be transmitted over the Internet. [uuencode] <i>ZLIB</i> – ZLIB compression [RFC1950].
<i>Disposition</i> ? <a href="#">Deprecated in JDF 1.2</a>	enumeration	Indicates what the device should do with the file when the process that uses this resource as an input resource completes. Possible values include: <i>Unlink</i> – The device should release the file. <i>Delete</i> – The device should attempt to delete the file. <i>Retain</i> – The device should do nothing with the file. In JDF 1.2 and beyond, retention of assets is specified in the <b>Disposition</b> element.
<i>DocumentNaturalLang</i> ?	language	The natural language of the document this <i>FileSpec</i> refers to. If the document contains more than one language, the value is the primary language of the document.
<i>FileFormat</i> ? <a href="#">Clarified in JDF 1.2</a>	string	A formatting string used with the <i>FileTemplate</i> attribute to define a sequence of URLs in a batch process, each of which has the same semantics as the <i>URL</i> attribute. If neither <i>URL</i> nor <i>UID</i> is present, both <i>FileFormat</i> and <i>FileTemplate</i> must be present, unless the resource is a pipe. If either <i>URL</i> or <i>UID</i> is specified, then <i>FileFormat</i> and <i>FileTemplate</i> must not be specified. For more information, see the text following this table.
<i>FileSize</i> ? <a href="#">Modified in JDF 1.2</a>	LongInteger	Size of the file in bytes. The datatype was changed from integer to LongInteger in JDF 1.2.

Name	Data Type	Description
<a href="#">FileTargetDeviceModel ?</a> <a href="#">New in JDF 1.2</a>	string	Identifies the model of the JDF device for which the document was formatted, including manufacturer name, when the file is device-dependent. The value of this attribute must exactly match the IEEE 1284-2000 Device ID string, except the length field must not be specified. See the Microsoft Universal Plug-and-Play [UPNP] section 2.2.6 <i>DeviceId parameter</i> for details. Here is an example showing only the required fields for a PostScript document formatted for a <i>LaserBeam 9</i> :  MANUFACTURER:ACME Co.;COMMAND SET:PS;MODEL:LaserBeam 9; (See [IEEE1284] clause 7.6)  If this attribute is not present, it is assumed that the file is device independent.
<a href="#">FileTemplate ?</a>	string	A template, used with <i>FileFormat</i> , to define a sequence of URLs in a batch process, each of which has the same semantics as the <i>URL</i> attribute. If neither <i>URL</i> nor <i>UID</i> is present, both <i>FileFormat</i> and <i>FileTemplate</i> must be present, unless the resource is a pipe.
<a href="#">FileVersion ?</a> <a href="#">New in JDF 1.1</a>	string	Version of the file referenced by this FileSpec.
<a href="#">MimeType ?</a> <a href="#">Modified in JDF 1.2</a>	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-mt] registered with IANA, that value must be used. [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.  See "FileSpec Attribute Examples for MimeType and MimeTypeVersion Attributes" on page 635 for examples in common use by JDF applications.
<a href="#">MimeTypeVersion ?</a> <a href="#">New in JDF 1.2</a>	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"  See "FileSpec Attribute Examples for MimeType and MimeTypeVersion Attributes" on page 635 for examples in common use by JDF applications.
<a href="#">OSVersion ?</a> <a href="#">Modified in JDF 1.2</a>	string	Version of the operating system specified by <i>AppOS</i> . The IANA Registry provides a list. See "AppOS and OSVersion Attributes" on page 651, for combinations of <i>AppOS</i> and <i>OSVersion</i> values.

Name	Data Type	Description
<a href="#">OverwritePolicy ?</a> <a href="#">New in JDF 1.2</a>	enumeration	Policy that specifies the policy to follow when a file already exists and the <b>FileSpec</b> is used as an output resource. One of: <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 <b>FileSpec</b> 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.
<a href="#">PageOrder ?</a>	enumeration	Indicates the order of pages in the file containing pages. Possible 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.
<a href="#">ResourceUsage ?</a>	NMTOKEN	If an element uses more than one <b>FileSpec</b> subelement, this attribute is used to refer from the parent element to a certain child element of this type, for example, see <b>FormatConversionParams</b> .
<a href="#">SearchDepth ?</a> <a href="#">New in JDF 1.2</a>	integer	Used when <i>ResourceUsage</i> ="SearchPath" to specify the maximum directory depth that will be recursively searched. 0 specifies this directory only, INF specifies an unlimited search.
<a href="#">UID ?</a> <a href="#">New in JDF 1.1</a>	string	Unique internal ID of the referenced file. This attribute is dependent on the type of file that is referenced: PDF – Variable unique identifier in the ID field of the PDF file's trailer. ICC Profile – Profile ID in bytes 84-99 of the ICC profile header. Others – Format specific. If neither <i>URL</i> nor <i>UID</i> is present on an input <b>FileSpec</b> , 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.
<a href="#">URL ?</a> <a href="#">Clarified in JDF 1.2</a>	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 <b>FileSpec</b> , 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 <b>RunList</b> with the <i>Compression</i> attribute without the <i>URL</i> attribute. See [RFC2396] and "Resolving RunList/@Directory and FileSpec/@URL URI references" on page 649 and "FileSpec mimeType, URL, and Compression attributes, and Container subelement" on page 641 for the syntax and examples. For the "file:" URL scheme see also [RFC1738] and [FileURL].

Name	Data Type	Description
<i>UserFileName</i> ?	string	A user-friendly name which may be used to identify the file.
<i>Container</i> ? <a href="#">New in JDF 1.2</a>	element	Specifies the container for this file. When a container <b>FileSpec</b> is pointed to by <b>Container</b> , that <b>FileSpec</b> must not also specify a <i>FileFormat/FileTemplate</i> attribute pair.  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 <i>RunList/@Directory</i> and <i>FileSpec/@URL</i> URI references" on page 649 for details.
<b>Disposition</b> ? <a href="#">New in JDF 1.2</a>	refelement	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 <b>FileSpec</b> must not be deleted by the Device. <b>FileSpec/Disposition</b> takes precedence over <b>RunList/Disposition</b> .
<i>FileAlias</i> *	element	Defines a set of mappings between file names that may occur in the document and URLs (which may refer to external files or parts of a MIME message).

### Structure of the Container Subelement

[New in JDF 1.2](#)

The **Container** specifies the containing file for a **FileSpec**, e.g. a zip file or tar archive. Containers may be specified recursively in their respective child **FileSpecs**.

Name	Data Type	Description
<b>FileSpec</b>	refelement	Link to another <b>FileSpec</b> 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 <b>FileSpec</b> resource). The link value is only to be used for locating that container <b>FileSpec</b> resource. See "Resolving <i>RunList/@Directory</i> and <i>FileSpec/@URL</i> URI references" on page 649 for details.

### Structure of FileAlias Subelement

Name	Data Type	Description
<i>Alias</i>	string	The filename which is expected to occur in the file.
<i>Disposition</i> ? <a href="#">Deprecated in JDF 1.2</a>	enumeration	Indicates what the device should do with the file referenced by this alias when the process that uses this resource as an input resource completes. Possible values are: <i>Unlink</i> – The device should release the file. <i>Delete</i> – The device should attempt to delete the file. <i>Retain</i> – The device should do nothing with the file. In JDF/1.2 and beyond, use <b>FileSpec/Disposition</b> .
<i>MimeType</i> ? <a href="#">Deprecated in JDF 1.2</a>	string	MIME type of the file. In JDF/1.2 and beyond, use <b>FileSpec/@MimeType</b> .
<i>RawAlias</i> ? <a href="#">New in JDF 1.2</a>	hexBinary	Representation of the original 8-bit byte stream of the Alias Name. Used to transport the original byte representation of an Alias name when moving JDF tickets between computers with different locales.
<i>URL</i> ? <a href="#">Deprecated in JDF 1.2</a>	URL	The URL which identifies the file the alias refers to. In JDF/1.2 and beyond, use <b>FileSpec/@URL</b> .
<b>FileSpec</b> ? <a href="#">New in JDF 1.2</a>	refelement	For JDF version 1.2 and beyond, this refolement must be present and reference a <b>FileSpec</b> resource that must contain a <i>URL</i> attribute, and may contain additional properties of the file, (e.g., <b>Disposition</b> , <i>MimeType</i> , <i>MimeTypeVersion</i> , etc.).

## Usage of FileFormat and FileTemplate Attributes to Define a Set of Files

The function defined when using the attributes *FileFormat* and *FileTemplate* is based on the standard C printf() function. (See [K&R].) *FileFormat* is the first argument and *FileTemplate* is a comma-separated list of the additional arguments. *FileTemplate* may contain the following operators: +,-,\*,/,%,(,) which are evaluated using standard C-operator precedence and the variables defined in the following table:

Table 7-4: Predefined variables used in FileTemplate

Name	Description
Acknowledge-Type	Corresponds to the JMF <i>AcknowledgeType</i> in the Acknowledge message. See "Acknowledge" on page 137.
all	Selects all matching elements. Valid only when <b>FileSpec</b> is used as an input resource.
CustomerID	CustomerID.
Date	Current <i>Date</i> in ISO 8601 format.
element	Integer iterator over all elements in a given page. Restarts at 0 for each page.
Generated	System generated file name.
i	Integer iterator over all files produced by this process. 0-based numbering.
input	Local file name of the input file. Valid only when <b>FileSpec</b> is used as an output resource.
jobID	Job ID string.
jobName	<i>DescriptiveName</i> of the Node that is being processed.
jobPartID	JobPartID string.
page	Integer iterator over the page number of a document. This is equivalent to r for the case that each run contains exactly one page.
PartVersion	PartVersion string of a partitioned resource.
r	Integer iterator over all <b>RunList</b> partitions with a partition key of "Run" in an input <b>RunList</b> .
ri	Integer iterator over all indices in an input "Run" of a <b>RunList</b> . This index is equivalent to looping over a RunIndex.
sep	Separation as defined in the separation PartIDKey(s) of a partitioned resource.
SheetName	SheetName string of a partitioned resource.
SignatureName	SignatureName string of a partitioned resource.
surf	Surface string, "Front" or "Back".
SystemRoot	Root of system directory file structure. This token provides an operating system, independent way to refer to the root.
TileX	X coordinate of a Tile.
TileY	Y coordinate of a Tile.
Time	Current <i>Time</i> in ISO 8601 format.

### Example:

```
<FileSpec FileFormat = "file://myserver.mydomain.com/next/%s/%4.i/m%4.i.pdf"
FileTemplate = "JobID,i/100,i%100"/>
... with JobID = "j001" and a RunList defining 2023 created files will iterate 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"
```



## 7.2.66 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 image onto media.

### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>InterpretingParams, LayoutPreparationParams</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>ClipOffset</i> ?	XYPair	Defines the offset (position) of the imaged area in the non-rotated source image when <i>SizePolicy</i> is <i>ClipToMaxPage</i> . The values 0.0 0.0 mean that the imaged area starts at the lower left point of the job. If absent, the imaged area is taken from the center of the image source. If <b>FitPolicy</b> is defined in the context of a <b>PageCell</b> , <b>ClipOffset</b> is ignored when <b>PageCell/@ImageShift</b> is specified.
<i>GutterPolicy</i> = "Fixed"	enumeration	Allows printing of NUp grids even if the media size does not match the requirements of the data. One of: <i>Distribute</i> – The gutters may grow or shrink to the value specified in <i>MinGutter</i> . <i>Fixed</i> – The gutters are fixed.
<i>MinGutter</i> ?	XYPair	Minimum width in points of the horizontal and vertical gutters formed between rows and columns of pages of a multi-up sheet layout. The first value specifies the width of all horizontal gutters and the second value specifies the minimum width of all vertical gutters.
<i>RotatePolicy</i> ?	enumeration	Specifies the policy for the device to automatically rotate the image to optimize the fit of the image to the container. <i>NoRotate</i> – Do not rotate. <i>RotateOrthogonal</i> – Rotate by 90° in either direction. <i>RotateClockwise</i> – Rotate clockwise by 90°. <i>RotateCounterClockwise</i> – Rotate counterclockwise by 90°.
<i>SizePolicy</i> ? <a href="#">Modified in JDF 1.1A</a>	enumeration	Allows printing even if the container size does not match the requirements of the data. <i>ClipToMaxPage</i> – The page contents should 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 should be scaled up or down to fit the container. The aspect ratio is maintained. <i>ReduceToFit</i> – The page contents should be scaled down but not scaled up to fit the container. The aspect ratio is maintained. <i>Tile</i> – the page contents should be split into several tiles, each printed on its own surface.

### 7.2.67 Fold

[New in JDF 1.1](#)

Fold describes an individual folding operation of the **Component**.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>FoldingIntent, FoldingParams</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>From</i>	enumeration	Edge from which the page is folded. Possible values are: <i>Front</i> <i>Left</i>
<i>To</i>	enumeration	Direction in which it is folded. Possible values are: <i>Up</i> – Upwards. <i>Down</i> – Downwards.
<i>Travel</i> ? <a href="#">Modified in JDF 1.2</a>	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> ? <a href="#">New in JDF 1.2</a>	double	Relative distance of the reference edge relative to <i>From</i> in the coordinates of the incoming <b>Component</b> . <i>RelativeTravel</i> is always based on the complete size of the input <b>Component</b> 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 <b>Component</b> .

### 7.2.68 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 25.

The specification of reference edges (e.g., *Front*, *Rear*, *Left*, and *Right*) for the description of an operation (e.g., the positioning of a tool) done by means of determined names. These names are case-sensitive. They must be written exactly as shown in Figure 7.8, below.

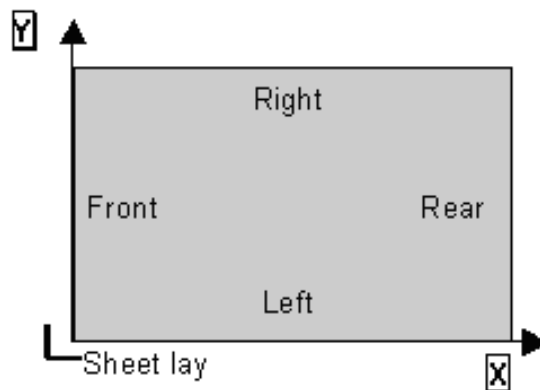


Figure 7.8: Names of the reference edges of a sheet in the *FoldingParams* resource

## Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>BlockName, RibbonName, SheetName, SignatureName, WebName</i>
Input of processes:	<b>Folding</b>
Output of processes:	—

## Resource Structure

Name	Data Type	Description
<i>DescriptionType</i> ? <a href="#">Deprecated in JDF 1.2</a>	enumeration	How the folding operations are described. Possible values are: <i>FoldProc</i> – Detailed description of each individual fold. <i>FoldCatalog</i> – Selection of fold procedure from <i>FoldCatalog</i> . In JDF 1.2 and beyond, the <i>FoldCatalog</i> defines the topology of the folding scheme. The specifics of each individual <b>Fold</b> may be described using <b>Fold</b> elements. If both <i>FoldCatalog</i> and <b>Fold</b> are specified, <b>Fold</b> takes precedence
<i>FoldCatalog</i> ? <a href="#">Clarified in JDF 1.2</a>	string	Describes the type of fold according to the folding catalog in the format “ <i>Fx-y</i> ” as shown in Table 7.9 and Table 7.10 on page 369. Imposition folds define finished pages. Thus, a sheet with a “ <i>F6-2</i> ” Z-fold is comprised of six (6) finished pages.
<i>FoldSheetIn</i> ? <a href="#">Deprecated in JDF 1.1</a>	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 <b>Component</b> , all coordinates of the folding procedure are scaled accordingly. The scaling factors in <i>X</i> and <i>Y</i> direction may differ. <b>Implementation Note:</b> This attribute should always match the <i>Size</i> attribute of the input <b>Component</b> , which is the default.
<i>SheetLay</i> = “ <i>Left</i> ”	enumeration	Lay of input media. Possible values are: <i>Left</i> <i>Right</i>
<b>Fold</b> * <a href="#">New in JDF 1.1</a> <a href="#">Clarified in JDF 1.2</a>	element	Describes the folding operations in the sequence in which they should be carried out. It is recommended to specify a set of subsequent <b>Fold</b> operations as multiple <b>Fold</b> elements in one <b>Folding</b> procedure, rather than specifying a <b>Combined</b> process that combines multiple <b>Folding</b> processes. If both <i>FoldCatalog</i> and <b>Fold</b> elements are specified, the <b>Fold</b> 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> ”.
<b>FoldOperation</b> * <a href="#">Deprecated in JDF 1.1</a>	element	Abstract element that describes the folding operations in the sequence in which they should be carried out. Replaced by the explicit element <b>Fold</b> * in JDF 1.1 and beyond.



Figure 7.9: Fold Catalog part 1

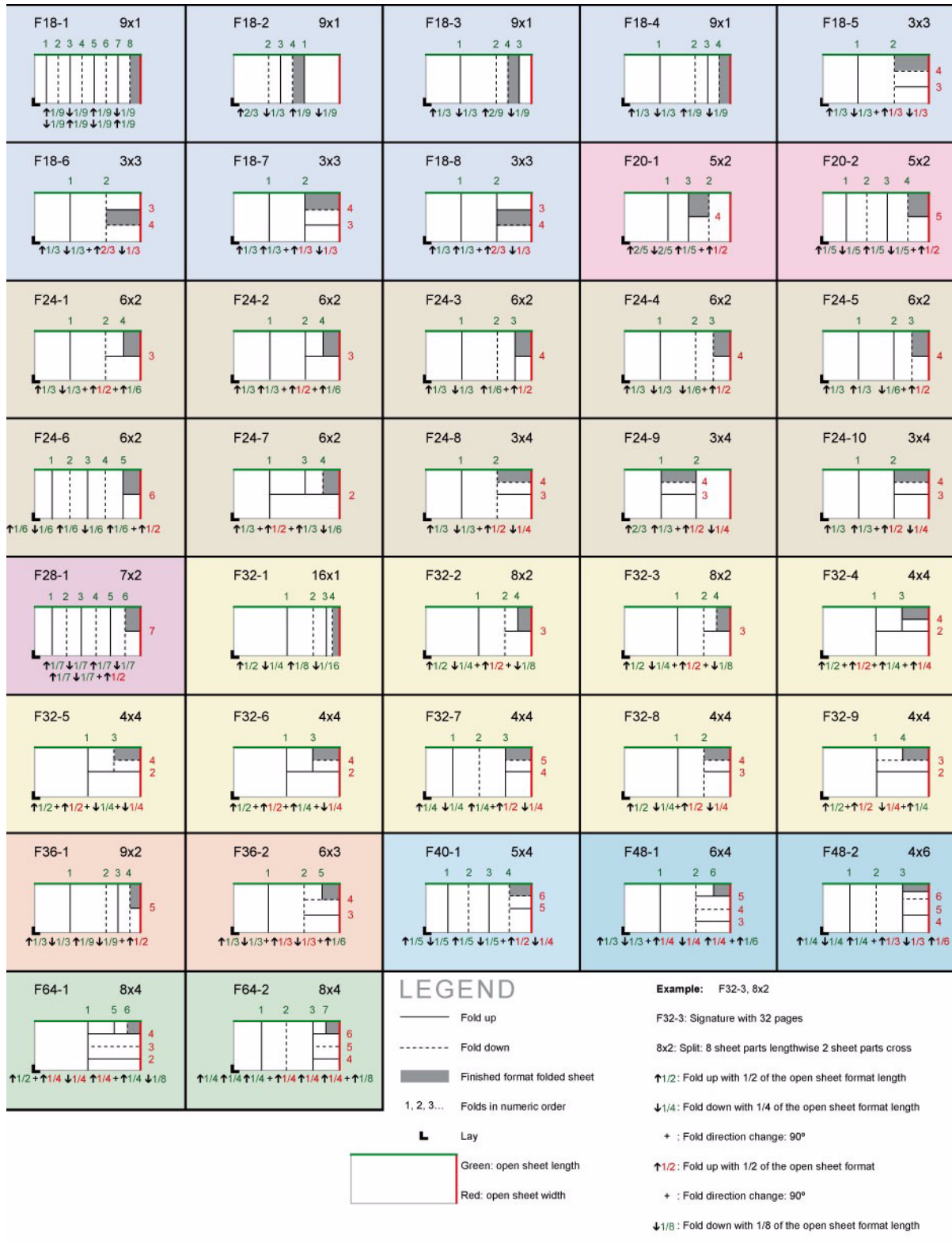


Figure 7.10: Fold Catalog part 2

## 7.2.69 FontParams

This resource describes how fonts must be handled when converting PostScript files to PDF.

### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>DocIndex, RunIndex, RunTags, SheetName, Side, SignatureName</i>
<b>Input of processes:</b>	<b>PDFToPSConversionParams</b>
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
<i>AlwaysEmbed</i> ?	NMTOKENS	One or more names of fonts that are always to be embedded in the PDF file. Each name must be the PostScript language name of the font. An entry that occurs in both the <i>AlwaysEmbed</i> and <i>NeverEmbed</i> lists constitutes an error.
<i>CannotEmbedFontPolicy</i> = "Warning"	enumeration	Determines what occurs when a font cannot be embedded. Possible values are: <i>Error</i> – Log an error and abort the process if any font can not be found or embedded. <i>Warning</i> – Warn and continue if any font cannot be found or embedded. <i>OK</i> – Continue without warning or error if any font can not be found or embedded.
<i>EmbedAllFonts</i> = "false"	boolean	If "true", specifies that all fonts, except those in the <i>NeverEmbed</i> list, are to be embedded in the PDF file.
<i>MaxSubsetPct</i> ?	integer	The maximum percentage of glyphs in a font that can be used before the entire font is embedded instead of a subset. This value is only used if <i>SubsetFonts</i> = "true".
<i>NeverEmbed</i> ?	NMTOKENS	One or more names of fonts that are never to be embedded in the PDF file. Each name must be the PostScript language name of the font. An entry that occurs in both the <i>AlwaysEmbed</i> and <i>NeverEmbed</i> lists constitutes an error.
<i>SubsetFonts</i> ?	boolean	If "true", font subsetting is enabled. If "false", it is not. Font subsetting embeds only those glyphs that are used, instead of the entire font. This reduces the size of a PDF file that contains embedded fonts. If font subsetting is enabled, the decision whether to embed the entire font or a subset is determined by number of glyphs in the font that are used and the value of <i>MaxSubsetPct</i> . <b>Note:</b> Embedded instances of multiple master fonts are always subsetted, regardless of the setting of <i>SubsetFonts</i> .

## 7.2.70 FontPolicy

This resource defines the policies that devices must follow when font errors occur while PDL files are being processed. When fonts are referenced by PDL files but are not provided, devices may provide one of the following two fallback behaviors:

- 1 The device may provide a standard default font which is substituted whenever a font cannot be found.
- 2 The device may provide 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. **FontPolicy** allows jobs to specify whether or not either of these fallback behaviors should be employed when missing fonts occur.

## Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>DocIndex, RunIndex, RunTags, SheetName, Side, SignatureName</i>
Input of processes:	<b>IDPrinting, Interpreting, Trapping</b>
Output of processes:	—

## Resource Structure

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 required font if a font cannot be found.

### 7.2.71 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:	<b>FormatConversion</b>
Output of processes:	—

## Resource Structure

Name	Data Type	Description
<b>FileSpec ?</b> ( <i>InputFormat</i> ) <a href="#">Deprecated in JDF 1.2</a>	refelement	The format of the original file is specified in a <b>FileSpec</b> with <i>ResourceUsage = InputFormat</i> . No URL should be specified because the list of files is given by the input <b>RunList</b> of the <b>FormatConversion</b> 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 <b>FileSpec</b> of the input <b>RunList</b> of the <b>FormatConversion</b> process.
<b>FileSpec ?</b> ( <i>OutputFormat</i> ) <a href="#">Deprecated in JDF 1.2</a>	refelement	The format of the converted file is specified in a <b>FileSpec</b> with <i>ResourceUsage = OutputFormat</i> . No URL should be specified because the list of files is given by the output <b>RunList</b> of the <b>FormatConversion</b> 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 <b>FileSpec</b> of the output <b>RunList</b> .
TIFFFormatParams ? <a href="#">New in JDF 1.2</a>	element	Parameters specific to conversion of rasters to TIFF files. (See below.) <b>FormatConversion</b> should not be used to convert non-raster files to TIFF. The appropriate <b>Interpreting</b> and <b>Rendering</b> processes should be used first.

Name	Data Type	Description
<b>ImageCompressionParams ?</b> <a href="#">New in JDF 1.2</a>	reference	Provides a set of controls that determines how images will be down-sampled and compressed in the converted documents
<b>ColorPool ?</b> <a href="#">New in JDF 1.2</a>	reference	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 contour 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.

### Structure of the TIFFFormatParams element

[New in JDF 1.2](#)

Name	Data Type	Description
<b>ByteOrder ?</b>	enumeration	Byte order of the TIFF file. Possible values are: <i>II</i> – Low byte first. <i>MM</i> – high byte first. The identifiers have been selected to match the identifier with the same purpose within the TIFF file itself.
<b>Interleaving = "1"</b>	integer	How the components of each pixel are stored. The values are taken from TIFF tag 284— <i>PlanarConfiguration</i> : <i>1</i> – “Chunky” format, which is pixel interleaved. <i>2</i> – “Planar” format, which is strip interleaved.
<b>WhiteIsZero = "true"</b>	boolean	When writing monochrome or grayscale files, this flag indicates whether the data should be written as “WhiteIsZero” or “BlackIsZero.”
<b>Segmentation ?</b>	enumeration	How the image data are segmented. Possible values are: <i>SingleStrip</i> – all data are included in one segment. This is encoded in the TIFF file by setting <i>RowsPerStrip</i> to a number equal to or larger than the number of pixel rows in the image. <i>Stripped</i> – Data are segmented into strips. <i>Tiled</i> – Data are segmented into tiles.
<b>RowsPerStrip ?</b>	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 <b>ByteMap</b> to TIFF, <b>ByteMap/@BandHeight</b> should be used as the default.
<b>TileSize ?</b>	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> .



Name	Data Type	Description
<i>SeparationNameTag</i> = "270"	integer	When color separations are stored in individual TIFF files it is often useful to mark each with the name of the colorant that it represents, but there is no universally accepted way to do this. In order to avoid the need for explicit partitioning, the tag to be used to encode the separation name (as a string) can be entered here as the TIFF tag number.  If the same TIFF tag number is also supplied as a TIFFtag sub element, then the TIFFtag element takes priority over <i>SeparationNameTag</i> .  The tag should only be filled in the resulting TIFF files if the name of the separation is known, (e.g., from a <b>ColorPool</b> resource supplied to <b>FormatConversion</b> , or because the <b>FormatConversion</b> process forms a part of a compound process with a Separation process). The default of "270" is the <i>TIFF</i> ImageDescription tag.
TIFFtag *	element	Specific tag values for inclusion in the TIFF file.
TIFFEmbeddedFile *	element	Files to be embedded within the created TIFF file. These might include an ICC profile, XMP data, etc.

The number of channels should be derived from the raster data to be converted.

When PhotometricInterpretation = 5 and InkSet = 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.

### Structure of the TIFFtag Subelement

[New in JDF 1.2](#)

Name	Data Type	Description
<i>TagNumber</i>	integer	Tag number of the required 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.)
<i>IntegerValue</i> ?	IntegerList	If the type of the tag should be BYTE, SHORT, LONG, SBYTE, SSHORT, or SLONG, then <i>IntegerValue</i> should be used to encode that data
<i>NumberValue</i> ?	DoubleList	If the type of the tag should be RATIONAL, SRATIONAL, FLOAT, or DOUBLE, then <i>NumberValue</i> must be used to encode that data
<i>StringValue</i> ?	string	If the type of the tag should be ASCII, then <i>StringValue</i> should be used to encode the data.
<i>BinaryValue</i> ?	hexBinary	If the type of the tag should be UNDEFINED, then <i>BinaryValue</i> should be used to encode the data

One and only 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.), but may include informational tags such as OPIProxy, ImageID, Copyright, DateTime, ImageDescription, etc.

## Structure of the TIFEmbeddedFile Subelement

[New in JDF 1.2](#)

Name	Data Type	Description
<i>TagNumber</i>	integer	Tag number of the required 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).
<b>FileSpec</b>	reference	Reference to the file to be embedded.

### 7.2.72 GatheringParams

This resource contains the attributes of the **Gathering** process.

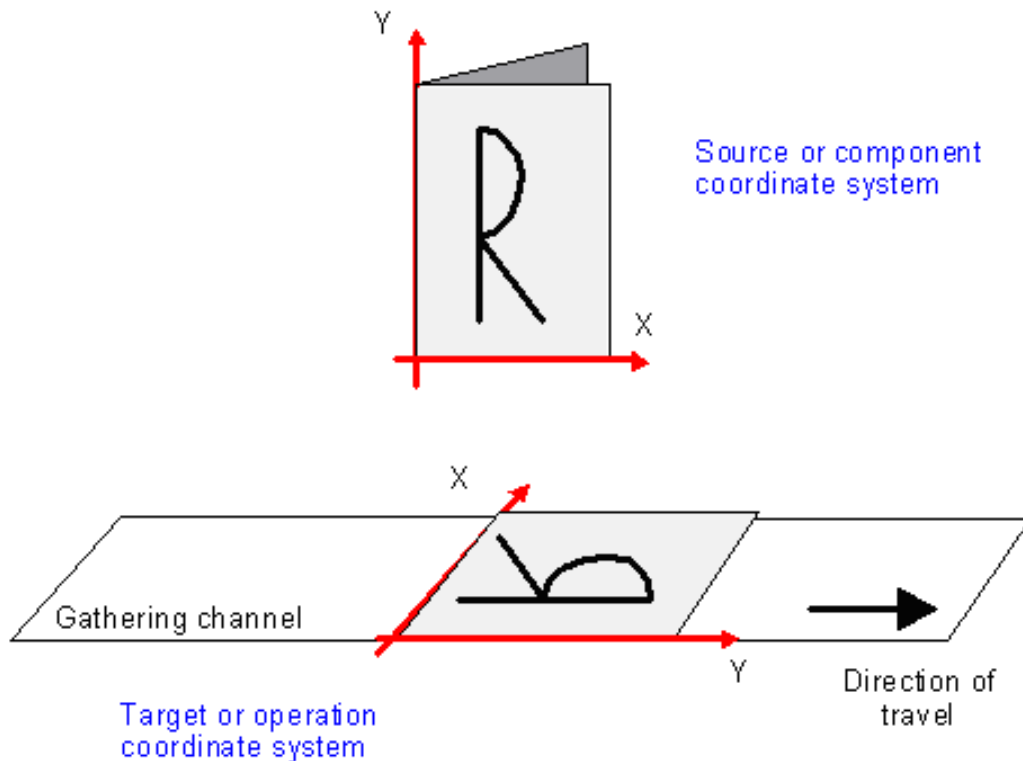


Figure 7.11: Coordinate system used for gathering

### Resource Properties

Resource class: Parameter  
 Resource referenced by: —  
 Input of processes: **Gathering**  
 Output of processes: —

### Resource Structure

Name	Data Type	Description
<b>Disjointing ?</b>	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.

## 7.2.73 GlueApplication

[New in JDF 1.1](#)

This resource specifies glue application in hard and soft cover book production.

### Resource Properties

Resource class: Parameter  
 Resource referenced by: **CoverApplicationParams, GluingParams, SpineTapingParams**  
 Input of processes: —  
 Output of processes: —

### Resource Structure

Name	Data Type	Description
<i>GluingTechnique</i>	enumeration	Type or technique of gluing application. Possible values are: <i>SpineGluing</i> <i>SideGluingFront</i> <i>SideGluingBack</i>
<b>GlueLine</b>	reference	Structure of the glue line.

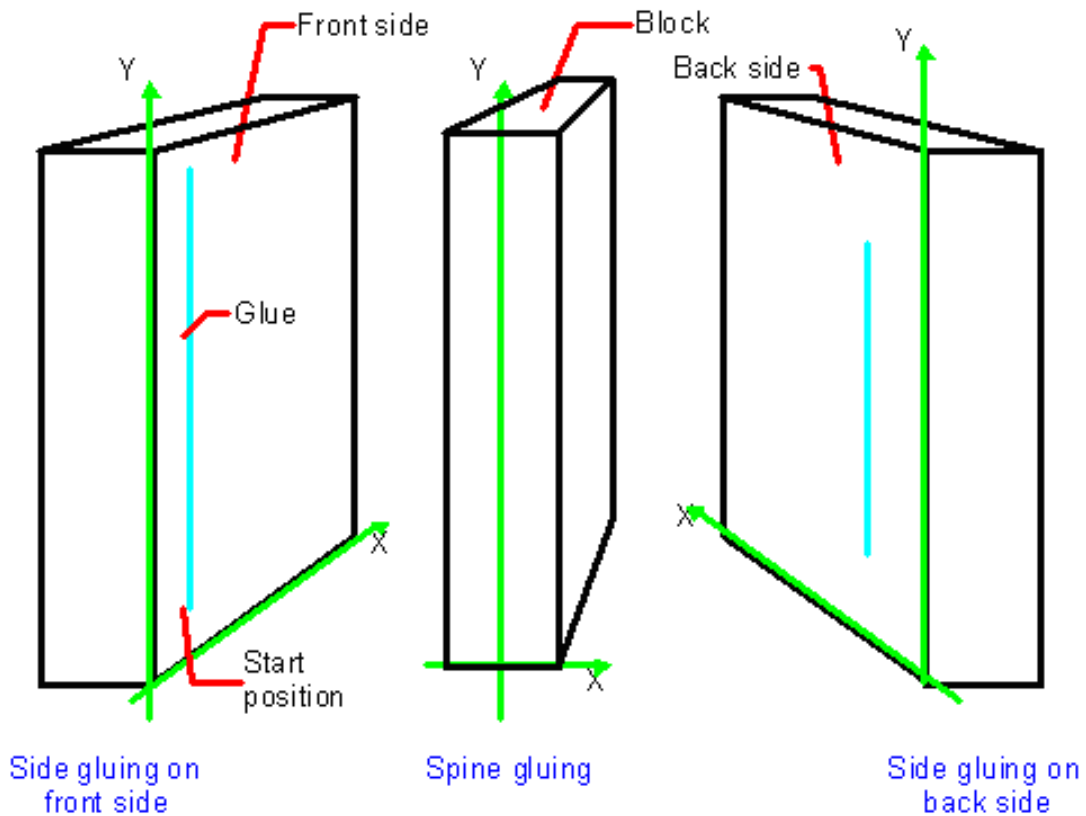


Figure 7.12: Parameters and coordinate system for glue application

## 7.2.74 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:	—
Input of processes:	<b>Gluing</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
Glue *	element	Definition of one or more Glue line applications.

### Properties of the Glue Element

The Glue element describes how to apply a line of glue.

Name	Data Type	Description
<i>WorkingDirection</i>	enumeration	Direction from which the tool is working. Possible values are: <i>Top</i> – From above. <i>Bottom</i> – From below.
<b>GlueApplication</b>	refelement	Describes the glue application.

## 7.2.75 GlueLine

This resource provides the information to determine where and how to apply glue.

### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>CaseMakingParams, EndSheetGluingParams, FoldingParams, CoverApplicationParams, InsertingParams, SpineTapingParams, ThreadSealingParams</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>AreaGlue</i> = "false" <a href="#">New in JDF 1.1</a>	boolean	Specifies that this GlueLine should cover the complete width of the <b>Component</b> 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 hole width.
<i>GluingPattern</i> ?	XYPair	Glue line pattern defined by the length of a glue line segment (X element) and glue line gap (Y element). A solid line is expressed by the pattern (1 0).

Name	Data Type	Description
<i>GlueType</i> ?	enumeration	Glue type. Possible values are: <i>ColdGlue</i> – Any type of glue that needs no heat treatment. <i>Hotmelt</i> – Hotmelt EVA (Ethyl-Vinyl-Acetate-Copolymer) <i>PUR</i> – Polyurethane
<i>MeltingTemperature</i> ?	integer	Required temperature for melting the glue, in degrees centigrade. Used only when <i>GlueType</i> = " <i>Hotmelt</i> " or <i>GlueType</i> = " <i>PUR</i> ".
<i>RelativeStartPosition</i> ? <a href="#">New in JDF 1.2</a>	XYPair	Relative starting position of the tool. <i>RelativeStartPosition</i> is always based on the complete size of the input <b>Component</b> 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 <b>Component</b> .
<i>RelativeWorkingPath</i> ? <a href="#">New in JDF 1.2</a>	XYPair	Relative working path of the tool beginning at <i>RelativeStartPosition</i> . <i>RelativeWorkingPath</i> is always based on the complete size of the input <b>Component</b> 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 <b>Component</b> .
<i>StartPosition</i> ? <a href="#">Modified in JDF 1.2</a>	XYPair	Start position of glue line. The start position is given in the coordinate system of the mother sheet. If both <i>StartPosition</i> and <i>RelativeStartPosition</i> are specified, <i>RelativeStartPosition</i> is ignored.
<i>WorkingPath</i> ? <a href="#">Modified in JDF 1.2</a>	XYPair	Relative working path of the gluing tool. If both <i>WorkingPath</i> and <i>RelativeWorkingPath</i> are specified, <i>RelativeWorkingPath</i> is ignored.

## 7.2.76 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:	<b><i>HeadBandApplication</i></b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>BottomBrand</i> ?	string	Bottom head band brand. If not specified, defaults to the value of <i>TopBrand</i> .
<i>BottomColor</i> ?	NamedColor	Color of the bottom head band. If not specified, defaults to the value of <i>TopColor</i> .
<i>BottomLength</i> ?	double	Length of the carrier material of the bottom head band along binding edge. If not specified, both head bands are on one carrier.
<i>TopBrand</i> ?	string	Top head band brand.
<i>TopColor</i> ?	NamedColor	Color of the top head band.
<i>TopLength</i> ?	double	Length of carrier material of the top head band along binding edge. If not specified, both head bands are on one carrier which has the length of the book block.

Name	Data Type	Description
<i>StripMaterial</i> ?	enumeration	Strip material. Possible values are: <i>Calico</i> <i>Cardboard</i> <i>CrepePaper</i> <i>Gauze</i> <i>Paper</i> <i>PaperlinedMules</i> <i>Tape</i>
<i>Width</i> ?	double	Width of the head bands and carrier.
<b>GlueLine</b> *	reference	The carrier may be applied to the bookblock with glue. The coordinate system for the <b>GlueLine</b> is defined in the Section 7.2.62, EndSheetGluingParams.

### 7.2.77 Hole

The Hole element describes an individual hole.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>HoleLine, HoleMakingIntent, HoleMakingParams</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>Center</i>	XYPair	Position of the center of the hole relative to the <b>Component</b> coordinate system. For more information, see Section 6.6.48.2, 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> <a href="#">Modified in JDF 1.1</a>	enumeration	Shape of the hole. Possible values are: <i>Elliptic</i> <i>Round</i> <i>Rectangular</i>

### 7.2.78 HoleLine

[New in JDF 1.1](#)

Line hole punching generates a series of holes with identical distance (pitch) running parallel to the edge of a web, which is mainly used to transport paper through continuous-feed printers and finishing devices (form processing). The final product typically is a web with two lines of holes, one at each edge of the web. The parameters for one line of holes are specified in the **HoleLine** element. The distance between holes within each line of holes is identical (constant pitch).

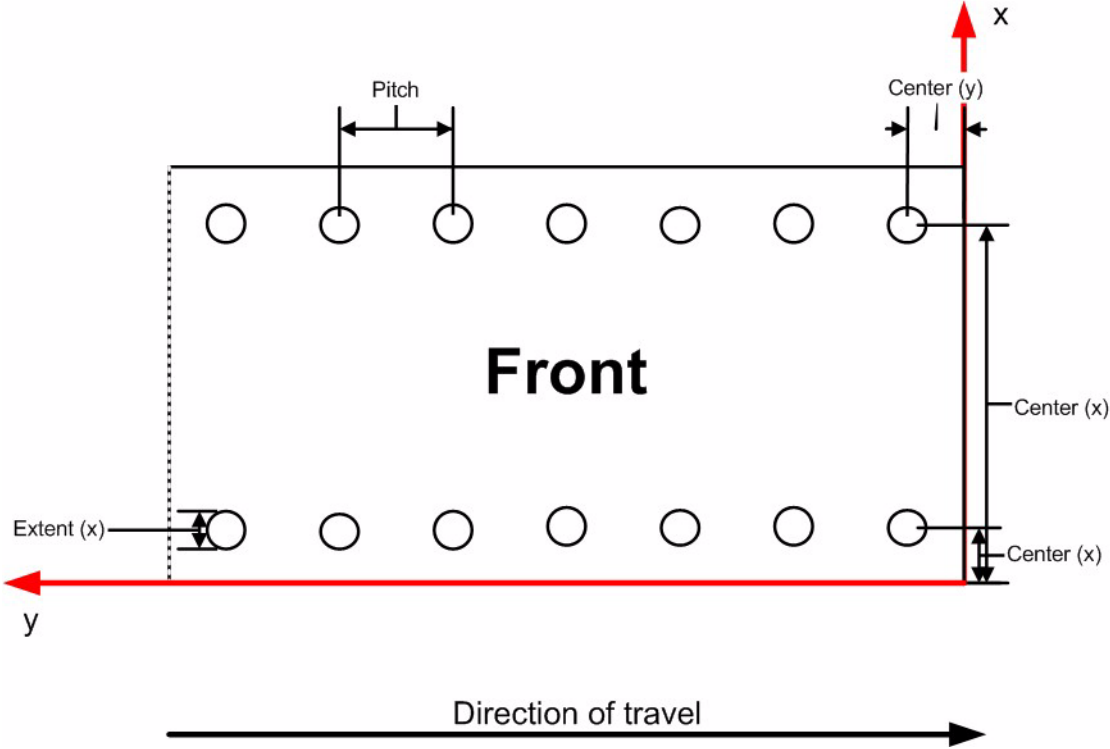
#### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>HoleMakingIntent, HoleMakingParams</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

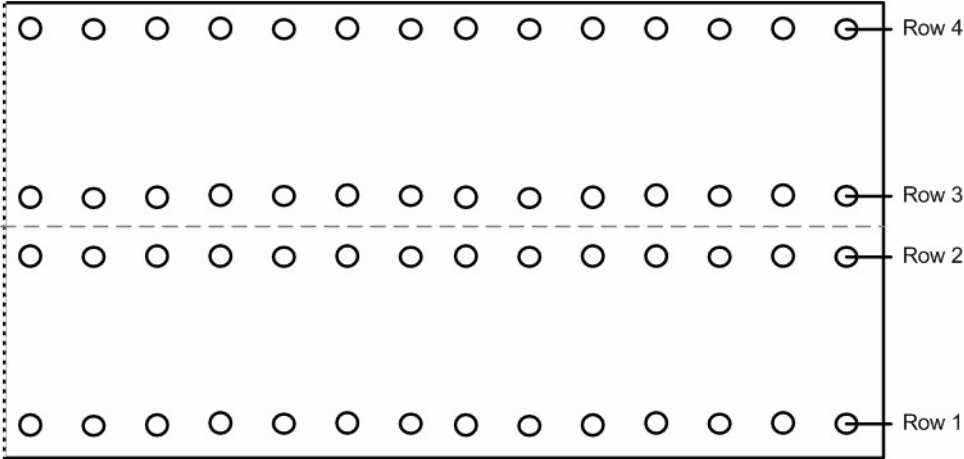
The parameters of the HoleLine element are:

**Resource Structure**

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.



However, sometimes Line Hole Punching is performed for multiple webs before dividing the web after the HoleMaking process as illustrated below:



### 7.2.79 HoleList

[Modified in JDF 1.2](#)

This resource is used to describe holes or rows of holes in intent resources. Note that it was an intent resource sub element prior to JDF 1.2.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>BindingIntent, HoleMakingIntent</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

#### Resource Structure.

Name	Data Type	Description
<b>Hole *</b> <a href="#">Modified in JDF 1.1</a>	refelement	Description of individual holes. See Section 7.2.77, Hole.
<b>HoleLine *</b> <a href="#">New in JDF 1.1</a>	refelement	Array of all HoleLine elements. See Section 7.2.78, HoleLine.

### 7.2.80 HoleMakingParams

This resource specifies where to make a hole of what shape in components. This information is used by the **HoleMaking** process.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>SheetName, SignatureName</i>
Input of processes:	<b>HoleMaking</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<b>Center ?</b> <a href="#">Modified in JDF 1.1</a>	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> .
<b>CenterReference =</b> <i>"TrailingEdge"</i> <a href="#">New in JDF 1.1</a>	enumeration	Defines the reference coordinate system for <i>Center</i> . One of: <i>TrailingEdge</i> – Physical coordinate system of the component. <i>RegistrationMark</i> – The center is relative to a registration mark.
<b>Extent ?</b>	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> .
<b>HoleCount ?</b> <a href="#">New in JDF 1.2</a>	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. <sup>a</sup> The first entry defines the number of holes, the second entry defines the number of spaces, and consecutive entries alternately define holes (h) and spaces (s), for instance: "2 2 2" = "h h s s h h". "0 3 3 3 3" = "s s s h h h s s s h h h".



Name	Data Type	Description
<a href="#">HoleReferenceEdge ?</a> <a href="#">New in JDF 1.1</a> <a href="#">Deprecated in JDF 1.2</a>	enumeration	The edge of the media relative to where the holes should be punched. Use with <i>HoleType</i> . Possible values are: <i>Left</i> <i>Right</i> <i>Top</i> <i>Bottom</i> <i>Pattern</i> – Specifies that the reference edge implied by the value of <i>HoleType</i> in "JDF/CIP4 Hole Pattern Catalog" on page 663 is used. The default if <i>HoleType</i> is <i>Explicit</i> , otherwise <i>Left</i> . <i>HoleReferenceEdge</i> has been replaced with an explicit <i>Transformation</i> or <i>Orientation</i> of the input <b>Component</b> . If both <i>Transformation</i> or <i>Orientation</i> and <i>HoleReferenceEdge</i> are specified, the result is the matrix product of both transformations. <i>Transformation</i> or <i>Orientation</i> must be applied first.
<a href="#">HoleType</a> <a href="#">New in JDF 1.1</a>	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). For details of hole types and a list of additional allowed values, refer to "JDF/CIP4 Hole Pattern Catalog" on page 663. Values are: <i>Explicit</i> – Holes are defined in an array of Hole elements. Additional values defined in "JDF/CIP4 Hole Pattern Catalog" on page 663 <a href="#">The following values are deprecated from JDF 1.0</a> <i>2HoleEuro</i> – Replaced by either <i>R2m-DIN</i> or <i>R2m-ISO</i> . <i>3HoleUS</i> – Replaced by <i>R3I-US</i> . <i>4HoleEuro</i> – Replaced by either <i>R4m-DIN-A4</i> or <i>R4m-DIN-A5</i> .
<a href="#">Shape ?</a> <a href="#">Modified in JDF 1.1</a>	enumeration	Shape of the holes if <i>HoleType</i> is not <i>Explicit</i> . Possible values are: <i>Elliptic</i> <i>Round</i> <i>Rectangular</i> If not specified, it defaults to the value implied by <i>HoleType</i> .
<b>Hole *</b>	element	Description of individual <b>Hole</b> elements.
<a href="#">HoleLine *</a> <a href="#">New in JDF 1.1</a>	element	Description of <b>HoleLine</b> elements.
<a href="#">RegisterMark ?</a> <a href="#">New in JDF 1.1</a>	refelement	Reference to the registration mark that defines the coordinate system origin for <b>HoleMaking</b> .

- a. **Application Note:** For dealing with the “default” case, intelligent systems will take into consideration job parameters like the length of the binding edge or distance of holes to the paper edges to calculate the appropriate number of holes. For production of the holes and selection/production of the matching binding element, the “system specified” values need to match 100% between the **HoleMaking** and the **Binding** process for obvious reasons. In practice, if no details are specified for **HoleMaking**, they should also be absent for **Binding**. In this case, either the operator provides the missing value when setting up the binding device for the job, or the device itself needs to have some kind of automatic hole detection mechanism.

### 7.2.81 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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>Disjointing, Sheet, Surface</b> , and any physical resource
<b>Example Partition:</b>	—
<b>Input of processes:</b>	<b>Collecting, Gathering, Inserting, Verification</b>
<b>Output of processes:</b>	—

#### Resource Structure

Name	Data Type	Description
<i>BoundingBox</i> ?	rectangle	Box that provides the boundaries in the coordinate system of the mark that indicates where the component is to be placed. If no <i>BoundingBox</i> is defined, the complete visible surface must be scanned for an appropriate bar code.
<i>Encoding</i>	enumeration	Encoding of the information. Possible values are: <i>ASCII</i> – Plain-text font. <i>BarCode1D</i> – One-dimensional bar code. <i>BarCode2D</i> – Two-dimensional bar code.
<i>EncodingDetails</i>	NMTOKEN	Details about the encoding type. An example is the bar code scheme. Possible values are: <i>Code39</i> <i>Interleave25</i> <i>Plessey</i> <i>EAN</i>
<i>Format</i> ? <a href="#">Modified in JDF 1.2</a>	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.
<i>Orientation</i> ?	matrix	Orientation of the contents within the <b>IdentificationField</b> . The coordinate system is defined in the system of the sheet or component where the <b>IdentificationField</b> resides.
<i>Position</i> ?	enumeration	Position with respect to the instance document or physical resource to which the resource refers. Possible values are: <i>Header</i> – Sheet before the document. <i>Trailer</i> – Sheet after the document. <i>Page</i> – A page of the document. <i>Top</i> – The top of the resource. <i>Bottom</i> – The bottom of the resource. <i>Left</i> – The left side of the resource. <i>Right</i> – The right side of the resource. <i>Front</i> – The front side of the resource. <i>Back</i> – The back side of the resource. <i>Any</i> – <a href="#">Deprecated in JDF 1.2</a>
<i>Page</i> ?	integer	If <i>Position</i> = <i>Page</i> , this refers to the page where the <b>IdentificationField</b> can be found. Negative values denote an offset relative to the last page in a stack of pages.

Name	Data Type	Description
<i>Purpose ?</i>	enumeration	Purpose defines the usage of the field. Possible values are: <i>Label</i> – Used to mark a product or component. <i>Separation</i> – Used to separate documents. <i>Verification</i> – Used for verification of documents.
<i>Value ?</i> <a href="#">New in JDF 1.1</a>	string	Fixed value of the <b>IdentificationField</b> , (e.g., on a label).

### 7.2.82 IDPrintingParams

[Deprecated in JDF 1.1](#) See "IDPrintingParams" on page 755 for details of this deprecated resource.

### 7.2.83 ImageCompressionParams

[Modified in JDF 1.2](#)

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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>Sheet</b>
<b>Example Partition:</b>	<i>DocIndex, RunIndex, RunTags, SheetName, Side, SignatureName</i>
<b>Input of processes:</b>	<b>ImageReplacement</b>
<b>Output of processes:</b>	—

#### Resource Structure

Name	Data Type	Description
ImageCompression *	element	Specifies how images are to be compressed.

#### Structure of ImageCompression Subelement

Name	Data Type	Description
<i>AntiAliasImages = "false"</i>	boolean	If " <i>true</i> ", anti-aliasing is permitted on images. If " <i>false</i> ", 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 down-sampling. Anti-aliasing is only performed if the image is actually down-sampled and if <i>ImageDepth</i> has a value greater than the number of bits per color component in the input image.
<i>AutoFilterImages = "true"</i> <a href="#">Modified in JDF 1.2</a>	boolean	Used only if <i>EncodeImages</i> is " <i>true</i> ". This attribute is not used if <i>ImageType = Monochrome</i> . If " <i>true</i> ", the filter defined by <i>ImageAutoFilterStrategy</i> is applied to photos and the <i>FlateEncode</i> filter is applied to screen shots. If " <i>false</i> ", the <i>ImageFilter</i> compression method is applied to all images.
<i>ConvertImagesToIndexed ?</i>	boolean	If " <i>true</i> ", 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 = Color</i> .

Name	Data Type	Description
<i>DCTQuality</i> = "0"	double	A value between 0 and 1 that indicates "how much" the process should compress images when using a <i>DCTEncode</i> filter. 0.0 means "do as lossless compression as possible." 1.0 means "do the maximum compression possible."
<i>DownsampleImages</i> = "false" <a href="#">Modified in JDF1.1A</a>	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> ? <a href="#">Deprecated in JDF 1.1</a>	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" <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF1.1A</a>	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> ? <a href="#">New in JDF 1.2</a>	NMTOKEN	Selects what image compression strategy to employ if passing through an image that is not already compressed. Possible values include: <i>JPEG</i> – Lossy JPEG compression for low-frequency images and lossless Flate compression for high-frequency images. <i>JPEG2000</i> – Lossy JPEG2000 compression for low-frequency images and lossless JPEG2000 compression for high-frequency images.
<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 may 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 $(72 * 1.5) = 108\text{dpi}$
<i>ImageDownsampleType</i> ?	enumeration	Downsampling algorithm for images. Possible values are: <i>Average</i> – The program averages groups of samples to get the new downsampled value. <i>Bicubic</i> – The program uses bicubic interpolation on a group of samples to get a new downsampled value. <i>Subsample</i> – The program picks the middle sample from a group of samples to get the new downsampled value.

Name	Data Type	Description
<a href="#">ImageFilter ?</a> <a href="#">Modified in JDF 1.2</a>	NMTOKEN	Specifies the compression filter to be used for images. Ignored if <i>AutoFilterImages</i> = "true" or if <i>EncodeImages</i> = "false". Possible values are: <i>CCITTFaxEncode</i> – Used to select CCITT Group 3 or 4 facsimile encoding. Used only if <i>ImageType</i> = <i>monochrome</i> . <i>DCTEncode</i> – Used to select JPEG compression. <i>FlateEncode</i> – Used to select ZIP compression. <i>JPEG2000</i> – Used to select JPEG2000/Wavelet compression. <a href="#">New in JDF 1.2</a> <i>LZWEncode</i> – LZW Compression. <i>PackBits</i> – A simple byte-oriented run length scheme. In JDF 1.1 and below, the data type was enumeration. It has been extended to NMTOKEN in order to allow for extensions.
<a href="#">ImageResolution ?</a>	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 downsamples only images that are above that resolution to that actual resolution.
<a href="#">ImageType</a>	enumeration	Specifies the kind of images that are to be manipulated. Possible values are: <i>Color</i> <i>Grayscale</i> <i>Monochrome</i>
<a href="#">JPXQuality ?</a> <a href="#">New in JDF 1.2</a>	integer	Specifies the required 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.
<a href="#">CCITTFaxParams ?</a> <a href="#">New in JDF 1.2</a>	element	The equivalent of the PostScript <i>Rows</i> and <i>BlackIs1</i> parameters, which are implicit in the raster data to be compressed.
<a href="#">DCTParams ?</a> <a href="#">New in JDF 1.2</a>	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.
<a href="#">FlateParams ?</a> <a href="#">New in JDF 1.2</a>	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.
<a href="#">LZWParams ?</a> <a href="#">New in JDF 1.2</a>	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

### Structure of the CCITTFaxParams Element

[New in JDF 1.2](#)

Name	Data Type	Description
<i>Uncompressed</i> = "false"	boolean	A flag to indicate whether the file generated may use uncompressed encoding when advantageous.
<i>K</i> = "0"	integer	An integer that selects the encoding scheme to be used. <0 – Pure two-dimensional encoding (Group 4, TIFF Compression = 4) 0 – Pure one-dimensional encoding (Group 3, 1-D, TIFF Compression = 2) >0 – Mixed one- and two-dimensional encoding (Group 3, 2-D, TIFF Compression = 3), in which a line encoded one-dimensionally can be followed by at most K - 1 lines encoded two-dimensionally

Name	Data Type	Description
<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

### Structure of the DCTParams Element

[New in JDF 1.2](#)

Name	Data Type	Description
<i>SourceCSs</i>	enumeration	<p>Identifies which of the incoming color spaces will be operated on. Possible values are:</p> <p><i>Calibrated</i>—Operates on CalGray and CalRGB color spaces. New in JDF 1.2</p> <p><i>CIEBased</i>—Operates on CIE-Based color spaces (CIEBasedA, CIEBasedABC, CIEBasedDEF, CIEBasedDEFG).</p> <p><i>CMYK</i>—Operates on deviceCMYK.</p> <p><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 the <b>DeviceNSpace</b> and <b>ColorPool</b> relements must also be present.</p> <p><i>DevIndep</i>—Operates on device independent color spaces (equivalent to Calibrated or CIE-Based or ICC-Based or Lab or YUV).</p> <p><i>Gray</i>—Operates on deviceGray.</p> <p><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].</p> <p>If IgnoreEmbeddedICC is "true", then nominally ICC-Based files or elements should be treated as being encoded in the alternate or underlying color space, and a ColorSpaceConversionOp where <i>SourceCS</i> = "DevIndep" will not be applied, unless that color space is also device independent.</p> <p><i>Lab</i>—Operates on Lab.</p> <p><i>RGB</i>—Operates on deviceRGB</p> <p><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 <b>RunList</b>.</p> <p><i>YUV</i>—Operates on YUV (Also known as YCbCr). See [CCIR601-2].</p> <p><b>Note:</b> 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.</p>
<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.

Name	Data Type	Description
<i>VSamples</i> ?	IntegerList	A sequence of vertical sampling factors—one entry per color channel in the raster data. If not specified, the implied default is "1" for every channel.
<i>QFactor</i> = "1.0"	double	A scale factor applied to the elements of <i>QuantTable</i> .
<i>QuantTable</i> ?	DoubleList	Quantization tables. If present there must be one <i>QuantTable</i> entry for each color channel.
<i>HuffTable</i> ?	DoubleList	Huffman tables for DC and AC components. If present there must be at least one <i>HuffTable</i> element for each color channel.
<i>ColorTransform</i> = "Automatic"	enumeration	Color transformation algorithm. Values are: <i>None</i> – Colors should not be transformed. <i>YUV</i> – RGB raster values should be transformed to YUV before encoding and from YUV to RGB after decoding. If four channels are present, transform CMYK values to YUVK before encoding and from YUVK to CMYK after decoding. <i>Automatic</i> – "YUV" for 3-channel raster data, "None" otherwise. Note that YUV is equivalent to YCbCr in TIFF terminology.

When the *DCTParams* element is a subelement of **ImageCompressionParams** used in a **FormatConversion** process to generate TIFF files, YUV is equivalent to YCbCr in TIFF terminology. The *HSamples* and *VSamples* values are used to set *YCbCrSubSampling* or *CIELabSubSampling*. This means that they are only relevant for data supplied as Lab, or data where *ColorTransform* is "YUV"; that the first element must be 1 in each case; that the fourth element must be 1 where CMYK data is to be compressed; and that the second and third elements must equal each other.

### Structure of the FlateParams Element

[New in JDF 1.2](#)

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 -1 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</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. <b>Note:</b> 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.

## Structure of the LZWParams Element

[New in JDF 1.2](#)

Name	Data Type	Description
<i>EarlyChange</i> = "1"	integer	<p>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.</p> <p><b>Note:</b> The default should not be used when this LZWParams element is in <b>ImageCompressionParams</b> used as an input resource to a <i>FormatConversion</i> process that is creating TIFF files.</p>
<i>Predictor</i> = "1"	integer	<p>A code that selects the predictor function:</p> <ul style="list-style-type: none"> <li>1 – No predictor (normal encoding or decoding).</li> <li>2 – TIFF Predictor 2.</li> <li>10 – PNG predictor, None function.</li> <li>11 – PNG predictor, Sub function.</li> <li>12 – PNG predictor, Up function.</li> <li>13 – PNG predictor, Average function.</li> <li>14 – PNG predictor, Path function.</li> <li>15 – PNG predictor in which the encoding filter automatically chooses the optimum function separately for each row.</li> </ul> <p><b>Note:</b> On 1X PNG predictors, these values select the specific PNG predictor function(s) to be used, as indicated above. When decoding, the predictor function is explicitly encoded in the incoming data.</p>

### 7.2.84 ImageReplacementParams

This resource specifies parameters required to control image replacement within production workflows.

#### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>DocIndex, RunIndex, RunTags, SheetName, Side, SignatureName</i>
<b>Input of processes:</b>	<b><i>ImageReplacement</i></b>
<b>Output of processes:</b>	—



## Resource Structure

Name	Data Type	Description
<a href="#">ImagePreScanStrategy ?</a> <a href="#">New in JDF 1.2</a>	NMTOKEN	Specifies the image pre-scanning strategy to be used on the input document data before starting the RIPing process. Possible values are: <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.
<a href="#">ImageReplacementStrategy</a>	enumeration	Identifies how externally referenced images will be handled within the associated process. Possible values are: <i>Omit</i> – Complete process maintaining only references to external data. <i>Proxy</i> – Complete process using available proxy images. <i>Replace</i> – Replace external references with image data during processing. <i>AttemptReplacement</i> – Attempt to replace external references with image data during processing. If replacement fails, complete the process using available proxy images.
<a href="#">MaxResolution ?</a> <a href="#">Deprecated in JDF 1.1</a>	double	Reduces the resolution of images with a resolution higher than <i>MaxResolution</i> . Replaced with a link to <i>ImageCompressionParams</i> in the process.
<a href="#">MinResolution ?</a>	double	Specifies the minimum resolution that an image must have in order to be embedded. If not specified, images of any resolution may be embedded.
<a href="#">ResolutionReductionStrategy ?</a> <a href="#">Deprecated in JDF 1.1</a>	enumeration	Identifies the mechanism used for reducing the image resolution. Possible values are: <i>Downsample</i> <i>Subsample</i> <i>Bicubic</i> Replaced with a link to <i>ImageCompressionParams</i> in the process.
<a href="#">IgnoreExtensions ?</a>	NMTOKENS	Identifies a set of filename extensions that will be trimmed during searches for high-resolution images. These extensions are what will be stripped from the end of an image name to find a base name. The leading dot “.” is included. Examples include: <i>.lay</i> <i>.e</i> <i>.samp</i>
<a href="#">MaxSearchRecursion ?</a>	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.
<a href="#">FileSpec + (SearchPath)</a> <a href="#">New in JDF 1.1</a>	refelement	Specification of the paths to search when trying to locate the referenced data. <i>Filespec</i> replaces the <i>SearchPath</i> text element.
<a href="#">SearchPath *</a> <a href="#">Deprecated in JDF 1.1</a>	telem	String that identifies the paths to search when trying to locate the referenced data.

## 7.2.85 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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	—
<b>Input of processes:</b>	<i>ImageSetting</i>
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
<i>AdvanceDistance</i> ?	double	Additional media advancement beyond the media dimensions on a roll-fed device.
<i>BurnOutArea</i> ? <a href="#">New in JDF 1.1</a>	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 may center an image if the device is capable of doing so. Possible values are: <i>None</i> – Do not center. <i>FeedDirection</i> – Image is centered around the feed-direction axis. <i>MediaWidth</i> – Image is centered around the media-width axis. <i>Both</i> – Image is centered around both axes.
<i>CutMedia</i> ?	boolean	Indicates whether or not to cut the media (roll-fed).
<i>ManualFeed</i> ? <a href="#">New in JDF 1.2</a>	boolean	Indicates whether the media will be fed manually.
<i>MirrorAround</i> ="None"	enumeration	This attribute specifies the axis around which a device must mirror an image if the device is capable of doing so. Possible values are: <i>None</i> – Do not mirror the image. <i>FeedDirection</i> – Image is mirrored around the feed-direction axis. <i>MediaWidth</i> – Image is mirrored around the media-width axis. <i>Both</i> – Image is mirrored around both possible axes.
<i>Polarity</i> = "Positive"	enumeration	Some devices can invert the image (in hardware). Possible values are: <i>Positive</i> <i>Negative</i>
<i>Punch</i> = "false"	boolean	If "true", indicates that the device must create registration punch holes.
<i>PunchType</i> ?	string	Name of the registration punch scheme, (e.g., <i>Bacher</i> ).
<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.

Name	Data Type	Description
<b>Sides =</b> "OneSidedFront" <a href="#">New in JDF 1.2</a>	enumeration	Indicates whether the content layout should be imaged on one or both sides of the media. Must only be used when <b>ImageSetterParams</b> describes output to a proofer. Possible values are: <i>OneSidedBackFlipX</i> – Page content is imaged on the back side of media so that the corresponding page cells back up to a blank front cell when flipping around the X axis. Equivalent to "WorkAndTumble" with a blank front side. <i>OneSidedBackFlipY</i> – Page content is imaged on the back side of media so that the corresponding page cells back up to a blank front cell when flipping around the Y axis. Equivalent to "WorkAndTurn" with a blank front side. <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 "WorkAndTumble". <i>TwoSidedFlipY</i> – Page content is imaged on both the front and back sides of media sheets so that the corresponding page cells back up to each other when flipping around the Y axis. Equivalent to "WorkAndTurn".
<b>SourceWorkStyle ?</b> <a href="#">New in JDF 1.2</a>	enumeration	When proofing in a "RIP once, output many" (ROOM) workflow, <b>SourceWorkStyle</b> specifies the direction in which the bytemaps have been prepared for press. The device should use this information to calculate a transformation that results in a proof that is identical to the press sheet. Possible values are identical to <b>ConventionalPrintingParams/@WorkStyle</b> .
<b>TransferCurve ?</b>	Transfer-Function	Area coverage correction of the device.
<b>Media ?</b> <a href="#">New in JDF 1.1</a>	refelement	Describes the media to be used. Different <b>Media</b> may be specified in different partition leaves to enable content driven <b>Media</b> selection.
<b>FitPolicy ?</b> <a href="#">New in JDF 1.2</a>	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.

## 7.2.86 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

<b>Resource class:</b>	Consumable
<b>Resource referenced by:</b>	<b>ConventionalPrintingParams</b>
<b>Example Partition:</b>	<i>FountainNumber, Separation, SheetName, Side, SignatureName, WebName</i>
<b>Input of processes:</b>	<b>ConventionalPrinting, DigitalPrinting</b>
<b>Output of processes:</b>	—

**Resource Structure**

Name	Data Type	Description
<i>ColorName</i> ? <a href="#">Clarified in JDF 1.2</a>	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 <b>Color</b> defined in a <b>ColorPool</b> resource that is linked to the process that is using the <b>Ink</b> resource. Instead of linking the <b>ColorPool</b> resource directly, it may be referenced by another resource that is linked to the process. <b>Note:</b> 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 576.
<i>Family</i> ?	NMTOKEN	Ink family. Possible values include: <i>HKS</i> <i>PANTONE</i> <i>Toyo</i> <i>ISO</i> – ISO 2846-1 (used by SWOP) <a href="#">Clarified in JDF 1.2</a> <i>InkJet</i> It is also possible to specify liquids that are similar to ink. Possible values of this type include: <i>Varnish</i> <i>Silicon</i> <i>Toner</i>
<i>InkName</i> ? <a href="#">Modified in JDF 1.1</a>	string	The name of ink is dependent on its <i>Family</i> . For example, the <i>InkName</i> "138 CVC" is a member of the <i>Pantone Family</i> .
<i>SpecialInk</i> ?	NMTOKEN	Specific ink attributes. Possible values include: <i>Metallic</i>
<i>SpecificYield</i> ?	double	Weight per area at total coverage in g/m <sup>2</sup> .

**7.2.87 InkZoneCalculationParams**

This resource specifies the parameters for the ***InkZoneCalculation*** process.

**Resource Properties**

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>TileID, WebName</i>
<b>Input of processes:</b>	<b><i>InkZoneCalculation</i></b>
<b>Output of processes:</b>	—

## Resource Structure

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 <b>Preview</b> 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> ? <a href="#">Modified in JDF 1.2</a>	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> ? <a href="#">Modified in JDF 1.2</a>	integer	The number of ink zones of the press.
<i>ZonesY</i> ?	integer	Number of ink zones in feed direction of the press.
<b>Device</b> ? <a href="#">New in JDF 1.2</a>	refelement	<b>Device</b> provides a reference to the press that the <b>InkZoneProfile</b> is defined for and is used to gather information about ink zone geometry.

### 7.2.88 InkZoneProfile

This resource specifies ink zone settings that are specific to the geometry of the printing device being used. **InkZoneProfiles** 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**

#### Resource Structure

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.
<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.

## 7.2.89 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.

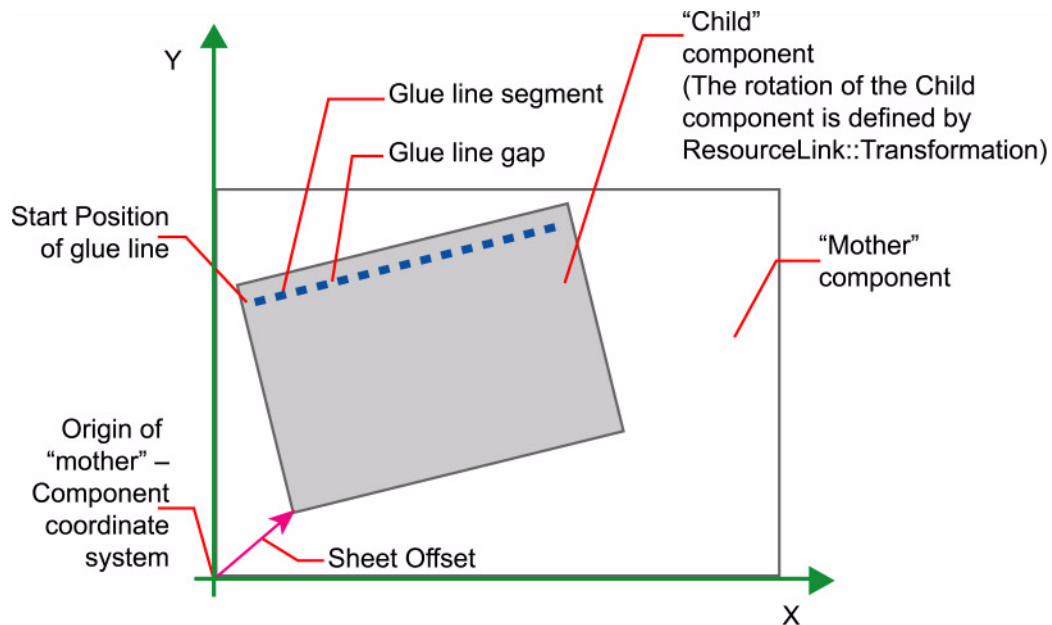


Figure 7.13: Parameters and Coordinate system used for Inserting

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:	<b>Inserting</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<a href="#">FinishedPage ?</a> <a href="#">New in JDF 1.2</a>	integer	Finished page number of the mother <b>Component</b> on which the child <b>Component</b> has to be placed, only if <i>InsertLocation</i> = "FinishedPage". Corresponds to <i>Folio</i> on <b>InsertingIntent</b> .
<a href="#">InsertLocation</a> <a href="#">Modified in JDF 1.2</a>	enumeration	Where to place the "child" sheet. Possible values are: <i>Back</i> <i>FinishedPage</i> – Place the child exactly onto the page specified in <i>FinishedPage</i> . <a href="#">New in JDF 1.2</a> <i>Front</i> <i>Overfold</i> – Place onto the overfold. Replaces <i>OverfoldLeft</i> and <i>OverfoldRight</i> . <a href="#">New in JDF 1.2</a> <i>OverfoldLeft</i> – <a href="#">Deprecated in JDF 1.2</a> <i>OverfoldRight</i> – <a href="#">Deprecated in JDF 1.2</a> <b>Note:</b> This was renamed from <i>Location</i> in JDF 1.2 due to a name clash with the <i>Location</i> partition key.

Name	Data Type	Description
<i>Method</i> = "BlowIn"	enumeration	Inserting method. Possible values are: <i>BindIn</i> – Apply glue to fasten the insert. <i>BlowIn</i> – Loose insert.
<i>SheetOffset</i> ? <a href="#">Deprecated in JDF 1.1</a>	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> .
<b>GlueLine</b> *	refelement	Array of all <b>GlueLine</b> elements. The coordinate system is defined by the mother <b>Component</b> .

## Location of Inserts

[New in JDF 1.2](#)

The following graphics depict the various values of **InsertingParams/@InsertLocation**.



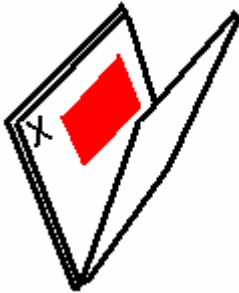
**Front.** Child on *Front* of mother component — is used for fixed inserts (e.g., glueing of inserts and so forth on signatures).



**Back.** Child on *Back* of mother component — is used for fixed inserts (e.g., glueing of inserts on signatures)..



**Overfold.** The mother component is opened at the overfold and the child is placed in the center of the of the mother. *Overfold* is used for loose inserts (e.g., inserts into newspapers).



**FinishedPage.** Child on *FinishedPage X* of mother component — can be used for loose and fixed inserts.

### 7.2.90 InsertSheet

**InsertSheet** resources define device generated images and sheets which may be produced along with the job. **InsertSheets** include separator sheets, error sheets, accounting sheets, and job sheets. The information provided on the sheet depends on the type of sheet. In some cases, an **Imposition** process may 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 should be inserted once the current **Sheet**, **Signature**, instance document, or job is completed.

**InsertSheets** may be used at the beginning or end of **RunLists** with a *SheetUsage* attribute of *Header* or *Trailer*. When an **InsertSheet** appears both in a **RunList** and in a **Layout** and/or **Sheet**, 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 InsertSheets** once, apply the **Surface**, **Sheet**, and **Signature InsertSheets**.

If the **RunList** of the **InsertSheet** does not supply enough content to fill a **Sheet**, **Signature**, or **Surface**, the **RunList** will be reapplied until no **PlacedObject** slots remain to be filled. When an **InsertSheet** is used in a **RunList** of a process that does not use a **Layout** or **LayoutPreparationParams** resource (i.e., that process has not been combined with **Imposition** or **LayoutPreparation**), only *Usage Header* or *Trailer* are valid.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>Disjointing, Layout, LayoutPreparationParams, RunList, Sheet</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—



## Resource Structure

Name	Data Type	Description
<a href="#">IncludeInBundleItem ?</a> <a href="#">New in JDF 1.2</a>	enumeration	<p>Defines bundle items when this <b>InsertSheet</b> is not a subelement of <b>RunList</b>. If this <b>InsertSheet</b> is a subelement of a <b>RunList</b>, then <i>IncludeInBundleItem</i> must be ignored, and <b>RunList/@EndOfBundleItem</b> must be used instead. As an example, <i>IncludeInBundleItem</i> controls whether the <b>InsertSheet</b> is to be included in a bundle item for purposes of finishing the <b>InsertSheet</b> with other sheets. Possible values are:</p> <p><i>After</i> – This <b>InsertSheet</b> is to be included in the <b>BundleItem</b> that occurs after this <b>InsertSheet</b>. "After" is equivalent to "None" if no <b>BundleItem</b> is defined after this <b>InsertSheet</b></p> <p><i>Before</i> – This <b>InsertSheet</b> is to be included in the <b>BundleItem</b> that occurs before this <b>InsertSheet</b>. "Before" is equivalent to "None" if no <b>BundleItem</b> is defined before this <b>InsertSheet</b></p> <p><i>None</i> – This <b>InsertSheet</b> is not included in a <b>BundleItem</b>.</p> <p><i>New</i> – A new <b>BundleItem</b> is created. This <b>InsertSheet</b> will be in the new <b>BundleItem</b> by itself unless another <b>InsertSheet</b> with <i>IncludeInBundleItem</i> = "Before" occurs immediately after this <b>InsertSheet</b>.</p>
<i>IsWaste ?</i>	boolean	Specifies whether the <b>InsertSheet</b> is waste that should be removed from the document before further processing. If "true", the <b>InsertSheet</b> is to be discarded when finishing the document.
<a href="#">MarkList ?</a> <a href="#">New in JDF 1.1</a>	NMTO-KENS	<p>List of marks that should be marked on this <b>InsertSheet</b>. Ignored if a Sheet is specified in this <b>InsertSheet</b>. 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>
<a href="#">SheetFormat ?</a> <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	NMTO-KEN	<p>Identifies that device-dependent information is to be included on the <b>InsertSheet</b>. Possible values include:</p> <p><i>Blank</i>  <i>Brief</i>  <i>Duplicate</i> – Valid for <i>SheetUsage</i> = "Interleaved" or "InterleavedBefore". 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 <b>InsertSheet</b> has both sides duplicated. <a href="#">New in JDF 1.2</a></p> <p><i>Full</i>  <i>Standard</i></p>

Name	Data Type	Description
<p><b>SheetType</b>  <a href="#">New in JDF 1.1</a></p>	<p>enumera- tion</p>	<p>Identifies the type of sheet. Possible 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 <b>ContentObjects</b> with no matching entry in the content <b>RunList</b>.</p> <p><i>InsertSheet</i> – A sheet that is inserted to the job, (e.g., a preprinted 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>
<p><b>SheetUsage</b>  <a href="#">New in JDF 1.1</a>  <a href="#">Modified in JDF 1.2</a></p>	<p>enumera- tion</p>	<p>Indicates where this <b>InsertSheet</b> is to be produced and inserted into the set of output pages. Possible values are:</p> <p><i>FillForceBack</i> – Valid for <i>SheetType</i> = "<i>FillSheet</i>". Contents of the <b>RunList</b> of the <b>InsertSheet</b> are used to fill the front surface of the current sheet before forcing the next page of the content <b>Runlist</b> to the back surface of the current sheet if not already on a back surface.</p> <p><i>FillForceFront</i> – Valid for <i>SheetType</i> = "<i>FillSheet</i>". Contents of the <b>RunList</b> of the <b>InsertSheet</b> are used to fill the back surface of the current sheet before forcing the next page of the content <b>Runlist</b> to the front surface of the next sheet if not already on a front surface.</p> <p><i>FillSheet</i> – Valid for <i>SheetType</i> = "<i>FillSheet</i>". Contents from the <b>RunList</b> of the <b>InsertSheet</b> are used to fill the current sheet.</p> <p><i>FillSignature</i> – Valid for <i>SheetType</i> = "<i>FillSheet</i>". Contents from the <b>RunList</b> of the <b>InsertSheet</b> are used to fill the current signature.</p> <p><i>FillSurface</i> – Valid for <i>SheetType</i> = "<i>FillSheet</i>". Contents from the <b>RunList</b> of the <b>InsertSheet</b> are used to fill the current surface.</p> <p><i>Header</i> – Valid for <i>SheetType</i> = "<i>InsertSheet</i>", "<i>JobSheet</i>", or "<i>SeparatorSheet</i>". 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 prepended before the current sheet, signature, layout, or <b>RunList</b> as defined by its context. Contents for the <b>Sheet</b> are drawn from the <b>RunList</b> included in this <b>InsertSheet</b> resource, if one is included. If a <b>RunList</b> is not included, the inserted sheet is filled with system-specified content defined by <i>SheetType</i>.</p> <p><i>Interleaved</i> – Valid for <i>SheetType</i> = "<i>SeparatorSheet</i>". The sheet is produced after each page, (e.g., used to insert sheets under transparencies). Contents for the <b>Sheet</b> are drawn from the <b>RunList</b> included in this <b>InsertSheet</b> resource, if one is included. If a <b>RunList</b> is not included, the inserted sheet is filled with system-specified content defined by <i>SheetType</i> = "<i>SeparatorSheet</i>".</p> <p><i>InterleavedBefore</i> – Valid for <i>SheetType</i> = "<i>SeparatorSheet</i>". The sheet is produced before each page, (e.g., used to insert sheets before transparencies). Contents for the <b>Sheet</b> are drawn from the <b>RunList</b> included in this <b>InsertSheet</b> resource, if one is included. If a <b>RunList</b> is not included, the inserted sheet is filled with system-specified content defined by <i>SheetType</i> = "<i>SeparatorSheet</i>". <a href="#">New in JDF 1.2</a></p>

Name	Data Type	Description
<i>SheetUsage</i> (Continued)		<p><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.</p> <p><i>Slip</i> – Valid for <i>SheetType</i> = "SeparatorSheet". The sheet is produced between each copy of each instance document. Contents for the <b>Sheet</b> are drawn from the <b>RunList</b> included in this <b>InsertSheet</b> resource, if one is included. If a <b>RunList</b> is not included, the inserted sheet is filled with system-specified content defined by <i>SheetType</i> = "SeparatorSheet".</p> <p><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 <b>RunList</b> has been consumed. Contents for the <b>Sheet</b> are drawn from the <b>RunList</b> included in this <b>InsertSheet</b> resource, if one is included. If a <b>RunList</b> is not included, the inserted sheet is filled with system-specified content defined by <i>SheetType</i> = "SeparatorSheet".</p> <p><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 <b>RunList</b> as defined by its context. Contents for the <b>Sheet</b> are drawn from the <b>RunList</b> included in this <b>InsertSheet</b> resource, if one is included. If a <b>RunList</b> is not included, the inserted sheet is filled with system specified content defined by <i>SheetType</i>.<sup>a</sup></p>
<i>Usage ?</i> <a href="#">Deprecated in JDF 1.1</a>	enumeration	Replaced by <i>SheetUsage</i> .
<b>RunList ?</b>	refelement	A <b>RunList</b> that provides the content for the <b>InsertSheet</b> . Any <b>InsertSheet</b> resources referenced by this <b>RunList</b> are ignored.
<b>Sheet ?</b>	refelement	Details of the <b>Sheet</b> that will be inserted. Contents for this <b>Sheet</b> are drawn from the <b>RunList</b> included in this <b>InsertSheet</b> , if any. If not specified, the system specified insert sheets are used. Any <b>InsertSheet</b> resources referenced by this <b>Sheet</b> are ignored.

- a. **Note:** Use *SheetType* = "ErrorSheet" and *SheetUsage* = "Trailer" to always produce a sheet that contains error or success information even if no errors or warnings occurred.

### 7.2.91 InterpretedPDLData

Represents the results of the PDL Interpretation 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:	<b>RunList</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

## 7.2.92 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, SheetName, Side, SignatureName</i>
Input of processes:	<b>Interpreting</b>
Output of processes:	—

### Structure of the InterpretingParams Resource

Name	Data Type	Description
<i>Center = "false"</i>	boolean	Indicates whether or not the finished page image should be centered within the imageable area of the media. <i>Center</i> is ignored if <b>FitPolicy</b> / <i>@SizePolicy = "ClipToMaxPage"</i> and clipping is required.
<i>FitToPage ?</i> <a href="#">Deprecated in JDF 1.1</a>	boolean	Specifies whether the finished page contents should be scaled to fit the media. In JDF 1.1 and beyond, use <b>FitPolicy</b> .
<i>MirrorAround = "None"</i>	enumeration	This attribute specifies the axis around which a RIP may mirror an image. Note that this is mirroring in the RIP and not in the hardware of the output device. Possible values are: <i>None</i> – The default. <i>FeedDirection</i> – Image is mirrored around the feed-direction axis. <i>MediaWidth</i> – Image is mirrored around the media-width axis. <i>Both</i> – Image is mirrored around both possible axes.
<i>Polarity = "Positive"</i>	enumeration	The image must be RIPed in the polarity specified. Note that this is a polarity change in the RIP and not a polarity change in the hardware of the output device. Possible values are: <i>Positive</i> <i>Negative</i>
<i>Poster ?</i>	XYPair	Specifies whether the page contents should 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 must be calculated for the poster tiles across the horizontal and vertical axes, respectively.
<i>PrintQuality = "Normal"</i> <a href="#">New in JDF 1.1</a>	enumeration	Generic switch for setting the quality of an otherwise inaccessible device. Possible values are: <i>High</i> – Highest quality available on the printer. <i>Normal</i> – The default quality provided by the printer. <i>Draft</i> – Lowest quality available on the printer.
<i>Scaling ?</i>	XYPair	A pair of positive real values that indicates the scaling factor for the page contents. Values between 0 and 1 specify that the contents are to be reduced, while values greater than 1 specify that the contents are to be expanded. This attribute is ignored if <i>FitToPage = "true"</i> or if <i>Poster</i> is present and has a value other than "1 1". Any scaling defined in <b>FitPolicy</b> must be applied after the scaling defined by this attribute.

Name	Data Type	Description
<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"
<b>FitPolicy</b> ? <a href="#">New in JDF 1.1</a>	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 <b>FitPolicy</b> element must be ignored in a combined process with <b>LayoutPreparation</b> .
<b>Media</b> * <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	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 <b>Interpreting</b> . 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 <b>Media</b> are specified, <b>Media/@MediaType</b> defines the type of <b>Media</b> . If multiple <b>Media</b> with <b>Media/@MediaType</b> = "Paper" are specified in a proofing environment, the first <b>Media</b> is the proofer paper and the second <b>Media</b> is the final device paper.
<b>ObjectResolution</b> *	refelement	Indicates the resolution at which the PDL contents will be interpreted in DPI. These elements may be different from the <b>ObjectResolution</b> elements provided in the resource.
<b>PDFInterpretingParams</b> ? <a href="#">New in JDF 1.1</a>	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.

### Structure of PDFInterpretingParams Subelement

[New in JDF 1.1](#)

Name	Data Type	Description
<i>EmitPDFBG</i> = "true"	boolean	Indicates whether BlackGeneration functions should be emitted.
<i>EmitPDFHalftones</i> = "true"	boolean	Indicates whether Halftones should be emitted.
<i>EmitPDFTransfers</i> = "true"	boolean	Indicates whether Transfer functions should be emitted
<i>EmitPDFUCR</i> = "true"	boolean	Indicates whether UnderColorRemoval functions should be emitted.
<i>HonorPDFOverprint</i> = "true"	boolean	Indicates whether or not overprint settings in the file will be honored. If "true", the setting for overprint will be honored. If "false", it is expected that the device does not directly support overprint and that the PDF is preprocessed to simulate the effect of the overprint settings
<i>ICColorAsDeviceColor</i> = "false"	boolean	Indicates whether colors specified by ICC color spaces should be treated as device colorants.
<i>PrintPDFAnnotations</i> = "false"	boolean	Indicates whether the contents of annotations on PDF pages should be included in the output. This only refers to annotations that are set to print in the PDF file.
<i>TransparencyRendering Quality</i> ?	double	Possible values are 0 to 1. 0 represents the lowest allowable quality. 1 represents the highest desired quality.

### 7.2.93 JacketingParams

[New in JDF 1.1](#)

Description of the setup of the jacketing machinery. Jacket height and width (1 and 3 in the figure below) are specified within the **Component** that describes the jacket.

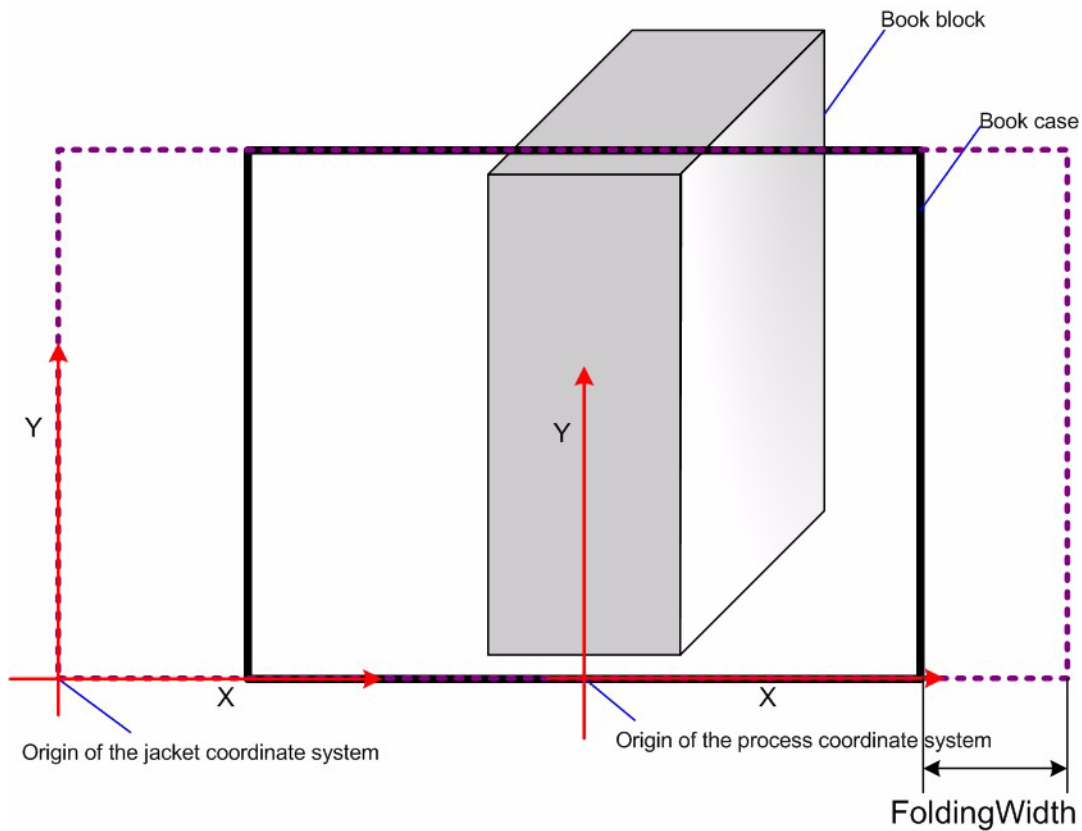
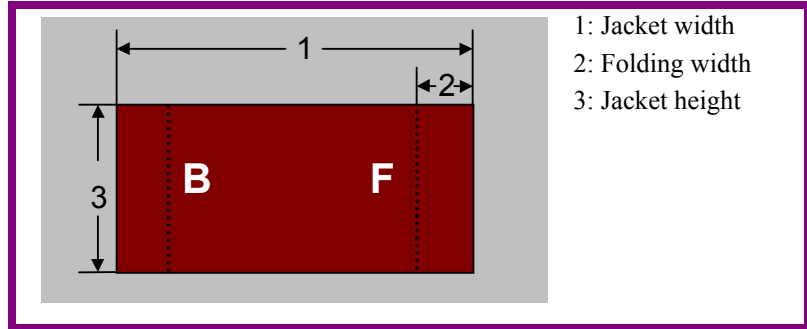


Figure 7.14: Parameters and Coordinate System for Jacketing

#### Resource Properties

Resource class: Parameter  
 Resource referenced by: —  
 Example Partition: —  
 Input of processes: **Jacketing**

#### Resource Structure

Name	Data Type	Description
<i>FoldingWidth</i>	double	Definition of the dimension of the folding width of the front cover fold (see “ <i>FoldingWidth</i> ” in the picture above). All other measurements are implied by the dimensions of the book.

## 7.2.94 JobField

[New in JDF 1.1](#)

A **JobField** is a Mark object that specifies the details of a job. **JobFields** are also referred to as slug lines.

### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>Surface</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<a href="#">ShowList</a> <a href="#">Modified in JDF 1.2</a>	NMTOKENS	List of elements to display in the <b>JobField</b> . Values include: <i>DeviceID</i> – ID of the device. This is a unique name within the workflow. <i>EndTime</i> – Actual end time of the job. <i>Error</i> – Errors that happened during the job. <i>ErrorStats</i> – Statistics on errors that happened during execution. <i>ExposedMediaName</i> – <i>DescriptiveName</i> of the exposed media, e.g. plate or proof that is being imaged. <i>FriendlyName</i> – <i>FriendlyName</i> of the device. <i>JobID</i> – JobID of the node that is executing. <i>JobName</i> – <i>DescriptiveName</i> of the node that is executing. <i>JobRecipientName</i> – Name of the recipient of the job. <i>JobSubmitterName</i> – Name of the submitter of the job. <i>StartTime</i> – Actual <i>StartTime</i> of the job. <i>MediaBrand</i> – Brand of the media that is being printed. <i>MediaType</i> – <i>DescriptiveName</i> of the media that is being printed. <i>MoonPhase</i> – Phase of the moon at the <i>StartTime</i> of the job. <i>Operator</i> – Name of the operator. <i>OperatorText</i> – Text from the operator as defined in <i>OperatorText</i> . <i>PrintQuality</i> – The quality of the printout. (High, Normal, Draft, or device specific name) <i>ProoferProfileName</i> – Name of the ICC profile for the proofing device. <i>PressProfileName</i> – Name of the ICC profile for the final printing (used as intermediate space during proofing). <i>Resolution</i> – Output resolution. <i>ResolutionX</i> – Output resolution in X direction. <i>ResolutionY</i> – Output resolution in Y direction. <i>ScreeningFamily</i> – Name of the screening family of the output. <i>UserText</i> – User-defined text as defined in <i>UserText</i> . <i>Warning</i> – Warnings that happened during the job. Warnings don't lose information in the resulting job, while errors do. In addition, the partition key names defined in Table 3-27, “Contents of the Partitionable Resource Element,” on page 78 are also supported.
<a href="#">OperatorText</a> <a href="#">?Modified in JDF 1.2</a>	string	Text from the operator. Note that this was erroneously described as text to the operator in JDF 1.1 and below.
<a href="#">UserText ?</a>	string	User-defined text to output with JobField.
<a href="#">DeviceMark ?</a>	refelement	<b>DeviceMark</b> defines the formatting parameters for the mark. If not specified, the <b>DeviceMark</b> settings defined in <b>LayoutPreparationParams</b> or in the <b>Layout</b> tree are assumed.

## 7.2.95 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:	<b>Labeling</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>Application ?</i>	NMTOKEN	Application method of the label. Values include: <i>Glue</i> – Glued onto the component. <i>Loose</i> – Loosely laid onto the component. <i>SelfAdhesive</i> – Self adhesive label. <i>Staple</i> – Stapled onto the component.
<i>CTM ?</i>	matrix	Position and orientation of the label lower-left-corner relative to the lower left corner of the component surface as defined by <i>Position</i> .
<i>Position ?</i>	enumeration	Position of the label on the bundle. One of: <i>Top</i> <i>Bottom</i> <i>Left</i> <i>Right</i> <i>Front</i> <i>Back</i>

## 7.2.96 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:	<b>Laminating</b>
Output of processes:	—

### Resource Structure

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 <b>Component</b> . 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.



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 <b>Component</b> to be laminated.
<i>LaminatingMethod</i> ?	enumeration	Laminating technology that is applied. One of: <i>CompoundFoil</i> <i>DispersionGlue</i> <i>Unknown</i> <a href="#">Deprecated in JDF1.2</a>
<i>Temperature</i> ?	double	Temperature used in the lamination process, in ° Centigrade.

## 7.2.97 Layout

Represents the root of the layout structure. **Layout** is used both for fixed-layout and for automated printing.

### Resource Properties

Resource class: Parameter

Resource referenced by: **Component**

Example Partition: It is strongly discouraged to partition the **Layout** tree, including **Sheet** and **Surface**.

Input of processes: **ConventionalPrinting, DigitalPrinting, Imposition, InkZoneCalculation, Proofing, SoftProofing**

Output of processes: **LayoutPreparation, Stripping**

### Resource Structure

Name	Data Type	Description
<i>Automated</i> = "false"	boolean	If "true", the <b>Imposition</b> process is expected to perform automated page consumption. Automated page consumption is equivalent to the PrintLayout functionality provided in PJTF.
<i>MaxDocOrd</i> = "1" <a href="#">New in JDF 1.1</a>	integer	Zero-based maximum number of instance documents that are consumed from a <b>RunList</b> each time the <b>Layout</b> is executed, assuming the <b>Imposition</b> process is automated.
<i>MaxOrd</i> ?	integer	Zero-based maximum number of placed objects that are consumed from a <b>RunList</b> each time the <b>Layout</b> is executed, assuming the <b>Imposition</b> process is automated. If not specified, it must be calculated from the <i>Ord</i> values of the ContentObjects in the <b>Layout</b> .
<i>MaxSetOrd</i> = "1" <a href="#">New in JDF 1.1</a>	integer	Zero-based maximum number of document sets that are consumed from a <b>RunList</b> each time the <b>Layout</b> is executed, assuming the <b>Imposition</b> process is automated.
<i>Name</i> ? <a href="#">New in JDF 1.1</a>	string	Unique name of the Layout. <i>Name</i> is used for external reference to a <b>Layout</b> .
<b>InsertSheet</b> *	refelement	Additional sheets that should be inserted before and/or after a document.
<i>LayerList</i> ? <a href="#">New in JDF 1.1</a>	element	List of <b>LayerDetails</b> elements.
<b>Media</b> ? <a href="#">New in JDF 1.1</a>	refelement	Describes the media to be used. <b>Media</b> must be specified within one of <b>Layout</b> , <b>Signature</b> , or <b>Sheet</b> within a <b>Layout</b> structure.
<b>MediaSource</b> ? <a href="#">Deprecated in JDF 1.1</a>	refelement	Describes the media to be used. Replaced by <b>Media</b> in JDF 1.1.
<b>Signature</b> *	element	The signatures that are defined by the layout.
<b>TransferCurvePool</b> ? <a href="#">New in JDF 1.1</a>	refelement	Describes the relationship of transfer curves and coordinate systems within the various processes.

## Structure of LayerList Subelement

[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.

Name	Data Type	Description
LayerDetails *	element	Details of the individual layers.

## Structure of LayerDetails Subelement

[New in JDF 1.1](#)

This element provides information about individual layers

Name	Data Type	Description
Name ?	string	Unique name of the layer.

## Structure of Signature Subelement

This element groups individual **Sheet** resources into one **Signature** subelement.

Name	Data Type	Description
Name ?	string	Unique name of the signature. <i>Name</i> is used for external reference to a signature, as in a <b>Part</b> element.
InsertSheet *	refelement	Specifies how to complete a signature in an automated printing environment.
Media ? <a href="#">New in JDF 1.1</a>	refelement	Describes the media to be used. Defaults to <b>Layout/Media</b> .
MediaSource ? <a href="#">Deprecated in JDF 1.1</a>	refelement	Describes the media to be used. Replaced by <b>Media</b> in JDF 1.1.
Sheet *	refelement	Resources that comprise the signature.

## 7.2.98 LayoutElement

This resource is needed for **LayoutElementProduction**. It describes some text, an image, one or more pages, or anything else that is used in the production of the layout of a product.

### Resource Properties

Resource class: Parameter

Resource referenced by: **RunList**, **Surface/PlacedObject**

Example Partition: *PageNumber*

Input of processes: **DBDocTemplateLayout**, **DBTemplateMerging**, **LayoutElementProduction**

Output of processes: **DBDocTemplateLayout**, **LayoutElementProduction**

### Resource Structure

Name	Data Type	Description
ClipPath ? <a href="#">Modified in JDF 1.2</a>	PDFPath	Path that describes the outline of the <b>LayoutElement</b> in the coordinate space of the <b>LayoutElement</b> of <i>ElementType</i> = "Page" that results from the <b>LayoutElementProduction</b> 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 available, must be concatenated.

Name	Data Type	Description
<i>ElementType</i> ? <a href="#">Modified in JDF 1.2</a>	enumeration	Describes the content type for this <b>LayoutElement</b> . Possible values are: <i>Auxilliary</i> – Any type of file that is needed to complete a layout but not explicitly displayed, (e.g., ICC profiles or fonts). <i>Composed</i> – Combination of elements that define an element that is not bound to a document page. <i>Document</i> – An ordered set of one or more pages. <i>Graphic</i> – Line art. <i>Image</i> – Bitmap image. <i>MultiDocument</i> – An ordered set of one or more Documents including document breaks, (e.g., PPML, PPML/VDX, MIME multipart/related). <i>MultiSet</i> – An ordered set of one or more Document sets including document set breaks, (e.g., PPML, PPML/VDX). <i>Page</i> – Representation of one document page. <i>Reservation</i> – Empty element. Content for this area of the page may 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> – <a href="#">Deprecated in JDF 1.2</a>
<i>HasBleeds</i> ? <a href="#">Modified in JDF 1.2</a>	boolean	If " <i>true</i> ", the file has bleeds. If not specified, the value of <b>PageList/PageData/@HasBleeds</b> is applied.
<i>IgnorePDLCopies</i> = " <i>false</i> " <a href="#">New in JDF 1.1</a>	boolean	If " <i>true</i> ", any PDL defined copy count must be ignored.
<i>IgnorePDLImposition</i> = " <i>true</i> " <a href="#">New in JDF 1.1</a>	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 may result in undesired behavior.
<i>IsBlank</i> ? <a href="#">New in JDF 1.2</a>	boolean	If " <i>false</i> ", the <b>LayoutElement</b> has no content marks and is blank. If not specified, the value of <b>PageList/PageData/@IsBlank</b> is applied.
<i>IsPrintable</i> ? <a href="#">Modified in JDF 1.2</a>	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> ". If not specified, the value of <b>PageList/PageData/@IsPrintable</b> is applied.
<i>IsTrapped</i> ? <a href="#">Modified in JDF 1.2</a>	boolean	If " <i>true</i> ", the file has been trapped. If not specified, the value of <b>PageList/PageData/@IsTrapped</b> is applied.
<i>PageListIndex</i> ? <a href="#">New in JDF 1.2</a>	Integer-RangeList	List of the indices of the <b>PageData</b> elements of the <b>PageList</b> specified in this <b>LayoutElement</b> . Note that this list may be overwritten by a <b>RunList</b> that contains this <b>LayoutElement</b> and refers to a subset of this <b>LayoutElement</b> . <b>PageList</b> must be specified if <i>PageListIndex</i> is defined.

Name	Data Type	Description
<a href="#">SourceBleedBox ?</a> <a href="#">Modified in JDF 1.2</a>	rectangle	A rectangle that describes the bleed area of the element to be included. This rectangle is expressed in the default user space. If not specified, the value of <b>PageList/PageData/@SourceBleedBox</b> is applied.
<a href="#">SourceClipBox ?</a> <a href="#">Modified in JDF 1.2</a>	rectangle	A rectangle that defines the region of the element to be included. This rectangle is expressed in the default user space of the source document page. If not specified, the value of <b>PageList/PageData/@SourceClipBox</b> is applied.
<a href="#">SourceTrimBox ?</a> <a href="#">Modified in JDF 1.2</a>	rectangle	A rectangle that describes the intended trimmed size of the element to be included. This rectangle is expressed in the default user space. If not specified, the value of <b>PageList/PageData/@SourceTrimBox</b> is applied.
<a href="#">Template ?</a> <a href="#">Modified in JDF 1.2</a>	boolean	<i>Template</i> is "false" when this layout element is self-contained. This attribute is "true" if the <b>LayoutElement</b> represents a template that must be completed with information from a database. If not specified, the value of <b>PageList/PageData/@Template</b> is applied.
<a href="#">ColorPool ?</a> <a href="#">New in JDF 1.2</a>	refelement	Definition of the color details.
<a href="#">Dependencies ?</a> <a href="#">New in JDF 1.2</a>	element	List of dependent references, (e.g., fonts, external images, etc.).
<a href="#">ElementColorParams ?</a> <a href="#">New in JDF 1.2</a>	refelement	Color details of the <b>LayoutElement</b> . If not specified, the value of <b>PageList/PageData/ElementColorParams</b> is applied.
<a href="#">FileSpec ?</a> <a href="#">Modified in JDF 1.2</a>	refelement	URL plus metadata about the physical characteristics of a file representing the <b>LayoutElement</b> . If not present, then only metadata is known but not the content file.
<a href="#">ImageCompressionParams ?</a> <a href="#">New in JDF 1.2</a>	refelement	Specification of the image compression properties. If not specified, the value of <b>PageList/PageData/ImageCompressionParams</b> is applied.
<a href="#">PageList ?</a> <a href="#">New in JDF 1.2</a>	refelement	Specification of page metadata for pages described by this <b>LayoutElement</b> .
<a href="#">ScreeningParams ?</a> <a href="#">New in JDF 1.2</a>	refelement	Specification of the screening properties. If not specified, the value of <b>PageList/PageData/ScreeningParams</b> is applied.
<a href="#">SeparationSpec *</a> <a href="#">Modified in JDF 1.2</a>	refelement	List of used separation names. If not specified, the value of <b>PageList/PageData/@SeparationSpec</b> is applied.

### Structure of Dependencies Subelement

[New in JDF 1.2](#)

This element provides a container for dependent references of the **LayoutElement**.

Name	Data Type	Description
<b>LayoutElement *</b>	refelement	Description of dependent elements, (e.g., fonts, images, etc.).

### 7.2.99 LayoutPreparationParams

[New in JDF 1.1](#)

This resource provides the parameters of the **LayoutPreparation** process, which provides the details of how finished page contents will be imaged onto media. This resource has a provision for specifying either a multi-up grid of content page cells or an imposition layout of finished pages.

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:	<b>LayoutPreparation</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>BackMarkList</i> ?	NMTOKENS	List of marks that should be marked on each back surface. The appearance of the marks are defined by the process implementation. For a list of pre-defined values, see <i>FrontMarkList</i> .
<i>CreepValue</i> ?	XYPair	This parameter determines horizontal and vertical creep compensation. The numbers specify the distance in points by which the respective gutter that creeps either increments or decrements in width from one sheet to the next for a given sequence of sheets related to the same bound component. If the value of a component of this attribute is positive, it specifies the amount in points by which the width of creeping gutters are incremented. If the value of a component of this attribute is negative then it specifies the amount in points by which the width of creeping gutters are decremented. If not specified, it may be calculated based on the information taken from <b>Media</b> .
<i>FinishingOrder</i> = "GatherFold"	enumeration	Specifies the order of operations for finishing a bound booklet created from multiple imposed sheets. The <b>LayoutPreparation</b> 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. Possible 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.

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 the format “Fx-y” as shown in Figure 7.9 and Figure 7.10.</p> <p>The LayoutPreparation process uses the fold description specified by this attribute in the determination of the proper distribution of pages onto the surfaces of the sheets in the context of the values of both the <i>PageDistributionScheme</i> and <i>FinishingOrder</i> attributes.</p> <p>If not present, no folding other than the folding that is implied by <i>PageDistributionScheme</i> = “<i>Saddle</i>” is assumed.</p>
<i>FrontMarkList</i> ?	NMTOKENS	<p>List of marks that should be marked on each front surface. The appearance of the marks are defined by the process implementation. Values include:</p> <p><i>CIELABMeasuringField</i>  <i>ColorControlStrip</i>  <i>ColorRegisterMark</i>  <i>CutMark</i>  <i>DensityMeasuringField</i>  <i>IdentificationField</i>  <i>JobField</i>  <i>PaperPathRegisterMark</i>  <i>RegisterMark</i>  <i>ScavengerArea</i></p>
<i>Gutter</i> ? <a href="#">Modified in JDF 1.2</a>	XYPair	<p>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 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 present 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 the <i>VerticalCreep</i> or <i>HorizontalCreep</i> attributes, then the value of <i>Gutter</i> specifies the initial gutter width where the gutter width may increment or decrement depending upon the explicit or implied value of the <i>CreepValue</i> attribute. The <i>Gutter</i> is applied in addition to any <i>Border</i> specified in the <i>PageCell</i>.</p>
<i>HorizontalCreep</i> ? <a href="#">Modified in JDF 1.2</a>	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 the <i>NumberUp</i> attribute. The value for an entry in this list must be between zero and two (2) less than the second value of the <i>NumberUp</i> attribute. If not specified, then no horizontal gutters will creep.</p>

Name	Data Type	Description
<p><i>NumberUp</i> ?  <a href="#">Clarified in JDF 1.2</a></p>	XYPair	<p>Specifies a regular, multi-up grid of <b>PageCells</b> 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 in tegers).</p> <p><b>Compatibility Warning.</b> 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 <b>RunList</b> 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>
<p><i>PageDistributionScheme</i> =  "Sequential"</p>	NMTOKEN	<p>Specifies how finished pages are to be distributed onto a multi-up grid of finished <b>PageCells</b> defined by the values of the <i>NumberUp</i> attribute. Possible values include:</p> <p><i>Saddle</i> – Distribute finished pages onto a sequence of one or more imposition layouts in proper order for saddle stitch binding. For this page distribution scheme, creep should only be applied to odd-numbered vertical gutters where any even-numbered gutters will 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 <b>Component</b> 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><b>Note:</b> 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><b>Note:</b> In all cases, the order of finished pages as represented by the content <b>RunList</b> must be either in reader order or in an order appropriate for multi-up saddle stitching. Refer to the <i>PageOrder</i> attribute.</p>
<p><i>PageOrder</i> =  "Reader"</p>	NMTOKEN	<p>The assumed ordering of the finished pages in the <b>RunList</b>.</p> <p><i>Booklet</i> – The finished pages are preordered in the <b>RunList</b> 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 <b>RunList</b>.</p>

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. Possible values are:</p> <p><i>FoldCatalog</i> – Finished pages are imaged so that the result is compatible with a finished product produced from the folding catalog as specified in <i>FoldCatalog</i>.</p> <p><i>XYZ</i>: Permutations of the letters XYZ and xyz so that exactly one of upper or lower case of x, y, and z define the order in which finished pages are flowed along each axis with respect to the coordinate system of the front side of the sheet.</p> <p><i>X</i> – Specifies flowing left to right across a sheet surface.  <i>x</i> – Specifies flowing right to left across a sheet surface.  <i>Y</i> – Specifies flowing bottom to top vertically across a sheet surface.  <i>y</i> – Specifies flowing top to bottom vertically across a sheet surface.  <i>Z</i> – Specifies flowing bottom of stack to top of stack through the stack.  <i>z</i> – Specifies flowing top of stack to bottom of stack through the stack.</p> <p>The following table specifies how cells are ordered on simplex 4-up depending on XYZ.</p> <table border="1" data-bbox="631 934 1156 1045"> <thead> <tr> <th><i>XYZ</i></th> <th><i>Zxy</i></th> <th><i>xyz</i></th> <th><i>XYZ</i></th> </tr> </thead> <tbody> <tr> <td>1 2 5 6</td> <td>4 2 3 1</td> <td>2 1 6 5</td> <td>3 4 7 8</td> </tr> <tr> <td>3 4 7 8</td> <td>8 6 7 5</td> <td>4 3 8 7</td> <td>1 2 5 6</td> </tr> </tbody> </table>	<i>XYZ</i>	<i>Zxy</i>	<i>xyz</i>	<i>XYZ</i>	1 2 5 6	4 2 3 1	2 1 6 5	3 4 7 8	3 4 7 8	8 6 7 5	4 3 8 7	1 2 5 6
<i>XYZ</i>	<i>Zxy</i>	<i>xyz</i>	<i>XYZ</i>											
1 2 5 6	4 2 3 1	2 1 6 5	3 4 7 8											
3 4 7 8	8 6 7 5	4 3 8 7	1 2 5 6											
<i>Rotate</i> = "Rotate0"	enumeration	<p>Orthogonal rotation including the implied translation to be applied to the grid of <i>PageCells</i> on the entire surface relative to the process coordinate system. One of:</p> <p><i>Rotate0</i></p> <p><i>Rotate90</i> – 90° counterclockwise rotation.  <i>Rotate180</i> – 180° rotation.  <i>Rotate270</i> – 90° clockwise rotation.</p> <p>For details of orthogonal rotations, refer to Table 2-3, "Matrices and Orientation values used to describe the orientation of a Component," on page 24. If a <i>RotatePolicy</i> value other than "NoRotate" is specified in, the value specified in <i>Rotate</i> may be modified accordingly.</p> <p><b>Note:</b> A rotation of the grid also rotates the gutters, (i.e., it is applied after all other parameters have been evaluated and applied).</p>												



Name	Data Type	Description										
<b>Sides =</b> <i>"OneSidedFront"</i>	enumeration	<p>Indicates whether the content layout should be imaged on one or both sides of the media. When a different value for the Sides 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 re-statement was not present. Possible values are:</p> <p><i>OneSidedBackFlipX</i> – Page content is imaged on the back side of media so that the corresponding page cells back up to a blank front cell when flipping around the X axis. Equivalent to <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>										
<b>StackDepth ?</b>	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>										
<b>StepDocs ?</b> <a href="#">Modified in JDF1.2</a>	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>NumberUp</i> = "4 4" and <i>StepRepeat</i> = "2 2" and <i>StepDocs</i> =:</p> <table border="1" data-bbox="623 1276 1422 1493"> <tr> <td data-bbox="623 1276 1036 1346">           "2 1" (Two documents in X, one in Y)         </td> <td data-bbox="1036 1276 1422 1346">           "1 2" (One document in X, two in Y)         </td> </tr> <tr> <td data-bbox="623 1346 1036 1377">A1 A1 B1 B1</td> <td data-bbox="1036 1346 1422 1377">A1 A1 A2 A2</td> </tr> <tr> <td data-bbox="623 1377 1036 1409">A1 A1 B1 B1</td> <td data-bbox="1036 1377 1422 1409">A1 A1 A2 A2</td> </tr> <tr> <td data-bbox="623 1409 1036 1440">A2 A2 B2 B2</td> <td data-bbox="1036 1409 1422 1440">B1 B1 B2 B2</td> </tr> <tr> <td data-bbox="623 1440 1036 1472">A2 A2 B2 B2</td> <td data-bbox="1036 1440 1422 1472">B1 B1 B2 B2</td> </tr> </table>	"2 1" (Two documents in X, one in Y)	"1 2" (One document in X, two 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" (Two documents in X, one in Y)	"1 2" (One document in X, two 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											

Name	Data Type	Description																									
<i>StepRepeat</i> ?	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 may be unknown. The following examples have <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	3 3 4 4	1 2 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																							
<i>SurfaceContentsBox</i> ? <a href="#">Modified in JDF 1.1A</a>	rectangle	<p>This box, specified in surface-coordinate space, defines the area into which <b>PageCells</b> 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 onto the surface. Note that <i>SurfaceContentsBox</i> does not imply clipping. Clipping is defined by <i>PageCell/@ClipBox</i>.</p> <p>If <i>SurfaceContentsBox</i> is not specified, a device may supply a <i>SurfaceContentsBox</i> that corresponds to the imageable area for the <b>Media</b> used by the device. Otherwise a rectangle with the origin at "0 0" and the dimensions of the <b>Media</b> defined in this resource is assumed. If no <i>Media/@Dimension</i> can be determined, the <i>SurfaceContentsBox</i> is assumed to have its origin at the lower left corner of the <b>Media</b> and to be unbounded in X and Y.</p>																									
<i>VerticalCreep</i> ?	IntegerList	<p>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 the <i>NumberUp</i> attribute.</p> <p>The value for an entry in this list must be between zero and two (2) less than the first value of the <i>NumberUp</i> attribute. An index value outside of this range is ignored. If not specified then no vertical gutters will creep.</p>																									
<i>ImageShift</i> ?	element	<p>Details how to place the grid of <b>PageCells</b> onto the media. The coordinate system is defined so that the X dimension is the first number of the <i>Media Dimension</i> attribute; Y is the second number. <i>ImageShift</i> must be applied before any transformations of the grid of <b>PageCells</b> as specified by <i>Rotate</i> or <i>FitPolicy</i>.</p>																									
<b>InsertSheet</b> *	refelement	Additional sheets to be inserted before, after, or within a job.																									
<b>DeviceMark</b> ?	refelement	Details how device-dependent marks should be generated. If not specified, the marks are device-dependent.																									
<b>FitPolicy</b> ?	refelement	Details how to fit the grid of <b>PageCells</b> onto the media.																									
<b>JobField</b> *	refelement	Specific information about this kind of mark object.																									
<b>Media</b> ?	refelement	Specific information about the media.																									
<b>PageCell</b> ? <a href="#">Modified in JDF 1.1A</a>	element	<b>PageCell</b> elements describe how page contents will be imaged onto individual page cells. Only one page cell size may be specified and is applied to all cells on both <b>Surfaces</b> of a <b>Sheet</b> .																									

## Structure of the PageCell Subelement

PageCell elements describe how page contents will be imaged onto individual page cells. Only one page cell size may be specified and is applied to all cells on both **Surfaces** of a **Sheet**.

Name	Data Type	Description
<p><i>Border ?</i></p> <p><a href="#">Modified in JDF 1.1A</a></p>	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 may 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 will 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 should be marked on each PageCell. The appearance of the marks are defined by the process implementation. Values include:</p> <p><i>CIELABMeasuringField</i>  <i>ColorControlStrip</i>  <i>ColorRegisterMark</i>  <i>CutMark</i>  <i>DensityMeasuringField</i>  <i>IdentificationField</i>  <i>JobField</i>  <i>PaperPathRegisterMark</i>  <i>RegisterMark</i>  <i>ScavengerArea</i></p>
<i>Rotate = "Rotate0"</i>	enumeration	<p>Orthogonal rotation to be applied to the contents in the PageCells. One of:</p> <p><i>Rotate0</i>  <i>Rotate90</i> – 90° counterclockwise rotation.  <i>Rotate180</i> – 180° rotation.  <i>Rotate270</i> – 90° clockwise rotation.</p> <p>For details of orthogonal rotation, refer to Table 2-3, "Matrices and Orientation values used to describe the orientation of a Component," on page 24. If a <i>RotatePolicy</i> value other than "NoRotate" is specified, the value specified in <i>Rotate</i> may be modified accordingly.</p>

Name	Data Type	Description
<a href="#">TrimSize ?</a> <a href="#">Modified in JDF 1.1A</a>	XYPair	Defines the dimensions of the <b>PageCell</b> . 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. <b>FitPolicy</b> defines the default <i>TrimSize</i> in the absence of an explicit <i>TrimSize</i> . If not specified, <i>TrimSize</i> is calculated by subtracting the gutters from the <b>LayoutPreparationParams/@SurfaceContentsBox</b> and dividing by the appropriate <i>NumberUp</i> value.
<b>Color ?</b>	refelement	Color of the border.
<b>DeviceMark ?</b>	refelement	Details how device dependent marks should be generated. Defaults to the value of <b>DeviceMark</b> in the parent <b>LayoutPreparationParams</b> .
<b>FitPolicy ?</b>	refelement	Details how page content is fit into the <b>PageCells</b> . If the dimensions of the page contents vary, <b>FitPolicy</b> is applied to the contents of each cell individually.
<b>ImageShift ?</b>	element	Element which describes how content should be placed into the <b>PageCells</b> . X and Y are specified in the coordinate system of the <b>PageCell</b> .

### Structure of the ImageShift Subelement

**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.

Name	Data Type	Description
<a href="#">PositionX ?</a> <a href="#">Modified in JDF 1.2</a>	enumeration	Indicates how images should be positioned horizontally. <i>ShiftBack</i> and <i>ShiftFront</i> are applied after <i>PositionX</i> and <i>PositionY</i> . Values are: <i>Center</i> – Center the images horizontally without regard to limitations of the printable area. <i>Left</i> – Position the left edge of the images so they are coincident with the left edge of the printable area. <i>Right</i> – Position the right edge of the images so they are coincident with the right edge of the printable area. <i>Spine</i> – Position the images so they are coincident with the vertical binding edge of the printable area. <a href="#">New in JDF 1.2</a> <i>None</i> – Place the images wherever the print data specify.
<a href="#">PositionY ?</a> <a href="#">Modified in JDF 1.2</a>	enumeration	Indicates how images should be positioned vertically. <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 images so they are coincident with the bottom edge of the printable area. <i>Center</i> – Center the images horizontally without regard to limitations of the printable area. <i>Top</i> – Position the top edge of the images so they are coincident with the top edge of the printable area. <i>Spine</i> – Position the images so they are coincident with the horizontal binding edge of the printable area. <a href="#">New in JDF 1.2</a> <i>None</i> – Place the images wherever the print data specify.
<b>ShiftBack ?</b>	XYPair	The amount in X and Y direction by which the image is to be shifted on the back side. If not specified, <i>ShiftBack</i> is calculated from <i>ShiftFront</i> .
<b>ShiftFront = "0 0"</b>	XYPair	The amount in X and Y direction by which the image is to be shifted on the front side.

## 7.2.100 LongitudinalRibbonOperationParams

[Deprecated in JDF 1.1](#). See "LongitudinalRibbonOperationParams" on page 765 for details of this deprecated resource.

## 7.2.101 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:	<b>ManualLabor</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>LaborType</i>	NMTOKENS	List of types of manual labor that are performed.

## 7.2.102 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:	<b>ExposedMedia, DigitalPrintingParams, InsertSheet, InterpretingParams, LayoutPreparationParams, RenderingParams, Sheet, StrippingParams, Tile</b>
Example Partition:	<i>Location, SheetName, Side, SignatureName, TileID, WebName</i>
Input of processes:	<b>ConventionalPrinting, ContactCopying, Cutting, DigitalPrinting, ImageSetting, Proofing</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>BackCoatings</i> ?	enumeration	Enumeration options are identical to <i>FrontCoatings</i> (see below), but applied to the back surface of the media. When not specified, defaults to the value of <i>FrontCoatings</i> .
<i>BackGlossValue</i> ? <a href="#">New in JDF 1.2</a>	double	Gloss of the back surface of the media in gloss units as defined by ISO 8254-1:1995 <i>Paper and board—Measurement of specular gloss—Part 1: 75° gloss with a converging beam, TAPPI method</i> . When not known, <i>BackGlossValue</i> defaults to the value of <i>FrontGlossValue</i> .
<i>Brightness</i> ? <a href="#">Clarified in JDF 1.2</a>	double	Reflectance percentage of diffuse blue reflectance as defined by ISO2470 - ISO 2470:1977 <i>Paper and board—Measurement of diffuse blue reflectance factor (ISO brightness)</i> . The reflectance is reported per ISO 2470 as the diffuse blue reflectance factor of the paper or board in percent to the nearest 0.5% reflectance factor. If one value is specified, Brightness applies to the front and back. If two values are specified the first value applies to the front and the second applies to the back. See also <i>CIEWhiteness</i> .

Name	Data Type	Description
<a href="#">CIETint ?</a> <a href="#">New in JDF 1.2</a>	double	Average CIE tint value. Average CIE tint is calculated according to equations given in TAPPI T 560 — “ <i>CIE Whiteness and Tint of Paper and Paper Board (using d/0 , diffuse illumination and normal viewing)</i> ”.
<a href="#">CIWhiteness ?</a> <a href="#">New in JDF 1.2</a>	double	Average CIE whiteness value. Average CIE whiteness is calculated according to equations given in TAPPI T 560 — “ <i>CIE Whiteness and Tint of Paper and Paper Board (using d/0 , diffuse illumination and normal viewing)</i> ”.
<a href="#">ColorName ?</a> <a href="#">New in JDF 1.1</a> <a href="#">Deprecated in JDF 1.2</a>	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 <b>Color</b> defined in a <b>ColorPool</b> resource that is linked to the process using this <b>Media</b> resource. In JDF 1.2 and beyond, use <i>MediaColorName</i> and <i>MediaColorNameDetails</i> .
<a href="#">Dimension ?</a> <a href="#">Modified in JDF 1.1</a>	XYPair	The X and Y dimensions of the chosen medium, measured in points. The X,Y values of <i>Dimension</i> establishes the user coordinate system into which content is mapped, (i.e., the origin is in the lower left corner of the rectangle defined by 0 0 X Y.) In case of <i>Roll</i> media, the X coordinate specifies the reel width and the Y coordinate specifies the length of the web in points. If a <i>Dimension</i> coordinate is unknown, the value must be “0”. If not specified, the dimension is unknown. If either or both X or Y = “0” (i.e., unknown), the default orientation is assumed to be portrait, (i.e., Y>X).
<a href="#">FrontCoatings ?</a> <a href="#">Modified in JDF 1.2</a>	enumeration	What preprocess coating has been applied to the front surface of the media. Possible values are: <i>None</i> – No coating. <i>Coated</i> – A coating of a system-specified type. <a href="#">New in JDF 1.2</a> <i>Glossy</i> <i>HighGloss</i> <i>InkJet</i> – A coating intended for use with inkjet technology. <a href="#">New in JDF 1.2</a> <i>Matte</i> <i>Satin</i> <i>Semigloss</i>
<a href="#">FrontGlossValue ?</a> <a href="#">New in JDF 1.2</a>	double	Gloss of the front side of the of the media in gloss units as defined by ISO 8254-1:1995 <i>Paper and board—Measurement of specular gloss—Part 1: 75° gloss with a converging beam, TAPPI method</i> . Refer also to TAPPI T 480 om-92 — “ <i>Specular gloss of paper and paper board at 75 degrees</i> ” for examples of gloss calculation.
<a href="#">Grade ?</a> <a href="#">Clarified in JDF 1.1A</a>	integer	The intended <i>Grade</i> of the media on a scale of 1 through 5. <i>Grade</i> is ignored if <i>MediaType</i> is not “ <i>Paper</i> ”. <i>Grade</i> of paper material is defined in accordance with the paper “types” set forth in [iso12647-2]. Offset printing paper types are defined with the following integer values: 1 – Gloss-coated paper. 2 – Matt-coated paper. 3 – Gloss-coated, web paper. 4 – Uncoated, white paper. 5 – Uncoated, yellowish paper. <b>Note:</b> ISO 12647-2 paper <i>type</i> attribute values do NOT align with U.S. GRACOL paper <i>grade</i> attribute values, (e.g., ISO 12647-2 type 1 does not equal U.S. GRACOL grade 1).

Name	Data Type	Description
<a href="#">GrainDirection ?</a> <a href="#">New in JDF 1.1</a>	enumeration	Direction of the grain in the coordinate system defined by <i>Dimension</i> . Possible values are: <i>ShortEdge</i> – Along the shorter axis as defined by <i>Dimension</i> . <i>LongEdge</i> – Along the longer axis as defined by <i>Dimension</i> .
<a href="#">HoleCount ?</a> <a href="#">Deprecated in JDF 1.1</a>	integer	The number of holes that should be punched in the media (either pre- or post-punched). In JDF/1.1, use <i>HoleType</i> , <i>Hole</i> or <i>HoleLine</i> , which includes the number of holes.
<a href="#">HoleType = "None"</a> <a href="#">New in JDF 1.1</a>	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 663. Allowed values are: <i>None</i> – No holes. <i>Explicit</i> – Holes are defined in an array of <i>Hole</i> or <i>HoleLine</i> . other values defined in "JDF/CIP4 Hole Pattern Catalog" on page 663.
<a href="#">ImagableSide ?</a>	enumeration	Side of the chosen medium that may be marked. Possible values are: <i>Front</i> <i>Back</i> <i>Both</i> <i>Neither</i>
<a href="#">LabColorValue ?</a> <a href="#">New in JDF 1.2</a>	Lab-Color	<i>LabColorValue</i> is the CIELAB color value of the media, computed as specified in TAPPI T527 — “Color of Paper and Paperboard (d/0 geometry)”
<a href="#">MediaColorName ?</a> <a href="#">Modified in JDF 1.1</a>	Named-Color	A name for the color. Allowed values are defined in Section A.3.3.2, <i>NamedColor</i> . If more specific, specialized, or site-defined media color names are needed, use <i>MediaColorNameDetails</i> .
<a href="#">MediaColorNameDetails ?</a> <a href="#">New in JDF 1.2</a>	string	A more specific, specialized, or site-defined name for the media color. If <i>MediaColorNameDetails</i> is supplied, <i>MediaColorName</i> must also be supplied.
<a href="#">MediaSetCount ?</a>	integer	When the input media is grouped in sets, identifies the number of pieces of media in each set. For example, if the <i>MediaTypeDetails</i> is “ <i>PreCutTabs</i> ”, a <i>MediaSetCount</i> of “5” would indicate that each set includes five tab sheets.
<a href="#">MediaType ?</a> <a href="#">Modified in JDF 1.2</a>	enumeration	Describes the medium being employed. Possible values are: <i>Disc</i> – CD or DVD disc to be printed on. <i>EndBoard</i> – End board used in the <b>Bundling</b> process. <i>EmbossingFoil</i> <i>Film</i> <i>Foil</i> <i>LaminatingFoil</i> <i>Other</i> – Not one of the defined values. <i>Paper</i> <i>Plate</i> <i>ShrinkFoil</i> <i>Transparency</i> <i>Unknown</i> – <a href="#">Deprecated in JDF 1.2</a>

Name	Data Type	Description
<p><a href="#">MediaTypeDetails ?</a> <a href="#">Modified in JDF 1.2</a></p>	NMTO-KEN	<p>Additional details of the chosen medium. If <i>MediaTypeDetails</i> is specified, <i>MediaType</i> must be specified. Possible values include:</p> <p><i>Aluminum</i> – Conventional press plate.</p> <p><i>Cardboard</i></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>CtPVisiblePhotoPolymer</i> – Visible light CtP plate with photo polymer process.</p> <p><i>CtPVisibleSilver</i> – Visible light CtP plate with silver halide process.</p> <p><i>CtPThermal</i> – Thermal CtP plate.</p> <p><i>DryFilm</i></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>ImageSetterPaper</i> – Contact paper as replacement for film.</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>PlateUV</i> – Press plate for the UV process.</p> <p><i>Polyester</i> – CtP press plate.</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, includes generic paper.</p> <p><i>TabStock</i> – Media with tabs, either precut or full-cut.</p> <p><i>Tractor</i> – Tractor feed with holes.</p> <p><i>WetFilm</i> – Conventional photographic film.</p>
<p><a href="#">MediaUnit = "Sheet"</a> <a href="#">Modified in JDF 1.2</a></p>	enumeration	<p>Describes the format of the media as it is delivered to the device. Possible values are:</p> <p><i>Continuous</i> – Continuously connected sheets which may be fan folded. <a href="#">New in JDF 1.2</a></p> <p><i>Roll</i></p> <p><i>Sheet</i> – Individual cut sheets.</p>



Name	Data Type	Description
<a href="#">Opacity ?</a> <a href="#">Modified in JDF 1.2</a>	enumeration	The opacity of the media. See <i>OpacityLevel</i> to specify the degree of opacity for any of these values. Possible values are: <i>Opaque</i> – The media is opaque. With two-sided printing the printing on the other side does not show through under normal incident light. <i>Translucent</i> – The media is translucent to a system specified amount. For example, translucent media can be used for back lit viewing. <a href="#">New in JDF 1.2</a> <i>Transparent</i> – The media is transparent.
<a href="#">OpacityLevel ?</a> <a href="#">New in JDF 1.2</a>	double	Normalized TAPPI opacity, (Cn), as defined and computed in ISO 2471:1998 “ <i>Paper and board—Determination of opacity (paper backing)—Diffuse reflectance method</i> ”. Refer also to TAPPI T 519 — “ <i>Diffuse opacity of paper (d/0 paper backing)</i> ” for calculation examples.
<a href="#">Polarity ?</a>	enumeration	Polarity of the chosen medium. Possible values are: <i>Positive</i> <i>Negative</i>
<i>PrePrinted = “false”</i>	boolean	Indicates whether the media is preprinted.
<a href="#">Recycled ?</a> <a href="#">Deprecated in JDF 1.2</a>	boolean	If “ <i>true</i> ”, recycled media is requested. If not specified, the <b>Media</b> may or may not have recycled content. In JDF 1.2 and beyond, use <i>RecycledPercentage</i> .
<a href="#">RecycledPercentage ?</a> <a href="#">New in JDF 1.2</a>	double	The percentage, between 0 and 100, of recycled material that the media must contain.
<a href="#">RollDiameter ?</a>	double	Specifies diameter of a roll, in points.
<a href="#">ShrinkIndex ?</a> <a href="#">New in JDF 1.1</a>	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.
<a href="#">StockType ?</a> <a href="#">New in JDF 1.1</a>	NMTO-KEN	Strings describing the available stock. Examples include: <i>Bristol</i> <i>Cover</i> <i>Bond</i> <i>Newsprint</i> <i>Index</i> <i>Offset</i> – This includes book stock. <i>Tag</i> <i>Text</i>
<a href="#">Texture ?</a> <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	NMTO-KEN	The intended texture of the media. Examples include: <i>Antique</i> – Rougher than vellum surface. <i>Calendared</i> – Extra smooth or polished, uncoated paper. <i>Linen</i> – Texture of coarse woven cloth. <i>Smooth</i> <i>Stipple</i> – Fine pebble finish. <i>Uncalendared</i> – Rough, unpolished, and uncoated papers. <a href="#">New in JDF 1.2</a> <i>Vellum</i> – Slightly rough surface.

Name	Data Type	Description
<i>Thickness ?</i>	double	The thickness of the chosen medium, measured in microns [ $\mu\text{m}$ ].
<i>UserMediaType ?</i> <a href="#">Deprecated in JDF 1.1</a>	NMTO-KEN	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. <i>UserMediaType</i> has been merged into <i>MediaTypeDetails</i> in JDF 1.1.
<i>Weight ?</i>	double	Weight of the chosen medium, measured in grams per square meter [ $\text{g}/\text{m}^2$ ]. See "North American Media Weight Explained" on page 627 for details on converting North American paper weights to $\text{g}/\text{m}^2$ .
<i>Color ?</i> <a href="#">Deprecated in JDF 1.1</a>	reference	A <b>Color</b> resource that provides the color of the chosen medium.
<i>ColorMeasurementConditions ?</i> <a href="#">New in JDF 1.2</a>	reference	Detailed description of the measurement conditions for color measurements used to measure <i>LacColorValue</i> .

### 7.2.103 MediaSource

[Deprecated in JDF 1.1](#) See "MediaSource" on page 766 for details of this deprecated resource.

### 7.2.104 MISDetails

[New in JDF 1.2](#)

MISDetails is a container for MIS related information. It is referenced by Audit elements and JMF messages.

#### Resource Properties

**Resource class:** ResourceElement

**Resource referenced by:** —

**Example Partition:** —

**Input of processes:** —

**Output of processes:** —

#### Resource Structure

Name	Data Type	Description
<i>CostType ?</i>	enumeration	Whether or not this <b>MISDetails</b> is chargeable to the customer or not. One of: <i>Chargeable</i> <i>NonChargeable</i>
<i>DeviceOperationMode ?</i>	enumeration	<i>DeviceOperationMode</i> shows the operation mode that the device is in. It is used to show if the production of a device is aimed at producing good products or not. The latter case applies when a device is used to produce a job for testing, calibration, etc. without the intention to produce good output. <i>Productive</i> – The device is used to produce good product. Any times recorded in this mode should 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 should 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>WorkType ?</i>	enumeration	<p>Definition of the work type for this <b>MISDetails</b>, (i.e., whether or not this <b>MISDetails</b> relates to originally planned work, an alteration, or rework). One of:</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 <b>MISDetails</b>, (i.e., why the work was done). Values include:</p> <p><i>CustomerRequest</i> – The customer requested change(s) requiring the work.</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>EquipmentMalfunction</i> – Equipment used to produce the resource malfunctioned, resource must be created again.</p> <p><i>UserError</i> – Incorrect operation of equipment or incorrect creation of resource requires creating the resource again.</p>

### 7.2.105 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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	—
<b>Input of processes:</b>	<b><i>Numbering</i></b>
<b>Output of processes:</b>	—

#### Resource Structure

Name	Data Type	Description
NumberingParam *	element	Set of parameters for one numbering machine

#### Structure of NumberingParam Subelement

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 = "1"</i>	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.

### 7.2.106 ObjectResolution

ObjectResolution defines a resolution depending on *SourceObjects* data types.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>InterpretingParams, RenderingParams, TrappingDetails</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>AntiAliasing</i> ? <a href="#">New in JDF 1.2</a>	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 may include: <i>AntiAlias</i> – Anti-aliasing must be applied. The algorithm is system specified. <i>None</i> – Anti-aliasing must not be applied.
<i>Resolution</i>	XYPair	Horizontal and vertical output resolution in DPI.
<i>SourceObjects</i> = "All"	enumerations	Identifies the class(es) of incoming graphical objects to render at the specified resolution. Possible values are: <i>All</i> <i>ImagePhotographic</i> – Contone images. <i>ImageScreenShot</i> – Images largely comprised of rasterized vector art. <i>LineArt</i> – Vector objects other than text. <i>SmoothShades</i> – Gradients and blends. <i>Text</i>

### 7.2.107 OrderingParams

Attributes of the **Ordering** process, which results in an acquisition.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>Ordering</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>Amount</i>	double	Amount of the ordered resource.
<i>Unit</i>	string	Unit of measurement for <i>Amount</i> .
Comment	telem	<b>OrderingParams</b> require a <b>Comment</b> element that contains a human-readable description of what to order.
<b>Company</b> ? <a href="#">Deprecated in JDF 1.1</a>	refelement	Address and further information of the <b>Company</b> responsible for this order. Replaced with <b>Contact/Company</b> in JDF 1.1.
<b>Contact</b> * <a href="#">New in JDF 1.1</a>	refelement	Address and further information of the <b>Contact</b> responsible for this order.

## 7.2.108 PackingParams

[Deprecated in JDF 1.1](#)

The PackingParams resource has been deprecated in JDF 1.1 and beyond. It is replaced by the individual resources used by the processes defined in Section 6.6.48.4, Numbering and Section 6.6.48.5, Packaging Processes. See "PackingParams" on page 767 for details of this deprecated resource.

## 7.2.109 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:	<b>LayoutElement, RunList</b>
Example Partition:	<i>PartVersion</i>
Input of processes:	—
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>AssemblyID</i> ?	string	ID of the <b>Assembly</b> or <b>AssemblySection</b> that this finished page belongs to.
<i>HasBleeds</i> ?	boolean	If " <i>true</i> ", the file has bleeds.
<i>IsBlank</i> ?	boolean	If " <i>false</i> ", the <b>PageData</b> has no content marks and is blank.
<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 default user space. 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 default user space of the source document page. 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 default user space. 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 <b>PageList</b> represents a template that must be completed with information from a database.

Name	Data Type	Description
<b>ColorPool ?</b>	refelement	Definition of the color details.
<b>ImageCompressionParams ?</b>	refelement	Specification of the image compression properties.
<b>PageData *</b>	element	Details of the individual finished page. <b>PageData</b> elements are referred to by their index in the <b>PageList</b> . <b>PageData</b> elements should, therefore, not be removed or inserted in a position other than the end of the list.
<b>ScreeningParams ?</b>	refelement	Specification of the screening properties.
<b>SeparationSpec *</b>	element	List of separation names defined in the element.
<b>ElementColorParams ?</b>	refelement	Color details of the page list.

### Properties of the PageData Subelement

**PageData** defines the additional metadata of individual finished pages such as pagination details. **PageData** elements are referred to by index of the **PageData** in the **PageList**.

If the **PageList** is partitioned, the index refers to **PageData** elements in the respective leaves of the partitioned **PageList**. The index restarts at 0 with each partitioned leaf.

Name	Data Type	Description
<b>AssemblyID ?</b>	string	ID of the <b>Assembly</b> or <b>AssemblySection</b> that this finished page belongs to. If not specified, defaults to the value of <b>PageList/@AssemblyID</b> .
<b>CatalogID ?</b>	string	Identification of the resource, (e.g., in a catalog environment). If not specified, defaults to the value of <b>PageList/@CatalogID</b> .
<b>CatalogDetails ?</b>	string	Additional details of a resource in a catalog environment. If not specified, defaults to the value of <b>PageList/@CatalogDetails</b> .
<b>FoldOutPages ?</b>	IntegerList	Page indices in the <b>PageList</b> 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 <b>PageData</b> . If not specified, the <b>PageData</b> does not describe a part of a foldout.
<b>HasBleeds ?</b>	boolean	If <i>"true"</i> , the file has bleeds. If not specified, defaults to the value of <b>PageList/@HasBleeds</b> .
<b>IsBlank ?</b>	boolean	If <i>"false"</i> , the <b>PageData</b> has no content marks and is blank. If not specified, defaults to the value of <b>PageList/@IsBlank</b> .
<b>IsPrintable ?</b>	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 = "false"</i> . If not specified, defaults to the value of <b>PageList/@IsPrintable</b> .
<b>IsTrapped ?</b>	boolean	If <i>"true"</i> , the file has been trapped. If not specified, defaults to the value of <b>PageList/@IsTrapped</b> .
<b>JobID ?</b>	string	ID of the job that this finished page belongs to. If not specified, defaults to the value of <b>PageList/@JobID</b> .
<b>PageLabel ?</b>	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 may be different than the position of the page in the finished document.

Name	Data Type	Description
<i>PageLabelPrefix</i> ?	string	Prefix of the identification of the reader page as it is displayed on the finished page. For instance "C - ", if the reader pages are labeled "C - 1", "C - 2", etc. If not specified, defaults to the value of <b>PageList/@PageLabelPrefix</b> .
<i>PageLabelSuffix</i> ?	string	Suffix of the identification of the reader page as it is displayed on the finished page. For instance " - a", if the pages are labeled "C - 1 - a", "C - 2 - a", etc. If not specified, defaults to the value of <b>PageList/@PageLabelSuffix</b> .
<i>ProductID</i> ?	string	An ID of the page as defined in the MIS system. If not specified, defaults to the value of <b>PageList/@ProductID</b> .
<i>SourceBleedBox</i> ?	rectangle	A rectangle that describes the bleed area of the page to be included. This rectangle is expressed in the default user space. If not specified, defaults to the value of <b>PageList/@SourceBleedBox</b> .
<i>SourceClipBox</i> ?	rectangle	A rectangle that defines the region of the finished page to be included. This rectangle is expressed in the default user space of the source document page. If not specified, defaults to the value of <b>PageList/@SourceClipBox</b> .
<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 default user space. If not specified, defaults to the value of <b>PageList/@SourceTrimBox</b> .
<i>Template</i> ?	boolean	Template is "false" when this page is self-contained. This attribute is "true" if the <b>PageList</b> represents a template that must be completed with information from a database. If not specified, defaults to the value of <b>PageList/@Template</b> .
<b>ImageCompressionParams</b> ?	refelement	Specification of the image compression properties. If not specified, defaults to the value of <b>PageList/ImageCompressionParams</b> .
<b>ScreeningParams</b> ?	refelement	Specification of the screening properties. If not specified, defaults to the value of <b>PageList/ScreeningParams</b> .
<b>SeparationSpec</b> *	element	List of separation names defined in the element. If none is specified, defaults to the values of <b>PageList/SeparationSpec</b> .
<b>ElementColorParams</b> ?	refelement	Color details of the PageData element. If not specified, defaults to the value of <b>PageList/ElementColorParams</b> .

### 7.2.110 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:	—

#### Resource Structure

Name	Data Type	Description
<i>Pattern</i> ?	string	Name of the palletizing pattern. Used to store a predefined pattern that defines the layers and positioning of individual component on the pallet.
<i>MaxHeight</i> ?	double	Maximum height of a loaded pallet in points.
<i>MaxWeight</i> ?	double	Maximum weight of a loaded pallet in grams.

### 7.2.111 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:	<b><i>Palletizing</i></b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>PalletType</i>	NMTOKEN	Type of pallet used. Examples include: <i>2Way</i> – Two-way entry. <i>4Way</i> – Four-way entry. <i>Euro</i> – Standard 1*1 m Euro pallet.
<i>Size ?</i>	XYPair	Describes the length and width of the pallet, in points, (e.g., 3500 3500). If not specified, the size is defined by <i>PalletType</i> .

### 7.2.112 PDFToPSConversionParams

This resource specifies a set of configurable options that may be used by processes that generate PostScript files from PDF files. Font controls are applied in the following order:

- 1 IncludeBaseFonts
- 2 IncludeEmbeddedFonts
- 3 IncludeType1Fonts
- 4 IncludeType3Fonts
- 5 IncludeTrueTypeFonts
- 6 IncludeCIDFonts

For example, an embedded Type-1 font follows the rule for embedded fonts, not the rule for Type-1 fonts. In other words, if *IncludeEmbeddedFonts* is "*true*", and *IncludeType1Fonts* is "*false*", embedded Type-1 fonts would be included in the PostScript stream.

#### Resource Properties

Resource class:	Parameter
Resources referenced:	—
Example Partition:	<i>DocIndex, RunIndex, RunTags, SheetName, Side, SignatureName</i>
Input of processes:	<b><i>PDFToPSConversion</i></b>
Output of processes:	—

#### Resource Structure

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 <i>BoundingBox</i> DSC comment, in <i>CenterCropBox</i> calculations, and for <i>SetPageDevice</i> .
<i>CenterCropBox = "true"</i>	boolean	If " <i>true</i> ", CropBox output is centered on the page when the Crop-Box < <i>MediaBox</i> .



Name	Data Type	Description
<i>GeneratePageStreams</i> = "false"	boolean	If "true", the process emits individual streams of data for each page in the <b>RunList</b> .
<i>IgnoreAnnotForms</i> = "false"	boolean	If "true", ignores annotations that contain an XObject form.
<i>IgnoreBG?</i> = "true" <a href="#">New in JDF 1.1</a>	boolean	Ignores the BG, BG2 parameters in the PDF ExtGState dictionary.
<i>IgnoreColorSeps</i> = "false"	boolean	If "true", ignores images for Level-1 separations.
<i>IgnoreDeviceExtGState?</i> <a href="#">Deprecated in JDF 1.1</a>	boolean	If "true", ignores all device-dependent extended graphic state parameters. This overrides <i>IgnoreHalftones</i> . The following parameters should 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</i> = "true"	boolean	If "true", ignores DSC (Document Structuring Conventions).
<i>IgnoreExternStreamRef</i> = "false"	boolean	If an image resource uses an external stream and <i>IgnoreExternStreamRef</i> = "true", ignores code that points to the external file.
<i>IgnoreHalftones</i> = "false"	boolean	If "true", ignores any halftone screening in the PDF file.
<i>IgnoreOverprint</i> = "true" <a href="#">New in JDF 1.1</a>	boolean	Ignores the OP parameters in the PDF ExtGState dictionary.
<i>IgnorePageRotation</i> = "false"	boolean	If "true", ignores a "concat" 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 should 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 <i>ShowPage</i> in PostScript files
<i>IgnoreTransfers</i> = "true" <a href="#">New in JDF 1.1</a>	boolean	Ignores the TR, TR2 parameters in the PDF ExtGState dictionary.
<i>IgnoreTTFontsFirst</i> = "false"	boolean	If "true", ignores TrueType fonts before any other fonts.
<i>IgnoreUCR</i> = "true" <a href="#">New in JDF 1.1</a>	boolean	Ignores the UCR, UCR2 parameters in the PDF ExtGState dictionary.

Name	Data Type	Description
<i>IncludeBaseFonts</i> = "IncludeNever"	enumeration	Determines when to embed the base fonts. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeCIDFonts</i> = "IncludeOncePerDoc"	enumeration	Determines when to embed CID fonts. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeEmbeddedFonts</i> = "IncludeOncePerDoc"	enumeration	Determines when to embed fonts in the document that are embedded in the PDF file. This attribute overrides the <i>IncludeType1Fonts</i> , <i>IncludeTrueTypeFonts</i> , and <i>IncludeCIDFonts</i> attributes. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeOtherResources</i> = "IncludeOncePerDoc"	enumeration	Determines when to include all other types of resources in the file. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeProcSets</i> = "IncludeOncePerDoc"	enumeration	Determines when to include ProcSets in the file. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeTrueTypeFonts</i> = "IncludeOncePerDoc"	enumeration	Determines when to embed TrueType fonts. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeType1Fonts</i> = "IncludeOncePerDoc"	enumeration	Determines when to embed Type-1 fonts. Possible values are: <i>IncludeNever</i> <i>IncludeOncePerDoc</i> <i>IncludeOncePerPage</i>
<i>IncludeType3Fonts</i> = "IncludeOncePerPage"	enumeration	Determines when to embed Type-3 fonts. Must always be set to <i>IncludeOncePerPage</i> . It is included here to complete the precedence hierarchy.
<i>OutputType</i> = "PostScript"	enumeration	Describes the kind of output to be generated. Possible values are: <i>PostScript</i> <i>EPS</i>
<i>PSLevel</i> = "2"	integer	Number that indicates the PostScript level.
<i>Scale</i> = "100"	double	Number that indicates the wide-scale factor of documents. Full size = "100".
<i>SetPageSize</i> = "false"	boolean	(PostScript Level 2 only) If "true", sets page size on each page automatically. Use media box for outputting PostScript files and crop box for EPS.

Name	Data Type	Description
<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>SupressRotate</i> = "true". On the other hand, if <i>SupressRotate</i> = "false", the value of the PDF Rotate key for each page is honored, regardless of the dimensions of the pages (as defined by the <i>MediaBox</i> 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.

### 7.2.113 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 may occur within PDL files or may 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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>ColorantControl</b>
<b>Example Partition:</b>	—
<b>Input of processes:</b>	<i>Interpreting</i>
<b>Output of processes:</b>	—

#### Resource Structure

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 element 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 <b>PDLResourceAlias</b> resource.

## 7.2.114 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:	<b>Perforating</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
Perforate *	element	Defines one or more Perforate lines.

### Structure of the Perforate element

Perforate describes one perforated line.

Name	Data Type	Description
<i>Depth</i> ? <a href="#">New in JDF 1.2</a>	double	Depth of the perforation, in microns [ $\mu\text{m}$ ].
<i>RelativeStartPosition</i> ? <a href="#">New in JDF 1.2</a>	XYPair	Relative starting position of the tool. <i>RelativeStartPosition</i> is always based on the complete size of the input <b>Component</b> 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 <b>Component</b> .
<i>RelativeWorkingPath</i> ? <a href="#">New in JDF 1.2</a>	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 <b>Component</b> 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 <b>Component</b> .
<i>StartPosition</i> ? <a href="#">Modified in JDF 1.2</a>	XYPair	Starting position of the tool. If both <i>StartPosition</i> and <i>RelativeStartPosition</i> are specified, <i>RelativeStartPosition</i> is ignored. At least one of <i>StartPosition</i> or <i>RelativeStartPosition</i> must be specified.
<i>TeethPerDimension</i> ?	double	Number of teeth in a given perforation extent in teeth/point. MicroPerforation is defined by specifying a large number of teeth ( <i>TeethPerDimension</i> >1000).
<i>WorkingDirection</i>	enumeration	Direction from which the tool is working. Possible values are: <i>Top</i> – From above. <i>Bottom</i> – From below.
<i>WorkingPath</i> ? <a href="#">Modified in JDF 1.2</a>	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.

### 7.2.115 Person

This resource provides detailed information about a person. It also has the ability to specify different communication channels to this person. The structure of the resource is derived from the vCard format. It contains all of the same name subtypes (N:) of the identification and the title of the organizational properties. The corresponding XML types of the vCard are quoted in the description field of the table below.

#### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>Contact, Employee</b>
<b>Example Partition:</b>	—
<b>Input of processes:</b>	—
<b>Output of processes:</b>	—

#### Resource Structure

Name	Data Type	Description
<i>AdditionalNames ?</i>	string	Additional names of the contact person (vCard: N:other).
<i>FamilyName ?</i>	string	The family name of the contact person (vCard: N:family).
<i>FirstName ?</i>	string	The first name of the contact person (vCard: N:given).
<i>JobTitle ?</i>	string	Job function of the person in the company or organization (vCard: title).
<i>NamePrefix ?</i>	string	Prefix of the name, may include title (vCard: N:prefix).
<i>NameSuffix ?</i>	string	Suffix of the name (vCard: N:suffix).
<b>Address ?</b> <a href="#">New in JDF 1.2</a>	refelement	Address of the person.
<b>ComChannel *</b>	refelement	Communication channels to the person.

### 7.2.116 PlaceholderResource

This resource is used to link *ProcessGroup* nodes when the exact nature of interchange resources is still unknown. In this way, a skeleton of process networks can be constructed, with the **PlaceholderResource** resources serving as place holders in lieu of the appropriate resources. This resource needs no structure besides that provided in an abstract **RESOURCE** element as it has no inherent value except as a stand-in for other resources.

#### Resource Properties

<b>Resource class:</b>	Placeholder
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	—
<b>Input of processes:</b>	any <i>ProcessGroup</i> nodes
<b>Output of processes:</b>	any <i>ProcessGroup</i> nodes

#### Resource Structure

The resource has no additional structure.

### 7.2.117 PlasticCombBindingParams

This resource describes the details of the **PlasticCombBinding** process.

#### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	—
<b>Input of processes:</b>	<b>PlasticCombBinding</b>
<b>Output of processes:</b>	—

## Resource Structure

Name	Data Type	Description
<i>Brand ?</i>	string	The name of the comb manufacturer and the name of the specific item.
<i>Color ?</i>	NamedColor	Determines the color of the plastic comb.
<i>Diameter ?</i>	double	The comb diameter is determined by the height of the block of sheets to be bound.
<i>Thickness ?</i>	double	The material thickness of the comb.
<i>Type ?</i> <a href="#">Modified in JDF 1.1</a> <a href="#">Deprecated in JDF 1.2</a>	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 663 exist: <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 <a href="#">The following values are deprecated in JDF 1.1.</a> <i>Euro</i> (Distance = 12 mm; Holes = 7 mm x 3 mm) <i>USA1</i> (Distance = 14.28 mm; Holes = 8 mm x 3 mm) In JDF 1.2 and beyond, use the value implied by <b>HoleMakingParams/@HoleType</b> .
<b>HoleMakingParams ?</b>	refelement	Details of the holes to be made. Note that <b>HoleMakingParams/@Shape</b> is always rectangular by design of the plastic combs.

### 7.2.118 PlateCopyParams

[Deprecated in JDF 1.1](#) See "PlateCopyParams" on page 768 for details of this deprecated resource.

### 7.2.119 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 744.

### 7.2.120 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 746.

### 7.2.121 PreflightParams

[New in JDF 1.2](#)

The **PreflightParams** resource specifies the tests for the **Preflight** process to run. These tests are defined using "Structure of the ActionPool Subelement" on page 504, which defines a list of reporting actions to have for given document object tests defined into a **Test**. (See "Structure of the TestPool Subelement" on page 524.) This section makes use of elements and attributes defined in "Device Capability Definitions" on page 502. It is suggested that readers familiarize themselves with that section and "Concept of the Preflight Process" on page 538.

## Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>Preflight</b>
Output of processes:	

## Resource Structure

Name	Data Type	Description
ActionPool +	refelement	References an ActionPool with its ID.

The ActionPool, as defined in "Structure of the ActionPool Subelement" on page 504, 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.

### Structure of the PreflightAction Subelement.

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 <i>SetRef</i> is not defined, the Test is applied to all the objects.
SetSplitBy = "RunList"	enumeration	This is used to group objects in different ways. Possible values include: <i>Page</i> – Tests are applied on objects page per page. <i>Document</i> – Tests are applied on objects document per document. <i>RunList</i> – All objects of all pages included in all documents are processed together. <i>SetSplitBy</i> is only used when <i>SetRef</i> is defined in order to create sets on a page-per-page or document-per-document basis. For instance, if you want to get the list of separations per page, <i>SetSplitBy</i> should be set to "Page". In such a case, the report's content (as long as the <i>PRItem</i> is defined properly for the Action ) will be grouped by page.

Test elements make use of Evaluation subelements that define various basic preflight testing functions that can be combined together in order to build preflight test. In order to specify basic preflight tests using Evaluation, the subelement BasicPreflightTest is used. **Note:** The BasicPreflightTest includes a PreflightArgument subelement that is defined below.

### Structure of the BasicPreflightTest Subelement

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.

Name	Data Type	Description
DevNS = "http://www.CIP4.org/JDFSchema_1_1"?	URI	Namespace of the test that is described by <i>Name</i> in this BasicPreflightTest element.
ListType?	enumeration	Specifies what type of list or object the basic preflight test describes. The allowed values are identical to those defined in the <i>ListType</i> attribute of "Structure of the Abstract State Subelement" on page 508.
MaxOccurs = "1"	integer	Maximum number of elements in the list described by this BasicPreflightTest, (e.g., the maximum number of integers in an integer list). If <i>MaxOccurs</i> is not "1", the BasicPreflightTest element refers to a list or RangeList of values, (e.g. a NameEvaluation will allow a list of NMTOKENS).

Name	Data Type	Description
<i>MinOccurs</i> = "1"	integer	Minimum number of elements in the list described by this <b>BasicPreflightTest</b> . Default = "1", (i.e., it is an individual value). If <i>MinOccurs</i> is not "1", the <b>BasicPreflightTest</b> element refers to a list or <b>RangeList</b> of values, (e.g., a <b>NameEvaluation</b> will allow a list of <b>NMTOKENS</b> ).
<i>Name</i> ?	NMTOKEN	Local name of the preflight constraint that is evaluated by this <b>BasicPreflightTest</b> . Valid <i>Name</i> values for the JDF <i>NameSpace</i> are defined in "PreflightParams" on page 434; preflight tests are defined through the use of constraints.
Preflightargument ?	element	Additional arguments for the preflight test. For details see "PreflightParams" on page 434 for the definition of <b>PreflightArgument</b> and constraints upon which preflight tests are defined.

### Structure of the PreflightArgument Subelement

This subelement is used by **BasicPreflightTest** when additional data are needed to determine object property.

Name	Data Type	Description
BoxArgument ?	element	Used by the <b>InsideBox</b> and <b>OutsideBox</b> tests.
BoxToBoxDifference ?	element	Used by the <b>BoxToBoxDifference</b> test.

### Structure of the BoxArgument Subelement

Name	Data Type	Description
<i>Box</i>	enumeration	The box type used to verify inclusion or exclusion. Refer to "Box Properties" on page 542 for a description of the valid types of boxes
<i>MirrorMargins</i> ?	enumeration	The <i>MirrorMargins</i> attribute allows the flip of the <i>Offset</i> value depending on the <b>RunList</b> index. When the index is even, the original <i>Offset</i> value is preserved. When the index is odd, the <i>Offset</i> value is flipped.  With a value of "Vertical", you turn [l b r t] into [r b l t]. With a value of "Horizontal", you turn [l b r t] into [l t r b]. If not specified, the value of <i>Offset</i> is not changed.
<i>Offset</i> ?	Rectangle	The offset to build real rectangle to which test is made.
<i>Overlap</i> = "false"	boolean	Explains if overlap is allowed to check inclusion or exclusion.

### Structure of BoxToBoxDifference Subelement.

Name	Data Type	Description
<i>FromBox</i> ?	enumeration	The "From" box used for <b>BoxToBoxDifference</b> calculation. Refer to "Box Properties" on page 542 for a description of the valid types of boxes
<i>ToBox</i> ?	enumeration	The "To" box used for <b>BoxToBoxDifference</b> calculation. Refer to "Box Properties" on page 542 for a description of the valid types of boxes

## 7.2.122 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 746.



## 7.2.123 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 502. It is suggested that reader's familiarize themselves with that section and "Concept of the Preflight Process" on page 538.

### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	all processes
Output of processes:	<b>Preflight</b>

### Resource Structure

Name	Data Type	Description
<b>PreflightParams</b>	refelement	References the <b>PreflightParams</b> that was used to create this report.
<b>PreflightReportRulePool</b>	refelement	References the <b>PreflightReportRulePool</b> that was used to create this report.
<b>RunList</b>	refelement	References the <b>RunList</b> that was used to create this report.
PRItem *	element	Describes the <b>Actions</b> that produced an error or a warning.
<i>ErrorState ?</i>	enumerations	Describes the type of errors that occurred during preflighting when the <b>Preflight</b> process does not understand certain preflight tests or cannot apply them to the given objects. Possible values include: <i>TestNotSupported</i> <i>TestWrongPDL</i> If not specified, no errors occurred.
<i>ErrorCount</i>	integer	The count of errors that were encountered while preflighting the job.
<i>WarningCount</i>	integer	The count of warnings that were encountered while preflighting the job.

### Structure of PRItem Subelement

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 PRRule.

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 may reduce the size of the PreflightReport by removing PRGroup and PROccurrence items within a PRGroup), this will not invalidate the **PreflightReport**.

Name	Data Type	Description
<i>ActionRef</i>	IDREF	References the <b>PreflightParams/ActionPool/Action</b> that triggered this PRItem.
<i>Occurences</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>	IntgerRange-List	All run indices where there is an object that gives an error on that page.

Name	Data Type	Description
PRError *	element	Describes the errors that were found while running this preflight test.
PRGroup *	element	Describes the ACTIONS that produced an error or a warning.

### Structure of the PRError Subelement

The PRError structure is used to describe generic errors that occurred while evaluating an object property while executing a Test.

Name	Data Type	Description
ErrorType	enumeration	Value is one of "TestWrongPDL" or "TestNotSupported".
Value	NMTOKEN	The name of the object property that was being tested when the process error occurred.

### Structure of the PRGroup Subelement

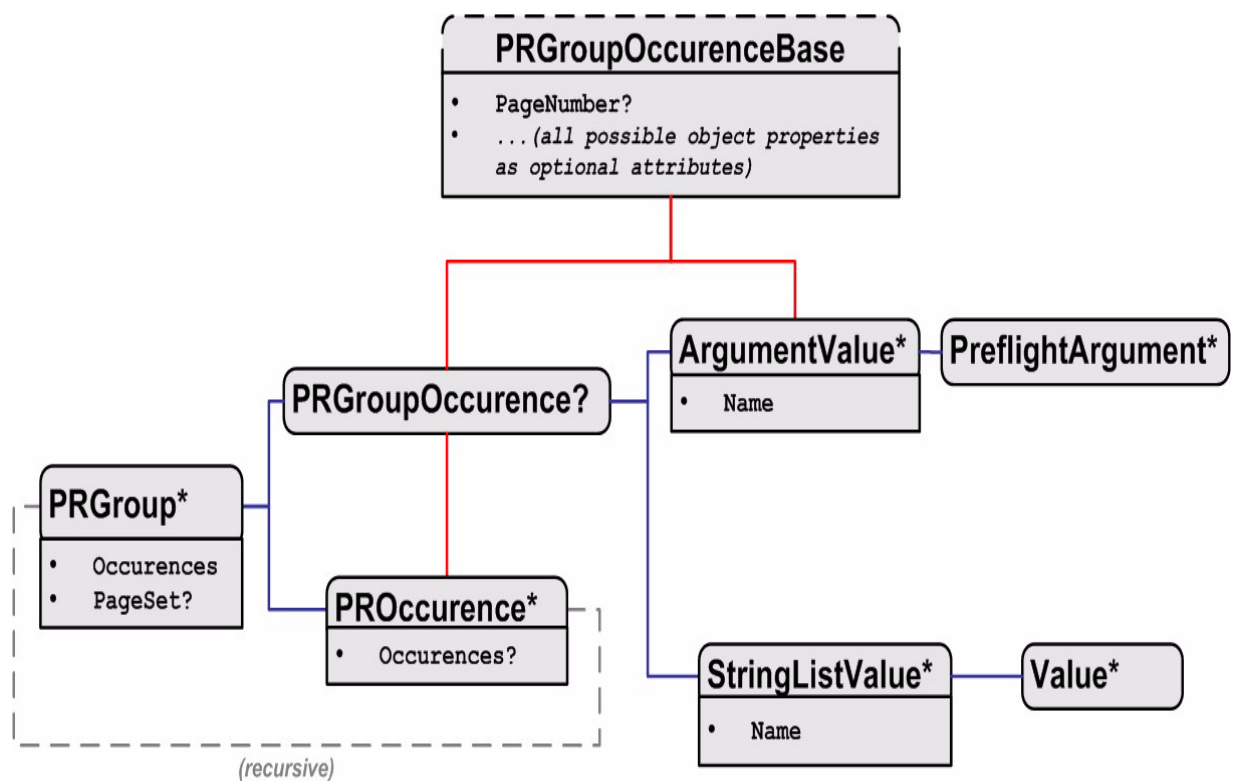


Figure 7.15: PRGroup Structure

The PRGroup structure is used to describe a group of document objects that share common properties and that failed the Action.

Name	Data Type	Description
Occurences	integer	The number of occurrences of objects of this group that failed the Action. When the Actions describes a set-test, this is the number of set-objects.
PageSet ?	IntgerRange-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 <b>PreflightReportRulePool/PRRule/@GroupBy</b> .
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 **PROccurrences**. These are used to describe all the document objects that failed the preflight test. **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 **PROccurrences** whereby the second level occurs as a child element of **PROccurrence**.
  - The top level describes the set objects that failed the preflight test. Just as in the non-set-test case, **PROccurrence** describes the actual set-objects while **PRGroup** is used to group those sets that share common properties. In the example below there are four page sets that failed the test, (e.g., pages 1, 4, 8, and 12).
  - The second level, which is a child element of the top level **PROccurrence**, describes the document objects that are part of the set. These document objects are grouped as well. In the example below page one consists of 20 objects: five text objects and 15 image objects.

### Example:

```
<PRItem Occurences="4">
  <PRGroup Occurences="1">
    <PRGroupOccurence PageNumber="1"/>
    <PROccurrence Occurences="20">
      <PRGroup Occurences="5">
        <PRGroupOccurence/>
        <PROccurrence TextSize="12"/>
      </PRGroup>
      <PRGroup Occurences="15">
        <PRGroupOccurence/>
        <PROccurrence EffectiveResolution="300 300"/>
      </PRGroup>
    </PROccurrence>
  </PRGroup>
  <PRGroup Occurences="1">
    <PRGroupOccurence PageNumber="4"/>
    <PROccurrence Occurences="20">
      <PRGroup Occurences="7">
        <PRGroupOccurence/>
        <PROccurrence NumberOfPathPoints="4"/>
      </PRGroup>
      <PRGroup Occurences="13">
        <PRGroupOccurence/>
        <PROccurrence EffectiveResolution="300 300"/>
      </PRGroup>
    </PROccurrence>
  </PRGroup>
  <PRGroup Occurences="1">
    <PRGroupOccurence PageNumber="8"/>
  </PRGroup>
  <PRGroup Occurences="1">
    <PRGroupOccurence PageNumber="12"/>
  </PRGroup>
</PRItem>
```

### Structure of the abstract **PRGroupOccurenceBase** element

**PRGroupOccurenceBase** is an abstract class that serves as container for properties that were evaluated during the preflight process.

Name	Data Type	Description
<i>All possible object properties as optional attributes.</i>	<i>As defined by the object property.</i>	An example is given below. See also section "Properties" on page 541 ff.
<i>PageNumber ?</i>	integer	Example of an integer attribute. The same format applies to boolean, Number, Name, NameList, enumeration, enumerations, and string data types.

When the object does not support a certain property, the corresponding attribute in `PROccurrence` and `PRGroupOccurrence` must not be specified.

### PRGroupOccurrence Structure

`PRGroupOccurrence` specifies the shared properties of all `PROccurrences` in a `PRGroup`. `PRGroupOccurrence` inherits from `PRGroupOccurrenceBase` and adds the following elements:

Name	Data Type	Description
<code>StringListValue *</code>	element	Describes the values of a <code>StringList</code> property.
<code>ArgumentValue *</code>	element	Describes the value of a property that is enhanced with additional arguments.

### Structure of `StringListValue` Subelement

`StringListValue` specifies a type that returns a set of strings.

Name	Data Type	Description
<code>Name</code>	NMTOKEN	The name of the subject property.
<code>Value *</code>	element	Element of type <code>StringEvaluation/Value</code> . (See "Structure of the <code>StringEvaluation</code> Subelement" on page 531.)

### Structure of `ArgumentValue` Subelement

`ArgumentValue` specifies a value that is specified with additional arguments. This inherits from `PRGroupOccurrenceBase` and adds the following values:

Name	Data Type	Description
<code>Name</code>	NMTOKEN	The name of the subject property.
<code>PreflightArgument</code>	element	The argument that was used to evaluate this property. This is a <code>PreflightArgument</code> element. (See "Structure of the <code>PreflightArgument</code> Subelement" on page 436.)

### Structure of the `PROccurrence` Subelement

`PROccurrence` describes an individual occurrence of a preflight action failure. This inherits from `PRGroupOccurrence` and adds the following values:

Name	Data Type	Description
<code>Occurrences ?</code>	integer	Only used when the subject occurrence is a set-object. It describes the number of objects in the set.
<code>PRGroup *</code>	element	When this occurrence describes a set-object, the <code>PRGroup</code> elements describe the objects that are part of the set.

## 7.2.124 PreflightReportRulePool

[New in JDF 1.2](#)

The `PreflightReportRulePool` resource specifies how the `PreflightReport` should 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 502. It is suggested that reader's familiarize themselves with that section and "Concept of the Preflight Process" on page 538.

### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>Preflight</i>
Output of processes:	

## Resource Structure

Name	Data Type	Description
<i>MaxOccurrences</i> ?	integer	An upper bound to the maximum number of <i>PROccurrences</i> that may be logged in the <b>PreflightReport</b> .
<i>ActionPools</i>	IDREFS	References the <b>ActionPool</b> whose reporting are defined by this rule.
PRRRule *	element	A list of available PRRules.
PRRRuleAttr ?	element	Defines the default behavior of all PRRule when not defined inside of a PRRule subelement.

## Structure of PRRule Subelement

The PRRule structure is used to define how the **PreflightReport** should log the events that were found during the execution of one Action.

Name	Data Type	Description
<i>ActionRefs</i> +	IDREFS	References the action for which the report behavior is defined in PRRuleAttr.
PRRRuleAttr	element	Defines the way to report this specific rule(s).

The format of the **PreflightReport** is defined by specifying PRRules for specific ACTIONS. Because *ActionRefs* may refer to multiple ACTIONS, a single rule applies to all referenced ACTIONS, (e.g., all color-related ACTIONS will use similar reporting).

## Structure of PRRuleAttr Subelement

Name	Data Type	Description
<i>GroupBy</i> = "Tested"	NMTO-KENS	Group objects having the same N-pair of attributes listed here. For a complete description, see explanations below this table.
<i>ReportAttr</i> = "Tested <i>Filename</i> <i>PageNumber</i> "	NMTO-KENS	When individual items are reported, these attributes are also reported. Attributes which are also being referred by <i>GroupBy</i> are ignored. See possible values and explanations below this table.
<i>LogErrors</i> ?	integer	When the <b>Preflight</b> 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 538.)
<i>MaxGroups</i> ?	integer	The maximum number of groups allowed in the report for this problem. When an object is encountered that fails the preflight test and it belongs to none of the existing groups and there are already <i>MaxGroups</i> , that occurrence is no longer reported individually and no new group is created, although it is added to the <i>Occurrences</i> count and the <i>PageSet</i> .
<i>MaxPerGroup</i> ?	integer	The maximum number of individual occurrences reported per group for this problem. When an object is encountered that fails the preflight test and it belongs to a group that already contains <i>MaxPerGroup</i> elements, that occurrence is no longer reported individually, although it is added to the <i>Occurrences</i> count and the <i>PageSet</i> .

The NMTOKENS for *GroupBy* and *ReportAttr* are one of:

- An object-specific attribute, (e.g., *ColorSpace*, *FontName*, etc.). At the time that we define the Test, we will almost automatically define these attributes.
- *Tested*, which refers to all the attributes that are referred to in the Test(s) used by the Action(s) listed in the *ActionRefs*.
- *TestRelated*, which refers to all the attributes referred in the Test(s) used by the Action(s) listed in *ActionRefs* and the ones that belong to the group of properties in which the tested property was found. For instance, if the *Creator* basic test was made, then all other document properties will be reported as well.

- *VerboseAppSpecific*, which refers to a large list of attributes that the preflight agent (with preflight agent-specific logic) finds interesting for the **Test(s)** used by the **Action(s)** listed in *ActionRefs* .
- *BriefAppSpecific*, which refers to a small list of attributes that the preflight agent (with preflight agent-specific logic) finds interesting for the **Test(s)** used by the **Action(s)** listed in *ActionRefs* .

When the report must be generated, the "*Tested*", "*VerboseAppSpecific*", and "*BriefAppSpecific*" terms are expanded depending on the context (i.e., the specific test and the specific preflight agent) so that the list of attributes only contain object specific attributes.

**Note:** The "*VerboseAppSpecific*" and "*BriefAppSpecific*" tokens can be dependent on the context of a specific test. It is expected that a preflight agent will have a default list of tokens that will always be added (e.g., "*PageNumber*"). In addition it is expected that a preflight agent will define separate lists for specific domains, (e.g., color, font). When a specific test covers some of these specific domains, the attributes of these lists are also added. When *ReportAttr* = "*Tested BriefAppSpecific PageNumber*", the attributes that are reported are dependent on the **Test(s)** used by the **Action(s)** and on the preflight agent as demonstrated in the table below.

Table 7.4: 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>FontSubsetted</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 />`.

### 7.2.125 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*).

#### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>PreviewType</i> , <i>Separation</i> , <i>SheetName</i> , <i>Side</i> , <i>TileID</i> , <i>WebName</i> , <i>RibbonName</i>
<b>Input of processes:</b>	<b><i>InkZoneCalculation</i></b> , <b><i>PreviewGeneration</i></b> , all other processes as thumbnails for UI purposes
<b>Output of processes:</b>	<b><i>PreviewGeneration</i></b>

## Resource Structure

Name	Data Type	Description
<a href="#">Compensation ?</a> <a href="#">Modified in JDF 1.2</a>	enumeration	Compensation of the image to reflect the application of transfer curves to the image. Possible values are: <i>Unknown</i> – <a href="#">Deprecated in JDF 1.2</a> <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.
<a href="#">CTM ?</a> <a href="#">New in JDF 1.1</a>	matrix	Orientation of the Preview w.r.t. the coordinate system of the device that is defined in <i>Compensation</i> . CTM is applied after any transformation defined within the referenced image file, (for example: the transformation defined in the CIP3PreviewImageMatrix of a PPF file).
<a href="#">Directory ?</a> <a href="#">New in JDF 1.1</a>	URL	Defines a base URL for the files that represent this <b>Preview</b> should be copied to or from. If <i>Directory</i> is specified, it must be an Absolute URI [RFC2396] that specifies a Base URI to resolve the <i>URL</i> attribute in the <b>Preview</b> . See "Resolving RunList/@Directory and FileSpec/@URL URI references" on page 649 and [FileURL] for examples.
<i>PreviewFileType</i> = "PNG" <a href="#">New in JDF 1.2</a>	enumeration	The file type of the preview. Possible values are: <i>PNG</i> – The Portable Network Graphics format. <i>CIP3Multiple</i> – The format as defined in the CIP3 PPF specification. One or more previews per CIP3 file are supported. <i>CIP3Single</i> – The format as defined in the CIP3 PPF specification. Only one preview per CIP3 file is supported. The CIP3 formats were added in JDF 1.2 only for backwards compatibility since many systems only support CIP3 format. The CIP3 formats must not be used except in <b>Preview</b> resources that are used as input resources to <i>InkZoneCalculation</i> .
<a href="#">PreviewType ?</a> <a href="#">Deprecated in JDF 1.2</a>	enumeration	Type of the preview. Possible values are: <i>Separation</i> – Separated preview in medium resolution. <i>SeparationRaw</i> – Separated preview in medium resolution. <i>SeparatedThumbNail</i> – Very low resolution separated preview. <i>ThumbNail</i> – Very low resolution RGB preview. <i>Viewable</i> – RGB preview in medium resolution. <i>PreviewType</i> is a partition key and should be used only as such — in JDF 1.2 and beyond, use <i>PreviewUsage</i> below.
<i>PreviewUsage</i> = "Separation" <a href="#">New in JDF 1.2</a>	enumeration	The kind of the preview. Possible values are: <i>Separation</i> – Separated preview in medium resolution. Separation is generally used in <i>InkZoneCalculation</i> . <i>SeparationRaw</i> – Separated preview in medium resolution. This is identical to <i>Separation</i> except that no compensation has been applied. <i>SeparationRaw</i> is generally used for closed loop color control. <i>SeparatedThumbNail</i> – Very low resolution separated preview. <i>ThumbNail</i> – Very low resolution RGB preview. <i>Viewable</i> – RGB preview in medium resolution. <i>PreviewUsage</i> defines the semantics of the preview. If both <i>PreviewType</i> and <i>PreviewUsage</i> are specified, they must match.

Name	Data Type	Description
<a href="#">URL</a> <a href="#">Modified in JDF 1.2</a>	URL	<p>URL identifying any preview file, (e.g., the PNG image or CIP3 PPF file that represents this <b>Preview</b>).</p> <p>See [RFC2396] and "Resolving RunList/@Directory and FileSpec/@URL URI references" on page 649 and "FileSpec MimeType, URL, and Compression attributes, and Container subelement" on page 641 for the syntax and examples. For the "file:" URL scheme see also [RFC1738] and [FileURL].</p> <p><b>Note:</b> A preview will generally be partitioned by separation, unless it represents an RGB viewable image or thumbnail. PPF files with multiple images may contain multiple Separations. In this case, the separation names defined in <b>CIP3ADMSeparationNames</b> define the separations and must match the <i>Separation</i> partition keys used in the JDF.</p>

### 7.2.126 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:** —

#### Resource Structure

Name	Data Type	Description
<a href="#">AspectRatio = "Ignore"</a> <a href="#">New in JDF 1.1</a>	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. One of:</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>
<a href="#">Compensation ?</a> <a href="#">Modified in JDF 1.2</a>	enumeration	<p>Compensation of the image to reflect the application of transfer curves to the image. Possible values are:</p> <p><i>None</i> – No compensation.</p> <p><i>Film</i> – Compensated until film exposure.</p> <p><i>Plate</i> – Compensated until plate exposure.</p> <p><i>Press</i> – Compensated until press.</p>



Name	Data Type	Description
<i>PreviewFileType</i> = "PNG" <a href="#">New in JDF 1.2</a>	enumeration	The file type of the preview to be generated. Possible values are: <i>PNG</i> – The Portable Network Graphics format. <i>CIP3Multiple</i> – The format as defined in the CIP3 PPF specification. One or more previews per CIP3 file are supported. <i>CIP3Single</i> – The format as defined in the CIP3 PPF specification. Only one preview per CIP3 file is supported. The CIP3 formats were added in JDF 1.2 only for backwards compatibility since many systems only support CIP3 format. The CIP3 formats must not be used except in <b>Preview</b> resources that are used as input resources to <i>InkZoneCalculation</i> .
<i>PreviewType</i> ? <a href="#">Deprecated in JDF 1.1</a>	enumeration	The kind of preview to be generated. Possible values are: <i>Separation</i> <i>Viewable</i> <i>PreviewType</i> is a partition key and should be used only as such. In JDF 1.2 and beyond, use <i>PreviewUsage</i> below.
<i>PreviewUsage</i> = "Separation" <a href="#">New in JDF 1.1</a> <a href="#">Modified in JDF 1.2</a>	enumeration	The kind of preview to be generated. Possible values are: <i>Separation</i> – Separated preview in medium resolution. <i>SeparationRaw</i> – Separated preview in medium resolution with no compensation. <i>SeparatedThumbNail</i> – Very low resolution separated preview. <i>ThumbNail</i> – Very low resolution RGB preview. <i>Viewable</i> – RGB preview in medium resolution. <i>PreviewUsage</i> defines the semantics of the preview. If both <i>PreviewType</i> 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.
<i>ImageSetterParams</i> ? <a href="#">New in JDF 1.1</a>	refelement	Details of the <b>ImageSetting</b> process. Needed for accessing information about coordinate transformations that are performed by the imagesetter hardware.

## 7.2.127 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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>SignatureName, SheetName, Side, Separation</i>
<b>Input of processes:</b>	<b><i>ConventionalPrinting, DigitalPrinting</i></b>
<b>Output of processes:</b>	—

## Resource Structure

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 <b>Color/@NeutralDensity</b> 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 <b>Color/PrintConditionColor/@Density</b> .
<i>Name</i>	string	Name of the <b>PrintCondition</b> . Used to reference a <b>PrintCondition</b> from a <b>Color/PrintConditionColor</b> element.
<b>ColorMeasurementConditions</b> ?	refelement	Describes measurement conditions for color measurement and density measurement. If not specified, it defaults to the value of <b>Color/PrintConditionColor/ColorMeasurementConditions</b>
<b>Device</b> ?	refelement	Specifies the Device or Device group that this <b>PrintCondition</b> applies to.
<b>FileSpec</b> ? ( <i>TargetProfile</i> )	refelement	A <b>FileSpec</b> 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 <b>Color/PrintConditionColor/FileSpec</b> ( <i>TargetProfile</i> ).

### Example:

```

<ResourcePool>
  <ColorMeasurementConditions Class="Parameter" ID="MyColorMeasCond"
Status="Available"/>
  <PrintCondition 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>

```

## 7.2.128 PrintRollingParams

[New in JDF 1.2](#)

### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	—
<b>Input of processes:</b>	<b>PrintRolling</b>
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
<i>Copies</i> ?	integer	Number of copies on the roll. Only one of <i>Copies</i> and <i>MaxDiameter</i> must be specified.
<i>MaxDiameter</i> ?	double	Maximal allowed diameter of roll. Only one of <i>Copies</i> and <i>MaxDiameter</i> must be specified.

## 7.2.129 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 768.

## 7.2.130 PStoPDFConversionParams

This resource contains the parameters that control the conversion of PostScript streams to PDF pages.

### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>DocIndex, RunIndex, RunTags, SheetName, Side, SignatureName</i>
<b>Input of processes:</b>	<b><i>PStoPDFConversion</i></b>
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
<i>ASCII85EncodePages = "false"</i>	boolean	If <i>true</i> , 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 <i>false</i> , they are not, resulting in a PDF file that may contain substantial amounts of binary data.
<i>AutoRotatePages ?</i>	enumeration	Allows the device to try to orient pages based on the predominant text orientation. Only used if the file does not contain “%%ViewingOrientation”, “%%PageOrientation”, or “%%Orientation” DSC comments. If the file does contain such DSC comments, it honors them. “%%ViewingOrientation” takes precedence over others, then “%%PageOrientation”, then “%%Orientation”. Possible values are: <i>None</i> – Turns <i>AutoRotatePages</i> off. <i>All</i> – Takes the predominant text orientation across all pages and rotates all pages the same way. <i>PageByPage</i> – Does the rotation on a page-by-page basis, rotating each page individually. Useful for documents that use both portrait and landscape orientations.
<i>Binding = "Left"</i>	enumeration	Determines how the printed pages would be bound. Specify <i>Left</i> for left binding or <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 <i>true</i> , use Flate compression.
<i>DefaultRenderingIntent ?</i> <a href="#">Modified in JDF 1.2</a>	enumeration	Selects the rendering intent for the current job. Possible values are: <i>Default</i> <a href="#">Deprecated in JDF 1.2</a> <i>Perceptual</i> <i>Saturation</i> <i>RelativeColorimetric</i> <i>AbsoluteColorimetric</i> See the <i>Portable Document Format Reference Manual</i> for more information on rendering intent.
<i>DetectBlend = "true"</i>	boolean	Enables or disables blend detection. If <i>true</i> and if <i>PDFVersion</i> is 1.3 or higher, then blends will be converted to smooth shadings.

Name	Data Type	Description
<i>DoThumbnails</i> = "true"	boolean	If "true", thumbnails are created.
<i>EndPage</i> ?	integer	Number that indicates the last page that is displayed when the PDF file is viewed. <i>EndPage</i> must be equal to anything less than <i>StartPage</i> or be greater than or equal to 1. If not, then it must be 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 or may not be distilled, but only <i>StartPage</i> to <i>EndPage</i> pages, inclusive, are opened and viewed in a PDF viewing application.
<i>ImageMemory</i> ? <a href="#">Deprecated in JDF 1.2</a>	integer	Number of bytes in the buffer used in sample processing for color, grayscale, and monochrome images. Its contents are written to disk when the buffer fills up.  This attribute was deprecated because it is an internal application setting and not a parameter setting.
<i>InitialPageSize</i> ? <a href="#">New in JDF 1.1</a>	XYPair	Defines the initial page dimensions assumed by the PS-to-PDF converter, in points. This will be overridden by any <i>PageSize</i> page device parameter found in the PostScript stream. The use of this attribute is strongly encouraged if the PS-to-PDF converter may be used to process EPS files.
<i>InitialResolution</i> ? <a href="#">New in JDF 1.1</a>	XYPair	Defines the initial horizontal and vertical resolution of the PS-to-PDF converter, in dpi. This will be overridden by any <i>HWResolution</i> page device parameter found in the PostScript stream. The use of this attribute is strongly encouraged if the PS-to-PDF converter may be used to process 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 <a href="http://partners.adobe.com/asn/developer/PDFS/TN/5044.ColorSep_Conv.pdf">http://partners.adobe.com/asn/developer/PDFS/TN/5044.ColorSep_Conv.pdf</a>
<i>Optimize</i> = "true"	boolean	If "true", the PS-to-PDF converter optimizes the PDF file. See the <i>Portable Document Format Reference Manual</i> for more information on optimization.
<i>PDFVersion</i> ?	double	Specifies the version number of the PDF file produced. Possible values include all legal version designators, (e.g., 1.2, 1.5).
<i>StartPage</i> ?	integer	Sets the first page that is displayed when the PDF file is opened with a PDF viewing application. <i>StartPage</i> must be greater than or equal to 1. If <i>EndPage</i> is not "-1", then it must be greater than or equal to <i>StartPage</i> .
AdvancedParams ?	element	Advanced parameters which control how certain features of PostScript are handled.
PDFXParams ? <a href="#">New in JDF 1.2</a>	element	PDF/X parameters.
ThinPDFParams ?	element	Parameters that control the optional content or form of PDF files that will be created.

## Structure of AdvancedParams Subelement

Name	Data Type	Description
<i>AllowPSXObject</i> s = "true" <a href="#">New in JDF 1.2</a>	boolean	If "true", allows PostScript XObjects.
<i>AllowTransparency</i> = "false" <a href="#">New in JDF 1.2</a>	boolean	If "true", allows transparency in the PDF.
<i>AutoPositionEPSInfo</i> = "true" <a href="#">Modified in JDF 1.1A</a>	boolean	If "true", the process automatically resizes and centers EPS information on the page.
<i>EmbedJobOptions</i> = "false" <a href="#">New in JDF 1.2</a>	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 distilling of the PS file.
<i>LockDistillerParams</i> = "true" <a href="#">Clarified in JDF 1.2</a>	boolean	If "true", any <b>PSToPDFConversionParams</b> settings configured by the PS content are ignored. If "false", the incoming PS content that specifies any of the <b>PSToPDFConversionParams</b> settings override those defined in <b>PSToPDFConversionParams</b> . <b>Compatibility warning:</b> In JDF 1.1A and previous versions, the definition of <i>LockDistillerParams</i> was accidentally inverted. It is now consistent with the PostScript <i>SetDistillerParams</i> operator.
<i>ParseDSCComments</i> = "true"	boolean	If "true", the process parses the DSC comments for any information that might be helpful for converting the file or for information that must be stored in the PDF file. If "false", the process treats the DSC comments as pure PS comments and ignores them.
<i>ParseDSCCommentForDocInfo</i> = "true"	boolean	If "true", the process parses the DSC comments in the PS file and extracts the document information. This information is recorded in the Info dictionary of the PDF file.
<i>PassThroughJPEGImages</i> = "false" <a href="#">New in JDF 1.2</a>	boolean	If "true", JPEG images are passed through without recompressing them.
<i>PreserveCopyPage</i> = "true"	boolean	If "true", the <i>CopyPage</i> operator of PostScript Level 2 is maintained. If "false", the PostScript Level 3 definition of copypage operator is used. In PostScript Levels 1 and 2, the copypage operator transmits the page contents to the current output device (similar to <i>ShowPage</i> ). However, <i>CopyPage</i> does not perform many of the re-initializations that <i>ShowPage</i> does. Many PostScript Level 1 and 2 programs used the <i>CopyPage</i> operator to perform such operations as printing multiple copies and implementing forms. These programs produce incorrect results when interpreted using the Level 3 <i>CopyPage</i> semantics. This attribute provides a mechanism to retain Level 2 compatibility for this operator.

Name	Data Type	Description
<i>PreserveEPSInfo</i> = "true"	boolean	If "true", preserves the EPS information in the PS file and stores it in the resulting PDF file.
<i>PreserveHalftoneInfo</i> = "false" <a href="#">New in JDF 1.1</a>	boolean	If "true", passes halftone screen information (frequency, angle, and spot function) into the PDF file. If "false", halftone information is not passed in.
<i>PreserveOverprintSettings</i> = "true" <a href="#">New in JDF 1.1</a>	boolean	If "true", Distiller passes the value of the <i>SetOverPrint</i> operator through to the PDF file. Otherwise, overprint is ignored.
<i>PreserveOPIComments</i> = "true"	boolean	If "true", encapsulates Open Prepress Interface (OPI) low resolution images as a form and preserves information for locating the high resolution images.
<i>TransferFunctionInfo</i> = "Preserve" <a href="#">New in JDF 1.1</a>	enumeration	Determines how transfer functions are handled. Possible values are: <i>Preserve</i> – Transfer functions are passed into the PDF file. <i>Remove</i> – Transfer functions are ignored. They are neither applied to the color values nor passed into the PDF file. <i>Apply</i> – Transfer functions are used to modify the data that are written to the PDF file, instead of writing the transfer function itself to the file.
<i>UCRandBGInfo</i> = "Preserve" <a href="#">New in JDF 1.1</a>	enumeration	Determines whether the arguments to the PostScript commands <i>SetUndercolorRemoval</i> and <i>SetBlackGeneration</i> are passed into the PDF file. Possible values are: <i>Preserve</i> – The arguments are passed into the PDF file. <i>Remove</i> – The arguments are ignored.
<i>UsePrologue</i> = "false"	boolean	If "true", the process must prepend a PostScript prologue file to the job and append a PostScript epilog file to 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.

### Structure of PDFXParams Subelement

[New in JDF 1.2](#)

Name	Data Type	Description
<i>PDFX1aCheck</i> = "false"	boolean	If "true", checks compliance with the PDF/X-1a standard (ISO 15930-1:2001).
<i>PDFX3Check</i> = "false"	boolean	If "true", checks compliance with the PDF/X-3 standard (ISO 15930-3:2002).
<i>PDFXBleedBoxtoTrimBoxOffset</i> ?	rectangle	If the <i>BleedBox</i> entry is not specified in the page object of the PostScript document, <i>BleedBox</i> is set to <i>TrimBox</i> with offsets. All numbers must be greater than or equal to 0.0. <i>BleedBox</i> will be completely outside <i>TrimBox</i> .
<i>PDFXCompliantPDFOnly</i> = "false"	boolean	If "true", produces a PDF document only if PDF/X compliance tests are passed.
<i>PDFXOutputCondition</i> ?	string	The string is an optional comment which is added to the PDF file. It describes the intended printing condition in a form that should be meaningful to a human operator at the site receiving the PDF document.

Name	Data Type	Description
<i>PDFXOutputIntentProfile</i> ?	string	If the PostScript document does not specify an output intent name, then this value is used. Possible values are: <i>None</i> – Used when it is required that the PostScript document specifies an intent, allows compliance checking to fail. <i>Euroscale Coated v2</i> <i>Euroscale Uncoated v2</i> <i>Japan Color 2001 Coated</i> <i>Japan Color 2001 Uncoated</i> <i>Japan Standard v2</i> <i>Japan Web Coated (Ad)</i> <i>U.S. Sheetfed Coated v2</i> <i>U.S. Sheetfed Uncoated v2</i> <i>U.S. Web Coated (SWOP) v2</i> <i>U.S. Web Uncoated v2</i> <i>Photoshop 4 Default CMYK</i> <i>Photoshop 5 Default CMYK</i>
<i>PDFXNoTrimBoxError</i> = "true"	boolean	If "true" and both "TrimBox" and "ArtBox" entries are not specified in the page object of the PostScript 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" may be obtained.
<i>PDFXSetBleedBoxToMediaBox</i> = "true"	boolean	If "true" and the "BleedBox" entry is not specified in the page object of the PostScript document, BleedBox is set to "MediaBox."
<i>PDFXTrapped</i> ?	enumeration	If a PostScript document does not specify a "Trapped" state, then the value provided here is used. "Unknown" should be used for workflows that require that the document specify a Trapped state and for which compliance checking should fail if it is not present in the document. Can be one of the following values: <i>Unknown</i> <i>false</i> <i>true</i>
<i>PDFXTrimBoxToMediaBoxOffset</i> ?	rectangle	If both the "TrimBox" and "ArtBox" entries are not specified in the page object of the PostScript document, TrimBox is set to "MediaBox" with offsets. All numbers must be greater than or equal to 0.0. TrimBox will be completely inside MediaBox.

### Structure of ThinPDFParams Subelement

Name	Data Type	Description
<i>FilePerPage</i> = "false"	boolean	If "true", the process generates 1 PDF file per page.
<i>SidelineEPS</i> = "false" <a href="#">New in JDF 1.2</a>	boolean	If "true", embedded EPS files are not converted but are stored in external files in the same location as the PDF itself.
<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.

### 7.2.131 QualityControlParams

[New in JDF 1.2](#)

This set of parameters identifies how the **QualityControl** process should operate. **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:	<b>QualityControl</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>TimeInterval</i> ?	duration	Time interval between individual tests.
<i>SampleInterval</i> ?	integer	Interval in number of samples between tests.
<b>BindingQualityParams</b> ?	element	Specification of the definition parameters of one individual resource.

#### Structure of the 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].

### 7.2.132 QualityControlResult

[New in JDF 1.2](#)

This set of parameters returns results of a **QualityControl** process. **QualityControlResult** defines the generic set of results from the quality control process. The specific measurements are returned in specialized subelements such as **BindingQualityParams**. Additional detailed quality control result types are anticipated in future versions of the JDF specification.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	—
Output of processes:	<b>QualityControl</b>

#### Resource Structure

Name	Data Type	Description
<i>Failed</i> ?	integer	Total number of failed measurements.
<i>Passed</i> ?	integer	Total number of passed measurements.
<b>BindingQualityParams</b> ?	element	Reference to the measurement setup definition.
<b>FileSpec</b> ?	refelement	Location of an external file that contains details of the quality control measurement.
<b>QualityMeasurement</b> *	element	One individual measurement result.



## Structure of the QualityMeasurement Element

QualityMeasurement elements describe an individual measurement.

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 should be set to " <i>destroyed</i> ".
<i>Start ?</i>	dateTime	Date and time of the start of the measurement. If not specified, the measurement time is not known.
BindingQualityMeasurement ?	element	Details of the BindingQualityMeasurement.

## Structure of the 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].

### 7.2.133 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:	<b>Surface</b>
Example Partition:	—
Input of processes:	Any printing process
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>Center</i>	XYPair	Position of the center of the register mark in the coordinates of the <b>MarkObject</b> that contains this mark.
<i>MarkType ?</i>	NMTOKEN	Type of <b>RegisterMark</b> . Possible values include: <i>Arc</i> <i>Circle</i> <i>Cross</i>
<i>MarkUsage ?</i> <a href="#">New in JDF 1.1</a>	enumerations	Specifies the usage of the <b>RegisterMark</b> . Allowed values are: <i>Color</i> – The mark is used for separation color registration. <i>PaperPath</i> – The mark is used for paper path synchronization.
<i>Rotation ?</i>	double	Rotation in degrees. Positive graduation figures indicate counter-clockwise rotation; negative figures indicate clockwise rotation.
<b>SeparationSpec</b> * <a href="#">Modified in JDF 1.2</a>	refelement	Set of separations to which the register mark is bound.

### 7.2.134 RegisterRibbon

[New in JDF 1.1](#)

Description of register ribbons. For the register ribbon the length should be given. There are two parameters:

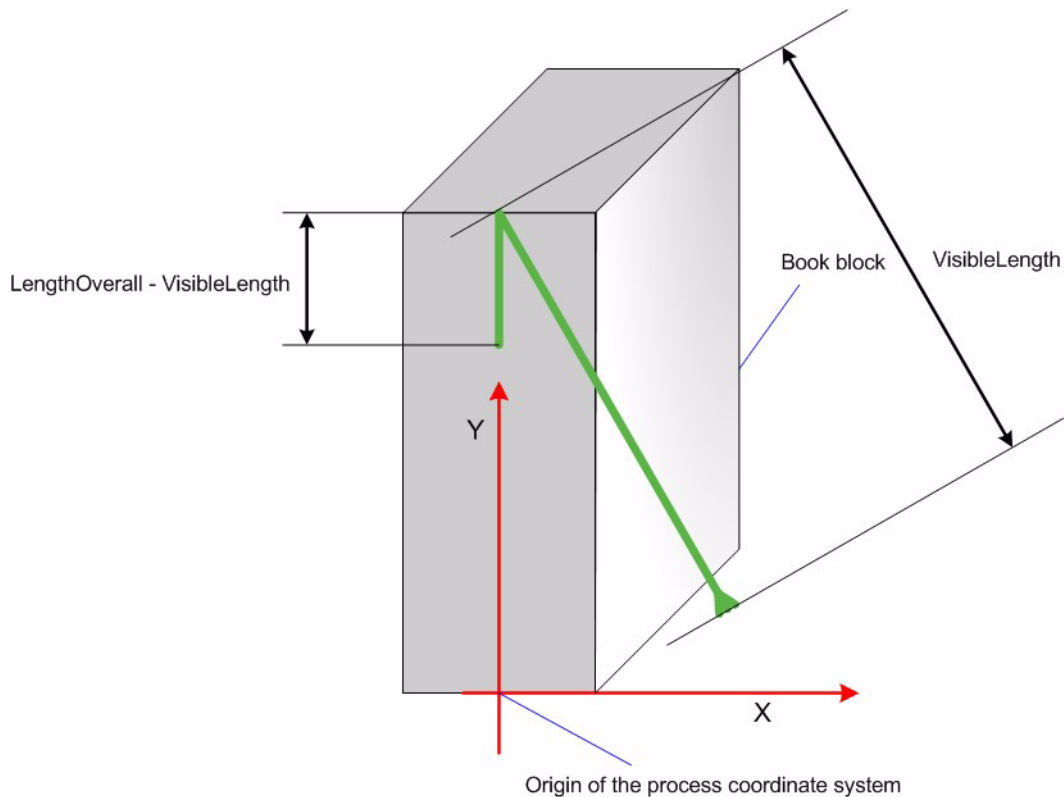


Figure 7.16: Parameters and Coordinate System for BlockPreparation

#### Resource Properties

Resource class:	Consumable
Resource referenced by:	<b>BlockPreparationParams</b>
Example Partition:	—
Input of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>LengthOverall</i>	double	Overall length of the register ribbon, (i.e., 1+2 in the picture above).
<i>Material ?</i>	string	Material of the register ribbon.
<i>RibbonColor ?</i>	NamedColor	Color of the ribbon.
<i>RibbonEnd ?</i>	NMTOKEN	End of the Ribbon. Values include: <i>Cut</i> <i>CutSealed</i> <i>Knot</i> <i>SealedOffset</i> – The ribbon is sealed a distance from the cut.
<i>VisibleLength</i>	double	Length of the register ribbon which will be seen when opening the book, (See picture above).

### 7.2.135 RenderingParams

This set of parameters identifies how the **Rendering** process should 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:	<i>DocIndex, RunIndex, RunTags, SheetName, Side, SignatureName</i>
Input of processes:	<b>Rendering</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>BandHeight</i> ?	integer	Height of output bands expressed in lines. For a frame device, the band height is simply the full height of the frame.
<i>BandOrdering</i> ?	enumeration	Indicates whether output buffers are generated in <i>BandMajor</i> or <i>ColorMajor</i> order. Possible values are: <i>BandMajor</i> – The position of the bands on the page is prioritized over the color. <i>ColorMajor</i> – All bands of a single color are played in order before progressing to the next plane. This is only possible with non-interleaved data.
<i>BandWidth</i> ?	integer	Width of output bands, in pixels.
<i>ColorantDepth</i> ? <a href="#">Clarified in JDF 1.2</a>	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.
<b>AutomatedOverPrintParams</b> ?	reference	Optional controls for overprint substitutions. Defaults to no automated overprint generation.
<b>ObjectResolution</b> * <a href="#">Modified in JDF 1.2</a>	reference	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 <b>ObjectResolution</b> is specified, the value is implied from the input data.
<b>Media</b> ? <a href="#">New in JDF 1.1</a> <a href="#">Deprecated in JDF 1.2</a>	reference	This resource provides a description of the physical media which will be marked. The physical characteristics of the media may affect decisions made during <b>Rendering</b> . In JDF 1.2 and beyond, a RIP should obtain <b>Media</b> information from <b>InterpretingParams/Media</b> .

### 7.2.136 ResourceDefinitionParams

This set of parameters identifies how the **ResourceDefinition** process should operate. Specifically, these parameters define how default parameters of applications and the input resource should be combined.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>ResourceDefinition</b>
Output of processes:	—

## Resource Structure

Name	Data Type	Description
<a href="#">DefaultID ?</a> <a href="#">Deprecated in JDF 1.1</a>	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.
<a href="#">DefaultJDF ?</a>	URL	Link to a JDF resource that defines preset values.
<a href="#">DefaultPriority = "DefaultJDF"</a>	enumeration	Defines whether preset values of the application or of the resource specified in <i>DefaultJDF</i> have priority. Possible values are: <i>Application</i> – The application default settings are used to fill the resource. <i>DefaultJDF</i> – The settings specified in <i>DefaultJDF</i> are applied.
<a href="#">ResourceParam +</a> <a href="#">New in JDF 1.1</a>	element	Specification of the definition parameters of one individual resource.

## Structure of the ResourceParam Subelement

[New in JDF 1.1](#)

Name	Data Type	Description
<a href="#">DefaultID ?</a>	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.
<a href="#">DefaultJDF ?</a>	URL	Link to a JDF resource that defines preset values. Defaults to the <i>DefaultJDF</i> specified in <b>ResourceDefinitionParams</b> .
<a href="#">DefaultPriority ?</a>	enumeration	Defines whether preset values of the application or of the Resource specified in <i>DefaultJDF</i> have priority. Possible values are: <i>Application</i> <i>DefaultJDF</i> Defaults to the <i>DefaultPriority</i> specified in the parent <b>ResourceDefinitionParams</b> .

### 7.2.137 RingBindingParams

This resource describes the details of the **RingBinding** process.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>RingBinding</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<a href="#">BinderColor ?</a>	NamedColor	Color of the ring binder.
<a href="#">BinderMaterial ?</a>	NMTOKEN	The following describe <b>RingBinding</b> 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.
<a href="#">BinderName ?</a>	string	The name of the binder manufacturer and the name of the specific item.

Name	Data Type	Description
<i>RingDiameter</i> ?	double	Diameter of the rings, in points.
<i>RingMechanic</i> ?	boolean	If " <i>true</i> ", a hand lever is available for opening.
<i>RingShape</i> ?	NMTOKEN	The possible values include the following <b>RingBinding</b> : <i>Round</i> <i>Oval</i> <i>D-shape</i> <i>SlantD</i>
<i>RingSystem</i> ? <a href="#">Deprecated in JDF 1.1</a>	enumeration	The following ring binding systems are used: <i>2HoleEuro</i> – In Europe <i>3HoleUS</i> – In North America <i>4HoleEuro</i> – In Europe In JDF 1.2 and beyond, use the value implied by <b>HoleMakingParams/@HoleType</b> .
<i>RivetsExposed</i> ?	boolean	The following <b>RingBinding</b> choice describes mounting of ring mechanism in binder case. If " <i>true</i> ", the heads of the rivets are visible on the exterior of the binder. If " <i>false</i> ", the binder covering material covers the rivet heads.
<i>SpineColor</i> ?	NamedColor	Color of the binders spine.
<i>SpineWidth</i> ?	double	The spine width is determined by the final height of the block of sheets to be bound.
<i>ViewBinder</i> ?	NMTOKEN	The possible values include the following <b>RingBinding</b> clear vinyl outer-wrap types on top of a colored base wrap: <i>Embedded</i> – Printed material is embedded by sealing between the colored and clear vinyl layers during the binder manufacturing. <i>Pocket</i> – Binder is designed so that printed material may be inserted between the color and clear vinyl layers after the binder is manufactured.
<b>HoleMakingParams</b> ? <a href="#">New in JDF 1.2</a>	refelement	Details of the holes in <b>RingBinding</b> .

### 7.2.138 RollStand

[New in JDF 1.2](#)

#### Resource Properties

Resource class:	Handling
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>PrintRolling</b>
Output of processes:	—

#### Resource Structure

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.
<b>Device</b> ?	refelement	Further details of the <b>RollStand</b> .

### 7.2.139 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 3.8.2, *Description of Partitionable Resources*.

**RunList** resources are used whenever an ordered set of page descriptions elements are required. Depending on the process usage of a **RunList**, only certain *Types* of **LayoutElement** may be valid. For example, a pre-RIP imposition process requires **LayoutElement** elements of *Type* "page" or "document", whereas a post-RIP imposition process requires **ByteMap** elements. The usage is detailed in the descriptions of the processes that use the **RunList** resource. **RunList** resources allow structuring of multiple *Pages* into *Documents*. Multiple *Documents* that have a joint context may be grouped into *Sets*.

#### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>PartVersion, Run, RunPage, Separation</i>
<b>Input of processes:</b>	RunLists are used as input resources by most processes that act on content data
<b>Output of processes:</b>	RunLists are used as output resources by most processes that act on content data

#### Resource Structure

Name	Data Type	Description
<i>ComponentGranularity</i> = "Document" <a href="#">New in JDF 1.2</a>	enumeration	Specifies which grouping of input <b>LayoutElement</b> PDL pages define the equivalent of an individual output <b>Component</b> 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 <b>DigitalPrinting</b> and <b>Stitching</b> node if <i>ComponentGranularity</i> = "Set". One of: <i>All</i> – The complete <b>RunList</b> , regardless of document or set breaks defines a new <b>Component</b> . <i>BundleItem</i> – Each <b>Component</b> must be defined by an implicit PDL-defined document break, or an explicit <i>EndofBundleItem</i> . <i>Document</i> – Each document as defined by an implicit PDL-defined document break, or an explicit <i>EndofDocument</i> , defines a new <b>Component</b> . <i>Page</i> – Each page in the <b>RunList</b> defines a new <b>Component</b> . <i>Set</i> – Each set as defined by an implicit PDL-defined set break, or an explicit <i>EndofSet</i> , defines a new <b>Component</b> .
<i>Directory</i> ? <a href="#">Clarified in JDF 1.2</a>	URL	Defines a directory where the files that are associated with this <b>RunList</b> should be copied to or from. If <i>Directory</i> is specified, it must be an Absolute URI [RFC2396] that specifies a Base URI to resolve the <b>FileSpec/URL</b> attribute in the <b>RunList</b> . See "Resolving RunList/@Directory and FileSpec/@URL URI references" on page 649 and [FileURL] for examples.
<i>DocCopies</i> = "1" <a href="#">New in JDF 1.1</a>	integer	Number of instance document copies that this <b>RunList</b> represents. Specifying <i>DocCopies</i> is equivalent to repeating the sequence of <b>RunList</b> leaves between <i>EndOfDocument</i> = "true" for a total of <i>DocCopies</i> times. <b>Note:</b> It is illegal to specify <i>DocCopies</i> with different values of various leaves of a <b>RunList</b> representing the same instance document.

Name	Data Type	Description
<i>DocNames</i> ?	NameRangeList	A list of named documents in a multi-document file that supports named access to individual documents. <i>DocNames</i> defaults to all documents. If <i>DocNames</i> occurs in the <b>RunList</b> , <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 <b>LayoutElement</b> element.
<i>EndOfBundleItem</i> ? <a href="#">New in JDF 1.2</a>	boolean	If " <i>true</i> ", the last page in the <b>RunList</b> is the last page of a <b>BundleItem</b> . The implied default value of <i>EndOfBundleItem</i> = " <i>false</i> ", except for the last <b>RunList</b> partition, which always has an implied default value of <i>EndOfBundleItem</i> = " <i>true</i> ". <i>EndOfBundleItem</i> must only be specified if <i>ComponentGranularity</i> = " <i>BundleItem</i> ".
<i>EndOfDocument</i> ? <a href="#">Clarified in JDF 1.2</a>	boolean	If " <i>true</i> ", the last finished page in the <b>RunList</b> is the last page of an instance document. The precise handling of instance document changes is defined in the <b>InsertSheet</b> resource. If the <b>RunList</b> 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 <b>RunList</b> partition leaf, which always has an implied default value of <i>EndOfDocument</i> = " <i>true</i> ".
<i>EndOfSet</i> ? <a href="#">New in JDF 1.1</a> <a href="#">Clarified in JDF 1.2</a>	boolean	If " <i>true</i> ", the last finished page in the <b>RunList</b> is the last page of a set of instance documents. The precise handling of instance document boundaries is defined in the <b>InsertSheet</b> resource. If the <b>RunList</b> 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 <b>RunList</b> 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 <b>RunList</b> . This attribute is generally used to describe pre-separated files.
<i>IsPage</i> = " <i>true</i> "	boolean	If " <i>true</i> ", the individual <b>RunList</b> element defines one or more page slots, (e.g., for filling <b>PlacedObjects</b> ). If " <i>false</i> ", the first parent partitioned <b>RunList</b> element 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 <b>RunList</b> .
<i>LogicalPage</i> ? <a href="#">Modified in JDF 1.1</a>	integer	The logical page number of the first finished page in a <b>RunList</b> . This attribute may be used to retain logical page indices when a partitioned <b>RunList</b> is spawned. It defaults to "1" plus the last finished page of the previous sibling <b>RunList</b> partition. If the <b>RunList</b> element is the first partition, <i>LogicalPage</i> defaults to "0". Note that is an error to specify <i>LogicalPage</i> to be less than the number of previously defined logical pages, since this defines overlapping finished pages within the <b>RunList</b> .
<i>NDoc</i> ? <a href="#">New in JDF 1.1</a> <a href="#">Deprecated in JDF 1.2</a>	integer	Total number of instance documents that are defined by the <b>RunList</b> . If <i>NDoc</i> is not specified, it defaults to all instance documents in the partitioned <b>RunList</b> elements that make up the <b>RunList</b> . In JDF 1.2 and beyond, only <i>Docs</i> is supported.

Name	Data Type	Description
<i>NPage</i> ?	integer	Total number of pages (placed object slots or <b>RunList</b> elements with <i>IsPage</i> = "true") that are defined by the <b>RunList</b> . If <i>NPage</i> is not specified, it defaults to all finished pages in the partitioned <b>RunList</b> elements that make up the <b>RunList</b> . If the <b>RunList</b> 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 <b>RunList</b> with <i>NPage</i> specified always refers to <i>NPage</i> pages, regardless of the number of pages of the referenced PDL.
<i>NSet</i> ? <a href="#">New in JDF 1.1</a> <a href="#">Deprecated in JDF 1.2</a>	integer	Total number of instance document sets that are defined by the <b>RunList</b> . If <i>NSet</i> is not specified, it defaults to all instance document sets in the partitioned <b>RunList</b> elements that make up the <b>RunList</b> . In JDF 1.2 and beyond, only <i>Sets</i> is supported.
<i>PageCopies</i> = "1" <a href="#">New in JDF 1.1</a>	integer	Number of finished page copies that this <b>RunList</b> represents. Specifying <i>PageCopies</i> is equivalent to repeating the <b>RunList</b> 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 uncollated when calculating the index in the logical <b>RunList</b> , (e.g., <i>PageCopies</i> = "2" would result in a logical page sequence of 0 0 1 1 2 2, etc.).
<i>PageListIndex</i> ? <a href="#">New in JDF 1.2</a>	IntegerRangeList	List of the indices of the <i>PageData</i> elements of the <b>PageList</b> specified in the <b>LayoutElement</b> referenced by this <b>RunList</b> . If not specified, the complete <i>PageListIndex</i> specified in the <b>LayoutElement</b> referenced by this <b>RunList</b> is applied.
<i>PageNames</i> ?	NameRangeList	A list of named pages in a multi-page file that supports named access to individual finished pages. <i>PageNames</i> defaults to all pages. If <i>PageNames</i> occurs in the <b>RunList</b> , <i>FirstPage</i> , <i>Npage</i> , <i>SkipPage</i> , and <i>Pages</i> must be ignored if any of them is also present.
<i>Pages</i> ? <a href="#">Modified in JDF 1.1A</a> <a href="#">Clarified in JDF 1.2</a>	IntegerRangeList	Zero-based list of indices in the documents specified by the <b>LayoutElement</b> element and the <i>Docs</i> , <i>DocNames</i> , <i>Sets</i> , and <i>SetNames</i> attributes. <i>Pages</i> need not be in document order. If <i>Pages</i> is present, <i>FirstPage</i> and <i>SkipPage</i> must be ignored. If neither <i>Pages</i> , <i>FirstPage</i> , <i>Npage</i> , <i>PageNames</i> , or <i>SkipPage</i> are present, all pages in the <b>LayoutElement</b> are selected.
<i>RunTag</i> ? <a href="#">New in JDF 1.1</a> <a href="#">Clarified in JDF 1.2</a>	NMTOKEN	Tag of a partition of a resource other than the <b>RunList</b> 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 <b>RunList</b> may have the same <i>RunTag</i> . If the <b>RunList</b> references a PDL that supports internal labels, <i>RunTag</i> may be implied from the PDL.
<i>SetCopies</i> = "1" <a href="#">New in JDF 1.1</a>	integer	Number of instance document set copies that this <b>RunList</b> represents. Specifying <i>SetCopies</i> is equivalent to repeating the sequence of <b>RunList</b> leaves between <i>EndOfSet</i> = "true" for a total of <i>SetCopies</i> times. Note that it is illegal to specify <i>SetCopies</i> with different values of various leaves of a <b>RunList</b> representing the same instance document.



Name	Data Type	Description
<a href="#">SetNames ?</a> <a href="#">New in JDF 1.1</a>	NameRangeList	A list of named document sets in a multi-document set file that supports named access to individual documents. <i>SetNames</i> defaults to all document sets specified by <i>Sets</i> . If <i>SetNames</i> occurs in the <b>RunList</b> , <i>Sets</i> is ignored if it is also present. <i>SetNames</i> is only valid if <b>LayoutElement/@ElementType</b> = "MultiSet".
<a href="#">Sets ?</a> <a href="#">New in JDF 1.1</a>	IntegerRangeList	Zero-based list of document set indices in a multi-document sets file specified by the <b>LayoutElement</b> element. If not present, all document sets are selected. <i>Sets</i> is only valid if <b>LayoutElement/@ElementType</b> = "MultiSet".
<a href="#">SkipPage ?</a>	integer	Used when the <b>RunList</b> 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 <b>RunList</b> element. 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.
<a href="#">Sorted ?</a>	boolean	Specifies whether the elements in the <b>RunList</b> are sorted in the document reader order.
<a href="#">ByteMap ?</a> <a href="#">Modified in JDF 1.2</a>	refelement	Describes the page or stream of pages. Only one of <b>ByteMap</b> , <b>InterpretedPDLData</b> , or <b>LayoutElement</b> must be specified in one <b>RunList</b> element. If <b>ByteMap</b> , <b>InterpretedPDLData</b> , nor <b>LayoutElement</b> are specified, the <b>RunList</b> entry specifies empty content.
<a href="#">Disposition ?</a>	refelement	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 <b>FileSpec</b> is retained indefinitely. <b>FileSpec/Disposition</b> takes precedence over <b>RunList/Disposition</b> .
<a href="#">DynamicInput *</a>	element	Replacement text for a <b>DynamicField</b> element. This information defines the contents of a dynamic mark on the automated page layout. 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. <b>DynamicInput</b> elements have one optional <i>Name</i> attribute that, when linked to the <i>ReplaceField</i> attribute of the <b>DynamicField</b> element, defines the string that should be replaced.
<a href="#">InsertSheet *</a>	refelement	Describes how <b>Sheets</b> and <b>Surfaces</b> may be completed and optional media which may be inserted at the beginning or end of this <b>RunList</b> element.
<a href="#">InterpretedPDLData ?</a> <a href="#">New in JDF 1.2</a>	refelement	Represents the results of the PDL interpretation process. Only one of <b>ByteMap</b> , <b>InterpretedPDLData</b> , or <b>LayoutElement</b> must be specified in one <b>RunList</b> element. If <b>ByteMap</b> , <b>InterpretedPDLData</b> , nor <b>LayoutElement</b> are specified, the <b>RunList</b> entry specifies empty content.
<a href="#">LayoutElement ?</a> <a href="#">Modified in JDF 1.2</a>	refelement	Describes the document, finished page or image. Only one of <b>ByteMap</b> , <b>InterpretedPDLData</b> , or <b>LayoutElement</b> must be specified in one <b>RunList</b> element. If <b>ByteMap</b> , <b>InterpretedPDLData</b> , nor <b>LayoutElement</b> are specified, the <b>RunList</b> entry specifies empty content.
<a href="#">PageList ?</a>	refelement	Specification of page metadata for pages described by this <b>RunList</b> .

## Structure of a DynamicInput Subelement

DynamicInput defines the contents of a dynamic mark on a **Surface** resource for automated page layout. 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. For details on dynamic marks, see the DynamicField element description in Section 7.2.158, Surface.

Name	Data Type	Description
<i>Name</i> ?	string	Label that must match the <i>ReplaceField</i> attribute of the appropriate DynamicField element
—	text	Defines the text string that should be inserted as a replacement for the text defined in <i>ReplaceField</i> of a DynamicField element.

## Examples of Partitioning of a RunList

The following examples illustrate how a **RunList** can be structured using partitioning Mechanisms. Note that the partitioning of a **RunList** often generates the values necessary to evaluate the partitioning of other resources, (e.g., the **RunIndex** into the **RunList**). Thus, the order in which the **RunLists** appear in the XML document is significant. It is interesting to note that the “Run” partitioning key has a string value, and is not required to be numeric.

### Simple unstructured Single-File Runlist

This example specifies all pages contained in “/in/colortest.pdf”.

```
<RunList Class="Parameter" ID="Link0003" Pages="0~-1" Status="Available">
  <LayoutElement>
    <FileSpec URL="File:///in/colortest.pdf"/>
  </LayoutElement>
</RunList>
```

### Simple Multi-File unseparated RunList using RunList/@Directory

This example specifies all pages contained in File1.pdf and File2.pdf, which are located in the directory “/Dir” that is specified in **RunList/@Directory**.

```
<RunList Class="Parameter" Directory="File:///Dir/" ID="Link0003" PartIDKeys="Run" Status="Available">
  <RunList Pages="0~-1" Run="1">
    <LayoutElement>
      <FileSpec URL="File1.pdf"/>
    </LayoutElement>
  </RunList>
  <RunList Pages="0~-1" Run="2">
    <LayoutElement>
      <FileSpec URL="File2.pdf"/>
    </LayoutElement>
  </RunList>
</RunList>
```

### Simple Multi-File unseparated RunList with independent spawning

This example specifies the first five pages contained in File1.pdf and File2.PDF. File2.pdf has been spawned and is being processed individually.

```
<RunList Class="Parameter" ID="Link0003" PartIDKeys="Run" Status="Available">
  <RunList Pages="0~4" Run="1">
    <LayoutElement>
      <FileSpec URL="File:///File1.pdf"/>
    </LayoutElement>
  </RunList>
  <RunList Pages="0~-1" Run="2" SpawnStatus="SpawnedRW">
    <LayoutElement>
      <FileSpec URL="File:///File2.pdf"/>
    </LayoutElement>
  </RunList>
```

```
</RunList>
```

This is the corresponding spawned RunList. Note the *LogicalPage* attribute, which specifies the number of skipped pages.

```
<RunList Class="Parameter" ID="Link0003" LogicalPage="5" Pages="0~-1"
  PartIDKeys="Run" Status="Available">
  <RunList Run="2">
    <LayoutElement>
      <FileSpec URL="File:///File2.pdf"/>
    </LayoutElement>
  </RunList>
</RunList>
```

### Simple Multi-File separated RunList

This example specifies all pages contained in Presep.pdf and following that, pages 1, 3, and 5 of each preprepared file.

```
<RunList Class="Parameter" ID="Link0003" PartIDKeys="Run Separation"
  Status="Available">
  <RunList Run="1" SkipPage="3">
    <LayoutElement>
      <FileSpec URL="File:///Presep.pdf"/>
    </LayoutElement>
    <RunList FirstPage="0" IsPage="false" Separation="Cyan"/>
    <RunList FirstPage="1" IsPage="false" Separation="Magenta"/>
    <RunList FirstPage="2" IsPage="false" Separation="Yellow"/>
    <RunList FirstPage="3" IsPage="false" Separation="Black"/>
  </RunList>
  <RunList IsPage="true" Pages="1 3 5" Run="2">
    <RunList IsPage="false" Separation="Cyan">
      <LayoutElement>
        <FileSpec URL="File:///Cyan2.pdf"/>
      </LayoutElement>
    </RunList>
    <RunList IsPage="false" Separation="Magenta">
      <LayoutElement>
        <FileSpec URL="File:///Magenta2.pdf"/>
      </LayoutElement>
    </RunList>
    <RunList IsPage="false" Separation="Yellow">
      <LayoutElement>
        <FileSpec URL="File:///Yellow2.pdf"/>
      </LayoutElement>
    </RunList>
    <RunList IsPage="false" Separation="Black">
      <LayoutElement>
        <FileSpec URL="File:///Black2.pdf"/>
      </LayoutElement>
    </RunList>
  </RunList>
</RunList>
```

## 7.2.140 SaddleStitchingParams

[Deprecated in JDF 1.1](#) See "SaddleStitchingParams" on page 770 for details of this deprecated resource.

## 7.2.141 ScanParams

This resource provides the parameters for the **Scanning** process.

### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>RunIndex</i>
<b>Input of processes:</b>	<b>Scanning</b>
<b>Output of processes:</b>	—

## Resource Structure

Name	Data Type	Description
<i>BitDepth</i>	integer	Bit depth of a one-color separation.
<i>CompressionFilter ?</i>	enumeration	Specifies the compression filter to be used. Possible values include: <i>CCITTFaxEncode</i> – Used to select CCITT Group 3 or 4 facsimile encoding. <i>DCTEncode</i> – Used to select JPEG compression. <i>FlateEncode</i> – Used to select Zip compression. <i>WaveletEncode</i> – Used to select Wavelet compression. <i>JBIG2Encode</i> – Used to select JBIG2 monochrome compression.
<i>DCTQuality ?</i>	double	A value between 0 and 1 that indicates “how much” the process should compress images. 0.0 means “do as loss-less compression as possible.” 1.0 means “do the maximum compression possible.”
<i>InputBox ?</i>	rectangle	Rectangle that describes the image section to be scanned, in points. The origin of the coordinate system is the lower left corner of the physical item to be scanned.
<i>Magnification =</i> “1 1”	XYPair	Size of the output/size of the input for each dimension.
<i>MountID ?</i>	string	ID of the drum or other mounting device upon which the media should be mounted.
<i>Mounting ?</i>	enumeration	Specifies how to mount originals. Possible values are: <i>Unfixed</i> – Original lies unfixed on the scanner tray/drum. <i>Fixed</i> – Original is fixed on the scanner tray/drum with transparent tape. <i>Wet</i> – Original is put in gel or oil and fixed on the scanner tray/drum. <i>Registered</i> – Original is fixed with registration holes. This value is used for copix.
<i>OutputColorSpace</i>	enumeration	Color space of the output images. Possible values are: <i>LAB</i> <i>RGB</i> <i>CMYK</i> <i>GrayScale</i>
<i>OutputResolution</i>	XYPair	X and Y resolution of the output bitmap, in dpi.
<i>OutputSize ?</i>	XYPair	X and Y dimension of the intended output image, in points.
<i>SplitDocuments ?</i>	integer	A number representing how many images are scanned before a new file is created.
<b>FileSpec ?</b> ( <i>CorrectionProfile</i> )	reference	A <b>FileSpec</b> resource pointing to an ICC profile that describes color corrections.
<b>FileSpec ?</b> ( <i>TargetProfile</i> )	reference	A <b>FileSpec</b> 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).
<b>FileSpec ?</b> ( <i>ScanProfile</i> )	reference	A <b>FileSpec</b> resource pointing to an ICC profile that describes the scanner.

## 7.2.142 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:	<b>Surface</b>
Example Partition:	—
Input of processes:	Any printing process
Output of processes:	—

### Resource Structure

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 * <a href="#">Modified in JDF 1.2</a>	refelement	Set of separations to which the scavenger area is bound.

## 7.2.143 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:	<b>ExposedMedia</b>
Example Partition:	<i>Separation, SheetName, Side, SignatureName</i>
Input of processes:	<b>Screening, ColorCorrection, ContoneCalibration</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>IgnoreSourceFile = "true"</i>	boolean	Specifies whether to ignore the screen settings (e.g., setscreen, setcolor-screen, 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> <a href="#">Deprecated in JDF 1.2</a>	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 * <a href="#">Modified in JDF 1.1</a>	element	List of screen selectors. A screen selector is included for each separation, including a default specification.

## Structure of ScreenSelector Subelement

Description of screening for a selection of source object types and separations.

Name	Data Type	Description
<i>Angle</i> ? <a href="#">Clarified in JDF 1.2</a>	double	Specifies the first angle of the screen when AM screening is used, otherwise <i>Angle</i> is ignored. Either <i>Angle</i> or <i>AngleMap</i> may be specified but not both. If neither <i>Angle</i> nor <i>AngleMap</i> are specified, the angle is determined by the default of the selected <i>ScreeningFamily</i> .
<i>AngleMap</i> ? <a href="#">New in JDF 1.1</a> <a href="#">Clarified in JDF 1.2</a>	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). Only one of <i>Angle</i> or <i>AngleMap</i> may be specified. This mapping is not transitive, so, when <i>Separation</i> already specifies a color with a known default <sup>a</sup> , it specifies the angle of the separation defined by <i>AngleMap</i> prior to that separation being mapped. The following example specifies that <i>Black</i> should be mapped to the <i>Cyan</i> default separation and <i>Cyan</i> to the <i>Black</i> default separation. The third line maps Spot1 to Magenta. <pre>&lt;ScreenSelector AngleMap="Black" Separation="Cyan"/&gt; &lt;ScreenSelector AngleMap="Cyan" Separation="Black"/&gt; &lt;ScreenSelector AngleMap="Magenta" Separation="Spot1"/&gt;</pre>
<i>DotSize</i> ? <a href="#">New in JDF 1.1</a> <a href="#">Clarified in JDF 1.2</a>	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> ? <a href="#">Modified in JDF 1.2</a>	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 may change as a function of gray level. In this case, the <i>Frequency</i> value is interpreted for a midtone (50%) gray level. If <i>Frequency</i> is not specified, the frequency is determined by the default of the selected <i>ScreeningFamily</i> .
<i>ScreeningFamily</i> ?	string	Vendor specific screening family name. <a href="#">Sample values removed in JDF 1.2</a>
<i>ScreeningType</i> ? <a href="#">Modified in JDF 1.2</a>	enumeration	General type of screening. Possible values are: <i>Adaptive</i> <i>AM</i> – May 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> should be applied to all separations that are not specified explicitly.
<i>SourceFrequency</i> ? <a href="#">Modified in JDF 1.2</a>	Double-Range	Specifies the line frequency of screens which should 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</i> = "All"	enumerations	Identifies the class(es) of incoming graphical objects on which to use the selected screen. Possible values are: <i>All</i> <i>ImagePhotographic</i> – Contone images. <i>ImageScreenShot</i> – Images largely comprised of rasterized vector art. <i>Text</i> <i>LineArt</i> – Vector objects other than text. <i>SmoothShades</i> – Gradients and blends.

Name	Data Type	Description
<b><i>SpotFunction ?</i></b> <a href="#">Clarified in JDF 1.2</a>	NMTOKEN	Specifies the spot function of the screen when AM screening is used. In general, it is common for a spot function to change its shape as a function of gray level. Response to these spot function names may be implementation-dependent. These example names are the same as the spot function names defined in PDF. Example values include: <i>Round</i> <i>Diamond</i> <i>Ellipse</i> <i>EllipseA</i> <i>InvertedEllipseA</i> <i>EllipseB</i> <i>EllipseC</i> <i>InvertedEllipseC</i> <i>Line</i> <i>LineX</i> <i>LineY</i> <i>Square</i> <i>Cross</i> <i>Rhomboid</i> <i>DoubleDot</i> <i>InvertedDoubleDot</i> <i>SimpleDot</i> <i>InvertedSimpleDot</i> <i>CosineDot</i> <i>Double</i> <i>InvertedDouble</i>

- a. In general this will be a CMYK process color, but it can also be another process color, (e.g., HexaChrome™).

### 7.2.144 SeparationControlParams

This resource provides the controls needed to separate composite color files.

#### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	—
<b>Input of processes:</b>	<b><i>Separation</i></b>
<b>Output of processes:</b>	—

#### Resource Structure

Name	Data Type	Description
<b><i>AutomatedOverPrintParams ?</i></b>	refelement	Optional controls for overprint substitutions. The default case is that no automated overprint generation is used.
<b><i>TransferFunctionControl ?</i></b>	refelement	Controls whether the device performs transfer functions and what values are used when doing so.

### 7.2.145 SeparationSpec

This resource specifies a specific separation, and is usually used to define a list or sequence of separations.

#### Resource Properties

Resource class: ResourceElement  
 Resource referenced by: **ColorantControl, LayoutElement, RegisterMark, TransferFunctionControl**

Example Partition: —

Input of processes: —

Output of processes: —

#### Resource Structure

Name	Data Type	Description
<i>Name</i>	string	Name of one specific separation.

### 7.2.146 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: —

#### Resource Structure

Name	Data Type	Description
Shape *	element	Details of each individual cut shape

#### Structure of Shape Subelement

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, while 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>Material</i> ?	string	Transparent material that fills a shape (e.g., an envelope window) that was cut out when <i>CutOut</i> = "true".
<i>CutType</i> = "Cut"	enumeration	Type of cut or perforation used. Possible values are: <i>Cut</i> – Full cut. <i>Perforate</i> – Interrupted perforation that does not span the entire sheet
<i>ShapeDepth</i> ?	double	Depth of the shape cut, measured in microns [µm]. If not specified, the shape is completely cut.



Name	Data Type	Description
<i>ShapeType</i>	enumeration	Describes any precision cutting other than hole making. Possible values are: <i>Rectangular</i> <i>Round</i> <i>Path</i>
<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).

### 7.2.147 Sheet

This resource provides a description of a sheet, as well as the marks on that sheet.

#### Resource Properties

Resource class:	Parameter
Resource referenced by:	<b>InsertSheet, Layout</b>
Example Partition:	<i>SheetName</i> .
Input of processes:	—
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>LockOrigins</i> = " <i>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> ? <a href="#">Clarified in JDF 1.2</a>	string	Name of the sheet. <i>Name</i> must be unique within a given <b>Layout</b> . <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 <b>Surfaces</b> in the <b>Sheet</b> . CTMs for <b>MarkObjects</b> or <b>ContentObjects</b> transform page contents or marks into this rectangle.
InsertSheet *	refelement	Specifies how to complete a sheet in an automated printing environment.
<b>Media</b> ? <a href="#">New in JDF 1.1</a>	refelement	Describes the media to be used. Defaults to <b>Layout/Signature/Media</b> .
<b>MediaSource</b> ? <a href="#">Deprecated in JDF 1.1</a>	refelement	Describes the media to be used. Replaced by <b>Media</b> in JDF 1.1.
<b>Surface</b> (Front) ?	refelement	Describes the front surface to be used. Two surfaces may be attached: one front surface and one back surface. The surface is defined by the <i>Side</i> attribute of the <b>Surface</b> resource. <b>Surface/@Side</b> must be <i>Front</i> .
<b>Surface</b> (Back) ?	refelement	Describes the back surface to be used. <b>Surface/@Side</b> must be <i>Back</i> .

### 7.2.148 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:	<b>Shrinking</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>Duration ?</i>	duration	Shrinking time.
<i>ShrinkingMethod</i> = "ShrinkHot"	enumeration	Specifics of the shrinking method for shrink wrapping. <i>ShrinkCool</i> <i>ShrinkHot</i>
<i>Temperature ?</i>	double	Oven temperature in ° Centigrade.

### 7.2.149 SideSewingParams

[Deprecated in JDF 1.1](#) See "SideSewingParams" on page 771 for details of this deprecated resource.

### 7.2.150 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:	<b>SpinePreparation</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>FlexValue ?</i> <a href="#">Deprecated in JDF 1.2</a>	double	Flex quality parameter, in [N/cm]. In JDF 1.2 and beyond, <i>FlexValue</i> is defined in <i>QualityControlParams/@BindingQuality</i> . See "QualityControlParams" on page 452 for details.
<i>MillingDepth ?</i> <a href="#">Modified in JDF 1.2</a>	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.

Name	Data Type	Description
<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. Possible values include: <i>Brushing</i> – Brushes away dust from the spine to improve the binding quality. <i>FiberRoughing</i> – The fibers of the paper on the spine are exposed without the risk of glazing the paper coating. This optimizes the spine preparation considering paper and adhesive types. <i>Leveling</i> – After milling the spine, any uneven areas are leveled to achieve an even surface. <i>Milling</i> – Cuts off part of the spine so the spine is not too even. A rough texture of the fibers is assured. This creates ideal conditions for stable anchoring of the sheets in the glue. <i>Notching</i> – This gives a clamping effect on the spine which is desirable for some products. <i>Sanding</i> – Is used for voluminous book papers. <i>Shredding</i> – Produces a relatively smooth surface. Further operations like <i>Notching</i> , <i>Leveling</i> , <i>FiberRoughing</i> , <i>Sanding</i> , or <i>Brushing</i> are necessary.
<i>PullOutValue ?</i> <a href="#">Deprecated in JDF 1.2</a>	double	Pull out quality parameter, in [N/cm]. In JDF 1.2 and beyond, <i>PulloutValue</i> is defined in <i>QualityControlParams/@BindingQuality</i> . See "QualityControlParams" on page 452 for details.
<i>StartPosition = "0"</i>	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.

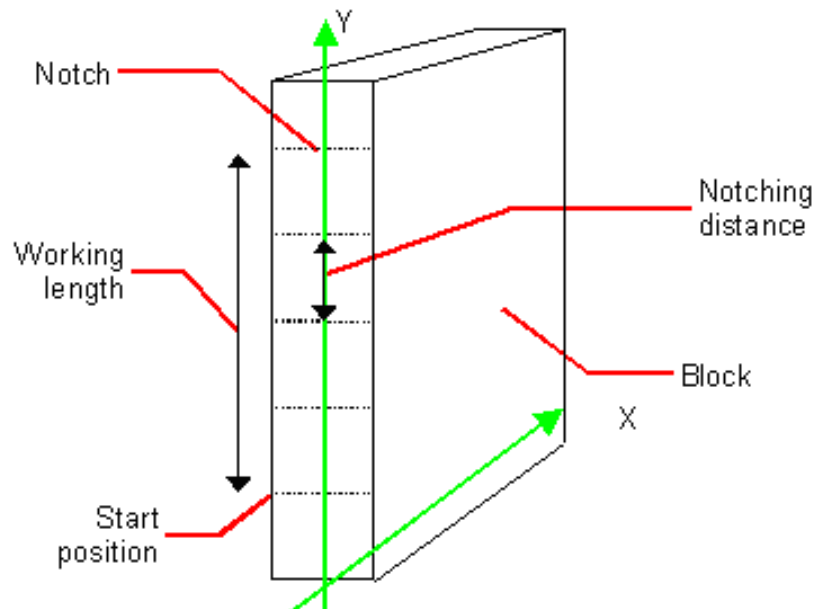


Figure 7.17: Parameters and coordinate systems for the SpinePreparation process

## 7.2.151 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:	<b>SpineTaping</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>HorizontalExcess</i> ?	double	Taping spine excess on each side. The tape is assumed to be centered between left and right.
<i>StripBrand</i> ?	string	Strip brand.
<i>StripColor</i> ?	NamedColor	Color of the strip.
<i>StripLength</i> ?	double	Length of strip material along binding edge. If not defined, the default case is that the <i>StripLength</i> be equivalent to the length of the spine.
<i>StripMaterial</i> ?	enumeration	Strip material. Possible values are: <i>Calico</i> <i>Cardboard</i> <i>CrepePaper</i> <i>Gauze</i> <i>Paper</i> <i>PaperlinedMules</i> <i>Tape</i>
<i>TopExcess</i> = "0.0"	double	Top spine taping excess. This value may be negative.
<i>GlueApplication</i> *	refelement	Describes where and how to apply glue to the book block.

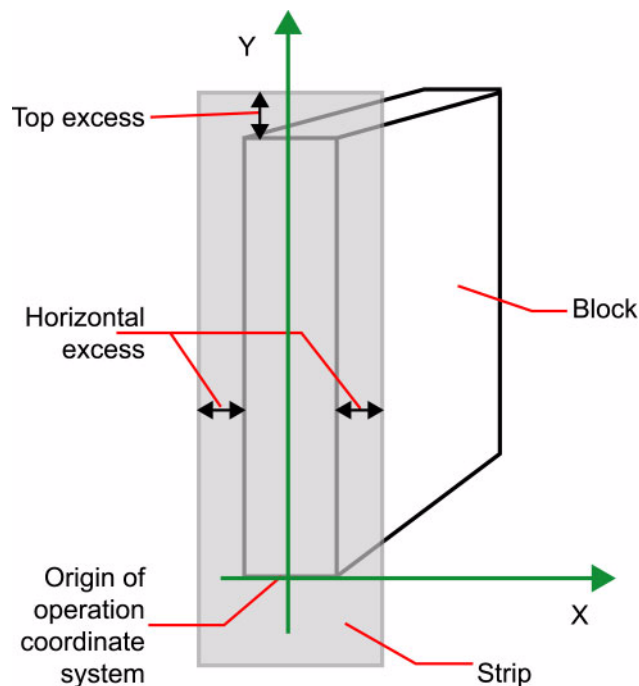
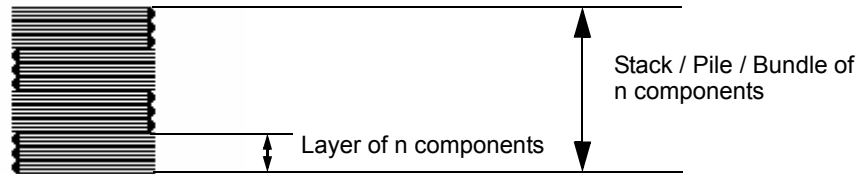


Figure 7.18: Parameters and coordinate system for the SpineTaping process

## 7.2.152 StackingParams

[New in JDF 1.1](#)

Settings for the **Stacking** process. A stack of components may be uneven and unstable, due to variations in thickness across each component. The thickness variations may be caused by folding, binding, or inserted components. A stack may be split into layers, with successive layers rotated by 180° to compensate for the unevenness.



If the thickest part is on an edge (e.g., a book binding), the components may be offset to separate the thick parts. Layer compensation and offsetting may be combined as in the following examples.

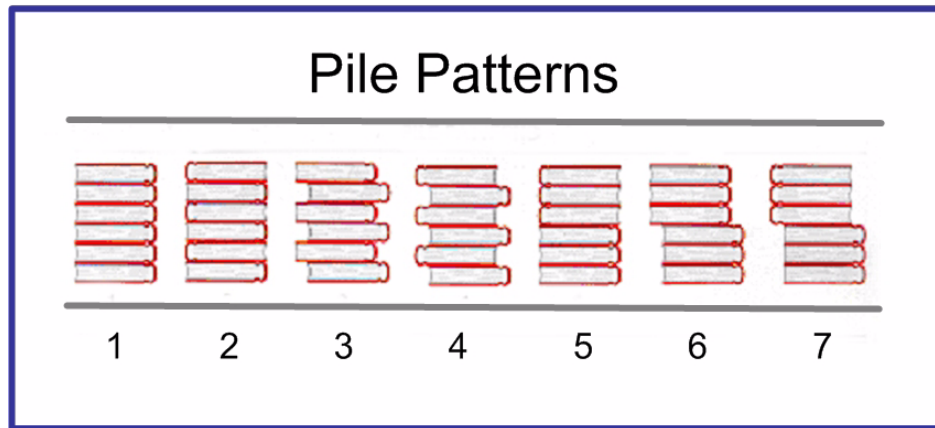


Table 7-5: Parameters in Stacking

File Pattern	StandardAmount	LayerAmount (Default = StandardAmount)	Compensate (Default = true)	Disjointing/ @Offset
1	6	6	true	0 0
2	6	1	true	0 0
3	6	1	false	x 0
4	6	1	true	x 0
5	6	3	true	0 0
6	6	3	false	x 0
7	6	3	true	x 0

If the number of components is not evenly divisible by standard stack size (*StandardAmount*) or the number of components in a bundle is not evenly divisible by layer size (*LayerAmount*), there will be a remainder, yielding one or more odd-count stacks or layers. By default, the odd-count stack or layer size may contain as few as one component. This may exceed equipment cycle times, and flimsy components (newspapers) may cause problems with downstream equipment such as strappers. *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.

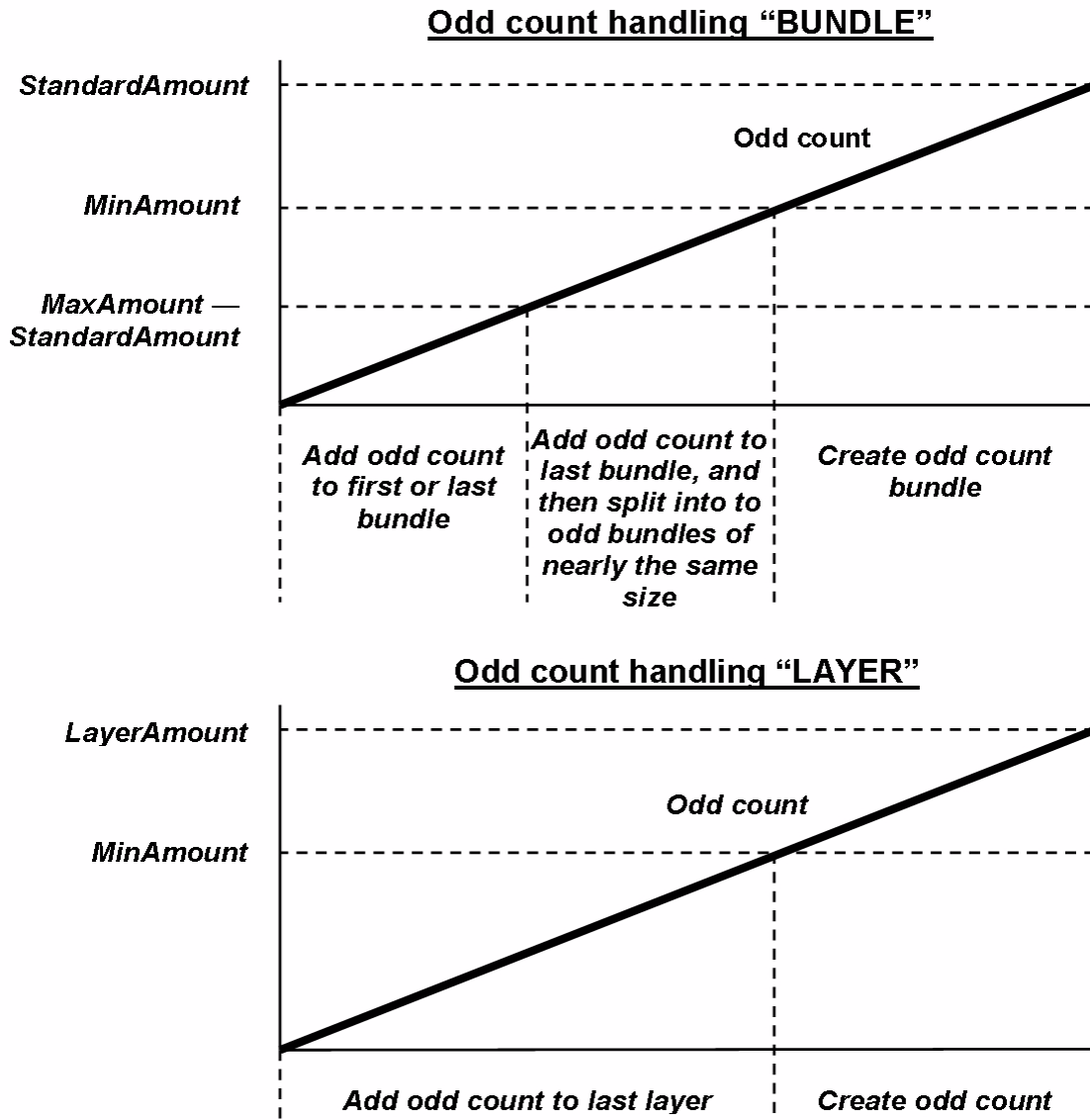


Figure 7.19: Odd Count Handling for Bundling

**Resource Properties**

Resource class: Parameter  
 Resource referenced by: —  
 Example Partition: —  
 Input of processes: **Stacking**  
 Output of processes: —

**Resource Structure**

Name	Data Type	Description
<u><a href="#">LayerAmount ?</a></u> <u><a href="#">Modified in JDF 1.2</a></u>	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> .

Name	Data Type	Description
<i>StandardAmount</i> ? <a href="#">Modified in JDF 1.2</a>	integer	Number of products in a standard stack.
<i>MaxAmount</i> ?	integer	Maximum number of products in a stack, <i>MaxAmount</i> $\geq$ <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> ) $\leq$ <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>MaxWeight</i> ?	double	Maximum weight of a stack in grams.
<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>Offset</i> ? <a href="#">Deprecated in JDF 1.2</a>	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 <b>Disjointing</b> in JDF 1.2 and beyond.
<b>Disjointing</b> ? <a href="#">New in JDF 1.2</a>	reference	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.

### 7.2.153 StitchingParams

[Clarified in JDF 1.2](#)

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.

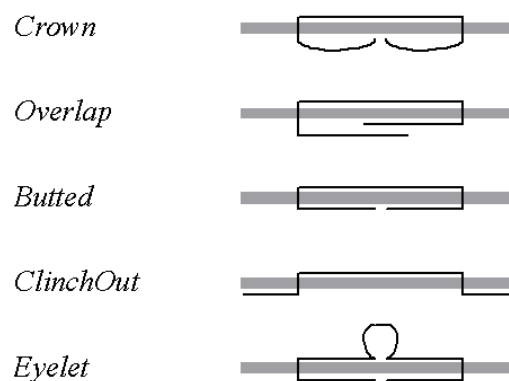


Figure 7.20: Staple shapes

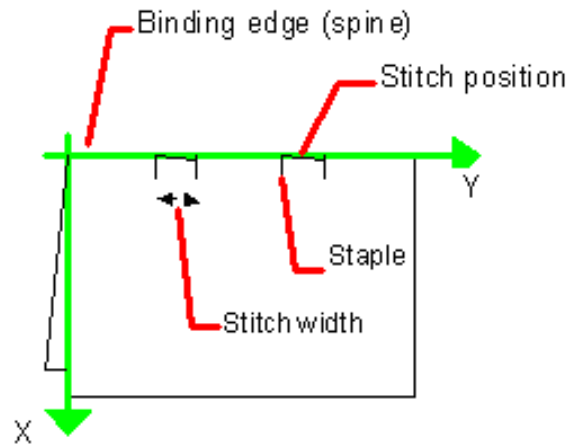


Figure 7.21: Parameters and coordinate system used for saddle stitching

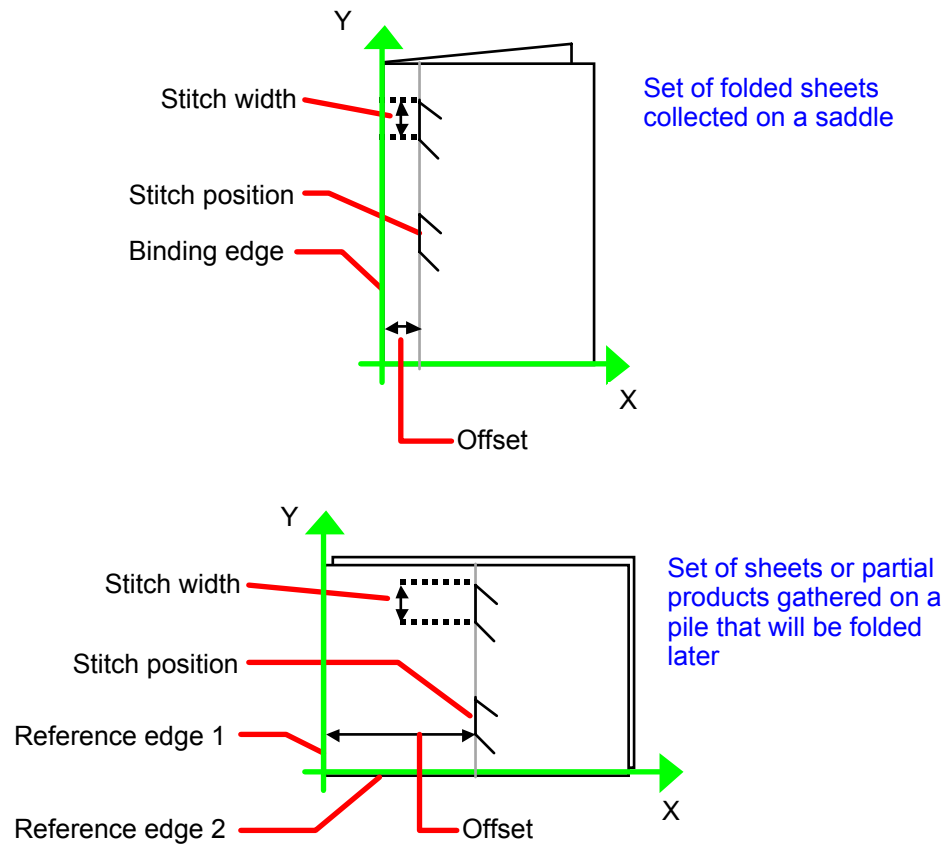


Figure 7.22: Parameters and coordinate system used for stitching



## Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>Stitching</b>
Output of processes:	—

## Resource Structure

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> ? <a href="#">Modified in JDF 1.2</a>	integer	Number of stitches. If not specified, use the system-specified number of stitches which may vary depending on other attributes set in this resource. Use a "0" value to use the stitcher without inserting any stitches. Use "NoOp" to bypass the stitcher altogether.
<i>ReferenceEdge</i> ? <a href="#">New in JDF 1.1</a> <a href="#">Deprecated in JDF 1.2</a>	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 <b>Stitching</b> process is combined with other processes (e.g., <b>DigitalPrinting</b> ) where, when combined, there is no input <b>Component</b> to be stitched. Possible values are: <i>Top</i> <i>Left</i> <i>Right</i> <i>Bottom</i> <i>ReferenceEdge</i> has been replaced in JDF 1.2 and beyond with an explicit <i>Transformation</i> or <i>Orientation</i> of the input <b>Component</b> . If both <i>Transformation/Orientation</i> and <i>ReferenceEdge</i> are specified, the result is the matrix product of both transformations. <i>Transformation/Orientation</i> must be applied first.
<i>Offset</i> ?	double	Distance between stitch and binding edge. If <i>StitchType</i> = "Saddle", <i>Offset</i> must be ignored. Note that it is possible to describe saddle stitching with an offset by defining <i>StitchType</i> = "Side" with a large <i>Offset</i> value.
<i>StapleShape</i> ?	enumeration	Specifies the shape of the staples to be used. Possible values are: <i>Crown</i> <i>Overlap</i> <i>Butted</i> <i>ClinchOut</i> <i>Eyelet</i> Representations of these values are displayed in Figure 7.20.
<i>StitchFromFront</i> ? <a href="#">Deprecated in JDF 1.2</a>	boolean	If "true", <b>Stitching</b> is done from front to back. Otherwise it is done from back to front. <i>StitchFromFront</i> has been replaced with an explicit <i>Transformation</i> or <i>Orientation</i> of the input <b>Component</b> .
<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> .

Name	Data Type	Description
<i>StitchType</i> ? <a href="#">Modified in JDF 1.2</a>	enumeration	Specifies the type of the <b>Stitching</b> operation. One of: <i>Corner</i> – Stitch in the corner that is at the clockwise end of the reference edge. For example, to stitch in the top right corner set <code>ComponentLink/@Orientation = "Rotate90"</code> . <i>Saddle</i> – Stitch on the middle fold which is on the saddle. <i>Side</i> – Stitch along the reference edge.
<i>StitchWidth</i> ?	double	Width of the stitch to be used. If not present or "0", means use the system-specified width of stitches which may vary depending on other attributes set in this resource.
<i>WireGauge</i> ?	double	Gauge of the wire to be used. If not present or "0", means use the system-specified wire gauge which may vary depending on other attributes set in this resource.
<i>WireBrand</i> ?	string	Brand of the wire to be used.

### 7.2.154 Strap

[New in JDF 1.1](#)

#### Resource Properties

Resource class:	Consumable
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>Strapping</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>StrapColor</i> ?	NamedColor	Color of the string or strap.
<i>Material</i>	enumeration	Strap material. Possible values are: <i>AdhesiveTape</i> <i>Strap</i> <i>String</i>

### 7.2.155 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:	<b>Strapping</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>StrappingType</i>	enumeration	Strapping pattern. Allowed values are: <i>Single</i> – One strap. <i>Double</i> – Two parallel single straps. <i>Cross</i> – Two crossed straps. <i>DoubleCross</i> – Two cross straps that strap each side of a box.

## 7.2.156 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:	—

### Resource Structure

Name	Data Type	Description
<i>Brand</i> ?	string	The name of the comb manufacturer and the name of the specific item.
<i>Distance</i> ? <a href="#">Deprecated in JDF 1.2</a>	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 <b>HoleMakingParams/@HoleType</b> .
<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.
<b>HoleMakingParams</b> ? <a href="#">New in JDF 1.2</a>	refelement	Details of the holes in <i>StripBinding</i> .

## 7.2.157 StrippingParams

[New in JDF 1.2](#)

The **StrippingParams** resource is a high-level description of how a **Component** is to be produced. It is typically produced by the MIS production planning module and consumed by a prepress workflow system, although its usage is not restricted to this example. There are enough optional attributes to use the same resource for the interface between estimation systems and production planning systems.

**StrippingParams** specifies how the surfaces of the **BinderySignatures** of a job are placed onto press sheets and also gives concrete values for the various **StripCellParams** defined by the **BinderySignature**.

The partitioning of **StrippingParams** defines the structure of the finished product and the structure of the **Layout** resource that is produced by the **Stripping** process. It is therefore strongly recommended to partition the **StrippingParams** resource by *SheetName*. Note that not all attributes and elements may 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:	—

## Resource Structure

Name	Data Type	Description
<i>AssemblyID</i> ?	string	Identification of the <b>Assembly</b> or <b>AssemblySection</b> to which the <b>StrippingParams</b> or partition belongs.
<i>JobID</i> ?	string	Identification of the original job to which the <b>StrippingParams</b> or partition belongs. If not specified, it defaults to the value specified or implied in the JDF node.
<i>SectionList</i> ?	IntegerList	List of numbered sections (of the <b>AssemblySections</b> with matching <i>JobID</i> and <i>AssemblyID</i> ) that should be flowed into the <b>BinderySignature</b> . If not specified, a linear sequence of sections is assumed. The section that matches the first entry is flowed into <b>SignatureCells</b> with <i>SectionIndex</i> = "0"; the section that matches the second entry is flowed into <b>SignatureCells</b> with <i>SectionIndex</i> = "1" and so forth. <i>SectionList</i> must not be specified at the <i>CellIndex</i> partition level.
<i>WorkStyle</i> ?	enumeration	The direction in which to turn the press sheet. Possible values are defined in <b>ConventionalPrintingParams/@WorkStyle</b> : <i>WorkStyle</i> must not be specified at partition levels lower than <i>SheetName</i> .
<b>BinderySignature</b>	refelement	Describes <b>BinderySignature</b> which are placed onto the sheets defined by <b>StrippingParams</b> . If multiple <b>BinderySignature</b> elements are placed on the same sheet, <b>StrippingParams</b> must be partitioned by <i>BinderySignatureName</i> . <b>BinderySignature</b> must not be specified at partition levels lower than <i>PartVersion</i> .
<b>Device</b> *	refelement	Devices that the MIS has planned to execute this <b>StrippingParams</b> . This may include prepress devices, presses, or finishing devices. Press devices must not be specified at partition levels lower than <i>SheetName</i> .
<b>Media</b> *	refelement	<b>Media</b> to be used for this <b>StrippingParams</b> . This may include paper, plate, or film media. Paper media must not be specified at partition levels lower than <i>SheetName</i> .
<b>Position</b> *	element	The <b>Position</b> element specifies how the <b>BinderySignature</b> is placed onto a sheet. Multiple <b>Position</b> objects in one <b>StrippingParams</b> specify multiple identical <b>BinderySignatures</b> with the same content. In case the <b>BinderySignature</b> is defined by <b>SignatureCells</b> , 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 <b>BinderySignature</b> is defined by <i>FoldCatalog</i> or <i>Folds</i> , then, by default, the lay is placed on the left front side of the sheet. Using the <i>Orientation</i> attribute one can influence this default behavior. <i>Position</i> must not be specified at partition levels lower than <i>PartVersion</i> .
<i>StripCellParams</i> ?	element	Specification of the parameters of the cells in the layout.

### Structure of the Position Subelement

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 *RelativeBoxes*, are added to calculate the final margin between objects.

Name	Data Type	Description
<i>MarginBottom ?</i>	Double	Bottom margin, in points, to be left outside of the <b>BinderySignature</b> that this <b>Position</b> applies to. The coordinate system is defined by the front side of the <b>StrippingParams</b> .
<i>MarginTop ?</i>	Double	Top margin, in points, to be left outside of the <b>BinderySignature</b> that this <b>Position</b> applies to. The coordinate system is defined by the front side of the <b>StrippingParams</b> .
<i>MarginLeft ?</i>	Double	Left margin, in points, to be left outside of the <b>BinderySignature</b> that this <b>Position</b> applies to. The coordinate system is defined by the front side of the <b>StrippingParams</b> .
<i>MarginRight ?</i>	Double	Right margin, in points, to be left outside of the <b>BinderySignature</b> that this <b>Position</b> applies to. The coordinate system is defined by the front side of the <b>StrippingParams</b> .
<i>Orientation ?</i>	Orientation	Named orientation describing the transformation of the orientation of the <b>BinderySignature</b> on the <b>StrippingParams</b> . For details, see Table 2-3, "Matrices and Orientation values used to describe the orientation of a Component," on page 24.
<i>RelativeBox ?</i>	Rectangle	Relative position of this <b>BinderySignature</b> on the front side of the <b>StrippingParams</b> . If not specified, the full media box "0 0 1.0 1.0" is applied.

### Structure of StripCellParams Subelement

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 generally default to "0". Nonetheless the parameters have no schema default.

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.
<i>Spine ?</i>	double	(S2) Amount of paper which is not cut-off from the spine.
<i>TrimHead ?</i>	double	(H2) Value for the trim distance at the head side.
<i>TrimFoot ?</i>	double	(T2) Value for the trim distance at the foot side.
<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.

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>Sides ?</i>	enumeration	Indicates whether contents should be printed on one or both sides of the media. Possible values are: <i>OneSided</i> – Page contents will only be imaged on one side of the media. <i>TwoSidedHeadToHead</i> – Impose pages upon the front and back sides of media sheets so that the head (top) of page contents back up to each other. <i>TwoSidedHeadToFoot</i> – Impose pages upon the front and back sides of media sheets so that the head (top) of the front backs up to the foot (bottom) of the back.

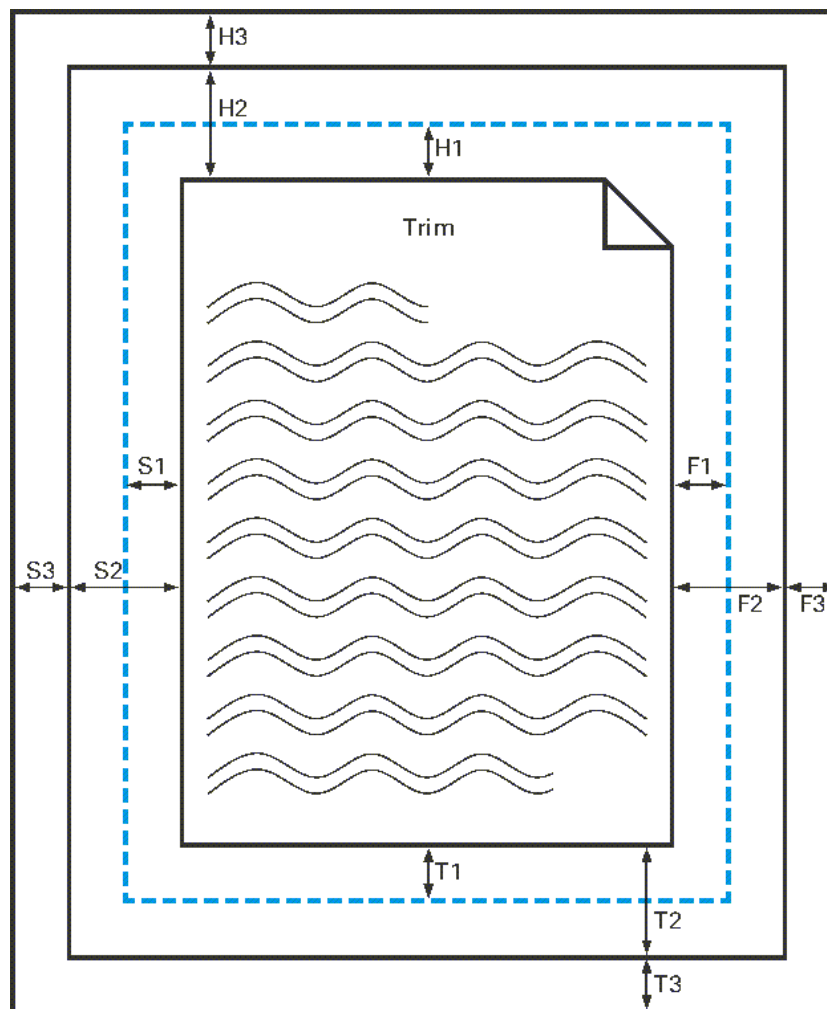


Figure 7.23: Definition of margins in StripCellParams

## 7.2.158 Surface

This resource describes the marks on a sheet surface. Up to two **Surface** resources may be defined for a **Sheet**.

### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>Sheet</b>
<b>Example Partition:</b>	<i>Side</i> . Otherwise it is strongly discouraged to partition the <b>Layout</b> tree, including <b>Surface</b> .
<b>Input of processes:</b>	—
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
<i>Side</i>	enumeration	The side of the <b>Sheet</b> that the <b>Surface</b> describes. Possible values are: <i>Front</i> <i>Back</i>
<i>SurfaceContentsBox</i> ?	rectangle?	This rectangle provides the region of the surface into which the contents of <b>ContentObjects</b> and <b>MarkObjects</b> are to be imaged. <b>Note:</b> The <i>SurfaceContentBox</i> also provides a translation for an object's <i>CTM</i> . (See Section 7.1.1.1, Structure of Abstract Span Subelement.)
<i>PlacedObject</i> *	element	Provides a list of the <b>ContentObject</b> and <b>MarkObject</b> elements to be placed on to the surface. Contains the marks on the surface in rendering order. See the description that follows. <b>Note:</b> <i>PlacedObject</i> is not a container but an abstract type.

### Structure of the Abstract PlacedObject Subelement

The marks that may be placed on the designated **Surface** come in two varieties: **ContentObject** or **MarkObject** elements. Both inherit characteristics from the abstract **PlacedObject** element type, and both are described below.

Name	Data Type	Description
<i>ClipBox</i> ?	rectangle	Clipping rectangle in the coordinates of the <i>SurfaceContentsBox</i> .
<i>CTM</i>	matrix	The coordinate transformation matrix ( <i>CTM</i> — a Postscript term) of the object in the <i>SurfaceContentsBox</i> . <i>CTM</i> is applied to the native coordinate system of the object. <b>Note:</b> <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 <b>ContentObject</b> . Specified in the coordinate systems of <i>SurfaceContentBox</i> .
<i>LayerID</i> ? <a href="#">New in JDF 1.1</a>	integer	If a layout supports layering (e.g., for versioning), <i>LayerID</i> may be used to identify the layer that a <b>ContentObject</b> belongs to, (e.g., the language layer version). The details of the layers are optionally specified in the <b>Layout/LayerList/@LayerDetails</b> key.
<i>OrdID</i> ? <a href="#">New in JDF 1.1</a>	integer	If a layout supports layering (e.g., for versioning), <i>OrdID</i> may be used to identify <b>ContentObjects</b> that belong to the same final page. These will have a matching <i>OrdID</i> .
<i>SourceClipPath</i> ?	PDFPath	Clip path for the <b>PlacedObject</b> in the coordinates of the source page. <i>SourceClipPath</i> is applied to the referenced source object in addition to any clipping that is internal to the object.

Name	Data Type	Description
<a href="#">TrimCTM ?</a> <a href="#">New in JDF 1.1</a>	matrix	The transformation matrix of the object's trim box in the <i>SurfaceContentsBox</i> . <i>TrimCTM</i> and <i>CTM</i> are identical if the trim box and dimension of the object in the <i>PlacedObject</i> are identical. Note that imposition programs that execute the <b>Layout</b> must recalculate the <i>CTM</i> in case the object is replaced with a new object with different dimensions, otherwise the position of the content inside the trim box will shift. This recalculation is based on <i>TrimCTM</i> , <i>TrimSize</i> and trim box.
<a href="#">TrimSize ?</a> <a href="#">New in JDF 1.2</a>	XYPair	The size of the object's trim box as viewed in the <i>SurfaceContentBox</i> ( <i>TrimCTM</i> scaling and rotation applied). Needed when replacing the object by a new object with a different dimension. <b>Note:</b> Recalculation of <i>PlacedObject/@CTM</i> is only necessary when the <b>Imposition</b> process needs to replace some pages from the provided <b>RunList</b> (using the <b>Layout</b> as a kind of imposition "template"). To ensure correct placement of a new page in the <b>Layout</b> , <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 <b>RunList</b> page trimming information (= <i>LayoutElement/@SourceTrimBox</i> when real trim box of the object is known). Usage of both <i>PlacedObject's TrimCTM</i> and <i>TrimSize</i> attributes will allow page replacements on any type of imposition <b>Layout</b> .
<a href="#">Type ?</a> <a href="#">Deprecated in JDF 1.1</a>	enumeration	Describes the kind of <i>PlacedObject</i> . Possible values are: <i>Content</i> <i>Mark</i>

## 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 should 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.

## 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 may include (but these boxes may not be correct in all cases):

- EPS – HiResBoundingBox then BoundingBox
- PDF – CropBox then MediaBox

### Structure of ContentObject Subelement

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.

Name	Data Type	Description
<i>DocOrd</i> ? <a href="#">New in JDF 1.1</a>	integer	Reference to an index of an instance document in the content <b>RunList</b> . This references an instance document with an index module. <b>Layout/@MaxDocOrd</b> equals <i>DocOrd</i> in an automated layout scenario. The index may either be known explicitly from a variable <b>Runlist</b> or implicitly from the index within an indexable content definition language, (e.g., PPML).
<i>Ord</i> ? <a href="#">Modified in JDF 1.1</a>	integer	A non-negative, zero-based reference to an index in the content <b>RunList</b> . The index is incremented for every page of the <b>RunList</b> with <i>IsPage</i> = "true". The <i>Ord</i> value of the first page of a <b>RunList</b> has the value "0".
<i>OrdExpression</i> ?	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. <i>Ord</i> or <i>DocOrd</i> and <i>OrdExpression</i> are mutually exclusive in one PlacedObject.
<i>SetOrd</i> ? <a href="#">New in JDF 1.1</a>	integer	A non-negative, zero-based reference to an index of a document set in the content <b>RunList</b> . This references an instance document with an index module. <b>Layout/@MaxSetOrd</b> = <i>SetOrd</i> in an automated layout scenario. The index may either be known explicitly from a variable <b>Runlist</b> or implicitly from the index within an indexable content definition language, (e.g., PPML).

### Using Ord to reference elements in RunLists

[New in JDF 1.1A](#)

The *Ord* attribute in ContentObject or MarkObject elements represents a reference to a *logical* element in a **RunList**. The reference is not changed by repartitioning the **RunList**. The content and marks **RunList** are referenced independently. The following examples illustrate the usage of *Ord*.

#### Simple Multi-File unseparated RunList

This example specifies all pages contained in File1.pdf and File2.pdf. File 1 has 6 pages, file 2 has an unknown number of pages.

```
<RunList Class="Parameter" ID="L3" PartIDKeys="Run" Status="Available">
  <RunList NPage="6" Pages="0~5" Run="1">
    <LayoutElement>
      <FileSpec URL=" File:///File1.pdf"/>
    </LayoutElement>
  </RunList>
  <RunList Pages="0~-1" Run="2">
    <LayoutElement>
      <FileSpec URL="File:///File2.pdf"/>
    </LayoutElement>
  </RunList>
</RunList>
```

Table 7-6: Example 1 of Ord in PlacedObjects

Ord	File	Page	Ord	File	Page
0	File1	0	1	File1	1
2	File1	2	3	File1	3
4	File1	4	5	File1	5
6	File2	0	7	File2	1
8	File2	2	(n)	File2	(n-6)

### Simple Multi-File separated RunList

This example specifies two pages contained in Presep.pdf and following that, pages 1, 3, and 5 of each preprepared file.

```
<RunList Class="Parameter" ID="Link0003" PartIDKeys="Run Separation"
  Status="Available">
  <RunList NPage="2" Run="1" SkipPage="3">
    <LayoutElement>
      <FileSpec URL="File:///Presep.pdf"/>
    </LayoutElement>
    <RunList FirstPage="0" IsPage="false" Separation="Cyan"/>
    <RunList FirstPage="1" IsPage="false" Separation="Magenta"/>
    <RunList FirstPage="2" IsPage="false" Separation="Yellow"/>
    <RunList FirstPage="3" IsPage="false" Separation="Black"/>
  </RunList>
  <RunList IsPage="true" Pages="1 3 5" Run="2">
    <RunList IsPage="false" Separation="Cyan">
      <LayoutElement>
        <FileSpec URL="File:///Cyan2.pdf"/>
      </LayoutElement>
    </RunList>
    <RunList IsPage="false" Separation="Magenta">
      <LayoutElement>
        <FileSpec URL="File:///Magenta2.pdf"/>
      </LayoutElement>
    </RunList>
    <RunList IsPage="false" Separation="Yellow">
      <LayoutElement>
        <FileSpec URL="File:///Yellow2.pdf"/>
      </LayoutElement>
    </RunList>
    <RunList IsPage="false" Separation="Black">
      <LayoutElement>
        <FileSpec URL="File:///Black2.pdf"/>
      </LayoutElement>
    </RunList>
  </RunList>
</RunList>
```

Table 7-7: Example 2 of Ord in PlacedObjects

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

Table 7-7: Example 2 of Ord in PlacedObjects

Ord	File	Page	Separation	Ord	File	Page	Separation
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

### Using Expressions in the OrdExpression Attribute

Expressions can use the operators +, -, \*, /, % and parentheses, operating on integers and two variables: *s* for signature number (starting at 0) and *n* for number of pages to be imposed in one document. Signature number denotes the number of times that a complete set of placed objects has been filled with content from the run list. The operators have the same meaning as in the C programming language. Expressions are evaluated with normal “C” operator precedence. Multiplication must be expressed by explicitly including the \* operator, (i.e., use “2\*s”, not “2 s”). Remainders are discarded.

### OrdExpression Examples (Saddlestitched booklet for variable page length documents)

The following describes the OrdExpressions for a booklet with varying page lengths. The example page assignments are for a book of 13-16 pages.

```
Front:
OrdExpression = "2*s"           0    2    4    6
OrdExpression = "4*((n+3)/4) - (s*2) - 1"  15   13   11   9
Back:
OrdExpression = "2*s+1"       1    3    5    7
OrdExpression = "4*((n+3)/4) - (s*2) - 2"  14   12   10   8
```

### DocOrd Usage Examples (Two-sided business cards 4/sheet)

The following describes the Ord + DocOrd usage for a 4-up step + repeat business card

MaxDocOrd=4

```
Front:
Ord=0 DocOrd=0
Ord=0 DocOrd=1
Ord=0 DocOrd=2
Ord=0 DocOrd=3
Back:
Front:
Ord=1 DocOrd=0
Ord=1 DocOrd=1
Ord=1 DocOrd=2
Ord=1 DocOrd=3
```

### Structure of MarkObject Elements

MarkObject elements describe containers for page marks on a surface. They are filled from the Marks **RunList** of the **Imposition** process. An individual MarkObject represents the content data of the Marks. The content data in individual MarkObjects may contain multiple logical marks: CIELABMeasuringField, ColorControlStrip, CutMark, DensityMeasuringField, IdentificationField, RegisterMark, and ScavengerArea.

Name	Data Type	Description
<i>LayoutElementPageNum</i> = "0" <a href="#">New in JDF 1.1</a>	integer	Page number to use from the PDL file described by the <i>LayoutElement</i> attribute.
<i>Ord</i> ? <a href="#">Modified in JDF 1.1A</a>	integer	A non-negative reference to an index in the marks <b>RunList</b> . The index is incremented for every page of the <b>RunList</b> with <i>IsPage</i> = "true". The first page of a <b>RunList</b> has the value 0.

Name	Data Type	Description
<b>CIELABMeasuringField</b> *	refelement	Specific information about this kind of mark object.
<b>ColorControlStrip</b> * <a href="#">Modified in JDF 1.1</a>	refelement	Specific information about this kind of mark object.
<b>CutMark</b> * <a href="#">Modified in JDF 1.1</a>	refelement	Specific information about this kind of mark object.
<b>DensityMeasuringField</b> * <a href="#">Modified in JDF 1.1</a>	refelement	Specific information about this kind of mark object.
<b>DeviceMark</b> ? <a href="#">New in JDF 1.1</a>	refelement	If neither <i>Ord</i> nor <b>LayoutElement</b> are specified, it is assumed that the device can independently generate the mark. <b>DeviceMark</b> defines a set of formatting parameters for the mark.
<b>DynamicField</b> *	refelement	Definition of text replacement for a <b>MarkObject</b> .
<b>IdentificationField</b> *	refelement	Specific information about this kind of mark object.
<b>JobField</b> * <a href="#">New in JDF 1.1</a>	refelement	Specific information about this kind of mark object.
<b>LayoutElement</b> ?	refelement	PDL description of the mark. <b>LayoutElement</b> and <i>Ord</i> are mutually exclusive within one <b>MarkObject</b> .
<b>RegisterMark</b> * <a href="#">Modified in JDF 1.1</a>	refelement	Specific information about this kind of mark object.
<b>ScavengerArea</b> * <a href="#">New in JDF 1.1</a>	refelement	Specific information about this kind of mark object

### Structure of DynamicField Subelement

Name	Data Type	Description
<i>Format</i>	string	Format string in C printf format that defines the replacement.
<i>InputField</i> ? <a href="#">Deprecated in JDF 1.1</a>	string	String that must be replaced by the <b>DynamicInput</b> element in the Contents <b>RunList</b> referenced by <i>Ord</i> or <i>OrdExpression</i> .
<i>Ord</i> ?	integer	Reference to an index in the Contents <b>RunList</b> that contains <b>DynamicInput</b> elements. Only one of <i>Ord</i> or <i>OrdExpression</i> may be specified.
<i>OrdExpression</i> ?	string	Expression to calculate the reference to an index in the Contents <b>RunList</b> that contains <b>DynamicInput</b> fields. For details, see the definition of <i>OrdExpression</i> in the description of the <b>PlacedObject</b> element. Only one of <i>Ord</i> or <i>OrdExpression</i> may 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>Ord</i> , <i>OrdExpression</i> . If <i>ReplaceField</i> is not specified, the Device that processes the <b>DynamicField</b> must format the <b>DynamicField</b> .
<i>Template</i>	string	Template to define a sequence of variables consumed by <i>Format</i> . A list of predefined values is found in the description of the resource. In addition, the <i>Name</i> attribute of <b>DynamicInput</b> elements of a <b>RunList</b> define further variables.
<b>DeviceMark</b> ? <a href="#">New in JDF 1.1</a>	refelement	<b>DeviceMark</b> defines the formatting parameters for the mark. If not specified, the <b>DeviceMark</b> settings defined in <b>LayoutPreparationParams</b> or in the <b>Layout</b> tree are assumed.

## DynamicField Subelement Properties

DynamicField provides a description of dynamic text replacements for MarkObjects. This element should 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 MarkObjects with optional additional DynamicField elements that define text replacement.

### Example usage of a DynamicField Element:

```
<RunList Class="Parameter" ID="L3" PartIDKeys="Run" Status="Available">
  <DynamicInput Name="i1">Joe</DynamicInput>
  <DynamicInput Name="i2">John</DynamicInput>
  <LayoutElement ElementType="Graphic">
    <FileSpec URL="File:///Variable.pdf"/>
  </LayoutElement>
</RunList>
<Surface Class="Parameter" ID="Link0003" Side="Front" Status="Available">
  <!--The MarkObject in the Layout hierarchy: -->
  <MarkObject CTM="1 0 0 1 0 0">
    <LayoutElement ElementType="Graphic">
      <FileSpec URL="File:///MyReplace.pdf"/>
    </LayoutElement>
    <DynamicField Format="Replacement Text for %s and %s go in here at %s on %s"
Ord="0" ReplaceField="___xxx___" Template="i1,i2,Time,Date"/>
  </MarkObject>
</Surface>
```

In the example above, the text “\_\_\_xxx\_\_\_” in the file MyReplace.pdf would be replaced by the sentence “Replacement Text for Joe and John go in here at 14:00 on Mar-31-2000”.

MyReplace.pdf is placed at the position defined by the *CTM* of the MarkedObject and Variable.pdf is placed at the position defined by the *CTM* of the PlacedObject.

## 7.2.159 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:	<b>ThreadSealing</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>BlindStitch</i> ?	boolean	If “ <i>true</i> ”, a blind stitch after last stitch is required.
<i>ThreadMaterial</i> ?	enumeration	Thread material. Possible values are: <i>Cotton</i> <i>Nylon</i> <i>Polyester</i>
<i>ThreadPositions</i> ? <a href="#">Modified in JDF 1.2</a>	DoubleList	Array containing the Y-coordinate of the center positions of the thread.
<i>ThreadLength</i> ? <a href="#">Modified in JDF 1.2</a>	double	Length of one thread.
<i>ThreadStitchWidth</i> ? <a href="#">Modified in JDF 1.2</a>	double	Width of one stitch.
<i>SealingTemperature</i> ?	integer	Temperature needed for sealing thread and sheets together, in degrees centigrade.

### 7.2.160 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).

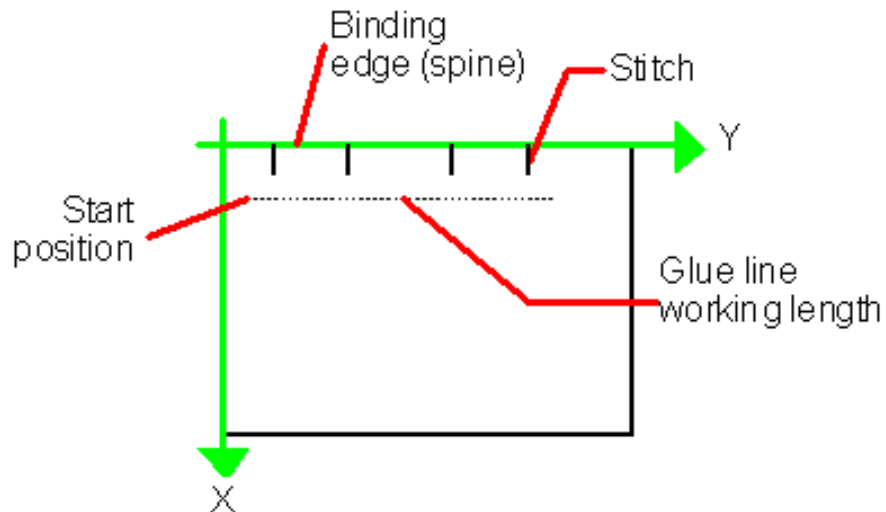


Figure 7.24: Parameters and coordinate system used for thread sewing

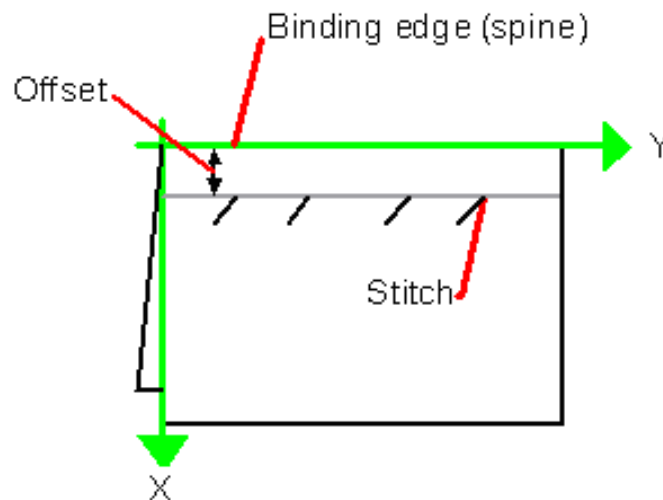


Figure 7.25: Parameters and coordinate system used for side sewing

## Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>ThreadSewing</i>
Output of processes:	—

## Resource Structure

Name	Data Type	Description
<i>BlindStitch</i> = "false"	boolean	If "true", a blind stitch after last stitch is required.
<i>CastingMaterial</i> ?	enumeration	Casting material of the thread being used. Possible values are: <i>Cotton</i> <i>Nylon</i> <i>Polyester</i>
<i>CoreMaterial</i> ?	enumeration	Core material of the thread being used. This attribute must be used to define the thread material if there is no casting. Possible values are: <i>Cotton</i> <i>Nylon</i> <i>Polyester</i>
<i>GlueLineRefSheets</i> ? <a href="#">Modified in JDF 1.2</a>	IntegerList	This entry is only required if <i>GlueLine</i> is defined. It contains the indices of the loose parts of the input component after which gluing should be applied. The index starts with 0.
<i>Offset</i> ? <a href="#">New in JDF 1.1</a>	double	Specifies the distance between the stitch and the binding edge. Used only for side stitching.
<i>NumberOfNeedles</i> ? <a href="#">Modified in JDF 1.2</a>	integer	Specifies the number of needles to be used.
<i>NeedlePositions</i> ?	DoubleList	Array containing the Y-coordinate of the needle positions. The number of entries must match the number given in <i>NumberOfNeedles</i> .
<i>Sealing</i> ?	boolean	If "true", thermo-sealing is required.
<i>SewingPattern</i> ?	enumeration	Sewing pattern. Possible values are: <i>Normal</i> <i>Staggered</i> <i>CombinedStaggered</i> <i>Side – Side</i> sewing.
<i>ThreadThickness</i> ?	double	Thread thickness.
<i>ThreadBrand</i> ?	string	Thread brand.
<b>GlueLine</b> *	element	Gluing parameters.

### 7.2.161 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 3.8.2, Description of Partitionable Resources) by *TileID*. Individual tiles are selected and matched by specifying the appropriate *TileID* attribute, which is described in Table 3-28, “Contents of the Part element,” on page 79.

### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>TileID</i>
Input of processes:	<b>Tiling</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>ClipBox</i>	rectangle	A rectangle that defines the bounding box of the <b>Surface</b> contents which will be imaged on this <b>Tile</b> . The <i>ClipBox</i> is defined in the coordinate system of the <b>Surface</b> .
<i>CTM</i>	matrix	A coordinate transformation matrix mapping the <i>ClipBox</i> for this <b>Tile</b> to the rectangle 0 0 X Y, where X and Y are the extents of the media that the <b>Tile</b> will be imaged onto.
<b>Media</b> ? <a href="#">New in JDF 1.2</a>	refelement	Describes the media to be used.
<b>MediaSource</b> ? <a href="#">Deprecated in JDF 1.2</a>	refelement	Describes the media to be used. Replaced with <b>Media</b> in JDF 1.2

### 7.2.162 Tool

[New in JDF 1.1](#)

A **Tool** resource defines a generic tool that is customized for needed for a given job, (e.g., an embossing stamp). The manufacturing process for the tool is not described within JDF.

### Resource Properties

Resource class:	Handling
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>Embossing, ShapeCutting</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>ToolAmount</i> ?	integer	Number of identical instances of the tool that the tool contains, (e.g., the number of cut forms in a die cutting die).
<i>ToolID</i>	string	ID of the tool. This is a unique name within the workflow.
<i>ToolType</i> ?	NMTOKEN	Type of the tool. Possible values include: <i>EmbossingCalendar</i> <i>EmbossingStamp</i> <i>CutDie</i>



### 7.2.163 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](http://www.cip4.org/documents/technical_info/cip3v3_0.pdf).

#### Resource Properties

**Resource class:** Parameter  
**Resource referenced by:** **Color, TransferCurvePool**  
**Example Partition:** *RibbonName, SheetName, Side, WebName*  
**Input of processes:**  
**Output of processes:** —

#### Resource Structure

Name	Data Type	Description
<i>Curve</i>	TransferFunction	The density mapping curve for the separation defined by Separation.
<i>Separation ?</i>	string	The name of the separation. If <i>Separation</i> = "All", this curve should be applied to all separations that are not explicitly defined.

### 7.2.164 TransferCurvePool

A transfer curve pool is a collection of TransferCurveSet elements that each contains information about a TransferCurve. Multiple TransferCurvesSets may exist at one time. For example, one may exist for the laser calibration of the imagesetter, one for the **ContactCopying** process and one for the printing process. Each TransferCurveSet consists of one or more TransferCurve elements. A TransferCurve element should be applied to the appropriate correlative *Separation*, or to all *Separations* when *Separation* = "All". The TransferCurveSets should be concatenated in the following order:

Film -> Plate -> Press -> Paper.  
 and  
 Proof.

In addition to the **TransferCurve** element, the TransferCurveSet elements contain device dependent geometrical information, (e.g., *CTM* definitions).

#### Resource Properties

**Resource class:** Parameter  
**Resource referenced by:** **TransferFunctionControl**  
**Example Partition:** —  
**Input of processes:** ***ContoneCalibration, DigitalPrinting, ImageSetting, InkZoneCalculation, PreviewGeneration, Stripping***  
**Output of processes:** ***LayoutPreparation***

#### Resource Structure

Name	Data Type	Description
TransferCurveSet *	element	The set of transfer curves.

#### Structure of TransferCurveSet Subelement

TransferCurveSet elements describe both the characteristic curve of transfer and the relation between the various process coordinate systems.

**TransferCurveSet**

Name	Data Type	Description
<i>CTM</i> ? <a href="#">New in JDF 1.1</a>	matrix	Defines the transformation of the coordinate system in the device as defined by <i>Name</i> .
<i>Name</i> <a href="#">Modified in JDF 1.2</a>	NMTOKEN	The name of the TransferCurveSet. Possible values include: <i>Film</i> – The transformation from the <b>Layout</b> system to the <i>Film</i> . In a CTP or <b>DigitalPrinting</b> 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 <b>DigitalPrinting</b> environment, this defaults to the identity matrix and the identity <b>TransferCurve</b> . <i>Paper</i> – The transformation from the <i>Press</i> system to the <i>Paper</i> . <i>Press</i> – The transformation from the <i>Plate</i> system to the <i>Press</i> . <i>Proof</i> – The transformation from the <b>Layout</b> system to the <i>Proof</i> . <a href="#">New in JDF 1.2</a>
<b>TransferCurve</b> * <a href="#">Modified in JDF 1.1</a>	refelement	List of TransferCurve entries.

**7.2.165 TransferFunctionControl****Resource Properties**

Resource class:	Parameter
Resource referenced by:	<b>SeparationControlParams</b>
Example Partition:	—
Input of processes:	<b>ContoneCalibration</b>
Output of processes:	—

**Resource Structure**

Name	Data Type	Description
<i>TransferFunctionSource</i>	enumeration	Identifies the source of transfer curves which should be applied during separation. <i>Document</i> – Use the transfer curves provided in the document. <i>Device</i> – Use transfer functions provided by the output device. When <b>Separation</b> is being performed pre-RIP, this may mean that no transfer curves will be applied. <i>Custom</i> – Use the transfer curves provided in the TransferCurvePool element of this element.
<b>TransferCurvePool</b> ?	refelement	Provides a set of transfer curves to be used by the process.

**7.2.166 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, SheetName, Side, SignatureName</i>
Input of processes:	<b>RIPing, Trapping</b>
Output of processes:	—

## Resource Structure

Name	Data Type	Description
<i>DefaultTrapping</i> = "false"	boolean	If "true", pages that have no defined <i>TrapRegions</i> are trapped using the set of <i>TrappingParams</i> . The <i>BleedBox</i> is used for the <i>TrapZone</i> . If "false", only pages that have <i>TrapRegions</i> are trapped.
<i>IgnoreFileParams</i> = "true" <a href="#">Clarified in JDF 1.2</a>	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 <b>TrappingDetails</b> (and the <b>Trapping</b> process) is not present, then the trapping defined in PostScript may still be applied.
<i>Trapping</i> ? <a href="#">Deprecated in JDF 1.2</a>	boolean	If "true", trapping is enabled. If "false", trapping is disabled. Use <i>NoOp</i> in JDF 1.2 and above.
<i>TrappingOrder</i> ?	element	Trapping processes will trap colorants as if they are laid down on the media in the order specified in <i>TrappingOrder</i> . The colorant order may affect which colors to spread, especially when opaque inks are used.
<i>TrappingType</i> ? <a href="#">Deprecated in JDF 1.2</a>	integer	Identifies the trapping method to be used by the trapping process. The number identifies the minor (last three digits) and major (any digits prior to the last three) version of the trapping type requested.
<b>TrappingParams</b> ?	refelement	A <b>TrappingParams</b> resource that is used to define the default trapping parameters when <i>DefaultTrapping</i> = "true".
<b>ObjectResolution</b> * <a href="#">New in JDF 1.1</a>	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.
<b>TrapRegion</b> *	refelement	A set of <b>TrapRegion</b> 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.

## Structure of the TrappingOrder Subelement

Name	Data Type	Description
<i>SeparationSpec</i> * <a href="#">Modified in JDF 1.2</a>	refelement	An array of colorant names.

### 7.2.167 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). The attributes of this resource that are optional in the sense that each implementation decides a default value for them.

#### Resource Properties

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	<b>TrapRegion, TrappingDetails</b>
<b>Example Partition:</b>	<i>DocIndex, RunIndex, RunTags, SheetName, Side, SignatureName</i>
<b>Input of processes:</b>	—
<b>Output of processes:</b>	—

## Resource Structure

Name	Data Type	Description
<i>BlackColorLimit</i> ?	double	A number between 0 and 1 that specifies the lowest color value required 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. <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> ? <a href="#">Deprecated in JDF 1.2</a>	boolean	If "true", trapping is enabled for zones that are defined with this parameter set. Use <i>NoOp</i> in JDF 1.2 and above.
<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 <b>PDLResourceAlias</b> 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 "true", the planes of color images are trapped against each other. If "false", the planes of color images are not trapped against each other.
<i>ImageResolution</i> ?	integer	A positive integer indicating the minimum resolution, in dpi, for downsampled images. Images can be downsampled by a power of 2 before traps are calculated. The downsampled image is used only for calculating traps, while the original image is used when printing the image.
<i>ImageMaskTrapping</i> ?	boolean	Controls trapping when the <i>TrapZone</i> contains a stencil mask. A stencil mask is a monochrome image in which each sample is represented by a single bit. The stencil mask is used to paint in the current color: image samples with a value of "1" are marked, samples with a value of "0" are not marked. When "false", none of the objects covered by the clipped bounding box of the stencil mask are trapped. No traps are generated between the stencil mask and objects that the stencil mask overlays. No traps are generated between objects that overlay the stencil mask and the stencil mask. For all other objects, normal trapping rules are followed. Two objects on top of the stencil mask that overlap each other may generate a trap, regardless of the value of this parameter. When "true", objects are trapped to the stencil mask, and to each other.
<i>ImageToImageTrapping</i> ?	boolean	If "true", traps are generated along a boundary between images. If "false", this kind of trapping is not implemented.
<i>ImageToObjectTrapping</i> ?	boolean	If "true", images are trapped to other objects. If "false", this kind of trapping is not implemented.

Name	Data Type	Description
<i>ImageTrapPlacement</i> ?	enumeration	Controls the placement of traps for images. Possible values are: <i>Center</i> – Trap is centered on the edge between the image and the adjacent object. <i>Choke</i> – Trap is placed in the image. <i>Normal</i> – Trap is based on the colors of the areas. <i>Spread</i> – Trap is placed in the adjacent object.
<i>ImageTrapWidth</i> ? <a href="#">New in JDF 1.2</a>	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 <b>ByteMap</b> defined in the input <b>RunList</b> 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> ? <a href="#">New in JDF 1.2</a>	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 <b>ByteMap</b> defined in the input <b>RunList</b> when <i>ImageToImageTrapping</i> , <i>ImageToObjectTrapping</i> and/or <i>ImageInternalTrapping</i> are set to <i>true</i> . The parameter applies only to non-black traps, if an image color on either side qualifies as black, the effective black trap width is used to compute the size of the trap, this is based on <i>TrapWidth</i> , <i>BlackWidth</i> , and <i>MinimumBlackWidth</i> .  Values must be greater than or equal to zero. A value of 0.0 disables non-black image trapping. Defaults to <i>ImageTrapWidth</i> .
<i>MinimumBlackWidth</i> = "0"	double	Specifies the minimum width, in points, of a trap that uses black ink. Allowable values are those greater than or equal to zero.
<i>SlidingTrapLimit</i> ?	double	A number between 0 and 1. Specifies when to slide traps towards a center position. If the neutral density of the lighter area is greater than the neutral density of the darker area multiplied by the <i>SlidingTrapLimit</i> , then the trap slides. This applies to vignettes and non-vignettes. No slide occurs at "1".
<i>StepLimit</i> ? <a href="#">Modified in JDF 1.2</a>	double	A non-negative number. Specifies the smallest step required 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 ( $\leq 1$ ) was removed in JDF 1.2.

Name	Data Type	Description
<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. Possible values include: <i>Miter</i> <i>Overlap</i> Other values may be added later as a result of customer requests.
<i>TrapJoinStyle</i> = "Miter"	NMTOKEN	Specifies the style of the connection between the ends of two traps created by consecutive segments along a path. Possible values include: <i>Bevel</i> <i>Miter</i> <i>Round</i>
<i>TrapWidth</i> ? <a href="#">Modified in JDF 1.2</a>	double	Specifies the trap width, in points in X direction (horizontal) of the PDF or <b>ByteMap</b> defined in the input <b>RunList</b> . Also defines the unit used in trap width specifications for certain types of objects such as <i>BlackWidth</i> .
<i>TrapWidthY</i> ? <a href="#">New in JDF 1.2</a>	double	Specifies the trap width, in points in Y direction (vertical). Also defines the unit used in trap width specifications for certain types of objects such as <i>BlackWidth</i> . If not specified, defaults to the value of <i>TrapWidth</i> .
<i>ColorantZoneDetails</i> *	element	<i>ColorantZoneDetails</i> subelements. Entries in this dictionary reflect the results of any named colorant aliasing specified. Each entry defines parameters specific for one named colorant. If the colorant named is neither listed in the <i>ColorantParams</i> array nor implied by the <i>ProcessColorModel</i> for the <i>ColorantControl</i> object in effect when these <i>TrappingParams</i> are applied, the entry is not used for trapping.

### Structure of ColorantZoneDetails Subelement

Name	Data Type	Description
<i>Colorant</i>	string	The colorant name that occurs in the <i>SeparationSpec/@Name</i> of the <i>ColorantParams</i> array of the <i>ColorantControl</i> object used by the process.
<i>StepLimit</i> ?	double	A number between 0 and 1. Specifies the smallest step required in the color value of a colorant to trigger trapping at a given boundary. If the higher color value at the boundary exceeds the lower value by an amount that is equal or greater than the larger of 0.05 or <i>StepLimit</i> times the lower value ( $low + \max(StepLimit * low, 0.05)$ ), then the edge is a candidate for trapping. The value 0.05 is set to avoid trapping light areas in vignettes. If omitted, the <i>StepLimit</i> attribute in the <b>TrappingParams</b> resource is used.
<i>TrapColorScaling</i> ?	double	A number between 0 and 1. Specifies a scaling of the amount of color applied in traps towards the neutral density of the dark area. A value of "1" means the trap has the combined color values of the darker and the lighter area. A value of "0" means the trap colors are reduced so that the trap has the neutral density of the darker area. If omitted, the <i>TrapColorScaling</i> attribute in the <b>TrappingParams</b> resource is used.

## 7.2.168 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:	<b>TrappingDetails</b>
Example Partition:	—
Input of processes:	—
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>TrapZone</i> ?	PDFPath	Each element within <i>TrapZone</i> is one subpath of a complex path. The <i>TrapZone</i> is the area that results when the paths are filled using the non-zero winding rule. When absent, the <i>MediaBox</i> array for the <b>RunList</b> defines the <i>TrapZone</i> .
<i>Pages</i>	IntegerRangeList	Identifies a set of pages from the <b>RunList</b> to trap using the specified geometry and trapping style.
<i>TrappingParams</i> ?	refelement	The set of trapping parameters which will be used when trapping in this region.

## 7.2.169 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).

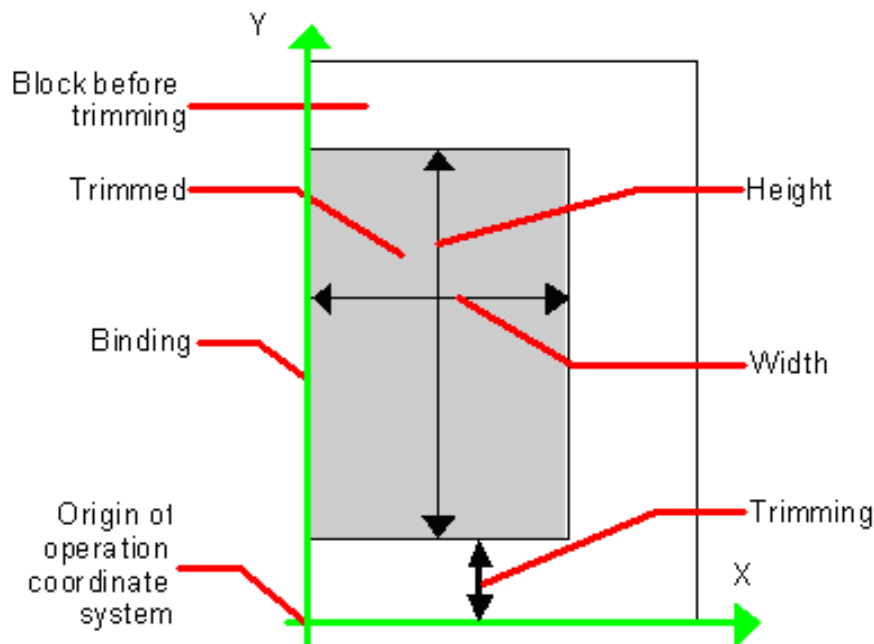


Figure 7.26: Parameters and coordinate system used for trimming

**Resource Properties**

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>Trimming</b>
Output of processes:	—

**Resource Structure**

Name	Data Type	Description
<i>Height</i> ?	double	Height of the trimmed product.
<i>TrimmingOffset</i> ?	double	Amount to be cut at bottom side.
<i>TrimmingType</i> ? <a href="#">New in JDF 1.1</a> <a href="#">Deprecated in JDF 1.2</a>	enumeration	Trimming operation to perform. Possible values are: <i>Detailed</i> – Cut the amount specified by <i>Height</i> , <i>Width</i> and <i>TrimmingOffset</i> . <i>SystemSpecified</i> – Cut the amount specified by the system.
<i>Width</i> ?	double	Width of the trimmed product.

**7.2.170 VerificationParams**

This resource provides the parameters of a **Verification** process.

**Resource Properties**

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>Verification</b>
Output of processes:	—

**Resource Structure**

Name	Data Type	Description
<i>FieldRange</i> ?	IntegerRangeList	Zero-based range list of integers that determines which characters of the data in <b>IdentificationField</b> should be applied to the field formatting strings. If not specified all characters are applied.
<i>InsertError</i> ?	string	Database insertion statement in <i>C printf</i> format defining how information read from the resource of the <b>Verification</b> process should be stored in case of verification errors. The database is defined by the <b>DBSelection</b> resource of the <b>Verification</b> 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 <b>IdentificationField</b> should be stored in case of verification success. The database is defined by the <b>DBSelection</b> 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 may 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

**7.2.171 WireCombBindingParams**This resource describes the details of the **WireCombBinding** process.**Resource Properties****Resource class:** Parameter**Resource referenced by:** —**Example Partition:** —**Input of processes:** **WireCombBinding****Output of processes:** —**Resource Structure**

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> ?	Named-Color	Determines the color of the comb.
<i>Diameter</i> ?	double	The comb diameter is determined by the height of the block of sheets to be bound.
<i>Distance</i> ? <a href="#">Deprecated in JDF 1.2</a>	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 <b>HoleMakingParams/@HoleType</b> .
<i>FlipBackCover</i> = "false" <a href="#">New in JDF 1.1</a>	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 required (e.g., inserting a CD) before closing the book.
<i>Material</i> ?	enumeration	The material used for forming the wire comb binding. Possible values are: <i>LaqueredSteel</i> <i>TinnedSteel</i> <i>ZincsSteel</i>
<i>Shape</i> = "Single"	enumeration	The shape of the wire comb binding. Possible values are: <i>Single</i> – Each “tooth” is made with one wire. <i>Twin</i> – The shape of each “tooth” is made with a double wire.
<i>Thickness</i> ?	double	The thickness of the comb material.
<b>HoleMakingParams</b> ? <a href="#">New in JDF 1.2</a>	refelement	Details of the holes in <b>WireCombBinding</b> .

## 7.2.172 WrappingParams

[New in JDF 1.1](#)

**WrappingParams** defines the details of *Wrapping*. Details of the material used for *Wrapping* can be found in the **Media** resource that is also an input of the *Wrapping* process.

### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<i>Wrapping</i>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>WrappingKind</i>	enumeration	<i>LooseWrap</i> – The wrap is loose around the component. <i>ShrinkWrap</i> – The wrap is shrunk around the component.

## 7.3 Device Capability Definitions

[New in JDF 1.1](#)

[Modified in JDF 1.2](#)

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 359. For more details on preflight, See “Preflight” on page 204.

Capabilities descriptions that are saved in files must be represented as a signal with a JMF response to the *KnownDevices* query.



### Preflighting in Device Capabilities

While the actions and tests described in this section as pertaining to “preflighting” may 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 may or may not be part of your 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.

### 7.3.1 Structure of the DeviceCap Subelement

[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.

Name	Data Type	Description
<i>CombinedMethod</i> = “None”	enumeration	Specifies how the processes specified in <i>Types</i> may be specified. One of: <i>Combined</i> – The list of processes in <i>Types</i> must be specified as a <i>Combined</i> process. <i>ProcessGroup</i> – The list of processes in <i>Types</i> must be specified as a <i>ProcessGroup</i> of individual processes. <i>CombinedProcessGroup</i> – The list of processes in <i>Types</i> may be specified either as a <i>Combined</i> process or as a <i>ProcessGroup</i> of individual processes. <i>None</i> – No support for <i>Combined</i> or <i>ProcessGroup</i> . Only one individual process type defined in <i>Types</i> is supported.

Name	Data Type	Description
<i>ExecutionPolicy</i> = "AllFound" <a href="#">New in JDF 1.2</a>	enumeration	Describes the policy for finding and executing JDF nodes as described in "Determining Executable Nodes" on page 110. <i>RootNode</i> – The device will execute the root JDF node only. It will not search the JDF tree for executable nodes. This will commonly be used for sub JDF nodes that have been spawned and targeted explicitly for the device. <i>FirstFound</i> – The device will execute the first node found in the JDF tree that is executable by this device. The search order is defined by the order in the XML. <i>AllFound</i> – The device will execute all executable nodes found in multiple passes of the JDF tree that are executable by this device. The results of executing a node are applied to the tree between passes.
<i>GenericAttributes</i> ?	NMTOKENS	List of generic attributes that are supported and unrestricted by the device implementation. Note that descriptions of attributes that appear in <i>State</i> elements (see the following Section 7.3.1.2.2.1, Structure of the Abstract State Subelement) overwrite the description in <i>GenericAttributes</i> .
<i>Lang</i> ? <a href="#">New in JDF 1.2</a>	languages	Specifies the localization(s) provided with the capabilities. If not specified, no localizations are provided.
<i>OptionalCombinedTypes</i> ? <a href="#">Deprecated in JDF 1.2</a>	NMTOKENS	List of optional JDF Node types. The entries of the list must be a subset of <i>Types</i> . For example, a RIP with optional in-RIP trapping would specify <i>OptionalCombinedTypes</i> = "Trapping" if <i>Types</i> = "Trapping Interpreting Rendering". Replaced by <i>TypeExpression</i> in JDF 1.2 and beyond
<i>Type</i> ? <a href="#">Deprecated in JDF 1.2</a>	NMTOKEN	JDF <i>Type</i> attribute of the supported process. Extension types may be specified by stating the <i>Namespace</i> prefix in the value. In JDF 1.2 and beyond, a single value of type is also defined in the <i>Types</i> attribute.
<i>TypeExpression</i> ? <a href="#">New in JDF 1.2</a>	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).
<i>TypeOrder</i> ? <a href="#">Deprecated in JDF 1.2</a>	enumeration	Ordering restriction for <i>Combined</i> or <i>ProcessGroup</i> nodes. <i>Fixed</i> – The order of process types specified in the <i>Types</i> attribute is ordered, and each type can be specified only once, (e.g., <i>Cutting, 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. Replaced by <i>TypeExpression</i> in JDF 1.2 and beyond.
<i>Types</i> ? <a href="#">Modified in JDF 1.2</a>	NMTOKENS	This attribute represents the list of supported JDF node <i>Type</i> values. If any of the node types are in a <i>Namespace</i> other than JDF, the <i>Namespace</i> prefix should be included in this list. The ordering is significant.

Name	Data Type	Description
<a href="#">ActionPool ?</a> <a href="#">New in JDF 1.2</a>	element	Container for zero or more <b>Action</b> elements for use as constraints.
<b>DevCaps *</b>	element	List of definitions of the accepted resources. The <b>DevCaps</b> elements are combined with a logical AND, (i.e., a JDF must fulfill all restrictions defined by the set of <b>DevCaps</b> ). Only resources that are specified within this list are honored by the device.
<a href="#">DisplayGroupPool ?</a> <a href="#">New in JDF 1.2</a>	element	List of <b>DisplayGroup</b> 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.
<a href="#">FeaturePool ?</a> <a href="#">New in JDF 1.2</a>	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.
<a href="#">MacroPool ?</a> <a href="#">New in JDF 1.2</a>	element	Container for zero or more macro elements, each of which contains an expression that may cause <i>State</i> attribute values (e.g., <i>CurrentValue</i> or <i>UserDisplay</i> ) to be changed.
<b>Performance *</b>	element	Specification of a devices performance capabilities.
<a href="#">TestPool ?</a> <a href="#">New in JDF 1.2</a>	element	Container for zero or more <b>Test</b> elements that are referenced from <b>Actions</b> .

### 7.3.1.1 Structure of the ActionPool Subelement

[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 **BasicPreflightReport** values. (See Structure of the Abstract Evaluation Subelement in "Structure of the abstract Term Subelement" on page 524. 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 may contain, is interdependent on **TestPool** and the **Test** and **Term** elements it may contain. For more information on **TestPool**, see "Structure of the **TestPool** Subelement" on page 524.

Name	Data Type	Description
<b>Action *</b>	element	A list of independent <b>Actions</b> .

#### 7.3.1.1.1 Structure of the Action Subelement

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.

Name	Data Type	Description
<i>Severity</i> = "Error"	enumeration	Indicates how the severity of the failure should be treated when the expression defined by <i>TestRef</i> is violated. One of: <i>Error</i> – The client should display an error message and not allow the conflicting settings to persist. <i>Warning</i> – The client should notify the user of the condition but allow the settings to persist if the user requests. <i>Information</i> – The client should allow the settings to persist but inform the user of the issue.

Name	Data Type	Description
<i>ID</i>	ID	Unique identifier of the <b>Action</b> element. This ID is used to refer to the <b>Action</b> element, (e.g., from a preflight report).
<i>TestRef</i>	IDREF	Reference to a <b>Test</b> element that is executed to evaluate this <b>Action</b> .
<i>Loc *</i>	element	Text to describe an error if the <b>Test</b> fails. (See “Structure of the Loc Subelement” on page 507.)
<i>PreflightAction ?</i>	element	Provides additional constraints that are specific to the <b>Preflight</b> process. See "PreflightParams" on page 434.

### 7.3.1.2 Structure of the DevCaps Subelement

[New in JDF 1.1](#)

The **DevCaps** element describes the valid parameter space of a JDF Resource, message, or resource link that is consumed, honored, or produced by a device. Note that **DevCaps** not only describes the structure of the individual resources and resource links but also of the **NodeInfo** element within a JDF node. The **DevCaps** element may be used to model product intent resources as well as process definition resources.

Name	Data Type	Description
<i>Availability = "Installed"</i> <a href="#">New in JDF 1.2</a>	enumeration	Specifies whether the feature described by this <b>DevCaps</b> element is available on the device. Possible values are: <i>Installed</i> – The feature is installed on the device and is available for use. <i>NotInstalled</i> – The feature has not been installed on the device. <i>NotLicensed</i> – The feature has been installed on the device but may not be used until licensed. <i>Disabled</i> – The feature is installed and licensed on the device but has been disabled.
<i>Context = "Resource"</i> <a href="#">New in JDF 1.2</a>	enumeration	Describes whether the <b>DevCaps</b> context is within a resource or a link to a resource (not applicable to <b>DevCaps</b> elements within messages). One of: <i>Resource</i> – The <b>DevCaps</b> context is describing a resource or generic element. <i>Link</i> – The <b>DevCaps</b> context is describing a link to a resource. <i>JMF</i> – The <b>DevCaps</b> context describes a JMF message.
<i>DevNS = "http://www.CIP4.org/JDFSchema_1_1"</i>	URI	<i>NameSpace</i> of the resource or message that is described.
<i>ID ?</i> <a href="#">New in JDF 1.2</a>	ID	ID of this <b>DevCaps</b> element. Used for reference from <b>Performance</b> elements.
<i>LinkUsage ?</i> <a href="#">New in JDF 1.2</a>	enumeration	Used when the <i>Context</i> of this <b>DevCaps</b> = “ <i>Resource</i> ” or “ <i>Link</i> ”. This field qualifies whether the <b>DevCaps</b> describes a resource used as an input to a process or as the output of a process. One of: <i>Input</i> – The <b>DevCaps</b> describes an input resource. <i>Output</i> – The <b>DevCaps</b> describes an output resource. If not specified, this <b>DevCaps</b> applies to both usages.

Name	Data Type	Description
<i>Name</i> <a href="#">Modified in JDF 1.2</a>	NMTOKEN	Fully qualified name of the element that is described. If <i>Name</i> = <i>"NodeInfo"</i> , it describes the structure of the <i>NodeInfo</i> information that is accepted by the device. When describing elements of a <i>ResourceLink</i> , <i>Name</i> must be the name of the referenced resource and <i>Context</i> = <i>"Link"</i> .  In order to distinguish between multiple resources that have varying <i>ResourceUsage</i> attributes or <i>ProcessUsage</i> attributes, the respective <i>DevCap</i> elements must specify a name state with a <i>Name</i> = <i>"ResourceUsage"</i> and an <i>AllowedValueRange</i> = <value of <i>ResourceUsage</i> > or <i>Name</i> = <Value of <i>ProcessUsage</i> >, <i>Context</i> = <i>"Link"</i> and <i>AllowedValueRange</i> = <value of <i>ProcessUsage</i> >.
<i>Required</i> = <i>"false"</i> <a href="#">New in JDF 1.2</a>	boolean	If <i>"true"</i> , the element described by this <i>DevCaps</i> 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 defined by the JDF specification when the specification requires the resource to be specified. If the JDF specification requires an attribute, <i>Required</i> must be <i>"true"</i> .
<i>ResourceUpdate</i> = <i>"None"</i>	NMTOKENS	Specifies the capability to handle partial updates defined in <i>ResourceUpdate</i> elements. Possible values include: <i>None</i> – <i>ResourceUpdate</i> is not supported. Must not be combined with any other value. <i>JMFID</i> – JMF <i>Resource</i> messages that reference <i>ResourceUpdates</i> that have been previously loaded to the device are accepted. <i>PDLID</i> – References from PDL data, (e.g., PPML <i>TicketRef</i> elements that reference <i>ResourceUpdates</i> that have been previously loaded to the device are accepted).
<i>TypeOccurrenceNum</i> ? <a href="#">New in JDF 1.2</a>	IntegerRange-List	Specifies which occurrence(s) of the name of this <i>DevCaps</i> element that is specified either within the <i>DeviceCap Types</i> attribute or by the <i>TypeExpression</i> attribute that this <i>DevCaps</i> element applies to. If not specified, elements belonging to all JDF nodes with a matching type that are not defined by other <i>DevCaps</i> entries.  Note: this is an index into the list of matching <i>Type</i> value and not an index into the complete list specified by <i>Types</i> or <i>TypeExpression</i> . The first occurrence is <i>"0"</i> , and the last occurrence is <i>"-1"</i> , etc.
<i>Types</i> ? <a href="#">Deprecated in JDF 1.2</a>	NMTOKENS	List of JDF Node types that a <i>DevCaps</i> applies to. The value of <i>Types</i> must be a subset of <i>Types</i> in <i>DeviceCap</i> . Replaced by <i>TypeOccurrenceNum</i> in JDF 1.2 and beyond.
<i>DevCap</i> + <a href="#">Clarified in JDF 1.2</a>	element	List of definitions of the accepted parameter space for resources and messages. The parameter spaces of multiple <i>DevCap</i> elements are combined as a superset of the individual <i>DevCap</i> elements. Only elements that are explicitly specified as <i>DevCap</i> elements within a <i>DevCaps</i> are supported.  When a capabilities description is constructed using constraints, each <i>DevCaps</i> should only contain a single <i>DevCap</i> element (although <i>DevCap</i> elements may still contain multiple <i>DevCap</i> sub-elements).
<i>Loc</i> * <a href="#">New in JDF 1.2</a>	element	The localization(s) of the resource, message, or resource link name as described by this <i>DevCaps</i> element. (See "Structure of the <i>Loc</i> Subelement" on page 507.)

### 7.3.1.2.1 Structure of the Loc Subelement

[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.

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, resource link, element, message, attribute name, or attribute value).

### 7.3.1.2.2 Structure of the DevCap Subelement

[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 7.3.1.2.2.1, Structure of the Abstract State Subelement. 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 resource link belonging to the resource, (e.g., *Transformation* or the various pipe control parameters may also be restricted).

Name	Data Type	Description
<i>Availability</i> ? <a href="#">New in JDF 1.2</a>	enumeration	Specifies whether the feature described by this DevCap element is available on the device. Possible values are: <i>Installed</i> – The feature is installed on the device and is available for use. <i>NotInstalled</i> – The feature has not been installed on the device. <i>NotLicensed</i> – The feature has been installed on the device but may not be used until licensed. <i>Disabled</i> – The feature is installed and licensed on the device but has been disabled. If not specified, the value specified in the parent DevCaps or DevCap element is applied.
<i>DevNS</i> = "http:// www.CIP4.o rg/ JDFSchema_ 1_1"	URI	<i>Namespace</i> of the element that is described by this DevCap.

Name	Data Type	Description
<i>MaxOccurs</i> = "1" <a href="#">Modified in JDF 1.2</a>	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.
<i>Name</i>	NMTOKEN	Fully qualified name of the resource that is described. <i>ResourceUsage</i> attribute or <i>ProcessUsage</i> of the respective resource within a JDF node. Default = the value of <i>Name</i> of the parent DevCaps element.
DevCap *	element	Definition of the accepted parameter space for the messages or resources subelements.
Loc * <a href="#">New in JDF 1.2</a>	element	The localization(s) of the element name. (See "Structure of the Loc Subelement" on page 507.)
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 resource that is not specified DeviceCaps/ <i>@GenericAttributes</i> . If a resource attribute has no matching State element in DevCap, it is not supported.

#### 7.3.1.2.2.1 Structure of the Abstract State Subelement

[New in JDF 1.1](#)

The following table describes the common, data type-independent parameters of all State objects.

Name	Data Type	Description
<i>Availability</i> ? <a href="#">New in JDF 1.2</a>	enumeration	Specifies whether the feature described by this State element is available on the device. Possible values are: <i>Installed</i> – The feature is installed on the device and is available for use. <i>NotInstalled</i> – The feature has not been installed on the device. <i>NotLicensed</i> – The feature has been installed on the device but may not be used until licensed. <i>Disabled</i> – The feature is installed and licensed on the device, but has been disabled. If not specified, the value specified or implied by the parent element is applied.
<i>ActionRefs</i> ? <a href="#">New in JDF 1.2</a>	IDREFS	Zero or more references to Actions that operate on the parameter. All Actions referenced must evaluate to "false" for the value of the State element to be valid. Any Actions referenced in ActionRefs should be evaluated whenever the attribute described by this State element is manipulated or changed in order to catch any attributes that become invalid due to the manipulation.
<i>DependentMacroRef</i> ? <a href="#">New in JDF 1.2</a>	IDREF	A reference to a macro that conditionally modifies the <i>UserDisplay</i> attribute of this State element. If present, this referenced macro should be executed when the State/ <i>@UserDisplay</i> "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 "Structure of the MacroPool Subelement" on page 522.



Name	Data Type	Description
<i>DevNS</i> = "http://www.CIP4.org/JDFSchema_1_1"	URI	<i>Namespace</i> of the attribute that is described by this <b>State</b> element.
<i>Editable</i> = "true" <a href="#">New in JDF 1.2</a>	boolean	When "true", the feature and its current value may be edited by the user. If "false", the user interface must not allow user modification of the <b>State</b> '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> ? <a href="#">New in JDF 1.2</a>	ID	An identification value to allow external reference.
<i>ListType</i> = "SingleValue" <a href="#">New in JDF 1.2</a>	enumeration	Specifies what type of list or object the <b>State</b> variable describes. One of: <i>CompleteList</i> – The <b>State</b> describes a list of individual values. Each value must occur exactly once. <i>CompleteOrderedList</i> – The <b>State</b> describes an ordered list of individual values. Each value must occur only once and in the specified order. <i>ContainedList</i> – The <b>State</b> describes a list of individual values. The <b>State</b> = "true" if at least one of the values occurs. This value is only expected to be used in <b>BasicPreflightTest</b> elements. <i>List</i> – The <b>State</b> describes a list of individual values. <i>OrderedList</i> – The <b>State</b> describes an ordered list of individual values. <i>OrderedRangeList</i> – The <b>State</b> describes an ordered RangeList of individual values. <i>RangeList</i> – The <b>State</b> describes a RangeList of values. <i>SingleValue</i> – The <b>State</b> describes an individual value. <i>Span</i> – The <b>State</b> describes a <b>Span</b> element in a product intent resource. <i>UniqueList</i> – The <b>State</b> describes a list of individual values. Each value may occur only once. <i>UniqueRangeList</i> – The <b>State</b> describes a RangeList of values. Each explicit or implied value may occur only once. <i>UniqueOrderedList</i> – The <b>State</b> describes an ordered list of individual values. Each value may occur only once. <i>UniqueOrderedRangeList</i> – The <b>State</b> describes an ordered RangeList of individual values. Each explicit or implied value may occur only once.
<i>MacroRefs</i> ? <a href="#">New in JDF 1.2</a>	IDREFS	Zero or more references to <b>macros</b> that operate on the parameter. These <b>macros</b> set other <b>State</b> attribute values as appropriate. Any <b>macros</b> referenced in <i>MacroRefs</i> should be evaluated whenever the attribute described by this <b>State</b> element is manipulated or changed to affect any necessary changes to other attributes. <b>Macros</b> may change attributes such as the <i>CurrentValue</i> attribute of a <b>State</b> or its <i>UserDisplay</i> attribute. For more information on <b>macro</b> definitions, see "Structure of the MacroPool Subelement" on page 522.
<i>MaxOccurs</i> = "1" <a href="#">New in JDF 1.2</a>	integer	Maximum number of elements in the list described by this <b>State</b> , (e.g., the maximum number of integers in an integer list). If <i>MaxOccurs</i> is not "1", the <b>State</b> element refers to a list or a range of list values, (e.g., a <b>NameState</b> will allow a list of NMTOKENS).

Name	Data Type	Description
<i>MinOccurs</i> = "1" <a href="#">New in JDF 1.2</a>	integer	Minimum number of elements in the list described by this <i>State</i> . If <i>MinOccurs</i> is not "1", the <i>State</i> element refers to a list or a range of list values, (e.g., a <i>NameState</i> will allow a list of NMTOKENS).
<i>Name</i> ?	NMTOKEN	Name of the attribute that is described by this <i>State</i> . If <i>Name</i> is omitted this <i>State</i> describes the element's text, (i.e., the text between the XML start and end tag).
<i>Required</i> ? <a href="#">New in JDF 1.2</a>	boolean	If "true", then the element described by this <i>DevCap</i> 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> ? <a href="#">New in JDF 1.1a</a> <a href="#">Deprecated in JDF 1.2</a>	boolean	A flag that describes whether the parameter is an intent span data type. For example a <i>State</i> element describing an <i>XYPairSpan</i> 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" <a href="#">New in JDF 1.2</a>	enumeration	Indicates whether the feature should be displayed in user interfaces. Possible values are: <i>Display</i> – The feature should be displayed. <i>Hide</i> – The feature should not be displayed. <i>Dependent</i> – The feature should be conditionally displayed depending on the action specified by the <i>macro</i> referenced by <i>DependentMacroRef</i> . Note that this action is only taken when the user interface is first initialized.
<i>Loc</i> * <a href="#">New in JDF 1.2</a>	element	The localization(s) of the <i>Name</i> of the attribute that is described by this <i>State</i> element. (See "Structure of the Loc Subelement" on page 507.)

The following types of *State* elements are defined:

Name	Data Type	Description
<i>BooleanState</i>	element	Describes a set of boolean values.
<i>DateTimeState</i> <a href="#">New in JDF 1.2</a>	element	Describes a set of dateTime values.
<i>DurationState</i> <a href="#">New in JDF 1.2</a>	element	Describes a set of duration values.
<i>EnumerationState</i>	element	Describes a set of enumeration values.
<i>IntegerState</i>	element	Describes a numerical range of integer values.
<i>MatrixState</i>	element	Describes a range of matrices. Generally used to define valid orientations of <b>Components</b> .
<i>NameState</i>	element	Describes a set of NMTOKEN values.
<i>NumberState</i>	element	Describes a numerical range of values.
<i>PDFPathState</i> <a href="#">New in JDF 1.2</a>	element	Describes a set of PDFPaths.
<i>RectangleState</i> <a href="#">New in JDF 1.2</a>	element	Describes a set of 4 value rectangle values.
<i>ShapeState</i>	element	Describes a set of 3 value shape values.
<i>StringState</i>	element	Describes a set of string values.
<i>XYPairState</i>	element	Describes a set of XYPair values.

### 7.3.1.2.2.1.1 Structure of the BooleanState Subelement

[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.

Name	Data Type	Description
<a href="#">AllowedValueList ?</a> <a href="#">Added in JDF 1.1A</a>	enumerations	A list of all legal values. Allowed list values are the boolean " <i>true</i> " and " <i>false</i> ".
<a href="#">CurrentValue ?</a>	boolean	Current value for the current running job set in the device.
<a href="#">DefaultValue ?</a>	boolean	Default value if not specified in a submitted JDF. Must be set if <i>HasDefault</i> = " <i>true</i> ".
<a href="#">PresentValueList ?</a> <a href="#">Added in JDF 1.1A</a>	enumerations	A list of all supported values that can be chosen without operator intervention. Allowed list values are the boolean " <i>true</i> " and " <i>false</i> ". If not specified, the value of <i>AllowedValueList</i> is applied.
<a href="#">ValueLoc *</a> <a href="#">New in JDF 1.2</a>	element	Localization(s) of " <i>true</i> " and/or " <i>false</i> " values. (See Structure of the ValueLoc Subelement under "Structure of the BooleanState Subelement" on page 511.)

### Structure of the ValueLoc Subelement

[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**.

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 <b>ValueLoc</b> is used in the context of a <b>MatrixState</b> ), <i>Value</i> must be an instance of the appropriate data type.
<i>Loc *</i>	element	The localization(s) of the attribute value. (See "Structure of the Loc Subelement" on page 507.)

### 7.3.1.2.2.1.2 Structure of the DateTimeState Subelement

[New in JDF 1.2](#)

This **State** subelement is used to describe ranges of date`Time` values. It inherits from the abstract **State** element described above.

Name	Data Type	Description
<a href="#">AllowedValueDurationList ?</a>	DurationRangeList	List of inclusive minimum and maximum allowed values relative to the current system time.
<a href="#">AllowedValueList ?</a>	DateTimeRangeList	A list of all supported values.
<a href="#">CurrentValue ?</a>	dateTime	Current value for the current running job set in the device.
<a href="#">DefaultValue ?</a>	dateTime	Default value if not specified in a submitted JDF. Must be set if <i>HasDefault</i> = " <i>true</i> ".
<a href="#">PresentValueDurationList ?</a>	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.
<a href="#">PresentValueList ?</a>	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.
<a href="#">ValueLoc *</a>	element	Localization(s) of specific dates. (See Structure of the ValueLoc Subelement under "Structure of the BooleanState Subelement" on page 511.)

### 7.3.1.2.2.1.3 Structure of the DurationState Subelement

[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.

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. Must be set 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 "Structure of the BooleanState Subelement" on page 511.)

### 7.3.1.2.2.1.4 Structure of the EnumerationState Subelement

[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.

Name	Data Type	Description
<i>AllowedValueList</i> ?	enumerations	A list of all supported values. Must match the enumeration defined in the resource. 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. Must match the enumeration defined in the resource.
<i>DefaultValue</i> ?	enumeration	Default value if not specified in a submitted JDF. Must match the enumeration defined in the resource. Must be set if <i>HasDefault</i> = "true".
<i>PresentValueList</i> ?	enumerations	A list of values that can be chosen without operator intervention. Must match the enumeration defined in the resource. If not specified, the value of <i>AllowedValueList</i> is applied.
ValueLoc * <a href="#">New in JDF 1.2</a>	element	Localizations of the enumerations listed in <i>AllowedValueList</i> and <i>PresentValueList</i> . (See Structure of the ValueLoc Subelement under "Structure of the BooleanState Subelement" on page 511.)

### 7.3.1.2.2.1.5 Structure of the IntegerState Subelement

[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.

Name	Data Type	Description
<i>AllowedValueList</i> ? <a href="#">Modified in JDF 1.2</a>	Integer-RangeList	A list of all supported values.
<i>AllowedValueMax</i> ? <a href="#">Deprecated in JDF 1.2</a>	integer	Inclusive maximum allowed value. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
<i>AllowedValueMin</i> ? <a href="#">Deprecated in JDF 1.2</a>	integer	Inclusive minimum allowed value. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
<i>AllowedValueMod</i> ? <a href="#">New in JDF 1.2</a>	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. <b>Note:</b> "Modulo" is the remainder of an integer division. For example: $4 \bmod 3 = 4 - 3 = 1$ ; $17 \bmod 3 = 17 - 5 * 3 = 2$ ; and $3 \bmod 3 = 3 - 3 = 0$ .
<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. Must be set if <i>HasDefault</i> = "true".
<i>PresentValueList</i> ? <a href="#">Modified in JDF 1.2</a>	Integer-RangeList	A list of values that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueList</i> is applied.
<i>PresentValueMax</i> ? <a href="#">Deprecated in JDF 1.2</a>	integer	Inclusive maximum allowed value that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueMax</i> is applied. Replaced by <i>PresentValueList</i> in JDF 1.2 and beyond.
<i>PresentValueMin</i> ? <a href="#">Deprecated in JDF 1.2</a>	integer	Inclusive minimum allowed value that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueMin</i> is applied. Replaced by <i>PresentValueList</i> in JDF 1.2 and beyond.
<i>PresentValueMod</i> ? <a href="#">New in JDF 1.2</a>	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.

Name	Data Type	Description
<a href="#">UnitType ?</a> <a href="#">New in JDF 1.2</a>	NMTO- KEN	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 may not display correctly if <i>UnitType</i> is not specified for attributes with units. Possible values include: <i>Angle</i> – The attribute is defined in degrees. <i>AngularVelocity</i> – Rotations / minute. <i>Area</i> – Area in square meters (m <sup>2</sup> ). <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 halftone, screened grayscale, and screened monotone bitmap images. <i>PaperWeight</i> – In grams per square meter (g/m <sup>2</sup> ). <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 (TIFF, BMP, etc.) 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, plate-setters, and proofers, the fundamental imaging unit, (e.g., one “on” laser or imaginghead 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.
<a href="#">ValueLoc *</a> <a href="#">New in JDF 1.2</a>	element	Localization(s) of specific values. (See Structure of the ValueLoc Subelement under "Structure of the BooleanState Subelement" on page 511.)

#### 7.3.1.2.2.1.6 Structure of the MatrixState Subelement

[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).

Name	Data Type	Description
<a href="#">AllowedRotateMod ?</a> <a href="#">New in JDF 1.2</a>	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.
<a href="#">AllowedShift ?</a> <a href="#">New in JDF 1.2</a>	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 2-3 on page 24 is subtracted from <i>AllowedShift</i> , thus all in-place rotations have an implied <i>AllowedShift</i> value of "0 0 0 0". (No shift = "0 0 0 0".) The first pair of numbers is the XY pair that defines the minimum shift, and the second pair is the XY pair that defines the maximum shift.

Name	Data Type	Description
<i>AllowedTransforms</i> ? <a href="#">New in JDF 1.2</a>	Orientations	List of valid orthogonal transformations of the matrix. Any of the eight predefined transforms for physical resources as defined in Table 2-3 on page 24.
<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. Must be set if <i>HasDefault</i> = "true".
<i>PresentRotateMod</i> ? <a href="#">New in JDF 1.2</a>	double	Present Modulo of the allowed rotations and offset in degrees that can be chosen without operator intervention. Examples: 360 – No rotation is allowed. 90 – Any orthogonal rotation. 0 – Any rotation is allowed. If not specified, the value of <i>AllowedRotateMod</i> is applied.
<i>PresentShift</i> ? <a href="#">New in JDF 1.2</a>	DoubleList	If <i>PresentTransforms</i> is specified, the implied shift defined in Table 2-3 on page 24 is subtracted from <i>PresentShift</i> , thus all in-place rotations have an implied <i>PresentShift</i> value of "0 0 0 0". If not specified, the value of <i>AllowedShift</i> is applied.
<i>PresentTransforms</i> ? <a href="#">New in JDF 1.2</a>	Orientations	Any of the eight predefined transforms for physical resources as defined in Table 2-3 on page 24. If not specified, the value of <i>AllowedTransforms</i> is applied.
<i>Value</i> *	element	A list legal values.

### Structure of the Value Element

Name	Data Type	Description
<i>AllowedValue</i>	matrix	A legal value for a matrix variable.
<i>PresentValue</i> ? <a href="#">Deprecated in JDF 1.2</a>	matrix	A legal value for a matrix variable that can be chosen without operator intervention. If not specified, the value of <i>AllowedValue</i> is applied. In JDF 1.2 and beyond, use <i>ValueUsage</i> .
<i>ValueUsage</i> ? <a href="#">New in JDF 1.2</a>	enumeration	Defines whether the value defined in <i>AllowedValue</i> may be used as a <i>Present</i> , <i>Allowed</i> , or both values. One of: <i>Present</i> – Present configuration is supported. <i>Allowed</i> – Allowed configuration is supported. If not specified, <i>Value</i> is valid for both <i>Present</i> and <i>Allowed</i> .
<i>Loc</i> * <a href="#">New in JDF 1.2</a>	element	The localization(s) of the string defined in <i>AllowedValue</i> . (See "Structure of the Loc Subelement" on page 507.)

#### 7.3.1.2.2.1.7 Structure of the NameState Subelement

[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.

Name	Data Type	Description
<i>AllowedRegExp</i> ? <a href="#">New in JDF 1.2</a>	regExp	Regular expression that limits the allowed values.
<i>AllowedValueList</i> ?	NMTOKENS	A list legal values.
<i>CurrentValue</i> ?	NMTOKEN	Current value for the current running job set in the device.

Name	Data Type	Description
<i>DefaultValue</i> ? <a href="#">New in JDF 1.2</a>	NMTOKEN	Default value if not specified in a submitted JDF. Must be set if <i>HasDefault</i> = "true".
<i>PresentRegExp</i> ? <a href="#">New in JDF 1.2</a>	regExp	Regular expression that limits the values that may be chosen without operator intervention. If not specified, the value of <i>AllowedRegExp</i> is applied.
<i>PresentValueList</i> ?	NMTOKENS	A list of values that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueList</i> is applied.
<i>ValueLoc</i> * <a href="#">New in JDF 1.2</a>	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 "Structure of the BooleanState Subelement" on page 511.)

### 7.3.1.2.2.1.8 Structure of the NumberState Subelement

[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.

Name	Data Type	Description
<i>AllowedValueList</i> ? <a href="#">Modified in JDF 1.2</a>	DoubleRange-List	A list of supported values.
<i>AllowedValueMax</i> ? <a href="#">Deprecated in JDF 1.2</a>	double	Inclusive maximum allowed value. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
<i>AllowedValueMin</i> ? <a href="#">Deprecated in JDF 1.2</a>	double	Inclusive minimum allowed value. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
<i>AllowedValueMod</i> ? <a href="#">New in JDF 1.2</a>	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. <b>Note:</b> "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. Must be set if <i>HasDefault</i> = "true".
<i>PresentValueList</i> ? <a href="#">Modified in JDF 1.2</a>	DoubleRange-List	A list of values that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueList</i> is applied.
<i>PresentValueMax</i> ? <a href="#">Deprecated in JDF 1.2</a>	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.
<i>PresentValueMin</i> ? <a href="#">Deprecated in JDF 1.2</a>	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.
<i>PresentValueMod</i> ? <a href="#">New in JDF 1.2</a>	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.



Name	Data Type	Description
<i>UnitType</i> ? <a href="#">New in JDF 1.2</a>	NMTO- KEN	Specifies the unit type that this <b>State</b> element represents. Used to enable an application to localize the representation of the units. <i>UnitType</i> must be specified if the <b>NumberState</b> represents a value that has units. Possible values are defined in "Structure of the IntegerState Subelement" on page 513 in the <i>UnitType</i> attribute definition. Some additional possible values that are typically used only with <i>Number</i> attributes are: <i>CMYKColor</i> – Four values representing a CMYK color. <i>LabColor</i> – Three values representing a Lab color. <i>sRGBColor</i> – Three values representing a sRGB color.
<i>ValueLoc</i> * <a href="#">New in JDF 1.2</a>	element	Localization(s) of specific values. (See Structure of the ValueLoc Subelement under "Structure of the BooleanState Subelement" on page 511.)

### 7.3.1.2.2.1.9 Structure of the PDFPathState Subelement

[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.

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.
<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. Must be set 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.
<i>Value</i> *	element	The localization(s) of the PDF path defined in <i>AllowedValue</i> .

### Structure of the 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> may be used as a <i>Present</i> , <i>Allowed</i> , or any value. One of: <i>Present</i> – Present configuration is supported. <i>Allowed</i> – Allowed configuration is supported. If not specified, <i>Value</i> is valid for both <i>Present</i> and <i>Allowed</i> .
<i>Loc</i> *	element	The localization(s) of the string defined in <i>AllowedValue</i> . (See "Structure of the Loc Subelement" on page 507.)

### 7.3.1.2.2.1.10 Structure of the RectangleState Subelement

[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.

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.

Name	Data Type	Description
<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. Must be set 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.
<i>ValueLoc</i> *	element	A list of supported values. The <i>ValueLoc/@Value</i> attribute must be a representation of a rectangle. This may also be used to localize (or provide names for) specific rectangles. (See Structure of the ValueLoc Subelement under "Structure of the BooleanState Subelement" on page 511.)

### 7.3.1.2.2.1.11 Structure of the ShapeState Subelement

[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.

Name	Data Type	Description
<i>AllowedValueList</i> ? <a href="#">Modified in JDF 1.2</a>	ShapeRangeList	A list of values that can be chosen.
<i>AllowedValueMax</i> ? <a href="#">Deprecated in JDF 1.2</a>	shape	Inclusive maximum allowed value. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
<i>AllowedValueMin</i> ? <a href="#">Deprecated in JDF 1.2</a>	shape	Inclusive minimum allowed value. Replaced by <i>AllowedValueList</i> in JDF 1.2 and beyond.
<i>AllowedX</i> ? <a href="#">New in JDF 1.2</a>	DoubleRangeList	Allowed X-Axis of the <i>Shape</i> .
<i>AllowedY</i> ? <a href="#">New in JDF 1.2</a>	DoubleRangeList	Allowed Y-Axis of the <i>Shape</i> .
<i>AllowedZ</i> ? <a href="#">New in JDF 1.2</a>	DoubleRangeList	Allowed Z-Axis of the <i>Shape</i> .
<i>CurrentValue</i> ?	shape	Current value for the current running job set in the device.
<i>DefaultValue</i> ?	shape	Default value if not specified in a submitted JDF. Must be set if <i>HasDefault</i> = "true".
<i>PresentValueList</i> ? <a href="#">Modified in JDF 1.2</a>	ShapeRangeList	A list of values that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueList</i> is applied.
<i>PresentValueMax</i> ? <a href="#">Deprecated in JDF 1.2</a>	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.
<i>PresentValueMin</i> ? <a href="#">Deprecated in JDF 1.2</a>	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.

Name	Data Type	Description
<a href="#">PresentX ?</a> <a href="#">New in JDF 1.2</a>	DoubleRange-List	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.
<a href="#">PresentY ?</a> <a href="#">New in JDF 1.2</a>	DoubleRange-List	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.
<a href="#">PresentZ ?</a> <a href="#">New in JDF 1.2</a>	DoubleRange-List	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.
<a href="#">ValueLoc *</a> <a href="#">New in JDF 1.2</a>	element	A list of supported shapes. (See Structure of the ValueLoc Subelement under "Structure of the BooleanState Subelement" on page 511.)

### 7.3.1.2.2.1.12 Structure of the StringState Subelement

[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.

Name	Data Type	Description
<a href="#">AllowedLength ?</a> <a href="#">New in JDF 1.2</a>	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").
<a href="#">AllowedRegExp ?</a> <a href="#">New in JDF 1.2</a>	regExp	Regular expression that limits the allowed values.
<a href="#">CurrentValue ?</a>	string	Current value for the current running job set in the device.
<a href="#">DefaultValue ?</a>	string	Default value if not specified in a submitted JDF. Must be set if <i>HasDefault</i> = "true".
<a href="#">PresentLength ?</a> <a href="#">New in JDF 1.2</a>	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.
<a href="#">PresentRegExp ?</a> <a href="#">New in JDF 1.2</a>	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.
<a href="#">Value +</a>	element	A list legal values.

### Structure of the Value element

[New in JDF 1.1](#)

Name	Data Type	Description
<a href="#">AllowedValue</a>	string	A legal value for a string variable.
<a href="#">PresentValue ?</a> <a href="#">Deprecated in JDF 1.2</a>	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> .
<a href="#">ValueUsage ?</a> <a href="#">New in JDF 1.2</a>	enumeration	Defines whether the value defined in <i>AllowedValue</i> may be used as a <i>Present</i> , <i>Allowed</i> , or any value. One of: <i>Present</i> – Present configuration is supported. <i>Allowed</i> – Allowed configuration is supported. If not specified, <i>Value</i> is valid for both <i>Present</i> and <i>Allowed</i> .
<a href="#">Loc *</a> <a href="#">New in JDF 1.2</a>	element	The localization(s) of the string defined in <i>AllowedValue</i> . (See "Structure of the Loc Subelement" on page 507.)

### 7.3.1.2.2.1.13 Structure of the XYPairState Subelement

[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.

Name	Data Type	Description
<a href="#">AllowedValueList</a> ? <a href="#">Modified in JDF 1.2</a>	XYPairRangeList	A list of values that can be chosen.
<a href="#">AllowedValueMax</a> ? <a href="#">Deprecated in JDF 1.2</a>	XYPair	Inclusive maximum allowed value. Replaced with <i>AllowedValueList</i> in JDF 1.2 and beyond.
<a href="#">AllowedValueMin</a> ? <a href="#">Deprecated in JDF 1.2</a>	XYPair	Inclusive minimum allowed value. Replaced with <i>AllowedValueList</i> in JDF 1.2 and beyond.
<a href="#">AllowedXYRelation</a> ? <a href="#">New in JDF 1.2</a>	XYRelation	Relative value of X vs. Y.
<a href="#">CurrentValue</a> ?	XYPair	Current value for the current running job set in the device.
<a href="#">DefaultValue</a> ?	XYPair	Default value if not specified in a submitted JDF. Must be set if <i>HasDefault</i> = "true".
<a href="#">PresentValueList</a> ? <a href="#">Modified in JDF 1.2</a>	XYPairRangeList	A list of values that can be chosen without operator intervention. If not specified, the value of <i>AllowedValueList</i> is applied.
<a href="#">PresentValueMax</a> ? <a href="#">Deprecated in JDF 1.2</a>	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.
<a href="#">PresentValueMin</a> ? <a href="#">Deprecated in JDF 1.2</a>	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.
<a href="#">PresentXYRelation</a> ? <a href="#">New in JDF 1.2</a>	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.
<a href="#">UnitType</a> ? <a href="#">New in JDF 1.2</a>	NMTOKEN	Specifies the unit type that this <b>State</b> element represents. Used to enable an application to localize the representation of the units. <i>UnitType</i> must be specified if the <i>IntegerState</i> represents a value that has units. Possible values are defined in "Structure of the IntegerState Subelement" on page 513 in the <i>UnitType</i> attribute definition.
<a href="#">ValueLoc</a> * <a href="#">New in JDF 1.2</a>	element	A list of supported shapes. (See Structure of the ValueLoc Subelement under "Structure of the BooleanState Subelement" on page 511.)

### 7.3.1.3 Structure of the DisplayGroupPool Subelement

[New in JDF 1.2](#)

The **DisplayGroupPool** element declares set(s) of related features that are intended to be displayed as a group in user interfaces. These declarations are references to individual features declared in **State** elements.

#### Example

```
<DeviceCap>
  <DisplayGroupPool>
    <DisplayGroup rRefs="btd cmp mag colorspace outputres">
      <Loc HelpText="Parameters for scanning configuration" Lang="en"
        Value="ScanningParameters"/>
    </DisplayGroup>
  </DisplayGroupPool>
</DeviceCap>
```

In this example, a single `DisplayGroup` is specified. This `DisplayGroup` declares that the `State` attributes with `ID`'s `"btd"`, `"cmp"`, `"mag"`, `"colorspace"`, and `"outputres"` should all be grouped together in any user interface. The English string `"ScanningParameters"` is associated with this `DisplayGroup`, though no explicit assumptions are made about how this group of attributes should be displayed. The `DisplayGroup` element merely states that there is a user-significant relationship between the attributes.

Name	Data Type	Description
<code>DisplayGroup *</code>	element	Declares a set of references to <code>State</code> elements that are intended to be displayed as a group in user interfaces.

#### 7.3.1.3.1 Structure of the DisplayGroup Subelement

Each `DisplayGroup` element declares a group of features that are intended to be displayed together in user interfaces.

Name	Data Type	Description
<code>rRefs</code>	IDREFS	References to <code>State</code> elements. (See "Structure of the Abstract State Subelement" on page 508 for details of the <code>State</code> element.)
<code>Loc *</code>	element	Localized strings describing the <code>DisplayGroup</code> .

#### 7.3.1.4 Structure of the FeaturePool Subelement

[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 "Structure of the MacroPool Subelement" on page 522.

These features may be mapped to `JDF/@NamedFeatures`. A feature from `JDF/@NamedFeatures` is selected by specifying an `NMTOKEN` pair that matches entries from `FeaturePool/EnumerationSpan/@Name` and `FeaturePool/EnumerationSpan/@AllowedValueList`

#### Example:

```
<DeviceCap>
  <FeaturePool>
    <EnumerationState AllowedValueList="Mono ColorTransparency Photo" ID="sm"
      MacroRefs="ScanModeMacro" Name="ScanMode" UserDisplay="Display"/>
  </FeaturePool>
</DeviceCap>
```

In this example, `ScanMode` is a feature that doesn't map directly to any JDF resource or attribute, but provides a "shell" feature that allows users to control a set of JDF resources and/or attributes to indicate a common or preferred grouping based on the user's desired task. The actual corresponding JDF resource attribute values are determined and set by the `ScanModeMacro` macro that is called when the `ScanMode` feature is manipulated.

Name	Data Type	Description
<code>State *</code>	element	Abstract <code>State</code> elements that define the accepted parameter space for the messages or resources subelements. These abstract subelements are identical in form to other <code>State</code> elements, but typically are only "macro" features that control other features through <code>macros</code> . For more information on <code>macro</code> definitions, see "Structure of the MacroPool Subelement" on page 522. For details of the <code>State</code> element, see "Structure of the Abstract State Subelement" on page 508.

### 7.3.1.5 Structure of the MacroPool Subelement

[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.

Name	Data Type	Description
macro *	element	A list of independent macros.

#### 7.3.1.5.1 Structure of the macro Subelement

[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 "Structure of the abstract Term Subelement" on page 524) and a set element. When the expression evaluates to "*true*", the action specified in the **set** element should be performed. If no evaluation in any **when** element in a **choice** evaluates to "*true*", the action(s) specified in the **otherwise** element should 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 should execute **choice**, **set**, and **call** elements in the order in which they are specified in the actual XML document. Note that this spec does not specify a required ordering, but the ordering provided in the actual capabilities description should be honored. The following shows the logical layout of the **macro** subelement:

Name	Data Type	Description
<i>ID</i>	ID	Unique identifier of a <b>macro</b> element. This <i>ID</i> is used to refer to the <b>macro</b> element.
choice *	element	A set of conditional operations that set (or not) feature values. At least one of <b>choice</b> , <b>set</b> , or <b>call</b> must be specified in <b>macro</b> .
set *	element	An element that sets one or more <b>State</b> attribute values. At least one of <b>choice</b> , <b>set</b> , or <b>call</b> must be specified in <b>macro</b> .
call *	element	An element that calls another <b>macro</b> , allowing for <b>macro</b> reuse and chaining. At least one of <b>choice</b> , <b>set</b> , or <b>call</b> must be specified in <b>macro</b> .

#### 7.3.1.5.1.1 Structure of the choice Subelement

The **choice** subelement is used to contain expressions that declare conditional operations that can cause **State** element attribute values to be changed. **Choice** includes one or more **when** statements that are evaluated in order, each of which contains a Boolean expression (as defined in "Structure of the abstract Term Subelement" on page 524) and a **set** element. When the expression evaluates to "*true*", the action specified in the **set** element should 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 should be performed.

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 <b>State</b> element attribute values if none of the <b>when</b> expressions evaluate to " <i>true</i> ".

### 7.3.1.5.1.1.1 Structure of the otherwise Subelement

The **otherwise** subelement sets one or more feature values if none of the **when** expressions in a choice element evaluate to *"true"*.

Name	Data Type	Description
set +	element	An element that sets one or more feature values.

### 7.3.1.5.1.1.2 Structure of the when Subelement

The **when** subelement is used to contain expressions that declare conditional operations to enforce sets of feature behaviors. **When** includes a Boolean expression (as defined in "Structure of the abstract Term Subelement" on page 524) and a **set** element. When the Term evaluates to *"true"*, the action specified in the set element should be performed.

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.

### 7.3.1.5.1.2 Structure of the set Subelement

The **set** subelement sets one or more **State** element attribute values

Name	Data Type	Description
<i>rRef</i>	IDREF	Reference to a <b>State</b> element referring to the feature value to set
FeatureAttribute ?	element	Specifies one or more attributes within the <b>State</b> element that should have their value changed (along with the value they change to).

### 7.3.1.5.1.2.1 Structure of the FeatureAttribute Subelement

**FeatureAttribute** specifies one or more attributes of a **State** element that should have their value changed. The following attributes may be changed:

Name	Data Type	Description
<i>CurrentValue ?</i>	string	The value to change the <i>CurrentValue</i> attribute of the <b>State</b> element to. Note that the mapping of the string to the actual data type of the <b>State</b> element must be performed by the application processing the capabilities.
<i>Editable ?</i>	boolean	When <i>"true"</i> , the feature and its current value may be edited by the user. If <i>"false"</i> , the user interface must not allow user modification of the current value of the <b>State</b> element.
<i>UserDisplay ?</i>	enumeration	Indicates under which conditions and whether the feature should be displayed in user interfaces. Possible values are the same as the <i>UserDisplay</i> attribute of the <b>State</b> element.

### 7.3.1.5.1.3 Structure of the call Subelement

The **call** subelement is used to call other macro elements, effectively using them as macro "templates."

Name	Data Type	Description
<i>rRef</i>	IDREF	Reference to a macro.

### 7.3.1.6 Structure of the Performance Subelement

[New in JDF 1.1](#)

The Performance element describes speed as the capability to consume or produce a JDF Resource.

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> ? <a href="#">New in JDF 1.2</a>	IDREF	Reference to the DevCaps 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> .
<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> ? <a href="#">Deprecated in JDF 1.2</a>	NMTOKEN	Name of the input resource type that is processed by the device, (e.g., <b>Media, Ink, RunList</b> ). Replaced with <i>DevCapsRef</i> in JDF 1.2 and beyond
<i>Unit</i> ?	NMTOKEN	Unit of measure of resource consumption per hour. Defaults to the resource's generic units as defined in Table 1-4, "Units used in JDF," on page 12.

### 7.3.1.7 Structure of the TestPool Subelement

[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.

Name	Data Type	Description
Test *	element	A list of independent Tests.

#### 7.3.1.7.1 Structure of the Test Subelement

The Test subelement is used to contain Boolean expressions that are for use only when referenced by another Test or Action and should not be evaluated independently. Its purpose is to simplify the description of other Tests and macros by representing a commonly used Boolean expression.

Name	Data Type	Description
ID	ID	Unique identifier of a Test element. This ID is used to refer to the Test element.
Term	element	Any element derived from an abstract Term, (e.g., "not", "and" or one of the explicit Evaluation elements).

#### 7.3.1.7.1.1 Structure of the abstract Term Subelement

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**. Terms are composed of Boolean combinations of three types of elements:

- Boolean expressions (i.e., nesting) comprising of “and”, “or”, “not” and “xor”. (See “Boolean Operators” on page 526.)
- Evaluation elements, which evaluate a JDF **State** attribute value to create a simple true or false Boolean expression, (e.g., “Is the value of BitDepth equal to 8?”). (See “Evaluation Subelements” on page 527.)
- TestRef — A reference to a constraint **Test** element. This referenced constraint is then used as a nested Boolean expression. (See “Structure of the TestRef Subelement” on page 532.)

Table 7.2: Term Elements

Name	Defined in Section	Description
and	See “Boolean Operators” on page 526.	Boolean AND operator.
not	See “Boolean Operators” on page 526.	Boolean negation.
or	See “Boolean Operators” on page 526.	Boolean OR operator.
xor	See “Boolean Operators” on page 526.	Boolean exclusive or (XOR) operator.
BooleanEvaluation	See “Evaluation Subelements” on page 527.	Describes operations on a set of Boolean values.
DateTimeEvaluation	See “Evaluation Subelements” on page 527.	Describes operations on a set of dateTime values.
DurationEvaluation	See “Evaluation Subelements” on page 527.	Describes operations on a set of duration values.
EnumerationEvaluation	See “Evaluation Subelements” on page 527.	Describes operations on a set of enumeration values.
IntegerEvaluation	See “Evaluation Subelements” on page 527.	Describes operations on a numerical range of integer values.
IsPresentEvaluation	See “Evaluation Subelements” on page 527.	Checks for the existence of a tag or feature.
MatrixEvaluation	See “Evaluation Subelements” on page 527.	Describes operations on a range of matrices. Generally used to define valid orientations of <b>Components</b> .
NameEvaluation	See “Evaluation Subelements” on page 527.	Describes operations on a set of NMTO-KEN values
NumberEvaluation	See “Evaluation Subelements” on page 527.	Describes operations on a numerical range of values.
PDFPathEvaluation	See “Evaluation Subelements” on page 527.	Describes operations on PDFPath.
RectangleEvaluation	See “Evaluation Subelements” on page 527.	Describes operations on a set of four-value rectangle values.
ShapeEvaluation	See “Evaluation Subelements” on page 527.	Describes operations on a set of three-value shape values.
StringEvaluation	See “Evaluation Subelements” on page 527.	Describes operations on a set of string values.
XYPairEvaluation	See “Evaluation Subelements” on page 527.	Describes operations on a set of XYPair values.
TestRef	See “Structure of the TestRef Subelement” on page 532.	Reference to a constraint <b>Test</b> to be evaluated as a nested Boolean expression inside a larger expression.

**Example:**

```

<DeviceCap>
  <TestPool>
    <Test ID="ctcmp">
      <!-- Can't CCITT compress anything but 1 bit grayscale -->
      <and>
        <not>
          <TestRef rRef="is1bit"/>
        </not>
        <EnumerationEvaluation ValueList="CCITTFaxEncode">
          <TestRef rRef="cmp"/>
        </EnumerationEvaluation>
      </and>
      <Loc HelpText="Only select CCITTFaxEncoding for 1 bit documents" Lang="en"
ShortValue="Ouch!" Value="CCITTFaxEncoding not supported on grayscale images"/>
    </Test>
    <Test ID="is1bit">
      <IntegerEvaluation ValueList="1">
        <TestRef rRef="btd"/>
      </IntegerEvaluation>
    </Test>
  </TestPool>
</DeviceCap>

```

**Note:** **Term** is an abstract element, so it will never appear in a JDF document. In the "ctcmp" constraint example, the **Term** is represented by the `<and>` element. Since the **Term** element itself is abstract, what will actually appear in constraints will be Boolean expressions. In this example, the logic is, "We can not use CCITT compression if the bit depth is not 1 bit." The check for compression type uses an **EnumerationEvaluation** element, which evaluates an **EnumerationState** value against "CCITTFaxEncode". If the value of the **EnumerationState** element referred to by "cmp" = CCITTFaxEncode, the **EnumerationEvaluation** evaluates as "true". The check for "btd" is accomplished through a **TestRef** to the "is1bit" constraint. The `<and>` and `<not>` elements behave according to the standard semantics for Boolean combinatorial logic.

**Note:** In the actual JDF schema, several abstract element definitions are used to create an appropriate inheritance structure. Rather than reproduce this here, only the actual non-abstract elements that will appear in JDF files will be described.

**7.3.1.7.1.2.1 Boolean Operators**

The **Boolean** operators that are defined in this section are instances of **Terms**, and, thus, they may be nested. They are used both in device capabilities and preflighting context.

**Structure of the "and" Subelement**

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.

Name	Data Type	Description
Term	element	Any element derived from an abstract <b>Term</b> .
Term +	element	Any element derived from an abstract <b>Term</b> .

**Structure of the "or" Subelement**

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.

Name	Data Type	Description
Term	element	Any element derived from an abstract <b>Term</b> .
Term +	element	Any element derived from an abstract <b>Term</b> .

### Structure of the “xor” Subelement

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.”

Name	Data Type	Description
Term	element	Any element derived from an abstract Term.
Term +	element	Any element derived from an abstract Term.

### Structure of the “not” Subelement

The not subelement inverts the Boolean state of a Term.

Name	Data Type	Description
Term	element	Any element derived from an abstract Term.

#### 7.3.1.7.1.2.2 Evaluation Subelements

Evaluation elements map generalized tests against a condition to form a true or false Boolean state that can be evaluated using the Boolean logic defined below.

When used in a device capabilities context, the Evaluation elements map to the State elements (i.e., BooleanState, IntegerState, etc.) which each declare individual JDF attributes for a device capabilities description. 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.

Name	Corresponding State Element	Description
BooleanEvaluation	BooleanState	Describes operations on a set of Boolean values.
DateTimeEvaluation	DateTimeState	Describes operations on a set of dateTime values.
DurationEvaluation	DurationState	Describes operations on a set of duration values.
EnumerationEvaluation	EnumerationState	Describes operations on a set of enumeration values.
IntegerEvaluation	IntegerState	Describes operations on a numerical range of integer values.
IsPresentEvaluation	all	Checks for the existence of a tag or feature.
MatrixEvaluation	MatrixState	Describes operations on a range of matrices. Generally used to define valid orientations of <b>Components</b> .
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.

### Structure of the Abstract Evaluation Element

The following table describes the common, data type-independent parameters of all **Evaluation** elements.

Name	Data Type	Description
<i>rRef</i> ?	IDREF	A reference to <b>State</b> elements when used in the context of device capability descriptions. Only one of <b>BasicPreflightTest</b> or <i>rRef</i> must be specified.
<b>BasicPreflightTest</b> ?	element	Definition of the preflight basic test to which the <b>Evaluation</b> refers. <b>BasicPreflightTest</b> is only valid when <b>Evaluation</b> elements are used in the context of preflighting. <b>Evaluations</b> in capability descriptions must reference the appropriate <b>State</b> element using <i>rRef</i> . For details of the <b>BasicPreflightTest</b> , see "PreflightParams" on page 434.

### Structure of the BooleanEvaluation Subelement

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.

Name	Data Type	Description
<i>ValueList</i> ?	enumerations	A list of all supported values. Allowed list values are the Boolean values "true" and "false".

### Structure of the DateTimeEvaluation Subelement

The **DateTimeEvaluation** element declares a Boolean value for comparison in an expression to a **DateTimeState** element in constraints.

Name	Data Type	Description
<i>ValueDurationList</i> ?	Duration-RangeList	List of inclusive minimum and maximum allowed values relative to the current system time.
<i>ValueList</i> ?	DateTime-RangeList	A list of all supported values.

### Structure of the DurationEvaluation Subelement

The **DurationEvaluation** element declares a Boolean value for comparison in an expression to a **DateTimeState** element in constraints. It inherits from the abstract **Evaluation** element described above.

Name	Data Type	Description
<i>ValueList</i> ?	Duration-RangeList	A list of all supported values.

### Structure of the EnumerationEvaluation Subelement

The **EnumerationEvaluation** element declares an enumeration value for comparison in an expression to an **EnumerationState** element in constraints.

Name	Data Type	Description
<i>ValueList</i> ?	enumerations	A list of all potential supported values. If not specified all enumerations defined by the XML schema are valid. In order to enable capabilities to be specified without access to the JDF XML schema, it is strongly recommended to specify <i>ValueList</i> , even when the entire range of schema-valid values is supported.

### Structure of the IntegerEvaluation Subelement

The IntegerEvaluation element declares an Integer value for comparison in an expression to a IntegerState element in constraints.

Name	Data Type	Description
<i>ValueList</i> ?	Integer-RangeList	A list of all supported values.
<i>ValueMod</i> ?	XYPair	X defines the Modulo and Y the offset of the allowed value. In other words, if <i>AllowedValueMod</i> = "10 2", only the values ... -8,2,12,22 ... are allowed. If not specified all values in the range are valid. If $((N\%X)-Y==0)$ then N is a valid value. <b>Note:</b> "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 $3 \text{ mod } 3 = 3 - 3 = 0$ .

### Structure of the IsPresentEvaluation Subelement

The IsPresentEvaluation element checks for the existence of a tag or feature. It inherits from the abstract Evaluation element described above and has no additional parameters.

### Structure of the MatrixEvaluation Subelement

The MatrixEvaluation element declares a matrix value for comparison in an expression to a MatrixState element in constraints.

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. <b>Note:</b> 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 2-3, "Matrices and Orientation values used to describe the orientation of a Component," on page 24 is subtracted from <i>Shift</i> , thus all in-place rotations have an implied Shift value of "0 0 0 0".
<i>Tolerance</i> = "0 0"	XYPair	The tolerance between the real and actual values that are defined as equal. Used to account for rounding errors and such. The first value is a positive value representing the negative tolerance, and the second value represents the positive tolerance. The tolerance applies to all of the matrix values.
<i>Transforms</i> ?	Orientations	Any of the eight predefined transforms for physical resources as defined in Table 2-3, "Matrices and Orientation values used to describe the orientation of a Component," on page 24.
<i>Value</i> *	element	A list supported values. The <i>Value/@Value</i> attribute must be a representation of a matrix.

### Structure of the Value element

Name	Data Type	Description
<i>Value</i>	matrix	A supported value for a matrix variable.

### Structure of the NameEvaluation Subelement

The NameEvaluation element declares a NMTOKEN value for comparison in an expression to a NameState element in constraints.

Name	Data Type	Description
<i>RegExp</i>	refExp	Regular expression that limits the allowed values.
<i>ValueList ?</i>	NMTOKENS	A list of supported values.

### Structure of the NumberEvaluation Subelement

The NumberEvaluation element declares a number value for comparison in an expression to a NumberState element in constraints.

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.
<i>ValueList ?</i>	Double-RangeList	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 = "10 2"</i> , 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. <b>Note:</b> "Modulo" is the remainder of an integer division. For example: $4 \bmod 3 = 4 - 3 = 1$ ; $17 \bmod 3 = 17 - 5 * 3 = 2$ ; and $3 \bmod 3 = 3 - 3 = 0$ .

### Structure of the PDFPathEvaluation Subelement

The PDFPathEvaluation element declares a PDF path value for comparison in an expression to a PDFPathState element in constraints.

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.

### Structure of the Value element

Name	Data Type	Description
<i>Value</i>	PDFPath	A supported value for a PDF path attribute.

### Structure of the RectangleEvaluation Subelement

The RectangleEvaluation element declares a Boolean value for comparison in an expression to a RectangleState element in constraints.

Name	Data Type	Description
<i>HWRelation ?</i>	XYRelation	Allowed relative value of width (X) vs. height (Y).
<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. The tolerance applies to both sides of the rectangle.
<i>ValueList ?</i>	RectangleRangeList	A list of ranges of allowed values that can be chosen.

### Structure of the ShapeEvaluation Subelement

The ShapeEvaluation element declares a shape value for comparison in an expression to a FeatureState element in constraints.

Name	Data Type	Description
<i>Tolerance</i> = "0 0"	XYPair	The tolerance between the real and actual values that are defined as equal. Used to account for rounding errors and such. The first value is a positive value representing the negative tolerance, and the second represents the positive tolerance. The tolerance applies to all values tested.
<i>ValueList</i> ?	ShapeRangeList	A list of ranges of values that can be chosen.
<i>X</i> ?	DoubleRangeList	Allowed X-Axis of the Shape.
<i>Y</i> ?	DoubleRangeList	Allowed Y-Axis of the Shape.
<i>Z</i> ?	DoubleRangeList	Allowed Z-Axis of the Shape.

### Structure of the StringEvaluation Subelement

The StringEvaluation element declares a string value for comparison in an expression to a StringState element in constraints.

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.

### Structure of the Value element

Name	Data Type	Description
<i>Value</i>	string	A supported value for a string attribute.

### Structure of the XYPairEvaluation Subelement

The XYPairEvaluation element declares a XYPair value for comparison in an expression to a XYPairState element in constraints.

Name	Data Type	Description
<i>Tolerance</i> = "0 0"	XYPair	The tolerance between the real and actual values that are defined as equal. Used to account for rounding errors and such. The first value is a positive value representing the negative tolerance, and the second represents the positive tolerance. These tolerance values apply to both the X and Y values of the evaluation being performed.
<i>ValueList</i> ?	XYPairRangeList	A list of values that can be chosen.
<i>XYRelation</i> ?	XYRelation	Relative value of X vs. Y.

•

## Structure of the TestRef Subelement

The TestRef element refers to another constraint that should be evaluated as part of the parent constraint.

Name	Data Type	Description
<i>rRef</i>	IDREF	Reference to a Test to be evaluated as a nested Boolean expression inside a larger expression.

## 7.3.2 Examples of Device Capabilities

[New in JDF 1.1](#)

[Modified in JDF 1.2](#)

All of the examples in this section are based on a simple definition of a scanner. The JMF based hand shaking is also illustrated. NodeInfo, **ExposedMedia**, and **ScanParams** are restricted.

### Device Description of a Scanner

This first example shows the general structure and provides an example of user interface localization (the query requests localization for the French language, and localizations are returned for the **ScanParams** resource).

#### Device Query:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_2" Version="1.2" TimeStamp="2004-04-05T16:45:43+02:00" SenderID="Controller">
  <Query ID="DeviceQuery" Type="KnownDevices">
    <DeviceFilter DeviceDetails="Capability" Localization="fre"/>
  </Query>
</JMF>
```

#### Device Response:

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_2" SenderID="Scanner" TimeStamp="2004-06-05T16:45:43+02:00" Version="1.2">
  <Response ID="xyz" Type="KnownDevices" refID="DeviceQuery">
    <DeviceList>
      <DeviceInfo>
        <Device Class="Implementation" DeviceID="Joe the Drum" ID="IDXYZ"
          KnownLocalizations="En Fre" ModelName="Bongo" Status="Available">
          <DeviceCap GenericAttributes="ID Class SettingsPolicy
BestEffortExceptionsOperatorInterventionExceptions 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="P1T0H1M" Name="ExposedMedia"/>
            <DevCaps Name="NodeInfo">
              <DevCap>
                <!-- NodeInfo only supports JobPriority and TargetRoute attributes -->
                <StringState Name="TargetRoute"/>
                <IntegerState Name="JobPriority"/>
              </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"/>
                </DevCap>
              </DevCap>
            </DevCaps>
            <DevCaps Name="ScanParams">
              <Loc HelpText="Les parametres pour commander le procede de balayage."
                Value="Les parametres de module de balayage"/>
            </DevCaps>
          </DeviceCap>
        </DeviceInfo>
      </DeviceList>
    </Response>
  </JMF>
```



```

<DevCap>
  <!-- Black and white 1 bit mode -->
  <IntegerState AllowedValueMax="1" AllowedValueMin="1" DefaultValue="8"
    Name="BitDepth"/>
  <EnumerationState AllowedValueList="CCITTFaxEncode None"
    Name="CompressionFilter">
    <Loc HelpText="Choisissez la compression pour reduire la taille de
donnees." Value="La compression de donnees"/>
    <ValueLoc Value="CCITTFaxEncode">
      <Loc Value="Compression de CCITT Fax"/>
    </ValueLoc>
    <ValueLoc Value="None">
      <Loc Value="Aucun compression"/>
    </ValueLoc>
  </EnumerationState>
  <NumberState AllowedValueMax="10" AllowedValueMin="1.e-002"
    Name="Magnification">
    <Loc ShortValue="Rapport optique" Value="Rapport de rapport optique
d'image"/>
  </NumberState>
  <EnumerationState AllowedValueList="GrayScale" Name="OutputColorSpace">
    <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">
    <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">
    <Loc ShortValue="Rapport optique" Value="Rapport de rapport optique
d'image"/>
  </NumberState>
  <EnumerationState AllowedValueList="GrayScale" Name="OutputColorSpace">
    <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 Locvalue="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>

```

## Device Description of a Scanner #2

This second example illustrates the use of constraints, macros, and DisplayGroups in a capability response. For the sake of simplicity, the only localizations returned are for the constraints.

### Device Query:

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Controller"
    TimeStamp="2004-04-05T16:45:43+02:00" Version="1.2">
    <Query ID="DeviceQuery" Type="KnownDevices">
        <DeviceFilter DeviceDetails="Capability" Localization="en"/>
    </Query>
</JMF>

```

**Device Response:**

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Scanner" TimeStamp="2004-10-17T09:30:47-05:00" Version="1.2">
  <Response ID="xyz" Type="KnownDevices" refID="DeviceQuery">
    <DeviceList>
      <DeviceInfo DeviceStatus="Idle">
        <Device Class="Implementation" DeviceID="Joe the Drum" ID="IDXYZ"
ModelName="Bongo" Status="Available">
          <DeviceCap GenericAttributes="ID Class SettingsPolicy BestEffortExceptions
OperatorInterventionExceptions MustHonorExceptions PartIDKeys DocIndex"
Type="Scanning">
            <!-- the scanner takes a minute to set up and scans
an average of 2 sheets a min. -->
            <Performance AverageAmount="120" Name="ExposedMedia"/>
            <FeaturePool>
              <EnumerationState AllowedValueList="Mono ColorTransparency Photo" ID="sm"
MacroRefs="ScanModeMacro" Name="ScanMode"/>
            </FeaturePool>
            <DisplayGroupPool>
              <DisplayGroup rRefs="btd cmp mag colorspace outputres">
                <Loc HelpText="Parameters for scanning configuration" Lang="en"
ShortValue="ScanningParameters"/>
              </DisplayGroup>
            </DisplayGroupPool>
            <ActionPool>
              <Action Severity="Error" TestRef="BD-bw" id="BD-bw-action">
                <Loc HelpText="For 1 bit grayscale, please select CCITTFaxEncoding"
Lang="en" ShortValue="Ouch!" Value="Flate and DCT Encoding not allowed on 1 bit
images"/>
              </Action>
              <Action Severity="Error" TestRef="ctcmp" id="ctcmp-action">
                <Loc HelpText="Only select CCITTFaxEncoding for 1 bit documents" Lang="en"
ShortValue="Ouch!" Value="CCITTFaxEncoding not supported on grayscale images"/>
              </Action>
              <Action Severity="Error" TestRef="cd" id="cd-action">
                <Loc HelpText="Choose a bit depth of 10 or less for color images"
Lang="en" ShortValue="Ouch!" Value="Bit depths higher than 10 are not supported for
color"/>
              </Action>
            </ActionPool>
            <TestPool>
              <Test ID="iscolor">
                <EnumerationEvaluation ValueList="RGB LAB CMYK" rRef="colorspace"/>
              </Test>
              <Test ID="islbit">
                <IntegerEvaluation ValueList="1" rRef="btd"/>
              </Test>
              <Test ID="BD-bw">
                <!-- Can't flate or DCT compress 1
bit grayscale -->
                <and>
                  <ConstraintRef rRef="islbit"/>
                  <EnumerationEvaluation ValueList="FlateEncode DCTEncode" rRef="cmp"/>
                </and>
              </Test>
              <Test ID="ctcmp">
                <!-- Can't CCITT compress anything
but 1 bit grayscale -->
                <and>
                  <not>
                    <ConstraintRef rRef="islbit"/>
                  </not>
                  <EnumerationEvaluation ValueList="CCITTFaxEncode" rRef="cmp"/>
                </and>
            </TestPool>
          </DeviceCap>
        </DeviceInfo>
      </DeviceList>
    </Response>
  </JMF>

```

```

        </Test>
        <Test ID="cd">
            <!-- Can't have a color depth greater
than 10 bits -->
            <and>
                <ConstraintRef rRef="iscolor"/>
                <IntegerEvaluation ValueList="1 10" rRef="btd"/>
            </and>
        </Test>
    </TestPool>
    <MacroPool>
        <macro id="ScanModeMacro">
            <choice>
                <when>
                    <EnumerationEvaluation ValueList="Mono" rRef="sm"/>
                    <set rRef="btd">
                        <FeatureAttribute CurrentValue="1"/>
                    </set>
                    <set rRef="colorspace">
                        <FeatureAttribute CurrentValue="GrayScale"/>
                    </set>
                    <set rRef="outputres">
                        <FeatureAttribute CurrentValue="1200 1200"/>
                    </set>
                </when>
                <when>
                    <EnumerationEvaluation ValueList="ColorTransparency" rRef="sm"/>
                    <set rRef="btd">
                        <FeatureAttribute CurrentValue="8"/>
                    </set>
                    <set rRef="colorspace">
                        <FeatureAttribute CurrentValue="RGB"/>
                    </set>
                    <set rRef="outputres">
                        <FeatureAttribute CurrentValue="600 600"/>
                    </set>
                </when>
                <when>
                    <EnumerationEvaluation ValueList="Photo" rRef="sm"/>
                    <set rRef="btd">
                        <FeatureAttribute CurrentValue="10"/>
                    </set>
                    <set rRef="colorspace">
                        <FeatureAttribute CurrentValue="LAB"/>
                    </set>
                    <set rRef="outputres">
                        <FeatureAttribute CurrentValue="200 200"/>
                    </set>
                </when>
            </choice>
        </macro>
    </MacroPool>
    <DevCaps Name="NodeInfo">
        <DevCap>
            <!-- NodeInfo only supports the
JobPriority and TargetRoute attributes -->
            <StringState Name="TargetRoute"/>
            <IntegerState Name="JobPriority"/>
        </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"/>
        </DevCap>
        </DevCap>
        </DevCaps>
        <DevCaps Name="ScanParams">
        <DevCap>
        <!-- all modes -->
        <IntegerState ActionRefs="BD-bw ctcmp cd" AllowedValueList="1 4 8 10 12"
DefaultValue="1" ID="btd" Name="BitDepth" UserDisplay="Hide"/>
        <EnumerationState ActionRefs="BD-bw ctcmp"
AllowedValueList="CCITTFaxEncode FlateEncode DCTEncode None" ID="cmp"
Name="CompressionFilter" UserDisplay="Hide"/>
        <NumberState AllowedValueMax="100" AllowedValueMin="1.e-002" ID="mag"
Name="Magnification"/>
        <EnumerationState ActionRefs="cd" AllowedValueList="GrayScale CMYK RGB
LAB" ID="colorspace" Name="OutputColorSpace"/>
        <XYPairState AllowedValueList="100 100 300 300 600 600 1200 1200 2400
2400" DefaultValue="600 600" ID="outputres" Name="OutputResolution"/>
        </DevCap>
        </DevCaps>
        </DeviceCap>
        </Device>
        </DeviceInfo>
        </DeviceList>
    </Response>
</JMF>

```

### JDF Node that is accepted by the scanner of the previous example

All parameters of the following Scanning node are compliant with the capabilities.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="GoodScan" Status="Waiting"
Type="Scanning" Version="1.2">
  <ResourcePool>
    <ScanParams BitDepth="8" Class="Parameter" ID="Link0007" OutputColorSpace="RGB"
OutputResolution="600. 600." Status="Available"/>
    <ExposedMedia Class="Handling" ID="Link0008" Status="Available">
      <Media Dimension="425.196850394 566.929133858"/>
    </ExposedMedia>
  </ResourcePool>
  <ResourceLinkPool>
    <ScanParamsLink Usage="Input" rRef="Link0007"/>
    <ExposedMediaLink Usage="Input" rRef="Link0008"/>
  </ResourceLinkPool>
</JDF>

```

### JDF node that is rejected by the scanner of the previous example

All parameters of the following Scanning node except **Magnification** are compliant with the device capabilities. Therefore, the device can not execute the job.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="BadScan" Status="Waiting"
Type="Scanning" Version="1.2">
  <ResourcePool>
    <ScanParams BitDepth="8" Class="Parameter" ID="Link0012" Magnification="1000.
1000." OutputColorSpace="RGB" OutputResolution="600. 600." Status="Available"/>
    <ExposedMedia Class="Handling" ID="Link0013" Status="Available">
      <Media Dimension="425.196850394 566.929133858"/>
    </ExposedMedia>
  </ResourcePool>
  <ResourceLinkPool>
    <ScanParamsLink Usage="Input" rRef="Link0012"/>
    <ExposedMediaLink Usage="Input" rRef="Link0013"/>
  </ResourceLinkPool>
</JDF>

```

## 7.4 Concept of the Preflight Process

[New in JDF 1.2](#)

**Note:** This section establishes elements, attributes, and attribute values that are used by the resources referenced by the *Preflight* process, including **PreflightParams**, **PreflightReportRulePool**, and **PreflightReport**, as well as extensions of testing methodology established **Action** and **Test** functions defined in "Structure of the DeviceCap Subelement" on page 502.

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 502. Each basic test is applied to one defined property with a given data type. Note that document properties defined in this section include one or more attributes that are extracted from documents (e.g., a client's PDF file) and used by one or more evaluations as part of a preflight test. Each data type can be tested on an object using its matching **Evaluation**. A document that is preflighted is made of objects. Some of them, like virtual boxes ("TrimBox" or "MediaBox") are not visible. In order to combine basic tests together, they have been classified by groups of properties. These groups do not necessarily match a class of an object. However, each class of object will implement one or more groups of properties.

The rules to combine basic tests into a **Test** can be built on both object classes and groups of properties. Each basic test takes an object as an input and has four different states in output: *false*, *true*, *TestWrongPDL*, or *TestNotSupported*. The two last values occur when a basic test has no meaning for the given object or when the application that is executing the test does not support that test. These four different states lead to a more open way of dealing with Boolean logic:

false	AND	TestWrongPDL	=	false
true	OR	TestWrongPDL	=	true
false	AND	TestNotSupported	=	false
true	OR	TestNotSupported	=	true
true	AND	TestWrongPDL	=	TestWrongPDL
false	OR	TestWrongPDL	=	TestWrongPDL
true	AND	TestNotSupported	=	TestNotSupported
false	OR	TestNotSupported	=	TestNotSupported
TestWrongPDL	OR	TestNotSupported	=	TestNotSupported
TestWrongPDL	AND	TestNotSupported	=	TestNotSupported
if (true)		Report according to action.		
if (false)		Do not report.		
if (TestWrongPDL)		Report problem if specified in PRRule.		
if (TestNotSupported)		Report problem if specified in PRRule.		

For instance, *TestWrongPDL* would occur when a test about font size is made on a page. *TestNotSupported* would happen when a JDF preflight agent does not support the concept of font size.

## 7.4.1 Object Classes

The following is the list of the real objects that can be preflighted in a document:

Table 7-8: Document Object Classes

Name	Description
Annotation	An annotation is a complex object that adds information to the page of a document. The characteristic of such object is that it is optional to print it. When an annotation is set to be printed, the graphical objects making the annotation are considered separated objects.
Document	The document, which is preflighted.
Font	A font is a set of characters that can be used to draw text. A font can be in a document without being used by any text of the document.
Image	An image is a graphic object drawn with colored pixels.
MaskUsingImage	This object is an object that masks another object using an image.
MaskUsingVector	This object is an object that masks another object using a vector path.
MaskUsingText	This object is an object that masks another object using text components.
Mask	A mask is an object used to mask or clip a graphic object.
Page	A document can be made of finished pages (but could be empty as well).
PageBox	In each finished page, some virtual boxes can be defined (page size and margins). Some tests can be done with these boxes.
PDL	A PDL object is a generic kind of object that can be specific to some types of documents. It is just a way to detect presence or not of such objects.
Shading	A shading is a graphic object drawn using a smooth color change from one point to another.
Text	A text is a set of characters that have exactly the same style, (i.e., same size, same font, same fill and stroke, etc.).
Vector	A vector is a graphic object drawn with vector curves. It is made of a fill and a stroke.

In the following table, you can see the list of object classes with the properties set that they implement.

Table 7-9: Properties Implemented by Class

Properties	Classes													
	Document	Page	Image	Vector	Text	Shading	ImageMask	Annotation	PageBox	Font	MaskUsingImage	MaskUsingVector	MaskUsingText	PDL
Logical	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Class	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Document	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Page		X	X	X	X	X	X	X	X	X	X	X	X	X
Reference			X	X										
Colorant	X		X	X	X	X	X							
Box		X	X	X	X	X	X	X	X		X	X	X	X
Graphic			X	X	X	X	X							
Fill				X	X		X							
Stroke				X	X									

Table 7-9: Properties Implemented by Class

Properties	Classes													
	Document	Page	Image	Vector	Text	Shading	ImageMask	Annotation	PageBox	Font	MaskUsingImage	MaskUsingVector	MaskUsingText	PDL
Image			X				X				X			
Vector				X								X		
Text					X								X	
Shading						X								
Font					X				X					
Annotation								X						
Page Box									X					
PDL Object														X

#### 7.4.1.1 Checking for the Presence of a Property

In most of the preflight process, only the “values” of properties are needed. (Please note that a property may incorporate one or more attributes, and it is the values (string, enumeration, etc.) of these attributes that are collectively referred to here as the “value” of the property.) In some cases, it is also useful to be able to check if a property has been defined. This happens in some types of documents where the property definition is optional. Before checking its value, you just want to check that this property was defined.

For all the basic tests described in this document where it makes sense to check if they are defined, they are checked “Yes” in the **Tag** column of properties definition tables below. Use the `IsPresentEvaluation` to check for the presence of a property.

This example checks if the *TrappedKey* is defined in a PDF document.

```
<Test ID="PT01">
  <IsPresentEvaluation>
    <BasicPreflightTest Name="TrappedKey"/>
  </IsPresentEvaluation>
</Test>
```

This example checks if the value of the *TrappedKey* = “Unknown” in a PDF document.

```
<Test ID="PT02">
  <EnumerationEvaluation ValueList="Unknown">
    <BasicPreflightTest Name="TrappedKey"/>
  </EnumerationEvaluation>
</Test>
```

Table 7-10: Mapping between property types (in the preflight spec) and Evaluations

Property Type	Evaluation	Expected usage for BasicPreflightTest ListType
presence	IsPresentEvaluation	-
boolean	BooleanEvaluation	SingleValue.
BooleanList	BooleanEvaluation	Any of <i>ListType</i> 's value that refers to a list.
DateTime	DateTimeEvaluation	SingleValue.
DateTimeList	DateTimeEvaluation	Any of <i>ListType</i> 's value that refers to a list.
enumeration	NameEvaluation	SingleValue.



Table 7-10: Mapping between property types (in the preflight spec) and Evaluations

Property Type	Evaluation	Expected usage for BasicPreflightTest ListType
enumerations	NameEvaluation	Any of <i>ListType</i> 's value that refers to a list.
integer	IntegerEvaluation	SingleValue.
integerList	IntegerEvaluation	Any of <i>ListType</i> 's value that refers to a list.
Name	NameEvaluation	SingleValue.
NameList	NameEvaluation	Any of <i>ListType</i> 's value that refers to a list.
Double	NumberEvaluation	SingleValue.
DoubleList	NumberEvaluation	Any of <i>ListType</i> 's value that refers to a list.
Rectangle	RectangleEvaluation	SingleValue.
RectangleList	RectangleEvaluation	Any of <i>ListType</i> 's value that refers to a list.
string	StringEvaluation	SingleValue.
stringList	StringEvaluation	Any of <i>ListType</i> 's value that refers to a list.
XYPair	XYPairEvaluation	SingleValue.
XYPairList	XYPairEvaluation	Any of <i>ListType</i> 's value that refers to a list.

#### 7.4.1.2 Basic tests on set of objects

Some properties can be applied to more than one object and have a value when applied to a list of objects which differs from their value when applied to a single object. For instance, this allows you to make tests on the number of separations of objects included in a given area. These properties have the column **Set** checked with “*Yes*.” In order to define a **Test** using such properties, a list of objects is filtered first, before applying the test. This is achieved using the **PreflightArgument** element.

#### 7.4.2 Properties

In the following pages, a full list of properties (and attribute definitions) is defined, that can be found, extracted, and evaluated from a document. The properties are grouped by classes (see Table 7-8, “Document Object Classes,” on page 539.)

### 7.4.2.1 Annotation Properties

Annotation objects are specific objects that can be optionally 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.

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: <i>Circle</i> <i>FileAttachment</i> <i>FreeText</i> <i>Highlight</i> <i>Ink</i> <i>Link</i> <i>Line</i> <i>Movie</i> <i>Popup</i> <i>PrinterMark</i> <i>Sound</i> <i>Square</i> <i>Squiggly</i> <i>Stamp</i> <i>StrikeOut</i> <i>Text</i> <i>TrapNet</i> <i>Underline</i> <i>Widget</i>	—	—	PDF
<i>TrapnetAnnotationPDFX</i>	NMTO- KENS	The PDF/X versions to which the <i>TrapNet</i> annotation complies, (e.g., "PDF/X-1a:2003").	—	—	PDF

### 7.4.2.2 Box Properties

All visible objects can be described at least by a box in which they can be contained. In a page, some kind of boxes can define some basic box properties that are extracted as attributes for use in a test. The allowed types of boxes are listed in the following table. Note that *BOX* is an attribute of the *BoxArgument* and the following box types are possible values of the *BOX* attribute.

Table 7-11: Allowed Box Types

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 should be clipped when output in a production environment. This may include any extra "bleed area" needed to accommodate the physical limitations of cutting, folding, and trimming equipment. The actual printed page may 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.

Table 7-11: Allowed Box Types

Box Type	Description
<i>MediaBox</i>	Defines the boundaries of the physical medium on which the page is to be printed. It may include any extended area surrounding the finished page for bleed, printing marks, or other such purposes. It may also include areas close to the edges of the medium that cannot be marked because of physical limitations of the output device. Content falling outside this boundary can safely be discarded without affecting the meaning of the file.
<i>SlugBox</i>	Defines an area where document related information and objects that will not be on the final document could be printed.
<i>TrimBox</i>	Defines the intended dimensions of the finished page after trimming. It may be smaller than the media box, to allow for production-related content such as printing instructions, cut marks, or color bars. In another type of document than PDF, this box represents the page size.

The following are other property attributes related to boxes:

Name	Type	Description	Set	Tag	Documents
<i>BoundingBox</i>	Rectangle	The bounding box of the object is the smallest rectangle containing the object. When used with group of objects, this is the smallest box containing boxes of all objects.	Yes	—	—
<i>DifferentBoxSize</i>	enumerations	This is the list of boxes, which are different on one page from the same boxes on another page. Refer to Table 7-11 on page 542 for a list of valid types of boxes.	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.	—	—	—

### Example:

The following is an example of *Test* using *InsideBox* and a *BoxArgument* subelement:

```
<Test ID="PT01">
  <BooleanEvaluation ValueList="true">
    <BasicPreflightTest Name="InsideBox">
      <PreflightArgument>
        <BoxArgument Box="TrimBox" Overlap="true"/>
      </PreflightArgument>
    </BasicPreflightTest>
  </BooleanEvaluation>
</Test>
```

### 7.4.2.3 Class Properties

Each object can define the name of the class of objects it belongs to:

Name	Type	Description	Set	Tag	Documents
<i>ClassName</i>	NMTOKEN	The name of the class to which the object belongs. Values include: <i>Annotation</i> <i>Document</i> <i>Font</i> <i>Image</i> <i>ImageMask</i> <i>MaskUsingImage</i> <i>MaskUsingText</i> <i>MaskUsingVector</i> <i>Page</i> <i>PageBox</i> <i>PDL</i> <i>Shading</i> <i>Text</i> <i>Vector</i>	—	—	—
<i>PropertyList</i>	enumerations	The list of properties the object has. Values are: <i>Annotation</i> <i>Box</i> <i>Class</i> <i>Colorant</i> <i>Document</i> <i>Fill</i> <i>Font</i> <i>Graphic</i> <i>Image</i> <i>Logical</i> <i>Page</i> <i>PageBox</i> <i>PDLObject</i> <i>Reference</i> <i>Shading</i> <i>Stroke</i> <i>Text</i> <i>Vector</i>	—	—	—

### 7.4.2.4 Colorant Properties

Every visible object or group of objects will imply a given number of separations.

Name	Type	Description	Set	Tag	Documents
<i>AliasSeparations</i>	boolean	Is " <i>true</i> " when some of the separations have different names but the same color values.	Yes	—	—
<i>AmbiguousSeparations</i>	boolean	Is " <i>true</i> " when some of the separations have the same name but different color values.	Yes	—	—
<i>InkCoverage</i>	double	This is the maximum percentage of ink coverage for one object. In case of a group of objects, this is the maximum amount of ink coverage for the list of objects. The method of calculation can be application-dependant and may 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.

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, whose value can be either: <i>Left</i> <i>Right</i>	—	Yes	PDF
<i>CreationDate</i>	dateTime	The date when the document was created according to the file system.	—	—	—
<i>CreationDateInDocument</i>	dateTime	The date when the document was created according to data inside the document.	—	Yes	—
<i>CreationID</i>	NMTOKEN	An NMTOKEN which can uniquely identify a document when created. In case of a PDF, it matches exactly the first element of ID array.	—	Yes	—
<i>Creator</i>	string	A string describing the creator of the document. This is usually the name and version of the authoring application used. In case of PS and PDF files, it matches exactly the Creator key.	—	Yes	—

Name	Type	Description	Set	Tag	Documents
<i>DocumentCompression</i>	enumerations	A list of all compression types used in the document (including image compression referenced by <i>CompressionTypes</i> in Image properties). See <i>CompressionTypes</i> for possible values.	—	—	—
<i>DocumentCorruption</i>	NMTO-KENS	The list of recoverable errors against the document format that were found in this document. An empty list means the document is not corrupted. Possible values include: <i>InvalidOffsets</i> – Some offsets are invalid, but the preflight agent was able to load the document nonetheless. Note that the absence of this value does not mean that all document structures are valid, only that the offsets are correct)	—	—	—
<i>DocumentEncoding</i>	enumeration	The document encoding which can be either: <i>ASCII</i> <i>Binary</i>	—	—	PS, PDF
<i>DocumentIsGoodCompression</i>	boolean	Is “ <i>true</i> ” when a strong compression algorithm is used (not just an ASCII filter) for all objects in the document where it makes sense to have compression.	—	—	—
<i>EncryptedDocument</i>	boolean	Is “ <i>true</i> ” if document is encrypted.	—	—	—
<i>EncryptionFilter</i>	NMTOKEN	The Filter name of encryption for a PDF file.	—	Yes	PDF
<i>EncryptionLength</i>	integer	The length of the encryption key of a PDF file in bits.	—	Yes	PDF
<i>EncryptionRestrictions</i>	NMTO-KENS	The actions that are forbidden by the encryption. Possible values are: <i>Assembly</i> – Inserting or removing pages. <i>Copying</i> – Extracting part of the content. <i>DisabledAccess</i> – Allowing copying specifically for providing access to the disabled. <i>EditingAnnotations</i> <i>EditingContent</i> <i>FillingIn</i> – Filling in forms. <i>HighResPrinting</i> <i>Printing</i>	—	—	PDF
<i>EncryptionSubFilter</i>	NMTOKEN	The SubFilter name of encryption for a PDF file.	—	Yes	PDF

Name	Type	Description	Set	Tag	Documents
<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 <i>"true"</i> 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.	—	—	—
<i>OutputIntentColorSpace = "None"</i>	NMTOKEN	The color space belonging to the output intent of the document. Possible values include: <i>None</i> – The default value to be used if this property is not present. <i>CMYK</i> <i>Gray</i> <i>RGB</i>	—	Yes	PDF
<i>OutputIntentStandard</i>	string	The standards the output intent is compliant with, (e.g., PDF/X-1a:2001). The version of the standard is assumed to be in the string accordingly to the standard's notation.	—	—	—
<i>PagesHaveSameOrientation</i>	boolean	Is <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 using MIME-type. <i>PDFType</i> is <i>"application/pdf"</i> for instance. See Table P-1 on page 635 and Table P-2 on page 638 for examples.	—	—	—
<i>PDLVersion</i>	string	The version of document according to the <i>PDLType</i> . See Table P-1 on page 635 and Table P-2 on page 638 for examples.	—	—	—
<i>Producer</i>	string	A string describing the producer of the document. This is usually the name of the software used to create file. In case of PDF files, it matches exactly the Producer key.	—	Yes	—

Name	Type	Description	Set	Tag	Documents
<i>SeparationFlag</i>	boolean	Is "true" if the document is made of separations or is not composite.	—	—	PS, PDF
<i>Subject</i>	string	A string describing the subject of the document.	—	Yes	—
<i>Title</i>	string	A string describing the title of the document.	—	Yes	—
<i>TrappedKey</i>	enumeration	A value explaining the use of trapping on the document. The values can be "true", "false", or "Unknown". It matches exactly the <i>TrappedKey</i> information of PDF.	—	Yes	—

### 7.4.2.6 Fill Properties

Fill property values are derived from graphic objects with vector primitives. They can have a fill color and a stroke color, with given colors. This is a list of properties that specifically apply to this kind of object:

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. Possible values are: <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.	—	—	—
<i>HasFillColor</i>	boolean	Is "true" if the vector object is drawn with a fill color.	—	—	—



### 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:

Name	Type	Description	Set	Tag	Documents
<i>EmbeddingRestrictionFlag</i>	boolean	Is <i>"true"</i> if a font cannot be embedded.	—	—	—
<i>FontCorrupted</i>	boolean	Is <i>"true"</i> if a font is corrupted or invalid. The implementation of this check may vary from one application to another.	—	—	—
<i>FontCreator</i>	string	The font creator.	—	—	—
<i>FontEmbedded</i>	boolean	Is <i>"true"</i> if a font is embedded into the document.	—	—	—
<i>FontIsStandardLatin</i>	boolean	Is <i>"true"</i> when all characters belong to the standard Latin character set.	—	—	—
<i>FontName</i>	string	The font name.	—	—	—
<i>FontNotUsed</i>	boolean	Is <i>"true"</i> if a font is not used to draw characters from the document.	—	—	—
<i>FontSubset</i>	boolean	Is <i>"true"</i> if a font is only a subset of a main font.	—	—	PS, PDF
<i>FontType = "Other"</i>	enumeration	This is the type of the font. Possible values referencing standardized types of fonts include: <i>CIDFontType0</i> <i>CIDFontType1</i> <i>CIDFontType2</i> <i>CIDFontType3</i> <i>CIDFontType4</i> <i>OpenType</i> <i>TrueType</i> <i>Type0</i> – PostScript Type0 without the CID <i>Type1</i> <i>Type1CMultipleMaster</i> <i>Type2C</i> <i>Type3</i> <i>PDFType3</i> <i>Type42</i> – Embedded TrueType into a PostScript font. <i>Unknown</i> – Type of font that can not be resolved for any reason, (i.e., missing font, etc.). <i>Other</i> – To be used when the property is not any of the values listed above.	—	—	—
<i>FontVendor</i>	string	The font vendor.	—	—	—
<i>IsFontScreenOnly</i>	boolean	Is <i>"true"</i> if a font referenced in the document contains only screen description.	—	—	Authoring
<i>PSFontName</i>	NMTOKEN	The PostScript font name.	—	—	PS, PDF

### 7.4.2.8 Graphic Properties

This is a list of property attributes that specifically apply to objects that can be displayed or printed.

Name	Type	Description	Set	Tag	Documents
<i>AlphaIsShape</i>	boolean	The <i>AlphaIsShape</i> of a PS or PDF object.	—	—	PS, PDF
<i>AlternateColorSpace</i>	enumeration	The alternate color space of the object is one of the given. Values are identical to those defined in <i>ColorSpace</i> , below.	—	Yes	PS, PDF
<i>BelongsToAnnotation</i>	boolean	Is "true" when this object belongs to an annotation.	—	—	—
<i>BlackGeneration</i>	enumeration	The <i>BlackGeneration</i> function of a PS or PDF object. Values may include either: <i>Identity</i> —Defines identity function. <i>Custom</i> —Used when the function is described.	—	Yes	PS, PDF
<i>BlendMode</i>	NMTOKEN	The <i>BlendMode</i> of a PS or PDF object.	—	—	PS, PDF
<i>ColorSpace</i>	enumeration	The color space of the object is one of the following values: <i>CalGray</i> <i>CalRGB</i> <i>CIEBasedA</i> <i>CIEBasedABC</i> <i>CIEBasedDEFG</i> <i>DeviceCMYK</i> <i>DeviceGray</i> <i>DeviceN</i> <i>DeviceRGB</i> <i>ICCBased</i> <i>Lab</i> <i>Separation</i>	—	—	PS, PDF
<i>EmbeddedPS</i>	boolean	Is "true" if a PDF object uses PostScript to be drawn.	—	—	PDF
<i>Flatness</i>	double	A number giving the value of PS or PDF <i>Flatness</i> .	—	Yes	PS, PDF
<i>HasSoftMask</i>	boolean	Is "true" when the object is using a soft-mask using pixel values.	—	—	—
<i>Halftone</i>	NMTOKEN	The value of the Halftone used in a document: "Named", "1", "5", "6", "10", "16".	—	Yes	PS, PDF
<i>HalftonePhase</i>	XYPair	The value of the <i>HalftonePhase</i> associated with the object.	—	Yes	PS, PDF
<i>HasColorLUT</i>	boolean	Is "true" when an object is using indexed colors in a table to describe color.	—	—	—
<i>NumberOfColorsInLUT</i>	integer	The number of colors in the color table used to display an indexed image.	—	—	—

Name	Type	Description	Set	Tag	Documents
<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 may include either: <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 should have the " <i>false</i> " value.	—	—	—
<i>UnderColorRemoval</i>	enumeration	The <i>UnderColorRemoval</i> function of a PS or PDF object. Values may include either: <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:

Name	Type	Description	Set	Tag	Documents
<i>AlternateImages</i>	NMTO-KENS	When to draw some of the alternate images that correspond with the given image. The PDF specification defines "Print" as a possible value, but any other application-specific value could be used.	—	Yes	PDF
<i>BitsPerSample</i>	integer	The number of bits used to represent color on every separation.	—	—	—
<i>CompressionRatio</i>	double	For all compression types to which it makes sense, the tests apply to the quality expressed as percentage of compression.	—	—	—

Name	Type	Description	Set	Tag	Documents
<i>CompressionTypes</i>	enumerations	The type of method used to compress or encode the image. Values may include: <i>ASCII85</i> <i>ASCIIHex</i> <i>CCITT</i> <i>JBIG2</i> <i>JPEG</i> <i>JPEG2000</i> <i>LZW</i> <i>None</i> <i>RunLength</i> <i>ZIP</i> Where <i>JPEG</i> , <i>JPEG2000</i> , and/or <i>JBIG2</i> are used, they may be concatenated and only <i>JPEG</i> 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. Possible values include: <i>None</i> <i>Horizontal</i> <i>Vertical</i>	—	—	—
<i>ImageMaskType</i>	enumeration	The type of masks used by image. The allowed values include: <i>NoMask</i> – Used when the image does not use specific mask. <i>BitmapMask</i> – Used when the image is masked using a bitmap image <i>ColorKeyMask</i> – Used when some colors are masked out to display the image (such like video chroma-key).	—	—	—
<i>ImageRotation</i>	integer	The number of degrees an image is rotated. A positive number represents a counterclockwise rotation. A negative number represents a clockwise rotation. <b>Note:</b> 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.	—	—	—

Name	Type	Description	Set	Tag	Documents
<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.

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.

Name	Type	Description	Set	Tag	Documents
<i>PageBoxType</i>	enumeration	Refer to "Box Properties" on page 542 for a list of the valid types of boxes. When not known, the default is to leave <i>PageBoxType</i> empty.	—	—	—

### 7.4.2.12 Pages Properties

This is the list of elements and attributes related to the page object in a document.

Name	Type	Description	Set	Tag	Documents
<i>BlankPage</i>	boolean	Is <i>true</i> when the trim box and the bleed box area, when defined, do not output any marks.	—	—	—
<i>BlendColorSpace</i>	enumeration	The page blend color space. For the list of possible values, see <i>ColorSpace</i> in Graphics Properties above.	—	Yes	PDF
<i>PageHasUnknownObjects</i>	boolean	Page contains unknown objects but the PDL was set to ignore these errors. Examples are the use of BX/EX in PDF.	—	—	—
<i>PageNumber</i>	integer	The page index in the <b>RunList</b> .	—	—	—
<i>ReversePageNumber</i>	integer	A special page numbering which starts from the last page. The last page is <i>-1</i> . This has been added to allow filtering of last page or the before last page, which is <i>-2</i> . 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>From</i> (top left bottom right) and <i>To</i> (top left bottom right). The calculation is made using the following formula: <i>To</i> (top)– <i>From</i> (top), <i>From</i> (left)– <i>To</i> (left), <i>From</i> (bottom)– <i>To</i> (bottom), <i>To</i> (right)– <i>From</i> (right). To define the two boxes used, options are given in <i>BoxToBoxDifference</i> argument.	—	—	—

#### Structure of the *BoxToBoxDifference* Subelement

Name	Type	Description
<i>From</i>	enumeration	The “from” box used in calculating the <i>BoxToBoxDifference</i> argument. See table “Allowed Box Types” on page 542 for a list of valid box types.
<i>To</i>	enumeration	The “to” box used in calculating the <i>BoxToBoxDifference</i> argument. See table “Allowed Box Types” on page 542 for a list of valid box types.

Note that *BoxToBoxDifference* element is always a subelement of a *PreflightArgument*.

#### Example:

```
<Test ID="PT01">
  <RectangleEvaluation ValueList="0 0 10 10">
    <BasicPreflightTest Name="BoxToBoxDifference">
      <PreflightArgument>
        <BoxToBoxDifference FromBox="TrimBox" ToBox="BleedBox"/>
      </PreflightArgument>
    </BasicPreflightTest>
  </RectangleEvaluation>
</Test>
```

### 7.4.2.13 PDLObject Properties

The PDL object is used to check whether select objects are defined or not defined in the document, but does not check anything else as these objects are specific to one given PDL

Name	Type	Description	Set	Tag	Documents
<i>PDLObjectType</i>	NMTOKEN	The type of specific PDL object. Possible values include: <i>AcroForm</i> – The PDF AcroForm. <i>Actions</i> – The PDF Actions. <i>Bookmarks</i> – The PDF Bookmarks. <i>JavaScript</i> – The PDF JavaScript. <i>Thread</i> – The PDF Thread. <i>Thumbnails</i> – The PDF Thumbnails.	—	—	PDF

### 7.4.2.14 Reference Properties

Reference property attributes describe objects that have links to external references on other objects. It only deals with OPI links and references in page to other graphical contents. This is not describing the font properties (see "Font Properties" on page 549).

Name	Type	Description	Set	Tag	Documents
<i>ExternalReferenceMissing</i>	boolean	Is " <i>true</i> " when the target of an external reference is missing.	—	—	—
<i>HasExternalReference</i>	boolean	Is " <i>true</i> " when some of the page graphical contents have a link on files.	—	—	—
<i>HasOPI</i>	boolean	Is " <i>true</i> " if there is OPI information associated with the object.	—	—	PS, PDF
<i>OPIMissing</i>	boolean	Is " <i>true</i> " when the target of OPI comments associated with the object is missing.	—	—	PS, PDF
<i>OPIType</i>	NMTOKEN	The OPI type of OPI comments associated with the object. Sometimes in PS, the comments are not OPI comments. Other values include: <i>OPIComments</i> <i>OtherComments</i>	—	—	PS, PDF
<i>OPIVersion</i>	NMTO-KENS	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.

Name	Type	Description	Set	Tag	Documents
<i>ShadingType</i>	enumeration	The type of shading. Value may be one of: <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:

Name	Type	Description	Set	Tag	Documents
<i>HasStrokeColor</i>	boolean	Is <i>"true"</i> if the vector object is drawn with a stroke color.	—	—	—
<i>StrokeAlternateColorSpace</i>	enumeration	The alternate color space of the stroke of one object. For the list of possible values, see <i>ColorSpace</i> in Graphics Properties above.	—	Yes	PS, PDF
<i>StrokeColorName</i>	string	The name of the color of the stroke of the vector object.	—	—	—
<i>StrokeColorSpace</i>	enumeration	The color space of the stroke of one object. For the list of possible values, see <i>ColorSpace</i> in Graphics Properties above.	—	—	PS, PDF
<i>StrokeColorType</i>	enumeration	This is an enumeration of known colors used to draw stroke. Refer to <i>FillColorType</i> for the list of possible values.	—	—	—
<i>StrokeOverprintFlag</i>	boolean	Is <i>"true"</i> when the stroke of one object has been set to overprint.	—	—	—
<i>StrokeShadingType</i>	enumeration	The type of shading used in the stroke. Values may be either: <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:

Name	Type	Description	Set	Tag	Documents
<i>CharacterProblem</i>	enumeration	Problem encountered to render character. The possible values are: <i>Corrupted</i> – Used when a character was found but could not be rendered. <i>IncorrectEncoding</i> – Used when encoding information is missing, incomplete, or otherwise incorrect. <i>Missing</i> – Use when the character could not be found in font. <i>Others</i> – Used in all other cases.	—	—	—
<i>MissingPrinterFont</i>	boolean	Is <i>"true"</i> if a referenced font has no printer information.	Yes	—	—
<i>MissingScreenFont</i>	boolean	Is <i>"true"</i> if a referenced font has no screen information.	—	—	—



Name	Type	Description	Set	Tag	Documents
<i>TextSize</i>	double	The size in points of the character.	—	—	—
<i>UseArtificialTextEffect</i>	enumerations	The artificial text effects list used to draw a character. Values may include: <i>Bold</i> <i>Italic</i> <i>Outline</i> <i>Shadow</i> <i>Underline</i> The authoring applications may apply the text effect directly, whereas in PS or PDF, the effect will be calculated.	—	—	—

#### 7.4.2.18 Vector Properties

Vector property attributes are derived from graphic objects with vector primitives. They can have a fill color and a stroke color, with given colors. This is a list of attributes that specifically apply to this kind of object:

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 may 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 110.

**Test run.** To reduce failures during processing, it is recommended that either individual devices or their controller support the testrun 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 may be based on hot folders. A Device that implements hot holders must define an input hot folder and an output folder for JDF. In addition, the “SubmitQueueEntry” message contains a URL attribute that allows specification of arbitrary JDF locators.

Implementation of JDF file-based protocol is simple, but it is important to note that the protocol does not support acknowledgement receipts for protocol error handling. It requires that the receiver polls the output folder of the processor. Finally, granting read/write access to your hot folder negates the security functions.

##### 8.2.1.1 JMF Transport Using The File Protocol

[New in JDF 1.2](#)

In order to allow JMF messaging based on a file protocol a set of additional conventions must be defined. There are some important differences between http and file-based protocols that must be taken into account:

- HTTP provides a URL to which the sender sends the file, while with the file protocol, the sender must provide 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 may 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 when using HTTPS.

### 8.2.2.1 Protocol Implementation Details

JDF Messaging will not specify a standard port.

#### Implementation of Messages

Only HTTP servers may be targeted by **Query** or **Command** messages. This is done with a standard HTTP Post request. The **JMF** is the body of the HTTP post message. The **Response** is the body of the initiated HTTP post response. **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.

## 8.3 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 (multi-part/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 bodies. In case of Multipart messages, the body consists of multiple messages, each identified by the individual MIME header and separated by a unique boundary string.

## 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 suggested to use .mjd for packages where a JDF is the first entity. Use .mjm 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 Fields

[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

This field 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 (part). The Multipart header uses this to identify itself as a multipart message and the subparts also have MIME types to identify their content. The following content types are defined for 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

This field is required for every part that is referenced by other parts in a multipart/related message. *Content-ID* identifies each different part within a multipart MIME message. Its value can be anything as long as it is defined using US-ASCII. Thus *Content-ID* may be a 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 Length

JDF allows a Content Length mechanism that may be used to enable fast scanning of MIME files of the bodyparts. Although this field is optional, it is recommended that it be included. Content Length is used to optimize the performance of scanning multipart messages. Each multipart bodypart may have an optional Content Length header field. Its syntax is identical to the syntax defined by [RFC2616].

When present, the Content Length identifies the number of octets of the encoded bodypart. With no encoding, as is the case with 7bit, 8bit, and binary, it represents the size of the bodypart. Otherwise it depends on what encoding method is used (e.g., base64, quoted-printable) and what the relationship is between the encoded size and the bodypart size. If an agent composing a MIME message can not derive a Content Length for its encoded body parts, it must omit the Content Length field.

An agent parsing such a message can use the Content Length field to seek the end of the body. This position is calculated by using the position of the first byte of the bodypart and adding the Content Length. At that position (one byte after the bodypart contents), the agent must check if the following characters are one of either “\r\n--boundary” or “--boundary.” If not, the agent must ignore the Content Length field and resume the normal MIME Multipart behavior, restarting scanning for the boundary from the beginning of the bodypart.

### 8.3.2.1.4 Content Transfer Encoding

This field is optional. [RFC2045] defines the following different encodings:

- "7bit"
- "quoted-printable"
- "base64"
- "8bit": This specifies that no additional encoding is applied to the data.
- "binary": This specifies that no additional encoding is applied to the data.

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 must be able to accept "base64". The other encodings are optional.

### 8.3.2.1.5 Content Disposition

This field is optional. See [RFC2183] It allows to define a filename. The *Disposition-Type* must be set to "attachment".

The Disposition filename parameter contains a suggested file name to store the attachment. This file name may be the original file name when creating the mime file and may be visible to the operator. Note that the standard mime escape rules must be used if the file name contains non US-ASCII characters.

Example:

```
Content-Disposition: attachment; filename=Coverpage.pdf;
```

### 8.3.2.2 Example Packaging of Individual JDF/JMF files in MIME

[New in JDF 1.2](#)

The following example displays MIME packaging of a JDF file as an individual MIME object:

```
MIME-Version: 1.0
Content-Type: multipart/related; boundary=abcdefg0123456789

--abcdefg0123456789
Content-Type: application/vnd.cip4-jdf+xml
Content-Length: 1234
<JDF ... >
<PreviewImage Separation = "PANTONE 128" URL="cid:123456.png" />
</JDF>
--abcdefg0123456789--
```

### 8.3.2.3 CID URL Scheme

[New in JDF 1.2](#)

One of the benefits of the MIME multipart/related *MediaType* is the ability to refer from one bodypart to another bodypart. This is done by using the cid: URL addressing scheme, specified in [RFC2392]. Please look at the example to see how it is used.

**Example:**

```
MIME-Version: 1.0
Content-Type: multipart/related; boundary=abcdefg0123456789

--abcdefg0123456789
Content-Type: application/vnd.cip4-jdf+xml
Content-Length: 1234

<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>
Content-Length: 12345
```

```
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 [RFC2396] does not permit in URLs; any such character (such as "+" or "&") must be hex-encoded using the %hh escape mechanism in the URL (see [RFC2396]). Therefore, matching the URL with the CID, must take account of the escaped equivalencies. Case-insensitive matching must be used.

### 8.3.2.4 Ordering of JDF/JMF in MIME Multipart/Related

#### [New in JDF 1.2](#)

The first section of the multipart MIME package must be the JMF submission command. Internal links are defined using the Content-ID (CID) label in MIME. Subsequent sections are the JDF jobs followed by the linked entities, such as the preview images shown in the following example:

**Example:** A multipart/related message is received that contains:

- Message.jmf
- Ticket01.jdf
- Pages.pdf

```
MIME-Version: 1.0
Content-Type: multipart/Related; boundary=unique-boundary
--unique-boundary
Content-type: application/vnd.cip4-jmf+xml
Content-Length: 1234
...
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFCClient"
TimeStamp="2000-11-07T13:15:56+01:00" Version="1.2">
  <Command ID="C0001" Type="SubmitQueueEntry">
    <QueueSubmissionParams Hold="true" URL="cid:JDF1@hostname.com"/>
  </Command>
</JMF>

--unique-boundary
Content-type: application/vnd.cip4-jdf+xml
Content-Length: 2345
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="1.2">
  <ResourcePool>
    <RunList Class="Parameter" DocCopies="1" FirstPage="0" ID="RunList4" IsPage="true"
NDoc="1" PageCopies="1" Status="Available">
```

```

    <LayoutElement ElementType="Document" HasBleeds="false" ID="LayoutElement_1"
IgnorePDLCopies="true" IgnorePDLImposition="true" IsPrintable="true">
    <FileSpec AppOS="Windows" Compression="None" Disposition="Retain"
ID="FileSpec_9" URL="cid:Asset01@hostname.com" UserFileName="Christmas Cards"/>
    </LayoutElement>
  </RunList>
</ResourcePool>
</JDF>

```

```

--unique-boundary
Content-type: application/pdf
Content-Length: 12345
Content-ID: <Asset01@hostname.com>
Content-Disposition: attachment; filename=Pages.pdf;
The pdf goes in here.
--unique-boundary--

```

When such a stream arrives at the server it must be 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 must be 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 may subset JDF but may not expand upon JDF. For instance, an ICS that covers desktop printers may either omit or prohibit all of the postpress processes related to case binding. ICS documents may 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 may be related to a specific edition of the JDF Specification, but may be released at a later date). Once approved, all published ICS documents will be available at [http://www.cip4.org/document\\_archive/ics.php](http://www.cip4.org/document_archive/ics.php).



---

# Appendix A Encoding

[Modified in JDF 1.2](#)

**Note:** This section has been completely rewritten in JDF 1.2. The work done on this section includes both new data types as well as many clarifications (and a better presentation of the information) that should make this section easier for the reader.

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 should be made to “XML Schema Part 2 - Datatypes” [XMLSchema].

### A.1.1 Ranges and RangeLists

Many of the following types can be considered to be a base type, a range for that type and a list of ranges or single values. A type will express a single value; a range for that type expresses a continuous inclusive range of values. The list type is a list of both single and range values.

Ranges are expressed as a pair of values separated by a ‘~’ character.

### 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 JDF1.2 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 required to support the mathematical concept of binary-valued logic:

#### Encoding

boolean attributes are encoded as either of the string values “*true*” or “*false*”. The XML Schema data type boolean values of “1” or “0” are not permitted.

#### Example

```
<Example Enable="true"/>
```

### A.2.2 CMYKColor

XML attributes of type CMYKColor are used to specify CMYK colors.

#### Encoding

CMYKColor attributes are primitive data types and are encoded as a string of four numbers (as doubles) in the range of [0...1.0] separated by whitespace. A value of 0.0 specifies no ink and a value of 1.0 specifies full ink.

#### Example:

```
<Color cmyk = "0.3 0.6 0.8 0.1"> (brick red)
```

### A.2.3 date

A calendar date, it represents a time period that starts at midnight on a specified day and lasts for 24 hours. Based on ISO 8601.

#### Encoding

It is represented identically to the XML Schema type: *date*

#### Example

```
<Example StartDate="1999-05-31"/>
```

### A.2.4 dateTime

Represents a specific instant of time. It must be a Coordinated Universal Time (UTC) or the time zone must be indicated by the offset to UTC. In other words, the time must be unique in all time zones around the world. It also allows infinity limits to allow for explicit ‘don’t care’ values, i.e. it must be finished before ‘anytime’.

#### Encoding

It is represented as a union of the XML Schema type: *dateTime*, and the infinity value tokens *INF* and *-INF*

#### Example

```
<Example Start="1999-05-31T18:20:00Z"/>
<Example Start="1999-05-31T13:20:00-05:00"/>
```

### A.2.5 DateTimeRange

[New in JDF 1.2](#)

XML attributes of type *DateTimeRange* are used to describe a range of points in time. More specifically, it describes a time span that has an absolute start and end. Unbounded ranges can use the infinity value tokens *INF* and *-INF*

#### Encoding

A *DateTimeRange* is represented by two *dateTime* or infinity tokens separated by the whitespace “~” whitespace sequence.

#### Example

```
<XXX range="1999-05-31T18:20:00Z ~ 1999-05-31T18:20:00Z"/>
<XXX range="1999-05-31T18:20:00Z ~ INF"/>
<XXX range="-INF ~ 1999-05-31T18:20:00Z"/>
```

### A.2.6 DateTimeRangeList

[New in JDF 1.2](#)

XML attributes of type *DateTimeRangeList* are used to describe a list of ranges of time 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:

```
<XXX RangeList="1999-05-31T18:20:00Z ~ 1999-05-31T18:20:00Z 1999-05-31T13:20:00-05:00 ~
INF"/>
```

### A.2.7 double

*double* Corresponds to IEEE double-precision 64-bit floating point type. It includes the infinity limit tokens *INF* and *-INF*, but does not allow the not a number token *NaN*.

#### Encoding

It is represented similarly to the XML Schema type: *double*. However string value *NaN*, is not permitted.

#### Example

```
<Example NegativePi="-3.14"/>
```

## A.2.8 DoubleList

[New in JDF 1.2](#)

XML attributes of type DoubleList are used to describe a variable length list of numbers (as doubles.) This type is used as the base for other JDF types that use a fixed length list of number, (e.g., CMYKColor which is restricted to four number in the list.)

### Encoding

A DoubleList is encoded as a string of whitespace-separated double values as defined in Section A.2.7, double.

### Example

```
<XXX list="3.14 1 .6"/>
```

## A.2.9 DoubleRange

[New in JDF 1.2](#)

XML attributes of type DoubleRange are used to describe a range of numbers (as doubles.) Mathematically spoken, the two numbers define a closed interval.

### Encoding

A DoubleRange is represented by two double values separated by a “~” (tilde) character and optional additional whitespace. Note: It is now recommended that the ‘~’ is surrounded by whitespace to aid validation and parsing.

### Example:

```
<XXX range="-3.14 ~ 5.13"/>
<XXX range="0 ~ INF"/>
```

## A.2.10 DoubleRangeList

[New in JDF 1.2](#)

XML attributes of type DoubleRangeList are used to describe a list of DoubleRanges and/or enumerated numbers as doubles).

### Encoding

A DoubleRangeList is a sequence of DoubleRanges and single double values separated by whitespace.

### Example:

```
<XXX list="-1 ~ -6 3.14 ~ 5.13 7 9 ~ 128 131 255 ~ INF"/>
```

## A.2.11 duration

Represents a duration of time. Based on ISO 8601. The single infinity limit token *INF* is permitted.

### Encoding

It is represented as a union of the XML Schema type: *duration*, and the string value *INF*

### Example

```
<Example Duration= "P1Y2M3DT10H30M"/>
```

## A.2.12 DurationRange

XML attributes of type DurationRange are used to describe a range of time durations. More specifically, it describes a time span that has a relative start and end.

### Encoding

A DurationRange is represented by two duration values, separated by the “~” (tilde) character and optional additional whitespace. Note: It is now recommended that the ‘~’ is surrounded by whitespace to aid validation and parsing.

### Examples:

```
<XXX range="P1Y2M3DT10H30M ~ P1Y2M3DT10H35M"/>
<XXX range="P1Y2M3DT10H30M ~ INF"/>
```

### A.2.13 DurationRangeList

[New in JDF 1.2](#)

XML attributes of type DurationRangeList are used to describe a list of ranges of time durations. More specifically, it describes a list of time spans that have a relative start and end.

#### Encoding:

A DurationRangeList is represented by sequence of DurationRanges and durations, separated by whitespace.

#### Example:

```
<XXX RangeList="P1Y2M3DT10H30M ~ P1Y2M3DT10H35M P1Y3M2DT10H30M"/>
```

### A.2.14 gYearMonth

Represents a specific Gregorian month in a specific Gregorian year. Based on ISO 8601.

#### Encoding

It is represented identically to the XML Schema type: *gYearMonth*

#### Example

```
<Example Month="2002-11"/>
```

### A.2.15 hexBinary

Represents arbitrary hex encoded binary data.

#### Encoding

It is represented identically to the XML Schema type: *hexBinary*

#### Example

```
<Example Hex="0A1C"/>
```

### A.2.16 ID

Represents the ID attribute from [XML]. It represents a name or string that contains no space characters and starts with a letter, ‘.’ or ‘\_’. Each ID value must be unique within a JDF document and thus uniquely identify the elements that bear them.

#### Encoding

It is represented identically to the XML Schema type: *ID*

#### Example

```
<Example ID="R-16"/>
```

### A.2.17 IDREF

IDREF Represents the IDREF attribute from [XML]. For a valid XML-document, an element with the ID value specified in IDREF must be present in the scope of the document.

#### Encoding

It is represented identically to the XML Schema type: *IDREF*

#### Example

```
<Example IDREF="R-16"/>
```

### A.2.18 IDREFS

IDREFS Represents the IDREFS attribute from [XML]. More specifically, this is a whitespace-separated list of IDREFs.

#### Encoding

It is represented identically to the XML Schema type: *IDREFS*

#### Example

```
<Example IDREFS="R-12 R-16"/>
```

### A.2.19 integer

Represents numerical integer values with tokens for representing infinity limits.

Implementation note: Except where explicitly noted otherwise, integers are not expected to exceed a value that can be represented as signed 32 bits.

#### Encoding

It is represented as a union of the XML Schema type: *integer*, and the infinity value tokens *INF* and *-INF*

#### Example

```
<Example Copies="36"/>
```

### A.2.20 IntegerList

XML attributes of type IntegerList are used to describe a variable length list of integer values.

#### Encoding

An IntegerList is encoded as a string of integers separated by whitespace.

#### Example

```
<XXX list="-INF 0 1 2 3 4 INF 1 3 0"/>
```

### A.2.21 IntegerRange

XML attributes of type IntegerRange are used to describe a range of integers. In some cases, ranges are defined for an unknown number of objects. In these cases, a negative value denotes a number counted from the end. For example, -1 is the last object, -2 the second to last, and so on. IntegerRanges that follow this convention are marked in the respective attribute descriptions.

If the first element of an IntegerRange specifies an element that is behind the second element, the Range specifies a list of integers in reverse order, counting backwards. For example “6 ~ 4” = “6 5 4” and “-1 ~ 0” = “last... 2 1 0”.

#### Encoding

An IntegerRange is represented by two integers, separated by a “~” (tilde) character and optional additional whitespace. Note: It is now recommended that the ‘~’ is surrounded by whitespace to aid validation and parsing.

#### Examples:

```
<XXX range="-3 ~ -5"/>
```

```
<XXX range="INF ~ -5"/>
```

### A.2.22 IntegerRangeList

XML attributes of type IntegerRangeList are used to describe a list of IntegerRanges and/or enumerated integers.

#### Encoding

A IntegerRangeList is represented by a sequence of IntegerRanges and integers, separated by whitespace.

#### Example:

```
<XXX list="-1 ~ -6 3 ~ 5 7 9 ~ 128 131"/>
```

### A.2.23 LabColor

XML attributes of type LabColor are used to specify absolute Lab colors. The Lab values are normalized to a Light of D50 and an angle of 2 degrees as specified in CIE Publication 15.2 — 1986 “Colorimetry, Second Edition” and ISO 13655:1996 “Graphic technology — Spectral measurement and colorimetric computation for graphic arts images.”

This corresponds to a white point of X = 0.9642, Y = 1.0000, and Z = 0.8249 in CIEXYZ color space. 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: “L a b”

#### Example:

```
<Color ... Lab="51.9 12.6 -18.9">
```

### A.2.24 language

Represents a natural language defined in IETF Request for Comment 1766. See <http://www.ietf.org/rfc/rfc1766.txt>.

#### Encoding

It is represented identically to the XML Schema type: *language*

#### Example

```
<Example Language="de"/> - German
<Example Language="de-CH"/> - Swiss German
<Example Language="en"/> - English
<Example Language="en-GB"/> - British English
```

### A.2.25 matrix

Coordinate transformation matrices are widely used throughout the whole printing process, especially in layout resources. They represent two dimensional transformations as defined by the PostScript and PDF reference manuals. 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”. Tx and Ty describe distances and are defined in points.

#### Example

```
<ContentObject CTM="1 0 0 1 3.14 21631.3" />
```

### A.2.26 NameRange

XML attributes of type NameRange are used to describe a range of NMTOKEN data that are acquired from a list of named elements, such as named pages in a PDL file. It depends on the ordering of the targeted list, which names are assumed to be included in the NameRange. The following two possibilities exist:

- 1 There is no explicit ordering. In this case, alphabetical ordering is implied.
- 2 There is explicit ordering, such as in a list of named pages in a **RunList**. In this case, the ordering of the **RunList** defines the order and all pages between the end pages are included in the NameRange.

#### Encoding

A NameRange typed attribute is represented by two NMTOKENs separated by a “~” (tilde) character and optional additional whitespace. Note: It is now recommended that the ‘~’ is surrounded by whitespace to aid validation and parsing.

#### Example

```
<XXX NameRange="Jack ~ Jill"/>
```

### A.2.27 NameRangeList

XML attributes of type NameRangeList are used to describe a list of NameRanges.

#### Encoding

A NameRangeList is represented by a sequence of NameRanges and NMTOKEN, separated by whitespace.

#### Example:

```
<XXX list="A brian ~ fred x z"/>
```

### A.2.28 NMTOKEN

Represents the NMTOKEN attribute type from [XML]. It represents a name or string that contains no space characters.

#### Encoding

It is represented identically to the XML Schema type: *NMTOKEN*

#### Example

```
<Example Alias="ABC_6"/>
```

### A.2.29 NMTOKENS

NMTOKENS Represents the NMTOKENS attribute type from [XML]. More specifically, this is a whitespace-separated list of NMTOKENS.

#### Encoding

It is represented identically to the XML Schema type: *NMTOKENS*

#### Example

```
<Example AliasList="ABC_6 ABCD_3 DEGF"/>
```

### A.2.30 PDFPath

[Modified in JDF 1.2](#)

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 Beziars.

#### 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 Section 8.6.1 “Path Construction Operators” in [PDF1.3] and [PDF1.4].

#### Example:

```
<ElementWithPath path="0 0 m 10 10 l 20 20 l"/>
```

### A.2.31 rectangle

XML attributes of type rectangle are used to describe rectangular locations on the page, sheet, or other printable surface. A rectangle is represented as an array of four numbers — *llx lly urx ury* — specifying the lower-left *x*, lower-left *y*, upper-right *x*, and upper-right *y* coordinates of the rectangle, in that order. This is equivalent to the ordering: Left Bottom Right Top. All numbers are defined in points.

#### Encoding

To maintain compatibility with PJTF, rectangles are primitive data types and are encoded as a string of four *numbers*, separated by whitespace: “*llx lly urx ury*” or “*l b r t*”.

#### Example:

```
<ContentObject ClipBox="0 0 3.14 21631.3" ... >
```

#### Implementation Remark

Since all numbers are real numbers, any comparison of boxes should take into account certain rounding errors. For example, different XYPairs may be considered equal when all numbers are the same within a range of 1 point.

### A.2.32 RectangleRange

[New in JDF 1.2](#)

XML attributes of type RectangleRange are used to describe a range rectangles.

#### Encoding

A RectangleRange is represented by one or two Rectangles, separated by a “~” (tilde) character and optional additional whitespace. Note: It is now recommended that the ‘~’ is surrounded by whitespace to aid validation and parsing.

#### Example:

```
<XXX range="1 2 3 4 ~ 5 6 7 8"/>
<XXX range="-INF -INF 3 4 ~ 0 1 INF INF"/>
```

### A.2.33 RectangleRange List

[New in JDF 1.2](#)

XML attributes of type RectangleRangeList are used to describe a list of rectangle ranges.

#### Encoding

A RectangleRangeList is represented by sequence of RectangleRanges and Rectangles, separated by whitespace.

#### Example:

```
<XXX RectangleRangeList="1 2 3 4 ~ 5 6 7 8 9 10 11 12 13 14 15 16"/>
```

### A.2.34 regExp

Represents a regular expression as defined in [XMLSchema].

#### Encoding

It is represented identically to the XML Schema type: *normalizeString*

#### Example

```
<Example expression="Foo({1|2}*)" />
```

### A.2.35 shape

XML attributes of type shape are used to describe a three dimensional box.

#### Encoding

A shape is represented as an array of three (positive or zero) *numbers* — *x y z* — specifying the Width *x*, height *y* and depth *z* coordinates of the shape, in that order.

#### Example:

```
<XXX Dimensions="10 20 40"/>
```

### A.2.36 ShapeRange

XML attributes of type ShapeRange are used to describe a range of shapes (three dimensional boxes). The range "*x1 y1 z1 ~ x2 y2 z2*" describes the area  $x1 \leq x \leq x2$  and  $y1 \leq y \leq y2$  and  $z1 \leq z \leq z2$ . Thus the shape "*2 3 4*" is within "*1 2 1 ~ 3 4 4*".

Note that this implies that all three values of the second entry must be  $\geq$  the corresponding values of the first entry. The following example is therefore invalid: "*1 2 1 ~ 0 4 4*".

#### Encoding

A ShapeRange is represented by two shapes, separated by a “~” (tilde) character and optional additional whitespace. Note: It is now recommended that the ‘~’ is surrounded by whitespace to aid validation and parsing.

#### Example:

```
<XXX Shaperange="1 2 3 ~ 4 5 6"/>
<XXX Shaperange="1 2 3 ~ 4 INF 6"/>
```



### A.2.37 ShapeRangeList

XML attributes of type ShapeRangeList are used to describe a list of ShapeRange and/or shapes.

#### Encoding

A ShapeRangeList is a sequence of ShapeRange and shapes separated by whitespace.

#### Example:

The brackets below the example illustrate the grouping of shapes and ShapeRanges.

```
<XXX Shapelist="100 200 300 ~ 110 220 330 150 300 150 2 3 0 ~ 3 4 5"/>
          (                               ) (                               ) (                               )
```

### A.2.38 sRGBColor

XML attributes of type sRGBColors are used to specify sRGB colors.

#### Encoding

sRGBColors are primitive data types and are encoded as a string of three numbers in the range of [0...1.0] separated by whitespace. A value of 0 specifies no intensity (black) and a value of 1 specifies full intensity:

```
"r g b"
```

#### Example:

```
<Color sRGB="0.3 0.6 0.8" ... >
```

### A.2.39 string

Represents character strings in XML.

#### Encoding

It is represented identically to the XML Schema type: *normalisedString* NB. This means that tabs, linefeeds, and so on are not valid characters.

#### Example:

```
<Example Name="Test With Space"/>
```

### A.2.40 TimeRange

[Deprecated in JDF 1.2](#)

### A.2.41 TransferFunction

XML attributes of type TransferFunction are functions that have a one-dimensional input and output. In JDF, they are encoded as a simple kind of sampled functions and used to describe transfer curves of image transfer processes from one medium to the next, (e.g. film to plate, or plate to press).

A transfer curve consists of a series of XY pairs where each pair consist of the stimuli (X) and the resulting value (Y). To calculate the result of a certain stimuli, the following algorithms must be applied:

- 1 If  $x \leq$  first stimuli, then the result is the y value of the first xy pair.
- 2 If  $x \geq$  the last stimuli, then the result is the y value of the last xy pair.
- 3 Search the interval in which x is located.
- 4 Return the linear interpolated value of x within that interval.

#### Encoding

A TransferCurve is encoded as a string of space-separated *numbers* (as doubles). The numbers are the XY pairs that build up the transfer curve.

#### Example:

```
<someElementWithTransferCurve someCurve="0 0 .1 .2 .5 .6 .8 .9 1 1"/>
```

### A.2.42 URI

Short for URI-reference. Represents a Uniform Resource Identifier (URI) Reference as defined in Section 4 of [RFC2396].

#### Encoding

It is represented identically to the XML Schema type: *anyURI*

#### Example

```
<Example URI="http://www.w3.org/1999/XMLSchema"/>
```

### A.2.43 URL

Short for URL-reference. Represents a Uniform Resource Locator (URL) Reference as defined in Section 4 of [RFC2396].

#### Encoding

It is represented identically to the XML Schema type: *anyURI*

#### Example

```
<Example URL="file://hubble/test.txt"/>
```

### A.2.44 XYPair

XML attributes of type XYPair are used to describe sizes like *Dimensions* and *PageSize*. They can also be used to describe positions on a page. All numbers that describe lengths are defined in points.

#### Encoding

XYPair attributes are primitive data types and are encoded as a string of two *numbers*, separated by whitespace: “x y”

#### Example:

```
<CutBlock BlockSize="612 792">
```

#### Implementation Remark

Since all numbers are real numbers, comparison of XYPairs should take into account certain rounding errors. For example, different XYPairs may be considered equal when all numbers are the same within a range of 1 point.

### A.2.45 XYPairRange

XML attributes of type XYPairRange are used to describe a range of XYPairs. The range “ $x_1\ y_1 \sim x_2\ y_2$ ” describes the area  $x_1 \leq x \leq x_2$  and  $y_1 \leq y \leq y_2$ . Thus the XYPair “2 3” is within “1 2 ~ 3 4”. Note that this implies that both values of the second entry must be  $\geq$  the corresponding values of the first entry. The following example is therefore invalid: “1 2 ~ 0 4”.

#### Encoding

An XYPairRange is represented by two XYPairs, separated by a “~” (tilde) character and optional additional whitespace. Note: It is now recommended that the ‘~’ is surrounded by whitespace to aid validation and parsing.

#### Example:

```
<XXX XYrange="1 2 ~ 3 4"/>
<XXX XYrange="-INF 2 ~ 3 INF"/>
```

### A.2.46 XYPairRangeList

XML attributes of type XYPairRangeList are used to describe a list of XYPairRange and/or XYPairs.

#### Encoding

A XYPairRangeList is a sequence of XYPairRange and XYPairs separated by whitespace.

#### Example:

The brackets below the example illustrate the grouping of XYPairs and XYPairRanges.

```
<XXX XYlist="100 200 ~ 110 220 150 300 150 350 200 300 ~ INF INF"/>
      (                ) (                ) (                ) (                )
```

## A.2.47 XPath

[New in JDF 1.2](#)

Represents an XPath expression. [XPath]

### Encoding

It is represented identically to the XML Schema type: *token*

### Example

```
<Example xpath= "JDF/AuditPool/Created/@TimeStamp" />
```

## A.3 Enumerations and Lists

### A.3.1 enumeration

Represents a closed set of values.

### Encoding

It is represented by an enumerated list of values derived from the XML Schema type: *NMTOKEN*

### Example

```
<Example Orientation="Flip90"/>
```

### A.3.2 enumerations

Represents a list of values taken from a closed set. Values may be repeated within the list. If there are any implications to the order of the values this will be detailed in the appropriate items description, otherwise none is implied.

### Encoding

It is represented by a whitespace-separated list of enumeration values derived from the XML Schema type: *NMTOKEN*

### Example

```
<Example Orientations="Rotate90 Flip90"/>
```

### A.3.3 Defined JDF enumeration Data Types

This section is a list of defined enumeration data types. These types are to be used wherever possible for enumerated values and lists of values.

#### A.3.3.1 JDFJMFVersion

Describes the schema version of a JDF or JMF instance.

Table A-1: JDFJMFVersion enumeration Values

Enumeration Value	Comment
1.1	JDF 1.1
1.2	JDF 1.2

### A.3.3.2 NamedColor

Colors of preprocessed products such as Wire-O binders and cover leaflets. The entries in the following table may be prefixed by either “Dark” or “Light”. The result may additionally be prefixed by “Clear” to indicate translucent material. For example, “*ClearDarkBlue*” indicates a translucent dark blue, “*ClearBlue*” a translucent blue and “Blue” indicates an opaque blue.

Table A-2: Named Colors

Color name/ Enumeration Value	Comment
<i>Black</i>	—
<i>Blue</i>	—
<i>Brown</i>	—
<i>Buff</i>	—
<i>Cyan</i>	<a href="#">New in JDF 1.2</a>
<i>Gold</i>	—
<i>Goldenrod</i>	—
<i>Gray</i>	—
<i>Green</i>	—
<i>Ivory</i>	—
<i>Magenta</i>	<a href="#">New in JDF 1.2</a>
<i>MultiColor</i>	<a href="#">New in JDF 1.1</a>
<i>Mustard</i>	<a href="#">New in JDF 1.1</a>
<i>NoColor</i>	—
<i>Orange</i>	—
<i>Pink</i>	—
<i>Red</i>	—
<i>Silver</i>	—
<i>Turquoise</i>	—
<i>Violet</i>	—
<i>White</i>	—
<i>Yellow</i>	—

### A.3.3.3 Orientation

Orientation of a physical resource. For details see Table 2-3, “Matrices and Orientation values used to describe the orientation of a Component,” on page 24.

Table A-3: Page Orientation

Enumeration Value	Comment
<i>Rotate0</i>	
<i>Rotate90</i>	
<i>Rotate180</i>	
<i>Rotate270</i>	
<i>Flip0</i>	
<i>Flip90</i>	
<i>Flip180</i>	
<i>Flip270</i>	

### A.3.3.4 Side

Describes one side of imageable material.

Table A-4: Side enumeration Values

Enumeration Value	Comment
<i>Front</i>	
<i>Back</i>	

### A.3.3.5 WorkStyle

Table A-5: WorkStyle enumeration Values

Enumeration Value	Comment
<i>Simplex</i>	No turning.
<i>Perfecting</i>	Many sheetfed printing presses have perfecting cylinder(s) built in. The leading edge of the print sheet changes as the sheet is turned by the perfecting cylinder, but the side lays remain unaltered. In this regard, this <i>WorkStyle</i> is similar to <i>WorkAndTumble</i> , but <i>Perfecting</i> is an in-line operation during the press run. Therefore, an additional plate (set) is required during this press run.
<i>WorkAndBack</i>	This <i>WorkStyle</i> describes the printing on both sides of the substrate with a different plate (set) in the second run. After the first run the side lays are altered but the front lays stay as they were. Lays can be turned by hand or using a pile reverser. Two-plate sets are necessary for <i>WorkAndBack</i> .
<i>WorkAndTurn</i>	<i>WorkAndTurn</i> refers to the turning of the first-run sheet for subsequent perfecting. The front lays remain unchanged but the side lays must be altered. The alteration can be made by hand or using a pile turner. Turning happens after the first press run and the plate (set) is used again in the second press run, imaging the other sheet surface.
<i>WorkAndTumble</i>	The <i>WorkAndTumble</i> method is also used for perfecting. The leading edge of the print sheet changes as the sheet is turned, but the side lays remain unaltered. Tumbling happens after the first press run and the plate (set) is used again in the second press run, imaging the other sheet surface.
<i>WorkAndTwist</i>	Done between two press runs. The palette is twisted 180 degree before the second run is performed so that the front lay and the side lay both change. The surface to be imaged is the same at both runs. Each run prints only part of the surface. The plate (set) stay in the machine. This <i>WorkStyle</i> is used for saving plate or film material. It is no longer a common <i>WorkStyle</i> .

### A.3.4 XYRelation

[New in JDF 1.2](#)

XML attributes of type XYRelation define the relationship between two ordered numbers.

Table A-6: XYRelation enumeration Values.

Enumeration Value	Comment
<i>gt</i>	$X > Y$
<i>ge</i>	$X \geq Y$
<i>eq</i>	$X = Y$
<i>le</i>	$X \leq Y$
<i>lt</i>	$X < Y$
<i>ne</i>	$X \neq Y$

## **A.4 JDF File Formats**

This section describes the specific file formats used by JDF. JDF uses TIFF and JPEG file formats, as well as the PNG image file format. The following sections explain in what ways PNG is used in JDF.

### **A.4.1 PNG Image Format**

JDF uses the PNG images for representing preview images. CIP3 defined two formats: composite CMYK and separated. With PNG, only the separated format is supported for color spaces other than RGB. The composite CMYK or spot color representations must be represented as separated CMYK or spot colors. Thus, preview images are stored as separate PNG images and JDF links them together. Viewable images and thumbnails can be represented as composite RGB PNG images.

References: <http://www.w3.org/Graphics/png>.

---

## Appendix B Schema

XML Schema for JDF (and JMF) will be published on: <http://www.CIP4.org>.

The XML Schema is not sufficient to completely validate a JDF job. For example, partitioned resources or process node types as defined in JDF cannot be validated by XML Schema processors. In other words, the structure of some elements depends on the context of usage which cannot be completely described by XML Schema. Thus, the XML Schema for JDF will be structured in a way that it enables a prevalidation of valid JDF-candidates but does not preclude all syntactically invalid files to be validated.

### B.1 Using xsi:type

[New in JDF 1.2](#)

XML Schema permits that multiple type definitions be derived from a base type. Wherever the schema has define an element of that base type, it is possible for the document to indicate to a validator the particular derived type that it has used. This it does by using the `xsi:type` attribute with a value of the name of the type, where the `xsi` tag is associated with the Schema Instance namespace that has to be declared in the document.

Note: Use of `xsi` as the tag is normal practice.

Note: The selected type is namespace qualified (which permits extensions)

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

[New in JDF 1.2](#)

When used with JDF nodes then all processes defined in Section 6 are supported. Furthermore the value to be used is identical to the process type, thus a JDF node that has a *Type* of “*DigitalPrinting*” can inform validators to use the schema definition for ***DigitalPrinting*** nodes by also setting *xsi:type* to “*DigitalPrinting*”.

Some JDF nodes are general in their nature and do not have a restricted definition, (i.e., Product, Combined, and so on.) General definitions with the appropriate name are provided to enable consistent use of `xsi:type`.

#### Example

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="BackCover" Status="InProgress"
Type="DigitalPrinting" Version="1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http:// www.CIP4.org/JDFSchema_1_1 JDF.xsd"
xsi:type="DigitalPrinting">
  <ResourceLinkPool>
    <DeviceLink Usage="Input" rRef="Entire_Book"/>
    <RunListLink Usage="Input" rRef="Entire_Book"/>
  </ResourceLinkPool>
</JDF>
```

If the JDF is not in the default namespace then the type name needs to be altered accordingly:

```
<jdf:JDF xmlns:jdf="http://www.CIP4.org/JDFSchema_1_1" ID="BackCover"
Status="InProgress" Type="DigitalPrinting" Version="1.2" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance" xsi:type="jdf:DigitalPrinting">
  <jdf:ResourcePool>
    <jdf:Device Class="Implementation" DeviceID="Unknown Device" ID="Device_001"
Status="Available"/>
    <jdf:RunList Class="Parameter" ID="RunList_001" Status="Unavailable"/>
  </jdf:ResourcePool>
  <jdf:ResourceLinkPool>
    <jdf:DeviceLink Usage="Input" rRef="Device_001"/>
    <jdf:RunListLink Usage="Input" rRef="RunList_001"/>
  </jdf:ResourceLinkPool>
</jdf:JDF>
```

The JDF Schema defines types for JDF Process nodes and JMF Messages. It is recommended that these types are used with `xsi:type`.



#### Using JDF Schema

Any JDF processor should be capable of validating whether or not a JDF Job meets JDF requirements. This can be accomplished by using a schema when parsing or by using an application derived from a schema. The schema itself may be sub-setted into multiple schemas that are used for validation purposes at different points in the workflow. For instance, a JMF schema subset may be used to test JDF-compliant devices on your shop floor. A product intent subset may be used to check customer submitted job specifications.

## B.1.2 Using `xsi:type` with JMF Messages

### [New in JDF 1.2](#)

JMF Messages are organized into families — Command, Acknowledge, etc. (See “Message Families” on page 131.)— and each of these families has messages for each message *Type* — Events, KnownControllers, etc. Because it is the convolution of these two that are the unique derived types, the name used in `xsi:type` has to be the convolution of the message family and Type.

To query an event a Query message with an Events/QueryTypeObj would be used. The type definition name employed by the JDF Schema would therefore be QueryEvents.

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="TestSender" TimeStamp="2003-11-07T12:15:56Z" Version="1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance">
  <Query ID="Message_001Q" Type="Events" xsi:type="QueryEvents">
    <NotificationFilter/>
  </Query>
  <Response ID="Message_001R" Type="Events" refID="Q001" xsi:type="ResponseEvents">
    <NotificationDef Classes="Error" Type="Barcode"/>
  </Response>
</JMF>
```

Note JMF messages also do not have to be in the default namespace as in the JDF Node example above.



---

## Appendix C Converting PJTF to JDF

This appendix is provided as a non-normative guide to developers writing applications that will consume PJTF version 1.2 jobs and produce JDF.

### C.1 PJTF Object Conversion

Many PJTF objects are directly translatable to JDF processes or resources. Others, especially those containing multiple keys, correspond to multiple processes and resources. For example, the **JobTicketContents** object corresponds to four JDF processes and three JDF resources. And still others, such as **AuditObject**, cannot be translated to JDF at all.

Listed below are the prominent PJTF objects and the JDF components to which they correspond. Each section heading contains the title of the object in question, and each section contains a descriptive table. The first column in the tables, entitled JDFKey or Object, contains a list of the keys or objects contained within the object being described. For example, the **Accounting** object contains an **Address** object, while the **Address** object contains an **Address** key. If no subobject or key is contained within the object, then the first column is left blank and the process or resource listed is assumed to correspond directly to that object.

#### C.1.1 Accounting

PJTF Key or Object	JDF Process	JDF Resource	Description
Address	—	<b>Address</b>	—

#### C.1.2 Address

PJTF Key or Object	JDF Process	JDF Resource	Description
Address	—	<b>Address</b>	Used whenever people or organizations need to be identified.

#### C.1.3 Analysis

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	<b>Analysis</b>	—

#### C.1.4 AuditObject

Audit objects must not be translated. PJTF Audit objects describe the results of operations on files, while JDF Audit elements describe the results of processes, so there is a basic incompatibility between the two. In addition, PJTF Audit objects will not be needed to direct further processing of the job after it is converted to JDF.

**C.1.5 ColorantAlias**

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	—	Maps to a subelement of the <b>ColorantControl</b> resource.

**C.1.6 ColorantControl**

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	<b>ColorantControl</b>	—

**C.1.7 ColorantDetails**

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	—	Keys in the PJTF ColorantDetails dictionary are a set of colorant names. The values are DeviceColorant objects.

**C.1.8 ColorantZoneDetails**

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	<b>TrappingParams</b>	DeviceColorant map to the ColorantZoneDetails subelement of the <b>TrappingParams</b> resource.

**C.1.9 ColorSpaceSubstitute**

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	—	Maps to a subelement of the <b>ColorantControl</b> resource.

**C.1.10 Delivery**

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	<b>Delivery</b>	<b>Address</b>	Specifies a quantity of a product to be delivered to an address.

**C.1.11 DeviceColorant**

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	—	Maps to a Color subelement of the <b>ColorPool</b> resource. The name is entered in the SeparationSpec of a <b>TrappingDetails</b> resource.

**C.1.12 Document**

**JobTicketContents**, **Document** and **PageRange** objects are decomposed into a number of different JDF objects.

Most of the key/value pairs translate into various resources.

PJTF Key or Object	JDF Process	JDF Resource	Description
bleed media trim	—	<b>RunList</b>	Maps to attributes of the <b>RunList</b> resource or to processes in which they are used.
ColorantControl	—	<b>ColorantControl</b>	—
Files	—	<b>RunList</b> <b>FileSpec</b>	Maps to <b>FileSpec</b> resources contained within <b>RunList</b> elements.
Finishing	<b>AdhesiveBinding</b> <b>EndSheetGluing</b> <b>SaddleStitching</b> <b>SideSewing</b> <b>Stitching</b> <b>ThreadSewing</b>	<b>AdhesiveBindingParams</b> <b>EndSheetGluingParams</b> <b>SaddleStitchingParams</b> <b>SideSewingParams</b> <b>StitchingParams</b> <b>ThreadSewingParams</b>	—
FontPolicy	—	<b>FontPolicy</b>	The resource is attached to the applicable processes.
IgnoreHalftone	—	—	Maps to the <i>IgnoreHalftone</i> attribute of the <b>PDFToPSConversionParams</b> resource.
InsertPage	<b>Imposition</b>	<b>RunList</b> <b>Sheet</b>	Occurs as an attribute either of <b>RunList</b> resources or of <b>Sheet</b> resources referenced by <b>Imposition</b> processes.
NewSheet	<b>Imposition</b>	<b>InsertSheet</b>	NewSheets become instances of <b>InsertSheet</b> resources on <b>RunLists</b> with a <i>SheetUsage</i> attribute of “ <i>Header</i> ”.
Media	—	<b>Media</b>	Maps to a subelement of the <b>ExposedMedia</b> resource.
MediaSource	—	—	Maps to a <b>Media</b> resource reelement of a <b>DigitalPrintingParams</b> resource.
MediaUsage	<b>Dividing</b>	<b>DividingParams</b>	Specifies controls for roll-fed media.
Rendering	<b>Rendering</b>	—	—
Trailer	<b>Imposition</b>	<b>InsertSheet</b>	Trailers become instances of <b>InsertSheet</b> resources on <b>RunLists</b> with a <i>Usage</i> attribute of “ <i>trailer</i> ”
Trapping	<b>Trapping</b>	—	—

### C.1.13 Finishing

Finishing operations are derived from CIP3 PPF. Conversion of PJTF **Finishing** objects is vendor-dependent, since the PJTF specification does not describe any detail for **Finishing** objects.

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	<b>AdhesiveBinding</b> <b>EndSheetGluing</b> <b>SaddleStitching</b> <b>SideSewing</b> <b>Stitching</b> <b>ThreadSewing</b>	<b>AdhesiveBindingParams</b> <b>EndSheetGluingParams</b> <b>SaddleStitchingParams</b> <b>SideSewingParams</b> <b>StitchingParams</b> <b>ThreadSewingParams</b>	—

### C.1.14 FontPolicy

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	<b>Interpreting</b>	<b>FontPolicy</b>	—

### C.1.15 InsertPage

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	<b>RunList</b>	<b>InsertPage</b> objects may generate a <b>InsertSheet</b> resource within a <b>RunList</b> .

### C.1.16 InsertSheet

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	<b>InsertSheet</b>	—

### C.1.17 Inventory

PJTF Key or Object	JDF Process	JDF Resource	Description
—	—	—	—

### C.1.18 Ticket

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys except Audit, Scheduling, PreflightResults	Any process	Any resource	Keys may be represented at various levels of the JDF tree. Contents are represented as processes, resources, and versions.

### C.1.19 JobTicketContents

**JobTicketContents**, **Document** and **PageRange** objects are decomposed into a number of different JDF objects. Most of the key/value pairs translate into various resources.

PJTF Key or Object	JDF Process	JDF Resource	Description
Accounting	—	—	Maps to the CustomerInfo element.
Administrator	—	—	Maps to the CustomerInfo element.
ColorantControl	—	<b>ColorantControl</b>	—
Delivery	<b>Delivery</b>	<b>DeliveryParams</b>	—

PJTF Key or Object	JDF Process	JDF Resource	Description
Documents	—	<b>RunList</b>	May require more than one <b>RunList</b> resource.
EndMessage	—	—	Maps to the <i>End</i> attribute of the <b>NodeInfo</b> element.
Finishing	<b>AdhesiveBinding</b> <b>EndSheetGluing</b> <b>SaddleStitching</b> <b>SideSewing</b> <b>Stitching</b> <b>ThreadSewing</b>	<b>AdhesiveBindingParams</b> <b>EndSheetGluingParams</b> <b>SaddleStitchingParams</b> <b>SideSewingParams</b> <b>StitchingParams</b> <b>ThreadSewingParams</b>	—
FontPolicy	<b>Interpreting</b> <b>PDFToPSConversion</b>	<b>FontPolicy</b>	The <b>FontPolicy</b> resource is attached to any process that uses it.
IgnoreHalftone	—	—	Maps to the <i>IgnoreHalftone</i> attribute of the <b>PDFToPSConversionParams</b> resource.
InsertPage	<b>Imposition</b>	<b>RunList</b> <b>Sheet</b>	Occurs as an attribute either of <b>RunList</b> resources or of <b>Sheet</b> resources referenced by <b>Imposition</b> processes.
JobName	—	—	<b>CustomerJobName</b> in the <b>CustomerInfo</b> element of the <b>JobInfo</b> node.
Layout	<b>Imposition</b>	<b>Layout</b>	—
MarkDocuments	<b>Imposition</b>	<b>RunList</b>	Requires one of two <b>RunList</b> resources, each of which is a resource of the <b>Imposition</b> process.
MediaSource	—	—	Maps to a <b>MediaSource</b> resource reelement of a <b>DigitalPrintingParams</b> resource.
MediaUsage	<b>Dividing</b>	<b>DividingParams</b>	Specifies controls for roll-fed media.
NewSheet	<b>Imposition</b>	<b>InsertSheet</b>	NewSheets become instances of <b>InsertSheet</b> resources on <b>RunLists</b> with a <i>Usage</i> attribute of “ <i>header</i> ”
PrintLayout	<b>Imposition</b>	—	Maps to a subelement of the <b>Layout</b> resource.
Rendering	<b>Rendering</b>	—	Maps to the attribute of the <b>Rendering</b> process.
Scheduling	—	—	The Scheduling object is not translated.
StartMessage	—	—	Maps to the <i>Start</i> attribute of the <b>NodeInfo</b> element.

PJTF Key or Object	JDF Process	JDF Resource	Description
Submitter	—	—	Maps to the <b>CustomerInfo</b> element.
Trailer	<b><i>Imposition</i></b>	<b>InsertSheet</b>	Trailers become instances of <b>InsertSheet</b> resources on <b>RunLists</b> with a <i>Usage</i> attribute of “ <i>trailer</i> ”
Trapping	<b><i>Trapping</i></b>	—	—

**C.1.20 JTFile**

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	—	In most cases, JTFile objects will become <b>FileSpec</b> resources. If a <b>FilesDictionary</b> is present, the resource may need to be partitioned by Separation. If a <b>PlaneOrder</b> is present, <b>RunLists</b> which reference the file will need to be partitioned by Separation and structured to reference the page in the file appropriately.

**C.1.21 Layout**

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	<b>Imposition</b>	<b>Layout</b>	—

**C.1.22 Media**

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	<b>Media</b>	Maps to a subelement of the <b>ExposedMedia</b> resource.

**C.1.23 MediaSource**

PJTF Key or Object	JDF Process	JDF Resource	Description
ManualFeed	—	—	Maps to the <i>ManualFeed</i> attribute of a <b>MediaSource</b> resource pointed to by a reelement of a <b>DigitalPrintingParams</b> or <b>IDPrintingParams</b> resource.
LeadingEdge	—	—	Maps to the <i>LeadingEdge</i> attribute of a <b>MediaSource</b> resource reelement of a <b>DigitalPrintingParams</b> or <b>IDPrintingParams</b> resource.
Media	—	—	Maps to a Media reelement of a <b>MediaSource</b> resource.
MediaClass	—	—	Maps to the <i>MediaTypeDetails</i> attribute of a <b>Media</b> resource of a <b>DigitalPrintingParams</b> or <b>IDPrintingParams</b> resource.
Position	—	—	Maps to the <i>MediaLocation</i> attribute of a <b>MediaSource</b> resource.

**C.1.24 MediaUsage**

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	<b>Dividing</b>	<b>DividingParams</b>	Specifies controls for roll-fed media.

**C.1.25 PageRange**

**JobTicketContents**, **Document** and **PageRange** objects are decomposed into a number of different JDF objects.

Most of the key/value pairs translate into various resources.

PJTF Key or Object	JDF Process	JDF Resource	Description
bleed media trim	—	<b>RunList</b>	Maps to attributes of the <b>RunList</b> resource or to processes in which they are used.
ColorantControl	—	<b>ColorantControl</b>	—
Delivery	<b><i>Delivery</i></b>	<b>DeliveryParams</b>	—
Files	—	<b>RunList</b> <b>FileSpec</b>	Maps to <b>FileSpec</b> resources contained within RunList elements.
Finishing	<b><i>AdhesiveBinding</i></b> <b><i>EndSheetGluing</i></b> <b><i>SaddleStitching</i></b> <b><i>SideSewing</i></b> <b><i>Stitching</i></b> <b><i>ThreadSewing</i></b>	<b>AdhesiveBindingParams</b> <b>EndSheetGluingParams</b> <b>SaddleStitchingParams</b> <b>SideSewingParams</b> <b>StitchingParams</b> <b>ThreadSewingParams</b>	—
FontPolicy	<b><i>Interpreting</i></b> <b><i>PDFToPSConversion</i></b>	<b>FontPolicy</b>	The <b>FontPolicy</b> resource is attached to any process that uses it.
IgnoreHalftone	—	—	Maps to the <i>IgnoreHalftone</i> attribute of the <b>PDFToPSConversionParams</b> resource.
InsertPage	<b><i>Imposition</i></b>	<b>RunList</b> <b>Sheet</b>	Occurs as an attribute either of <b>RunList</b> resources or of <b>Sheet</b> resources referenced by <b><i>Imposition</i></b> processes.
Media	—	<b>Media</b>	Maps to a subelement of the <b>ExposedMedia</b> resource.
MediaSource	—	—	Maps to a <b>Media</b> resource reelement of a <b>DigitalPrintingParams</b> resource.
MediaUsage	<b><i>Dividing</i></b>	<b>DividingParams</b>	Specifies controls for roll-fed media.
NewSheet	<b><i>Imposition</i></b>	<b>InsertSheet</b>	NewSheets become instances of <b>InsertSheet</b> resources on <b>RunLists</b> with a <i>Usage</i> attribute of “ <i>header</i> ”
Rendering	<b><i>Rendering</i></b>	—	—
Trailer	<b><i>Imposition</i></b>	<b>InsertSheet</b>	Trailers become instances of <b>InsertSheet</b> resources on <b>RunLists</b> with a <i>Usage</i> attribute of “ <i>trailer</i> ”



PJTF Key or Object	JDF Process	JDF Resource	Description
Trapping	<b>Trapping</b>	—	—
Which	—	<b>RunList</b>	The <i>Pages</i> attribute or combination of <i>FirstPage</i> and <i>SkipPage</i> in <b>RunLists</b> reflect the values of Which. Note: More than one <i>PageRange</i> may generate <i>Pages</i> entries for a single Run.

### C.1.26 PlacedObject

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	—	Maps to a subelement of the <b>Surface</b> resource.

### C.1.27 PlaneOrder

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	<b>RunList</b>	See Section C.4, Translating the Contents Hierarchy.

### C.1.28 Preflight

Note that the definition of **Preflight** has been completely redefined in JDF 1.2. A one to one translation of PJTF preflight to JDF preflight is no longer possible.

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	<b>Preflight</b>	—	—

### C.1.29 PreflightConstraint

Note that the definition of **Preflight** has been completely redefined in JDF 1.2. A one to one translation of PJTF preflight to JDF preflight is no longer possible.

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	—	Maps to a subelement of the <b>PreflightProfile</b> resource.

### C.1.30 PreflightDetail

Note that the definition of **Preflight** has been completely redefined in JDF 1.2. A one to one translation of PJTF preflight to JDF preflight is no longer possible.

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	—	Maps to a subelement of the <b>PreflightAnalysis</b> resource.

### C.1.31 PreflightInstance

Note that the definition of **Preflight** has been completely redefined in JDF 1.2. A one to one translation of PJTF preflight to JDF preflight is no longer possible.

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	—	Subelement of the <b>PreflightAnalysis</b> resource

### C.1.32 PreflightInstanceDetail

Note that the definition of **Preflight** has been completely redefined in JDF 1.2. A one to one translation of PJTF preflight to JDF preflight is no longer possible.

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	—	Subelement of the <b>PreflightAnalysis</b> resource

### C.1.33 PreflightResults

Note that the definition of **Preflight** has been completely redefined in JDF 1.2. A one to one translation of PJTF preflight to JDF preflight is no longer possible.

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	—	This object is not translated.

### C.1.34 PrintLayout

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	<b>Imposition</b>	—	Maps to a subelement of the <b>Layout</b> resource.

### C.1.35 Profile

Note that the definition of **v** has been completely redefined in JDF 1.2. A one to one translation of PJTF preflight to JDF preflight is no longer possible.

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	<b>Preflighting</b>	<b>PreflightProfile</b>	

### C.1.36 Rendering

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	<b>Rendering</b>	<b>RenderingParams</b>	—

### C.1.37 ResourceAlias

PJTF Key or Object	JDF Process	JDF Resource	Description
Location		<b>PDLResourceAlias</b>	<b>Location</b> is Device
File		<b>PDLResourceAlias</b>	<b>File</b> is supported via the <b>SourceFile</b> fileref.
This		<b>PDLResourceAlias</b>	<b>This</b> is supported via the <b>SourceFile</b> fileref.
ResourceName		<b>PDLResourceAlias</b>	This key is not used. References to the aliased resource run via the <b>ResourceLink</b> element.
SourceFile		<b>PDLResourceAlias</b>	Source file maps to an attribute of this resource.

PJTF **ResourceAlias** objects provide a unified namespace that allows each PJTF object to refer to the resources it needs to execute the job of which it is a part. More specifically, PJTF version 1.1 supports the use of **ResourceAlias** objects to allow references to halftones and colorspace.

For the **ResourceAlias::Location** key, the **File** and **This** keys are supported by a **SourceFile** attribute whose value is a fileref. The translator must provide a reference to the original PJTF file (for this) or a copy that contains the referenced resources.

### C.1.38 Scheduling

**Scheduling** objects are not translated. It is presumed that translation of PJTF jobs into JDF is performed to allow the reuse of PJTF jobs that have been archived. Thus, the original scheduling information embedded in the PJTF is irrelevant.

### C.1.39 Signature

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	—	Maps to a subelement of the <b>Layout</b> resource.

## C.2 Sheet

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	<b>Sheet</b>	—

### C.2.1 SlipSheet

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	<b>InsertSheet</b>	SlipSheets become an <b>InsertSheet</b> resource which may define new media, and which has a <i>Usage</i> attribute of “ <i>trailer</i> ”.

### C.2.2 Surface

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	<b>Surface</b>	—

### C.2.3 Tile

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	<b>Tiling</b>	<b>Tile</b>	—

### C.2.4 Trapping

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	<b>Trapping</b>	<b>TrappingParams</b>	—

### C.2.5 TrappingDetails

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	<b>TrappingDetails</b>	See the PJTF DeviceColorant object entry for details on how it is translated.

### C.2.6 TrappingParameters

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	<b>TrappingParams</b>	—

### C.2.7 TrapRegion

PJTF Key or Object	JDF Process	JDF Resource	Description
All keys	—	<b>TrapRegion</b>	—

## C.3 Translating Values

The PJTF version 1.1 specification lists twelve data types that may occur for the values of keys in PJTF objects. The

following table describes how each of these datatypes must be represented in JDF.

PJTF Data Type	JDF Representation	Comment
boolean	boolean	—
Number	double	—
Name	name	—
Dictionary	element	All PJTF objects are dictionaries. These dictionaries generally become resources or processes as specified above. In addition, some PJTF objects contain embedded dictionaries whose keys are not specified (examples include <i>TrappingParameters</i> and <i>ColorantDetails</i> ). These dictionaries are converted to arrays of elements, with the key name from the PJTF dictionary becoming an attribute of the subelement.
Stream	URL	PJTF supports PDF streams by reference to an object in a PDF file. The same mechanism is supported in JDF, with the JDF URL data type being used to identify the PDF file.
Rectangle	rectangle	—
Filespec	URL	—
Text	string	—
String	string	—
Date	date	—
Phone number	Phone number	The standard for the representation of phone numbers in PJTF is used here as well.

## C.4 Translating the Contents Hierarchy

The contents of a PJTF job are represented in the “contents hierarchy”. The hierarchy is headed by the *JobTicketContents* object, with *Document*, *PageRange* and *JTFile* objects occurring below. The hierarchy implicitly specifies the sequence of source pages for the job.

The contents sequence comprises all the pages specified by the first, then second, then last *PageRange* for the first *Document*, followed by the pages specified by the first, then last *PageRange* for the second *Document*, followed by the pages for the first, then last *PageRange* for the last *Document*. This sequence of source pages is consumed when the job is printed via *PrintLayout* (discussed below).

The contents hierarchy must be translated into a JDF **RunList** resource. Each *LayoutElement* entry in a **RunList** can reference a file via the *FileSpec/@URL* attribute and a set of pages in the file via the *Pages* element. There are several additional issues related to this translation which are discussed below.

## C.5 Representing Pages

In PJTF, source pages are represented as a hierarchy of *Document* and *PageRange* objects. Pages are referenced by page number out of files; files are represented in *JTFile* objects. *PageRange* objects can reference a single page, or a set of contiguous pages.

In JDF, source pages are represented as a set of partitions of the **RunList**, which reference files via URL, and pages from the files via an *IntegerRangeList* (such as ‘1 3~5 7~ -1’).

As a consequence of this difference, pages from more than one PJTF *PageRange* object can be represented in a single **RunList** resource, assuming that all the other keys for the multiple *PageRanges* have the same values.

## C.6 Representing Preseparated Documents

In preseparated workflows, all planes of each page may occur in the same file, or there may be a separate file for each plane. When all the planes occur in a single file, PJTF JTFFile objects use a *PlaneOrder* object to specify which pages in the file represent each colorant plane for each source page. When each plane occurs in a separate file, the JTFFile objects use a FilesDictionary to associate files with each colorant.

In JDF, both of these cases are handled through the **RunList** resource. In the case where the planes occur in separate files, the RunList is partitioned; and each partition contains the name of the colorant and the URL for the file for that colorant. In the case where the colorant planes are intermingled via *PlaneOrder* objects, the RunLists are partitioned, but only a single URL is used for each RunList partition. Each *PlaneOrder* object will become one RunList partition.

## C.7 Representing Inherited Characteristics

In PJTF, many of the characteristics of source pages—including *MediaBox*, *ColorantControl*, and *InsertPage*—may occur at all levels of the contents hierarchy. This inheritance scheme is not provided in JDF. Therefore, the correct values for each of the attributes must be translated to the appropriate element for each **RunList** element.

## C.8 Translating Layout

PJTF provides two mechanisms to image a set of source pages onto a larger surface for printing: Layout and PrintLayout. Layout is a mechanism for explicitly associating specific source pages with specific locations on the surface. PrintLayout is a method for automatically positioning a sequence of source pages onto a series of surfaces.

Layout is represented as a hierarchy of PJTF objects: Signatures, Sheets, Surfaces and PlaceObjects. The Layout hierarchy may have one or more Signature objects. Each Signature must have one or more Sheets. Each Sheet must have 1 or 2 Surfaces. Each Surface may have 0 or more PlacedObjects.

PlacedObjects directly reference source pages by referring to a Document object via its Doc key, and a specific page within the sequence of pages specified by all the PageRanges in Pages arrays for that Document.

JDF defines resources which are direct translations of Signature, Sheet and Surface. PlacedObjects and MarkObjects are subelements of the Surface resource. Note: PlacedObjects identify specific source pages via a combination of Ord and either Doc or MarkDoc. Ord identifies one page out of the sequence of pages specified by all the PageRange objects for the document identified by either Doc or MarkDoc.

In the JDF PlacedObject subelement, the Ord attribute is an index into the entire sequence of pages specified by all the partitions with *IsPage = true* in the **RunList**. So there is a translation required between the PJTF Ord value and the JDF Ord attribute.

Similarly, in the JDF MarkObject subelement, the Ord attribute is an index into the entire sequence of pages specified by all the partitions in the **RunList** for marks. So there is a translation required between the PJTF Ord value and the JDF Ord attribute.

## C.9 Translating PrintLayout

PrintLayout uses the same hierarchy of objects as Layout, but with the restriction that there can be only a single Signature. The Signature is used as a template that is repeated to consume all the source pages specified by the contents hierarchy for the job.

In addition, the PlacedObjects that occur in a PrintLayout hierarchy are not references to specific source pages. Instead, they represent the intent that a page from the sequence of source pages specified by the contents hierarchy be consumed and placed onto the Surface each time the Signature is executed.

In JDF, PrintLayout is represented via the same set of resources as Layout, except that the top of the hierarchy is an AutomatedLayout resource instead of Layout. This resource is constrained to have only one Signature resource. Note that when translating PJTF PlacedObjects to PlacedObject subelements of a Surface resource in the AutomatedLayout hierarchy, the Ord values from the PJTF PlacedObjects need not be modified. However, as in the creation of

Layout, the Ord attribute for JDF MarkObject subelements are indices into the entire sequence of pages specified by all the partitions in the **RunList** for marks. So there is a translation required between the PJTF Ord value and the JDF Ord attribute.

## C.10 Translating Trapping

Trapping controls are represented in PJTF as several objects: Trapping, TrappingDetails, ColorantDetails and Device-Colorants; TrappingParameters and ColorantZoneDetails; and TrapRegions. These objects can occur in multiple places in the PJTF job, and they work together to determine, for each page in the job, whether it will be trapped and how. There is also a key in the JobTicketContents object, TrappingSourceSelector, which determines which set of trapping controls will be honored.

The trapping controls in PJTF are the same, whether the trapping will be done pre-RIP or in-RIP. In translating PJTF trapping controls to JDF, there are several tasks to perform:

- Create the required Trapping node
- Add the resources to represent the TrappingParameters which will be used
- Create the resources which represent the TrapRegions which will be used
- Determine the pages to be trapped
- Determine which controls to use for each page
- Add references to the pages in the **RunList** in the TrapRegion resource

Note: The contents hierarchy for the PJTF job must be translated into **RunLists** before trapping objects can be translated. Paths in JDF are specified as a set of path operators. PJTF TrapZone paths are a sequence of coordinates with an implied moveto at the beginning, and an implied closepath the end.





---

## Appendix D Converting PPF to JDF

This appendix gives non-normative advice on how to convert CIP3 PPF 3.0 files to JDF encoded files. Since JDF was designed with the intention of providing the highest possible level of compatibility with PPF, many of these conversions are relatively straightforward. From the point of view of JDF, CIP3's PPF is mainly resource-based. Most of the PPF structures were, therefore, translated to JDF resources of a corresponding process. Meanwhile, the PPF product definition operations are easily translated to JDF processes of the same name, as quoted in **CIP3ProductOperation**. This kind of conversion is possible because the component structure of PPF is adopted by JDF, with some enhancements. Parameters of PPF product definition operations (**CIP3ProductParams**) are given the abbreviated name "Params," and this name is appended to the **CIP3ProductOperation** name. Thus SideSewing becomes SideSewingParams.

In many cases, PPF key names became JDF attribute or element names with the "CIP3" prefix removed. An example of this kind of translation is provided below, and the CIP3 product structure shown in the example is expressed as a JDF process in Figure D.1, following the example.

### Example: A CIP3 PPF product definition operation

```
/CIP3Products [  
<<  
/CIP3ProductName (sewed book block)  
/CIP3ProductOperation /ThreadSewing  
/CIP3ProductParams <<  
/NumberOfNeedles 4  
/GlueLineRefSheets [ 0 ]  
/GlueLine <<  
...  
>>  
/BlindStitch false  
/Sealing false  
>>  
/CIP3ProductComponents  
[  
<<  
/SourceType /PartialProduct  
/SourceProduct (book block)  
...  
>>  
]  
>>  
<<  
/CIP3ProductName (book block)  
% ... the definition of the book block operation would go here ...  
>>  
] def
```

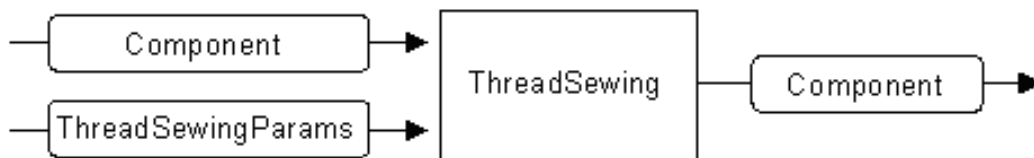


Figure D.1: JDF node of a CIP3 product structure

In Figure D.1, the input **Component** represents the “book block,” the output **Component** represents the “sewed book block,” and **ThreadSewingParams** covers all information of the **CIP3ProductParams** structure. Whenever possible, the formal conversion and translation conventions described above were followed, but because extensions and operations new to PPF are included in JDF, some exceptions were made. These exceptions are explained in detail for each PPF structure in the sections that follow. Before they are explained, however, a translation of PPF data types is provided.

## D.1 Converting PPF Data Types

The following table shows all PPF data types, and how they are transformed. All measuring units of CIP3 must be converted to the JDF native unit point (1/72 inch). Comments are only provided when there is something unusual or noteworthy about the translation; thus, not all translations require comment.

Table D-1: Conversion of PPF Data Types

PPF Data Type	JDF Data Type	Comments
boolean	boolean	—
Integer	integer	—
Real	double	The exponent symbol must be a capital “E” in XML.
Number	double	The exponent symbol must be a capital “E” in XML.
Name	enumeration <i>or</i> NMTOKEN	When PPF Names are used as a closed set of predefined values, they are converted to an enumeration. Otherwise, they are converted to an NMTOKEN.
String	string	Some PostScript string characters cannot be used in XML.
Array	Sequence of elements <i>or</i> IntegerList <i>or</i> DoubleList	If the array consists of homogeneous integers or doubles, it is converted to an IntegerList or DoubleList, otherwise to a sequence of corresponding elements.
Dictionary	element	In most cases, the structure of a Dictionary is directly converted to a XML element. Exceptions to this rule are described in the following sections.

## D.2 PPF Product Definitions

The information stored in **CIP3Products** and **CIP3FinalProducts** is implicitly expressed by the structure of the JDF tree. Each product definition step is converted to a JDF node, and a product node is created for every final product of a PPF file. This is also the case for each partial product that is used in two or more final products. The following table provides information that explains how to accomplish these transformations and make these conversions. The content of the entities **CIP3ProductJobName**, **CIP3ProductJobCode**, **CIP3ProductCopyright** and **CIP3ProductCustomer** must also be copied to the parent product node. The sections that follow contain information about the conversion requirements of prominent postpress processes.

Table D-2: JDF Representation of a product definition step

PPF Key	JDF Representation	Comments
<b>CIP3ProductName</b>	This is expressed by an output resource link.	—
<b>CIP3ProductOperation</b>	JDF node	See Section 3.1, JDF Nodes.
<b>CIP3ProductParams</b>	Resource identified by the name of the JDF node + “Params”	For example, during a <b>CIP3ProductOperation</b> of the type “ <b>SaddleStitching</b> ”, the JDF representation of the <b>CIP3ProductParams</b> is <b>SaddleStitchingParams</b>
<b>CIP3ProductComponent</b>	<b>Component</b>	See Section D.2.1, Comparison of the PPF Component to the JDF Component, below

Table D-2: JDF Representation of a product definition step

<b>CIP3ProductJobName</b>	Comment element of the JDF node	—
<b>CIP3ProductJobCode</b>	<i>JobID</i> or <i>JobPartID</i> attribute of the JDF node	If the output of this step is a final product and it is only final product, it should be converted into <i>JobID</i> of the root node. Otherwise, it is converted into a <i>JobPartID</i> of the corresponding process node.
<b>CIP3ProductCopyright</b>	Comment element of the JDF node	—
<b>CIP3ProductCustomer</b>	CustomerInfo element of the JDF node	Note that the CustomerInfo element is structured, while the <b>CIP3ProductCustomer</b> is not.
<b>CIP3ProductVolume</b>	<i>Amount</i> attribute of the output <b>Component</b> resource link	—

## D.2.1 Comparison of the PPF Component to the JDF Component

The structure of the PPF **Component** is very similar to the structure of the JDF **Component**, so it is easy to convert one to the other. The following table gives advice on how to do this. Some information stored in the PPF **Component** must be used for linking the correct resources to a process. Other implicit information, such as the bounding box of the component or an overfold, must be calculated and explicitly specified in the subelements of the **Component**. Furthermore, the appropriate algorithms can be very complex for some operations, such as folding. For further information about the **Component** resource, see Section 7.2.33, Component.

Table D-3: Converting a PPF Component

PPF Key	JDF Representation	Comments
<b>SourceType</b>	<i>ComponentType</i> attribute of <b>Component</b>	—
<b>SourceSheet</b>	<i>SourceSheet</i> attribute of <b>Component</b>	—
—	<i>SheetPart</i> attribute of <b>Component</b>	Calculable out of the cut block structure.
<b>SourceBlock</b>	Expressed by an input resource link to an output <b>Component</b> of a previous <b>Cutting</b> process.	see Section D.3.6, Cutting Data
<b>SourceProduct</b>	Expressed by an input resource link to a <b>Component</b> .	—
<b>Params</b>	<i>Transformation</i> attribute of <b>Component</b>	In most CIP3 operations, there is only one component parameter called <b>Orientation</b> . This matrix is renamed to <i>Transformation</i> . The only exception is the <b>EndSheetGluing</b> process. See <b>EndSheetGluing</b> for more information.

## D.2.2 Collecting

To convert a **Collection** operation, follow the previous descriptions. This process contains no special considerations to take into account.

## D.2.3 Gathering

To convert a **Gathering** operation, follow the previous descriptions. This process contains no special considerations to take into account.

### D.2.4 ThreadSewing

Convert the entries of **CIP3ProductParams** structure directly to the **ThreadSewingParams** resource. Add this resource as an input resource link to the originated **ThreadSewing** process. See Section 7.2.160, **ThreadSewingParams** for more information.

### D.2.5 SaddleStitching

Convert the entries of **CIP3ProductParams** structure directly to the **StitchingParams** resource. Set *StitchType* = “*Saddle*”. Add this resource as an input resource link to the originated **Stitching** process. See **Stitching** for more information.

### D.2.6 Stitching

Convert the entries of **CIP3ProductParams** structure directly to the **StitchingParams** resource. Set *StitchType* = “*Side*”. Add this resource as an input resource link to the originated **Stitching** process. See **Stitching** for more information.

### D.2.7 SideSewing

Convert the entries of **CIP3ProductParams** structure directly to the **ThreadSewingParams** resource. Add this resource as an input resource link to the originated **ThreadSewing** process. See **ThreadSewing** for more information.

### D.2.8 EndSheetGluing

The **EndSheetGluing** CIP3 operation is the only operation that requires more information than **Orientation** in the PPF Component **Params**. This additional information of the front and the back end sheet components is transferred to the **EndSheetGluingParams** resource, as described in the following table. See Section 7.2.62, **EndSheetGluingParams** for more information.

Table D-4: Converting the PPF EndSheetGluing operation to JDF

PPF Key	JDF Representation	Comments
<b>Offset</b>	<i>Offset</i> attribute of the EndSheet element of <b>EndSheetGluingParams</b>	—
<b>GlueLine</b>	GlueLine element of the EndSheet element of <b>EndSheetGluingParams</b>	See Section 7.2.62, <b>EndSheetGluingParams</b> for information on how to convert the <b>GlueLine</b> structure.

### D.2.9 AdhesiveBinding

The PPF main adhesive binding operation dictionary is translated to the **AdhesiveBindingParams** resource. All single suboperations that were resident in the PPF **Processes** array are converted to special elements inside the **AdhesiveBindingParams** (see Section 7.2.3, **AdhesiveBindingParams**). For each type of adhesive binding suboperation there exists one extra element. The suboperations **SpinePreparation** and **GlueApplication** can simply be translated by removing the **ProcessType** entry and converting all other entries directly to the appropriate element.

The following tables show how to convert the main operation and its other suboperations. Because new features were added, the CIP3 **Lining** operation was renamed to **SpineTaping**.

Table D-5: Converting the PPF AdhesiveBinding operation to JDF

PPF Key	JDF Representation	Comments
<b>Processes</b>	Several single process:	See description above.
• <b>BackPreparation</b>	<b>SpinePreparation</b>	
• <b>GlueApplication</b>	<b>Gluing</b>	
• <b>Lining</b>	<b>SpineTaping</b>	
• <b>CoverApplication</b>	<b>CoverApplication</b>	

Table D-5: Converting the PPF AdhesiveBinding operation to JDF

<b>PullOutValue</b>	<i>PullOutValue</i> attribute of all <b>SpinePreparationParams</b> resources, which are part of the <b>AdhesiveBinding</b> process chain.	—
<b>PullOutMake</b>	—	Not needed.
<b>FlexValue</b>	<i>FlexValue</i> attribute of <b>AdhesiveBindingParams</b>	—
<b>FlexMake</b>	—	Not needed.

The following tables show how to convert the main operation and its other sub-operations. Because new features were added, the CIP3 **Lining** operation was renamed to **SpineTaping**. Convert the PPF AdhesiveBinding sub-operation Lining to a **SpineTaping** process. Copy the parameters of the sub-operation to the equivalent attributes of the **SpineTapingParams** resource and link them with the process.

Table D-6: Converting the PPF AdhesiveBinding suboperation Lining

PPF Key	JDF Representation	Comments
<b>ProcessType</b>	Name of the JDF process.	
<b>TopLiningExcess</b>	<i>TopExcess</i> attribute of <b>SpineTapingParams</b>	—
<b>LiningExcess</b>	<i>HorizontalExcess</i> attribute of <b>SpineTapingParams</b>	—
<b>LiningLength</b>	<i>StripLength</i> attribute of <b>SpineTapingParams</b>	—
<b>LiningMaterial</b>	<i>StripMaterial</i> attribute of <b>SpineTapingParams</b>	—
<b>LiningBrand</b>	<i>StripBrand</i> attribute of <b>SpineTapingParams</b>	—

Table D-7: Converting the PPF AdhesiveBinding suboperation CoverApplication

PPF Key	JDF Representation	Comments
<b>ProcessType</b>	—	There is an extra element for each type of <b>AdhesiveBinding</b> suboperation.
<b>CoverOffset</b>	<i>CoverOffset</i> attribute of <b>CoverApplication</b>	—
<b>ScoringOffsets</b> and <b>ScoringSide</b>	Several <b>Score</b> elements inside of <b>CoverApplication</b>	The <b>Score</b> element is much more structured than these two single entries.

## D.2.10 Trimming

Convert the entries of **CIP3ProductParams** structure directly to the **TrimmingParams** resource. Add this resource as an input resource link to the originated **Trimming** process. See Section 6.6.48.9, **Trimming** for more information.

## D.2.11 GluingIn

Because extended features have been added, the PPF **GluingIn** operation was renamed to the **Inserting** process. Consequently, the parameters of this CIP3 operation are transformed into the **InsertingParams** resource. For more information see Section 7.2.89, **InsertingParams**.

Table D-8: Converting the PPF GluingIn operation to JDF

PPF Key	JDF Representation	Comments
<b>SheetOffset</b>	<i>SheetOffset</i> attribute of <b>InsertingParams</b>	—
—	<i>Location</i> attribute of <b>InsertingParams</b>	Must be <i>Front</i>

Table D-8: Converting the PPF GluingIn operation to JDF

<b>GlueLines</b>	Several GlueLine elements in <b>InsertingParams</b>	See Section 7.2.89, InsertingParams for information on how to convert the GlueLine structure.
<b>Sample</b>	Comment of the corresponding <b>Component</b>	Converted to an input <b>Component</b> of Type <i>PartialProduct</i>

Most of the entries of the PPF **GlueLine** structure can be directly mapped to the **GlueLine** element. Note that the *GluingPattern* attribute cannot have an empty array to describe a solid glue line. For this purpose, use an array of “1 0”.

## D.2.12 Folding

Like all formats, JDF follows a structured approach in the description of the folding process. That is why every sub-operation has its own element type and has no need of the function entry. Normally, the names of the CIP3 fold functions was taken for the name of the respective corresponding process names. One of the specialized processes ...

- **Folding**,
- **Creasing**,
- **Cutting**,
- **Perforating**, and
- **Gluing** ...

is created for each folding sub-operation.

Because of inherent naming obscurities, the CIP3 functions **Groove** and **Lime** were renamed to **Crease** and **Gluing** in JDF. The following tables give advice on how to convert the PPF structures to JDF elements.

Table D-9: Converting the PPF Folding operation to JDF

PPF Key	JDF Representation	Comments
<b>CIP3FoldDescription</b>	—	If required, it can be expressed by the <i>FoldCatalog</i> attribute or by the fold operations.
<b>CIP3FoldSheetIn</b>	—	In CIP3 the parameters of the folding procedure will be scaled, if the value of the CIP3FoldSheetIn array is different from the dimension of the input component. In JDF a scaling mechanism is not supported.
<b>CIP3FoldProc</b>	Several processes: <b>Folding</b> <b>Gluing</b> <b>Cutting</b> <b>Creasing</b> <b>Perforating</b>	See previous description
<ul style="list-style-type: none"> <li>• <b>Fold</b></li> <li>• <b>Lime</b></li> <li>• <b>Cut</b></li> <li>• <b>Groove</b></li> <li>• <b>Perforate</b></li> </ul>		

The PPF Folding suboperation is translated to a **Folding** process. The parameters of the PPF command are copied into a **Fold** element inside the **FoldingParams** resource. The table below shows how to assign the parameters of the PPF Fold command to the equivalent attributes inside the **Fold** element.

Table D-10: Converting the PPF Folding suboperation of type Fold

PPF Key	JDF Representation	Comments
<b>travel</b>	<i>Travel</i> attribute of <b>Fold</b>	—
<b>from</b>	<i>From</i> attribute of <b>Fold</b>	—

Table D-10: Converting the PPF Folding suboperation of type Fold

<b>to</b>	<i>To</i> attribute of Fold	—
<b>function</b>	—	—

For every lime operation, a **Gluing** process is generated. Create a **GluingParams** resource and add a Glue element. Insert the value of the working-direction attribute into the *WorkingDirection* attribute. Attach a GlueApplication element. To this element add a GlueLine element. The attributes start-position and working-path can put into the equivalent attributes *StartPosition* and *WorkingPath* inside the GlueLine.

Table D-11: Converting the PPF Folding suboperation of type Lime

PPF Key	JDF Representation	Comments
<b>start-position</b>	<i>StartPosition</i> attribute of the GlueLine element of the Gluing element	JDF uses the GlueLine element because of the advantage of more optional attributes of this type of element.
<b>working-path</b>	<i>WorkingPath</i> attribute of the GlueLine element of the Gluing element	JDF uses the GlueLine element because of the advantage of more optional attributes of this type of element.
<b>working-direction</b>	<i>WorkingDirection</i> attribute of the Gluing element	—
<b>function</b>	—	—

The remaining operation types can converted to one of the following processes:

- **Cutting**. Create a **CuttingParams** resource and link it to the process. Transfer the parameters of the PPF Cut command into equivalent attributes of a Cut element and insert this into the **CuttingParams** resource.
- **Creasing**. The same as above except that there is a **CreasingParams** resource with a Crease element inside which will fill with the converted parameters of the PPF Groove command.
- **Perforate**. The same as above except that there is a **PerforatingParams** resource with a Perforate element inside which will fill with the converted parameters of the PPF Perforate command.

Table D-12: Converting the PPF Folding suboperation of all other types

PPF Key	JDF Representation	Comments
<b>start-position</b>	<i>StartPosition</i> attribute of the respective Cut / Crease / Perforate element	—
<b>working-path</b>	<i>WorkingPath</i> attribute of the respective Cut / Crease / Perforate element	—
<b>working-direction</b>	<i>WorkingDirection</i> attribute of the respective Cut / Crease / Perforate element	—
<b>function</b>	—	There is an extra element for each type of a <b>Folding</b> suboperation. The extra elements are: Cut, Crease, and Perforate

### D.3 PPF Sheet Structure

The conversion of the PPF sheet structures is much more complex than the conversion of the product operations. A JDF layout structure, which is not directly specified in PPF, must be built up in order to place the mark objects such as register mark or density measuring field. All other sheet information is stored in specialized resources. These resources are often partitionable to specify the sheet, surface and separation to which they belong (see Section 3.8.2, Description of Partitionable Resources). The result is an inheritance of attributes comparable to the inheritance process in CIP3.

To build the layout structure, create a **Layout** resource that includes one **Signature** element with a unique *Name*. For each PPF **Sheet**, add one **Sheet** resource to the **Signature**. Set the *Name* of the corresponding **Sheet** to the value of **CIP3AdmSheetName**. For each surface (front or back) initiate a **Surface** resource with one **PlacedObjects** element. In order to define a mark object, (i.e., **CutMark**, **CIELABMeasuringField**, **DensityMeasuringField**, **ColorControlStrip**, or **RegisterMark**), build a **MarkObject** element inside **PlacedObjects**. In that element, define *CTM* and an appropriate *LayoutElement*. The CIP3 information is added to the **MarkObject** by including the mark-specific element, (e.g., **RegisterMark** for a register mark). Note: The coordinate system of the JDF **Sheet** is specified by the *SurfaceContentsBox*, which defaults to the page coordinates and the coordinate system of the CIP3 **Sheet** is the PSExtent coordinates.

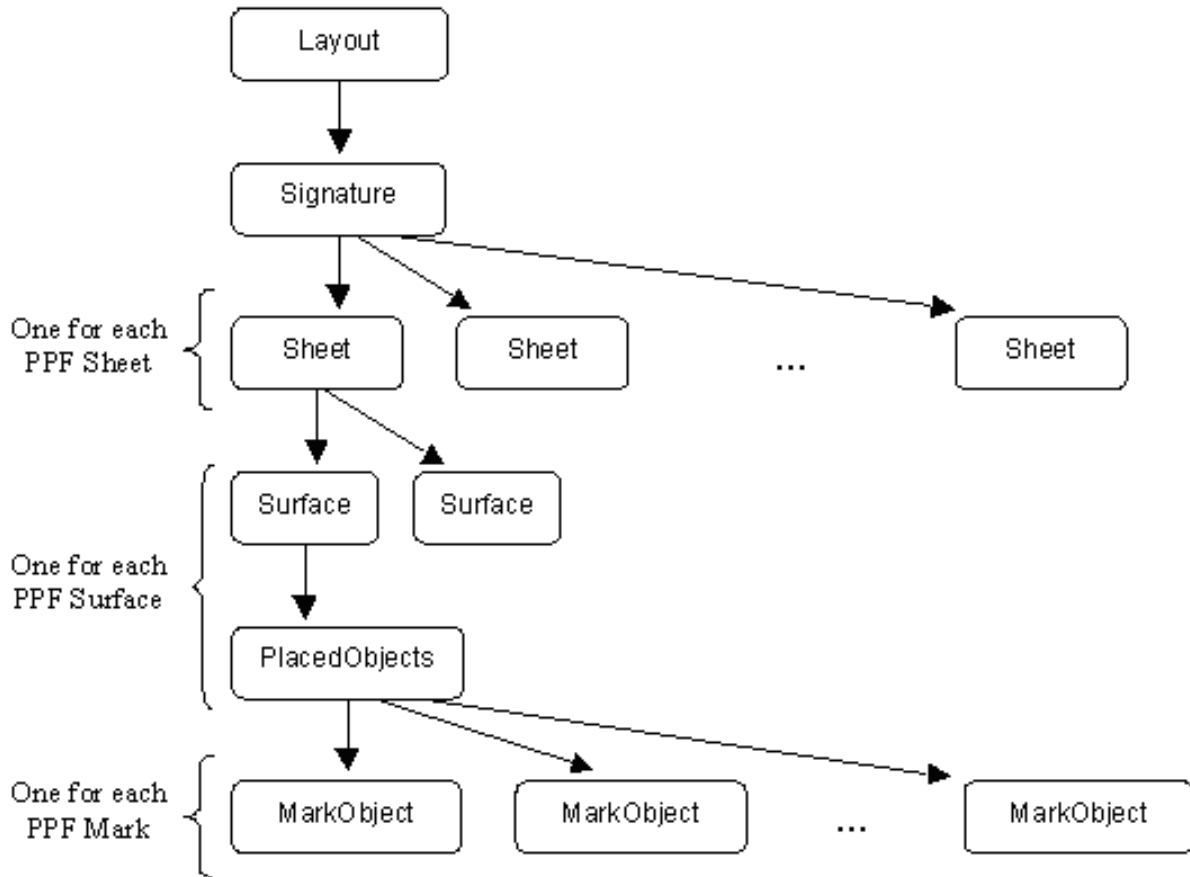


Figure D.2: JDF representation of sheets

If there are no product definitions in the PPF file, create JDF product nodes which are the results of all cutting and folding information in the sheet structure.



### D.3.1 Administration Data

The following table defines how to convert the administration data of CIP3. In some situations, it may not be clear whether or not conversion is necessary. Processes such as **CIP3AdmFilmType**, for example, contain limited information, making it difficult to tell.

Table D-13: Converting administration data

PPF Key	JDF Representation	Comments
<b>CIP3AdmSheetName</b>	<i>Name</i> attribute of the corresponding <b>Sheet</b>	If there is no <b>CIP3AdmSheetName</b> , define a unique new one.
<b>CIP3AdmJobName</b>	Comment of the corresponding product node	—
<b>CIP3AdmJobCode</b>	<i>JobPart</i> of the corresponding product node	May conflict with <b>CIP3ProductJobCode</b> .
<b>CIP3AdmMake</b>	—	Not supported.
<b>CIP3AdmModel</b>	—	Not supported.
<b>CIP3AdmSoftware</b>	—	Not supported.
<b>CIP3AdmCreationTime</b>	—	Not supported.
<b>CIP3AdmArtist</b>	Comment of the corresponding product node	—
<b>CIP3AdmCopyright</b>	Comment of the corresponding product node	—
<b>CIP3AdmCustomer</b>	<b>CustomerInfo</b> element of the corresponding product node	May conflict with <b>CIP3ProductCustomer</b> . Note: The <b>CustomerInfo</b> element is structured while the <b>CIP3AdmCustomer</b> is not.
<b>CIP3AdmPSExtent</b>	indirect	—
<b>CIP3AdmTypeOfScreen</b>	see description	Not possible to convert appropriately.
<b>CIP3AdmFilmType</b>	<i>Brand</i> attribute of the corresponding resource	<i>MediaType</i> of the <b>Media</b> is <i>Film</i> .
<b>CIP3AdmFilmExtent</b>	<i>Dimension</i> attribute of the corresponding resource	—
<b>CIP3AdmFilmTrf</b>	<b>TransferCurveSet/@CTM</b>	<b>TransferCurveSet/@Name</b> = "Film"
<b>CIP3AdmPlateType</b>	<i>Brand</i> attribute of the corresponding resource	<i>MediaType</i> of the <b>Media</b> is <i>Plate</i> .
<b>CIP3AdmPlateExtent</b>	<i>Dimension</i> attribute of the corresponding resource	—
<b>CIP3AdmPlateTrf</b>	<b>TransferCurveSet/@CTM</b>	<b>TransferCurveSet/@Name</b> = "Plate"
<b>CIP3AdmPaperGrade</b>	<i>Grade</i> attribute of the corresponding resource	<i>MediaType</i> of the <b>Media</b> is <i>Paper</i>
<b>CIP3AdmPaperGrammage</b>	<i>Weight</i> attribute of the corresponding resource	See <b>CIP3AdmPaperGrade</b> .
<b>CIP3AdmPaperThickness</b>	<i>Thickness</i> attribute of the corresponding resource	See <b>CIP3AdmPaperGrade</b> .
<b>CIP3AdmPaperColor</b>	<i>Lab</i> attribute of the <b>Color</b> element of the corresponding resource	See <b>CIP3AdmPaperGrade</b> .
<b>CIP3AdmPaperExtent</b>	<i>Dimension</i> attribute of the corresponding resource	—
<b>CIP3AdmPaperTrf</b>	<b>TransferCurveSet/@CTM</b>	<b>TransferCurveSet/@Name</b> = "Paper"

Table D-13: Converting administration data

<b>CIP3AdmSeparationNames</b>	see description	Create a <b>ConventionalPrinting</b> process (see Section 6.5.1, ConventionalPrinting) and a corresponding <b>ColorantControl</b> resource. Fill the <i>ColorantOrder</i> parameter.
<b>CIP3AdmSheetLay</b>	<i>SheetLay</i> attribute of the corresponding <b>ConventionalPrintingParams</b> or <b>FoldingParams</b> resource	—
<b>CIP3AdmPrintVolume</b>	<i>Amount</i> attribute of the output <b>Component</b> resource link of the printing process	—
<b>CIP3AdmPressTrf</b>	TransferCurveSet/@CTM	TransferCurveSet/@Name = “Press”
<b>CIP3AdmPressExtent</b>	indirect	—
<b>CIP3AdmInkInfo</b>	<i>Name</i> attribute of the Color element of the corresponding resource. The value of <b>Ink/@ColorName</b> color should match the <i>Name</i> attribute of a Color defined in a <b>ColorPool</b> resource that is linked to the process that is using this <b>Ink</b> resource. This does not refer to the A.2.8 NamedColor attribute.	Create a partitioned <b>Ink</b> matching the side and separation. Add the <b>Ink</b> to the <b>ConventionalPrinting</b> process of <b>CIP3AdmSeparationNames</b>
<b>CIP3AdmInkColors</b>	<i>LabColor</i> attribute of the Color element defined by the <i>ColorName</i> of the <b>Ink</b> resource. The value of <b>Ink/@ColorName</b> color should match the <i>Name</i> attribute of a Color defined in a <b>ColorPool</b> resource that is linked to the process that is using this <b>Ink</b> resource. <i>LabColor</i> defines values for that Color/@Name.	see CIP3AdmInkInfo

### D.3.2 Preview Images

In PPF, preview images are coded as an in-line image. This is not possible in version 1.0 of XML, so JDF uses the *URL* attribute within the **Preview** resource (see Section 7.2.125, Preview), which points to an external PNG file. The following table shows how to translate the PPF preview structure to the PNG header. Use the partition feature to assign a preview image to a specific separation and surface.

Table D-14: PPF preview representation as PNG

PPF Key	JDF Representation	Comments
<b>CIP3PreviewImageWidth</b>	“Width” of the “IHDR” chunk of the PNG file	—
<b>CIP3PreviewImageHeight</b>	“Height” of the “IHDR” chunk of the PNG file	—
<b>CIP3PreviewImageBitsPerComp</b>	“Bit depth” of the “IHDR” chunk of the PNG file	—

Table D-14: PPF preview representation as PNG

<b>CIP3PreviewImageComponents</b>	—	Because of a lack of CMYK composite support by PNG, PPF previews of this type must be separated.
<b>CIP3PreviewImageImageMatrix</b>	—	Not needed. Convert image data to the PNG native sequence.
<b>CIP3PreviewImageResolution</b>	“pHYs” chunk of the PNG file	Use the meter unit and convert DPI to DPM.
<b>CIP3PreviewImageEncoding</b>	—	Not needed.
<b>CIP3PreviewImageCompression</b>	—	Not needed. Use PNG’s own compression.
<b>CIP3PreviewImageFilterDict</b>	—	Not needed.
<b>CIP3PreviewImageByteAlign</b>	—	Not needed.
<b>CIP3PreviewImageDataSize</b>	—	Not needed.

To calculate ink zones, JDF uses a process chain of *PreviewGeneration* and *InkZoneCalculation* processes. Add the converted CIP3 previews as an input resource to *InkZoneCalculation*. The *ProfileOffset* attribute of *InkZoneCalculationParams* can be calculated out of the different CIP3 coordinate systems.

### D.3.3 Transfer Curves

Simply convert all CIP3 transfer curves to elements of a partitioned **TransferCurvePool** (see Section 7.2.161, Tile). Add this **TransferCurvePool** as an input resource to a corresponding *InkZoneCalculation* process.

### D.3.4 Register Marks

The table provides information about how to create a JDF **RegisterMark** and place this element inside the respective MarkObject.

Table D-15: Converting the parameter of the CIP3PlaceRegisterMark command

PPF Key	JDF Representation	Comments
<b>translate-x</b> and <b>translate-y</b>	<i>Center</i> attribute of <b>RegisterMark</b>	Apply all transformations of the CIP3 coordinate systems to get from the PS system to the <b>Layout</b> system.
<b>rotation</b>	<i>Rotation</i> attribute of <b>RegisterMark</b>	—
<b>type</b>	<i>MarkType</i> attribute of <b>RegisterMark</b>	—
Current <b>CIP3SetRegisterMarkSeparations</b> context	Several <i>SeparationSpec</i> elements inside the <b>RegisterMark</b>	—

### D.3.5 Color and Ink Control

In CIP3, the two types of measuring fields are specified by an entry of the data dictionary in the **CIP3PlaceMeasuringField** command. In JDF, this approach is replaced by two different types of JDF elements: *CIELABMeasuringField* and *DensityMeasuringField*. All parameters of the **CIP3PlaceMeasuringField** command are merged into these elements. See the following tables as well as Section 7.2.19, *CIELABMeasuringField* and Section 7.2.48, *DensityMeasuringField* for further information. All PPF entries that are

not explicitly listed in the following tables can be directly converted. Place the originated element inside the appropriate MarkObject.

Table D-16: Converting PPF color-measuring data

PPF Key	JDF Representation	Comments
<b>position-x</b> and <b>position-y</b> of the respective <b>CIP3PlaceMeasuringField</b> command	<i>Center</i> attribute of <b>CIELABMeasuringField</b>	Apply all transformations of the CIP3 coordinate systems to get from the PS system to the <b>Layout</b> system.
<b>Type</b>	—	There is an extra resource for each type of CIP3 measuring field.
<b>CIE-L*</b> , <b>CIE-a*</b> and <b>CIE-b*</b>	<i>CIELab</i> attribute of <b>CIELABMeasuringField</b>	—

Table D-17: Converting PPF density-measuring data

PPF Key	JDF Representation	Comments
<b>position-x</b> and <b>position-y</b> of the respective <b>CIP3PlaceMeasuringField</b> command	<i>Center</i> attribute of <b>DensityMeasuringField</b>	Apply all transformations of the CIP3 coordinate systems to get from the PS system to the <b>Layout</b> system.
<b>Type</b>	—	There is an extra resource for each type of CIP3 measuring field.
<b>DensityCyan</b> , <b>DensityMagenta</b> , <b>DensityYellow</b> and <b>DensityBlack</b>	<i>Density</i> attribute of <b>DensityMeasuringField</b>	—

Like the measuring fields, the **CIP3PlaceColorControlStrip** command is translated to a structured element. All parameters of this command can be converted to the **ColorControlStrip** element (see Section 7.2.25, **ColorControlStrip**) by following the instructions in table D.18, below.

Table D-18: Converting the parameter of the CIP3PlaceColorControlStrip command

PPF Key	JDF Representation	Comments
<b>position-x</b> and <b>position-y</b>	<i>Center</i> attribute of <b>ColorControlStrip</b>	Apply all transformations of the CIP3 coordinate systems to get from the PS system to the <b>Layout</b> system.
<b>rotation</b>	<i>Rotation</i> attribute of <b>ColorControlStrip</b>	—
<b>width</b> and <b>height</b>	<i>Size</i> attribute of <b>ColorControlStrip</b>	—
<b>data</b>	Sequence of <b>DensityMeasuringField</b> elements within the <b>ColorControlStrip</b>	The entries of the <b>data</b> parameter have to be converted to <b>DensityMeasuringField</b> elements.
<b>name</b>	<i>StripType</i> attribute of <b>ColorControlStrip</b>	—

### D.3.6 Cutting Data

CIP3's cut block structure is translated to JDF by defining **Cutting** processes. Since CIP3 has the ability to create nested cut blocks, one separate **Cutting** process is needed for each nested block set. Simply follow the instructions

in the following table, and add all originated **CutBlock** resources as input the corresponding **Cutting** process. The **CIP3CutModel** entry is not used in JDF.

Table D-19: Converting the Cutting Data structure

PPF Key	JDF Representation	Comments
<b>CIP3BlockTrf</b>	<i>BlockTrf</i> attribute of <b>CutBlock</b>	If the <b>CutBlock</b> is at the uppermost level, apply all transformations of the CIP3 coordinate systems to get from the PS system to the <b>Layout</b> system.
<b>CIP3BlockSize</b>	<i>BlockSize</i> attribute of <b>CutBlock</b>	—
<b>CIP3BlockElementSize</b>	<i>BlockElementSize</i> attribute of <b>CutBlock</b>	—
<b>CIP3BlockSubdivision</b>	<i>BlockSubdivision</i> attribute of <b>CutBlock</b>	Determines how many <b>Components</b> are produced.
<b>CIP3BlockType</b>	<i>BlockType</i> attribute of <b>CutBlock</b>	—
<b>CIP3BlockElementType</b>	<i>BlockElementType</i> attribute of <b>CutBlock</b>	—
<b>CIP3BlockName</b>	This is expressed by resource links	Not needed in JDF.
<b>CIP3BlockFoldingProcedure</b>	A <b>Folding</b> process	See Section 6.6.17, Folding

For cut marks, follow the instructions in the table below. Place the originated element inside the appropriate MarkObject.

Table D-20: Converting the parameter of the CIP3PlaceCutMark command

PPF Key	JDF Representation	Comments
<b>position-x</b> and <b>position-y</b>	<i>Center</i> attribute of <b>CutMark</b>	Apply all transformations of the CIP3 coordinate systems to get from the PS system to the <b>Layout</b> system.
<b>mark-type</b>	<i>MarkType</i> attribute of <b>CutMark</b>	—

### D.3.7 Folding Data

When a CIP3 cut block has a folding operation defined (**CIP3BlockFoldingProcedure**), append a JDF **Folding** process which uses the respective output **Component** of the respective **Cutting** process as an input **Component**. See Section 6.6.17, Folding for more information on how to translate the CIP3 folding procedure, which is used to fold the cut block.

### D.3.8 Comments and Annotations

PPF comments can either be converted to an XML comment or to a human-readable form by transforming them into a **Comment** telem of the next element. In most cases, PPF comments can simply be ignored. Annotations are not supported by JDF.

### D.3.9 Private Data and Content

For your private data, you should first examine if one of the new JDF elements or attributes fits your requirements. If not, please use the extension capabilities of JDF to express your needs. They are described in Section 3.10, JDF Extensibility.



---

# Appendix E Modeling IfraTrack in JDF

## Introduction

Job tracking and production control are integral parts of a workflow system. IFRA, described in this section, has defined a job tracking system called IfraTrack that fulfills a large number of the job tracking requirements of a production scenario and is especially effective in newspaper production. The JDF messaging system generalizes the IfraTrack approach, expanding its focus from a newspaper workflow to one that encompasses the entire graphic arts industry. This appendix provides further detail about the way in which JDF expands upon the existing IfraTrack technology.

## E.1 IFRA Objects and JDF Nodes

IfraTrack traces the status of objects, and these objects are modified by processes that are only generic. JDF, on the other hand, precisely defines process nodes that create output resources. These JDF output resources are equivalent to IfraTrack objects, so tracking the state of a JDF node conveys a superset of the information communicated by tracking the state of an IfraTrack. The sections that follow define the mapping of IFRA concepts to JDF concepts in greater detail.

### E.1.1 Object Identification

IfraTrack defines objects with an object path. The object path, in turn, may be a unique identifier, or UID. JDF also supports UIDs for internal linking of objects, although these UIDs should not be exported beyond the scope of a JDF document. External references to JDF nodes should be made the JobID/JobPartID pair. These values may be defined by an external system, such as MIS, and can be used to uniquely track JDF nodes.

### E.1.2 IFRA Object Hierarchy

IfraTrack defines an explicit hierarchy to define a newspaper, from Issue through Edition, EditionVersion, and so on. JDF, on the other hand, defines a generic hierarchy of products containing a description attribute that allows the products to be named. An IfraTrack-conforming JDF job consequently includes a product hierarchy with product nodes that contain the appropriate description fields. Furthermore, the abstract IFRA Element type is mapped to the JDF **LayoutElement** type.

### E.1.3 Object States

IFRA defines object states that define the status of a resource, although they also define the status of the process that defines a resource. JDF defines explicit states for both processes and resources. In addition, JDF defines a descriptive string to denote the details of each status. The mapping is defined in the following table.

Table E-1: IFRA object states

IFRA Object Status	JDF Node Status	JDF Resource Status	Description
<i>Not Started</i>	<i>Waiting</i>	<i>Unavailable</i>	Status prior to <i>InProgress</i> .
	<i>Ready</i>	<i>Unavailable</i>	JDF defines a test-run mode that allows generalized preflighting. <i>Ready</i> is the status after <i>TestRun</i> .
<i>In Progress</i>	<i>Setup</i>	<i>Unavailable</i>	A process is <i>InProgress</i> but not yet producing any output.
	<i>InProgress</i>	<i>Unavailable</i>	A process is <i>InProgress</i> .
	<i>Cleanup</i>	<i>Available</i>	A process is running after all output has been produced.
<i>On Hold</i>	<i>Stopped</i>	<i>Unavailable</i>	A process is active but not currently producing, as when maintenance is run during a job.
<i>Completed</i>	<i>Completed</i>	<i>Available</i>	Completed
<i>Aborted</i>	<i>Aborted</i>	<i>Unavailable</i> <sup>a</sup>	Fatal Error

- a. Unless aborted during cleanup

### **E.1.4 Deadlines and Scheduling**

In IfraTrack, activities may be linked to deadlines. JDF defines deadlines in the `NodeInfo` element of every node. The definition of deadline values is identical.

IFRA defines an integer value for deadline level. JDF defines four explicit enumerations for *DueLevel* in order to assure that devices in a heterogeneous system have the same concept of deadline level.

## **E.2 JMF Messages that Translate IfraTrack Messages**

The messages explained in Section 5.5.2, *Device/Operator Status and Job Progress Messages* can be used to emulate IfraTrack functionality. Specifically the messages:

- Section 5.5.2.8, *Status*
- Section 5.5.2.9, *Track*



---

## Appendix F Mapping between JDF and IPP

The mapping between JDF and IPP is specified in Appendix F in JDF/1.0 using the IDPrinting process. However, for JDF 1.1, the IDPrinting process is deprecated. Thus for JDF 1.1 and beyond, mapping between JDF and IPP should be done with the *DigitalPrinting* process and other JDF processes as a combined process node.

### F.1 IPP References

The documents below give detailed information about IPP attributes.

- IPP Model and Semantics, RFC 2911, September 2000
- Internet Printing Protocol (IPP): The “collection” attribute syntax, RFC 3382, September 2002.
- Production Printing Attributes - Set1, IEEE-ISTO 5100.3-2001, <ftp://ftp.pwg.org/pub/pwg/standards/pwg5100.3.pdf>, .doc, .rtf, February 17, 2001
- Override Attributes for Documents and Pages, IEEE-ISTO 5100.4-2001, <ftp://ftp.pwg.org/pub/pwg/standards/pwg5100.4.pdf>, .doc, .rtf, February 7, 2001
- IPP/1.0 & 1.1: “Output-bin” attribute extension, IEEE-ISTO 5100.2-2001, <ftp://ftp.pwg.org/pub/pwg/standards/pwg5100.2.pdf>, .doc, .rtf, February 7, 2001
- IPP/1.1: finishings attribute values extension, IEEE-ISTO 5100.1-2001, <ftp://ftp.pwg.org/pub/pwg/standards/pwg5100.1.pdf>, .doc, .rtf, February 5, 2001
- Internet Printing Protocol (IPP): Job Progress Attributes, RFC 3381, September 2001.



## Appendix G 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*. *StatusDetails* values are never localized and not intended for direct GUI display. Note that *JDF/@Status* = "Setup", "Cleanup" and "Stopped" may include the description of a device with no job assigned to it.

Table G-1: StatusDetails and Status mapping for generic devices

StatusDetails	JDF/ @Status	DeviceS tatus	Description
<i>BreakDown</i>	<i>Stopped</i>	<i>Down</i>	Breakdown of the device, repair required.
<i>Calibrating</i>	<i>Setup</i>	<i>Setup</i>	The Device is calibrating, either manually or automatically.
<i>ControlDeferred</i>	__ <sup>a</sup>	<i>Stopped</i>	The device is controlled by a master device and cannot be accessed.
<i>Failure</i>	<i>Stopped</i>	<i>Stopped</i>	Failure of the device. Requires some maintenance in order to restart the device.
<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>Maintenance</i>	<i>Stopped</i>	<i>Stopped</i>	Maintenance of the device:
<i>MissResources</i>	<i>Stopped</i>	<i>Stopped</i>	Production has been stopped because resources are missing or unavailable. Waits for new resources, subterm of Pause.
<i>PaperJam</i>	<i>Stopped</i>	<i>Stopped</i>	Media jam in the device, subterm of Failure.
<i>Pause</i>	<i>Stopped</i>	<i>Stopped</i>	Machine paused, restart is possible:
<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 (may 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 required approval is still missing, subterm of Pause.
<i>WarmingUp</i>	<i>Setup</i>	<i>Setup</i>	Device is warming up after power up or power saver mode wakeup.
<i>Waste</i>	<i>InProgress</i>	<i>Running</i>	Production of products in progress, good copy counter is off, waste copy counter is on.
<i>WasteFull</i>	<i>Stopped</i>	<i>Stopped</i>	The Device waste receptacle is full.

a. "—" means that the *JDF/@Status* is unknown if the device is not accessible.

Table G-2: Printing Device specific StatusDetails

StatusDetails	JDF/ @Status	DeviceStatus	Description
<i>BlanketChange</i>	<i>Stopped</i>	<i>Stopped</i>	Changing of blankets, subterm (e.g., a 'specialization') for <i>Maintenance</i> .
<i>BlanketWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of the blanket, subterm of <i>WashUp</i> .
<i>CleaningInkFountain</i>	<i>Cleanup</i>	<i>Cleanup</i>	Cleaning of the ink fountain, subterm of <i>WashUp</i> .
<i>CylinderWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of impression cylinders, subterm of <i>WashUp</i> .

Table G-2: Printing Device specific StatusDetails

StatusDetails	JDF/ @Status	DeviceStatus	Description
<i>DampeningRollerWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of the dampening roller, subterm of <i>WashUp</i> .
<i>FormChange</i>	<i>Setup</i>	<i>Setup</i>	In conventional printing, changing of plates; in direct imaging printing, imaging or reimaging of plates.
<i>InkRollerWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of the inking roller, subterm of <i>WashUp</i> .
<i>PlateWash</i>	<i>Cleanup</i>	<i>Cleanup</i>	Washing of the plate, subterm of <i>WashUp</i> .
<i>SleeveChange</i>	<i>Stopped</i>	<i>Stopped</i>	Changing of sleeves, subterm for <i>Maintenance</i> .
<i>WashUp</i>	<i>Cleanup</i>	<i>Cleanup</i>	Machine is washed before, during or after production. <i>WashUp</i> is a super-term (e.g., a “generalization”) for <i>BlanketWash</i> , <i>CylinderWash</i> , <i>CleaningInkingUnit</i> , or <i>CleaningInkFountain</i> . <i>WashUp</i> is the default which is assumed if <i>StatusDetails</i> is not specified.

Table G-3: PostPress Device specific StatusDetails

StatusDetails	JDF/ @Status	DeviceStatus	Description
<i>DoubleFeed</i> <a href="#">New in JDF 1.2</a>	<i>Stopped</i>	<i>Stopped</i>	Double feeds on a feeder, subterm of <i>Failure</i> .
<i>BadFeed</i> <a href="#">New in JDF 1.2</a>	<i>Stopped</i>	<i>Stopped</i>	Bad feed on a feeder, subterm of <i>Failure</i> .
<i>BadTrim</i> <a href="#">New in JDF 1.2</a>	<i>Stopped</i>	<i>Stopped</i>	Bad trimmed components, subterm of <i>Failure</i> .
<i>ObliqueSheet</i> <a href="#">New in JDF 1.2</a>	<i>Stopped</i>	<i>Stopped</i>	Oblique sheets on components, subterm 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> <a href="#">New in JDF 1.2</a>	<i>Stopped</i>	<i>Stopped</i>	Incorrect components on a feeder, subterm of <i>Failure</i> .
<i>IncorrectThickness</i> <a href="#">New in JDF 1.2</a>	<i>Stopped</i>	<i>Stopped</i>	Incorrect thickness of components, subterm of <i>Failure</i> .

---

## Appendix H ModuleType Supported Strings

Both the `ModuleStatus` element (see Table 5-61, “Contents of the `ModuleStatus` element,” on page 167) and the `ModulePhase` element (see Table 3-35, “Contents of the `ModulePhase` element,” on page 94) contain a `ModuleType` attribute that defines individual modules within a machine. The following table defines individual `ModuleType` values.

Table H-1: `ModuleType` definition for conventional printing devices

ModuleType	Description
<i>Feeder</i>	Feeder module, feeds the device with paper.
<i>PrintModule</i>	Unit for printing a color.
<i>CoatingModule</i>	Unit for coatings, for example, full coating of varnish.
<i>Drier</i>	Module for drying the previously printed color or varnish.
<i>PerfectingModule</i>	Unit for perfecting, reversing device.
<i>ExtensionModule</i>	Unit for extending the distance between modules, for example to increase the distance between the last printing module and the delivery module.
<i>Delivery</i>	Delivery module, unit for gathering the printed sheets.
<i>Imaging</i>	Imaging Module in a direct to plate machine.
<i>Numbering</i>	Numbering unit.

Table H-2: `ModuleType` definition Gathering / Collecting

ModuleType	Description
<i>Feeder</i> <a href="#">New in JDF 1.2</a>	Feeder module, feeds the device with paper.
<i>Chain</i> <a href="#">New in JDF 1.2</a>	Transport chain or conveyer to transport gathered / collected product.
<i>PaperPath</i> <a href="#">New in JDF 1.2</a>	Paper path module, path that paper follows through the machine.

Table H-3: `ModuleType` definition for DigitalPrinting

ModuleType	Description
<i>Fuser</i> <a href="#">New in JDF 1.2</a>	Fuser module — fuses the toner onto the media.



## Appendix I 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.

Table I-1: Return codes for JMF

ReturnCode	Description
0	Success
1 – 99	Protocol errors
1	General error
2	Internal error
3	XML parser error, (e.g., if a MIME file is sent to an XML controller).
4	XML validation error
5	Query/command not implemented
6	Invalid parameters
7	Insufficient parameters
8	Device not available (controller exists but not the device or queue)
9	Message incomplete. Message Service is busy
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	Job ID not known by controller
104	JobPartID not known by controller
105	Queue entry not in queue
106	Queue request failed because the queue entry is already executing
107	The queue entry is already executing. Late change is not accepted
108	Selection or applied filter results in an empty list
109	Selection or applied filter results in an incomplete list. A buffer cannot provide the complete list queried for.
110	Queue request of a job submission failed because the requested completion time of the job cannot be fulfilled.
111	Subscription request denied.
112	Queue request failed because the Queue is closed and does not accept new entries. <a href="#">New in JDF 1.1</a>
113	Queue entry is already in the resulting status. <a href="#">New in JDF 1.2</a>
114	Queue entry is already completed or aborted and therefore does not accept changes. <a href="#">New in JDF 1.2</a>
115	Queue entry is not running. <a href="#">New in JDF 1.2</a>
200 – ...	Job and pipe specific errors
200	Invalid resource parameters
201	Insufficient resource parameters
202	PipeID unknown
203	Unlinked resource link





---

## Appendix J NotificationDetails

The Notification element is used for messaging and logging of events. It is defined in Section 3.9.1.2, Notification. Notifications are grouped into five classes: *event*, *information*, *warning*, *error*, and *fatal*. For notification classes see Section 4.6.1, Classification of Notifications. In addition to the classes, the *Type* attribute and abstract NotificationDetails element provide a container for detailed information about the notification.

Elements derived from the abstract NotificationDetails element represent a structured and extensible data type. It is defined in Section 3.9.1.2.1, NotificationDetails. The structure of various predefined NotificationDetails-types and their descriptions are listed in the following sections.

### J.1 Predefined NotificationDetails

This section defines elements that are derived from the abstract element.

#### J.1.1 Barcode

A bar code has been scanned.

Table J-1: Contents of the Barcode element

Name	Data Type	Description
<i>Code</i>	string	Contains the scanned bar code.

#### J.1.2 FCNKey

A function key has been activated at a console.

Table J-2: Contents of the FCNKey element

Name	Data Type	Description
<i>Key</i>	integer	Contains the number of that function key.

#### J.1.3 SystemTimeSet

The system time of a device/controller/agent has been set, (e.g., readjusted, changed to daylight saving time, etc.).

Table J-3: Contents of the SystemTimeSet element

Name	Data Type	Description
<i>NewTime</i>	dateTime	Contains the new time.
<i>OldTime ?</i>	dateTime	Contains the old time.

#### J.1.4 CounterReset

The production counter of a device has been reset.

Table J-4: Contents of the 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.

**J.1.5 Error**

This element provides additional information for common errors.

*Table J-5: Contents of the Error element, derived from NotificationDetails*

Name	Data Type	Description
<i>ErrorID</i>	string	Internal Error ID of the application that declares the error.
<i>ReturnCode</i> ? <a href="#">New in JDF 1.2</a>	integer	JDF defined return code for an error. See “Supported Error Codes in JMF and Notification elements” on page 619.

**J.1.6 Event**

[New in JDF 1.2](#)

This element provides additional information for common events.

*Table J-6: Contents of the Event element, derived from NotificationDetails*

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.

## Appendix K MessageEvents Values

Table K.1:

MessageEvents Values	JDF Process	JDF Intent Resource	Description
<i>Accepted</i>	<b>DigitalDelivery</b>	<b>ArtDeliveryIntent</b>	The receiver acknowledged that the files are accessible for their destination.
<i>Delivered</i>	<b>DigitalDelivery</b>	<b>ArtDeliveryIntent</b>	The files were delivered to the destination.
<i>DeviceStopped</i>	All	—	The device that executes the node has been stopped.
<i>JobCompletedSuccessfully</i>	All	All	Job completed successfully.
<i>JobCompletedWithErrors</i>	All	All	Job completed with errors.
<i>JobCompletedWithWarnings</i>	All	All	Job completed with warnings.
<i>JobInProgress</i>	All	All	Job is in progress status.
<i>PostPressCompleted</i>	—	All	All Postpress nodes of the job have been completed. Postpress nodes are defined according to Section 6.6, Postpress Processes.
<i>PostPressInProgress</i>	—	All	At least one of the Postpress nodes of the job is in progress status.
<i>PrepressCompleted</i>	—	All	All Prepress nodes of the job have been completed. Prepress nodes are defined according to Section 6.4, Prepress Processes.
<i>PrepressInProgress</i>	—	All	At least one of the Prepress nodes of the job is in progress status.
<i>PressCompleted</i>	—	All	All Press nodes of the job have been completed. Press nodes are defined according to Section 6.5, Press Processes.
<i>PressInProgress</i>	—	All	At least one of the Press nodes of the job is in progress status.
<i>ShippingCompleted</i>	<b>Delivery</b>	<b>DeliveryIntent</b>	Final product was delivered to the customer or distributors.
<i>ShippingInProgress</i>	<b>Delivery</b>	<b>DeliveryIntent</b>	Final product was shipped.

### Example:

```

<CustomerInfo CustomerJobName="Job title ...">
  <CustomerMessage Language="FR" MessageEvents="JobCompletedSuccessfully
JobCompletedWithErrors" ShowList="StartTime EndTime Error" UserText="This is the Mighty
Mouse brochure job that should be approved by Walt Disney">
    <ComChannel ChannelType="Email" Locator="admin@mycompany.com"/>
  </CustomerMessage>
</CustomerInfo>

```

This is an instruction to generate an e-mail message in French to admin@mycompany.com when the job defined in the JDF node that includes **CustomerInfo/CustomerMessage**, is completed. Two e-mail messages are requested — when job is completed either successfully or with errors. The e-mail 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”.

# Appendix L 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 307.) that are intended to allow simple, late-in-the-workflow, minor adjustments to the overall color appearance of a job or portions of a job.

Note: These color adjustments are not available in any product intent resource, such as `ColorIntent`. In order to request such adjustment in a product intent job ticket supplied to a print provider, attach to a product intent node an incomplete `ColorCorrection` process with a `ColorCorrectionParams` resource specifying the requested `ColorCorrectionOp` element attributes.

## L.1 Adjustment Using Direct Attributes

This section describes the following attributes that provide direct adjustments to various aspect of the color space:

Attribute Name	Allowed Value Range
<i>AdjustCyanRed</i>	-100 to +100
<i>AdjustMagentaGreen</i>	-100 to +100
<i>AdjustYellowBlue</i>	-100 to +100
<i>AdjustContrast</i>	-100 to +100
<i>AdjustHue</i>	-100 to +100
<i>AdjustLightness</i>	-100 to +100
<i>AdjustSaturation</i>	-100 to +100

These attributes can be applied at a point where an abstract profile would be applied (following any abstract profiles used) in the order: *AdjustLightness*, *AdjustContrast*, *AdjustSaturation*, *AdjustHue*, {*AdjustCyanRed/AdjustMagentaGreen/AdjustYellowBlue*}. The operation of each adjustment attribute is described in relation to colors expressed in the L\*a\*b\* connection color space (with L\* expressed on a scale of 0 to 100).

- *AdjustLightness* offsets the L channel. [ $L^* += AdjustLightness$ ]
- *AdjustContrast* scales the L\* channel about mid-scale ( $L^* = 50$ ). [ $L^* = 50 + (L^* - 50) * (AdjustContrast / 100 + 1)$ ]
- *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 may 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 TR 001 print characterization) are:

	Red-cyan	Green-Magenta	Blue-yellow
Angle	-129.9	-5.3	94.5
a*	0.641	-0.996	0.078
b*	0.767	0.092	-0.997

So ...

$$a^* += 0.641 * AdjustCyanRed$$

$$\begin{aligned}
 & - 0.996 * \text{AdjustMagentaGreen} \\
 & + 0.078 * \text{AdjustYellowBlue} \\
 \mathbf{b}^* & += 0.767 * \text{AdjustCyanRed} \\
 & + 0.092 * \text{AdjustMagentaGreen} \\
 & - 0.997 * \text{AdjustYellowBlue}
 \end{aligned}$$

*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.

- $a^{*'} = a^* * \cos(\text{AdjustHue}) - b^* * \sin(\text{AdjustHue})$
- $b^{*'} = a^* * \sin(\text{AdjustHue}) + b^* * \cos(\text{AdjustHue})$

## L.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.

### L.3 Adjustment using an ICC Abstract Profile Attribute

A color adjust can be encapsulated in an ICC abstract profile that is applied in ICC Profile Connection Space (PCS). The FileSpec element of the ColorCorrectionOp element with the *ResourceUsage* attribute set to "*AbstractProfile*" references an ICC profile to be used in this manner.

### L.4 Adjustment using an ICC DeviceLink Profile Attribute

A color adjust can be encapsulated in an ICC DeviceLink profile that is applied in device space. The FileSpec element of the ColorCorrectionOp element with the *ResourceUsage* attribute set to "*DeviceLinkProfile*" references an ICC profile to be used in this manner.

# Appendix M North American Media Weight Explained

[New in JDF 1.2](#)

In North America, each grade of paper has one basic size used to compute its basis weight. For example, bond basic size is 17" x 22", text basic size is 25" x 38", offset basic size is 25" x 38", coated basic size is 25" x 38", and cover basic size is 20" x 26".

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/m<sup>2</sup>) known as the sheet's grammage. Papers made to metric standards don't convert to basis weights familiar to North Americans. For example, 100 gm<sup>2</sup> 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* must be multiplied with to calculate *Weight* for various stock types.

Stock Type	Basis size in Inches	Weight / USWeight
Bond	17" x 22"	3.76
Text	25" x 38"	1.48
Offset	25" x 38"	1.48
Coated	25" x 38"	1.48
Cover	20" x 26"	2.70
Bristol	22½" x 28½"	2.19
Index	25½" x 30½"	1.81

In the following table, the right columns list common basis weights for North American papers while the left columns list their corresponding grammages. 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 M.1: Example Grammage and equivalent North American Paper Weight

Grammage (g/m <sup>2</sup> )	Basis Weight	Grammage (g/m <sup>2</sup> )	Basis Weight
30	20# Book	150	40# ledger
34	9# manifold	152	60# cover
36	24# book	163	90 # index
44	30# book	163	100 # tag
45	12# manifold	175	80# bristol
49	13# bond	176	65# cover
49	33# book	178	120# book
52	35# book	197	90# bristol
59	40# book	199	110# index
60	16# bond	216	80# cover
67	45# bond	218	125# tag
74	50# book	219	100# bristol
75	20# bond	244	150# tag

Table M.1: Example Grammage and equivalent North American Paper Weight

Grammage (g/m <sup>2</sup> )	Basis Weight	Grammage (g/m <sup>2</sup> )	Basis Weight
81	55# book	253	140# index
89	60# book	263	120# bristol
90	24# bond	270	100# cover
104	70# book/text	285	175# tag
105	28# ledger	307	140# bristol
108	40# cover	307	170# index
118	80# book/text	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/text	438	200# bristol
		488	300# tag



---

## Appendix N Media Sizes

The following table defines a set of named media sizes as defined by [http://partners.adobe.com/asn/developer/pdfs/tt/5003.PPD\\_Spec\\_v4.3.pdf](http://partners.adobe.com/asn/developer/pdfs/tt/5003.PPD_Spec_v4.3.pdf).

### Key for Notes

- I — Size is defined by ISO standards.
- J — Size is defined by JIS standards.
- E — This is an envelope size.

Media Size	Size in Points	Size in Millimeters	Size in Inches	Notes
A0	2384 x 3370	841 x 1189	33.11 x 46.81	I, J
A1	1684 x 2384	594 x 841	23.39 x 33.11	I, J
A2	1191 x 1684	420 x 594	16.54 x 23.39	I, J
A3	842 x 1191	297 x 420	11.69 x 16.54	I, J
A3Extra	913 x 1262	322 x 445	12.67 x 17.52	
A4	595 x 842	210 x 297	8.27 x 11.69	I, J
A4Extra	667 x 914	235.5 x 322.3	9.27 x 12.69	
A4Plus	595 x 936	210 x 330	8.27 x 13	
A5	420 x 595	148 x 210	5.83 x 8.27	I, J
A5Extra	492 x 668	174 x 235	6.85 x 9.25	
A6	297 x 420	105 x 148	4.13 x 5.83	I, J
A7	210 x 297	74 x 105	2.91 x 4.13	I, J
A8	148 x 210	52 x 74	2.05 x 2.91	I, J
A9	105 x 148	37 x 52	1.46 x 2.05	I, J
A10	73 x 105	26 x 37	1.02 x 1.46	I, J
AnsiC	1224 x 1584	431.8 x 558.8	17 x 22	
AnsiD	1584 x 2448	558.8 x 863.6	22 x 34	
AnsiE	2448 x 3168	863.6 x 1118	34 x 44	
ARCHA	648 x 864	228.6 x 304.8	9 x 12	
ARCHB	864 x 1296	304.8 x 457.2	12 x 18	
ARCHC	1296 x 1728	457.2 x 609.6	18 x 24	
ARCHD	1728 x 2592	609.6 x 914.4	24 x 36	
ARCHE	2592 x 3456	914.4 x 1219	36 x 48	
B0	2920 x 4127	1030 x 1456	40.55 x 57.32	J
B1	2064 x 2920	728 x 1030	28.66 x 40.55	J
B2	1460 x 2064	515 x 728	20.28 x 28.66	J
B3	1032 x 1460	364 x 515	14.33 x 20.28	J
B4	729 x 1032	257 x 364	10.12 x 14.33	J
B5	516 x 729	182 x 257	7.17 x 10.12	J
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

Media Size	Size in Points	Size in Millimeters	Size in Inches	Notes
B10	91 x 127	32 x 45	1.26 x 1.77	J
C4	649 x 918	229 x 324	9.02 x 12.75	I, E
C5	459 x 649	162 x 229	6.38 x 9.02	I, E
C6	323 x 459	113 x 162	4.49 x 6.38	I, E
Comm10	297 x 684	104.8 x 241.3	4.125 x 9.5	E
DL	312 x 624	110 x 220	4.33 x 8.66	I, E
DoublePostcard	567 x 419	200 x 148	7.87 x 5.83	
Env9	279 x 639	98.4 x 225.4	3.875 x 8.875	E
Env10	297 x 684	104.8 x 241.3	4.125 x 9.5	E
Env11	324 x 747	113.3 x 263.5	4.5 x 10.375	E
Env12	342 x 792	120.7 x 279.4	4.75 x 11	E
Env14	360 x 828	127 x 292.1	5 x 11.5	E
EnvC0	2599 x 3676	917 x 1297	36.10 x 51.06	I, E
EnvC1	1837 x 2599	648 x 917	25.51 x 36.10	I, E
EnvC2	1298 x 1837	458 x 648	18.03 x 25.51	I, E
EnvC3	918 x 1296	324 x 458	12.75 x 18.03	I, E
EnvC4	649 x 918	229 x 324	9.02 x 12.75	I, E
EnvC5	459 x 649	162 x 229	6.38 x 9.02	I, E
EnvC6	323 x 459	113 x 162	4.49 x 6.38	I, E
EnvC65	324 x 648	113 x 229	4.5 x 9	E
EnvC7	230 x 323	81 x 113	3.19 x 4.49	I, E
EnvChou3	340 x 666	120 x 235	4.72 x 9.25	E
EnvChou4	255 x 581	90 x 205	3.54 x 8	E
EnvDL	312 x 624	110 x 220	4.33 x 8.66	I, E
EnvInvite	624 x 624	220 x 220	8.66 x 8.66	E
EnvISOB4	708 x 1001	250 x 353	9.84 x 13.9	E
EnvISOB5	499 x 709	176 x 250	6.9 x 9.8	E
EnvISOB6	499 x 354	176 x 125	6.9 x 4.9	E
EnvItalian	312 x 652	110 x 230	4.33 x 9	E
EnvKaku2	680 x 941	240 x 332	9.45 x 13	E
EnvKaku3	612 x 785	216 x 277	8.5 x 10.9	E
EnvMonarch	279 x 540	98.43 x 190.5	3.875 x 7.5	E
EnvPersonal	261 x 468	92.08 x 165.1	3.625 x 6.5	E
EnvPRC1	289 x 468	102 x 165	4 x 6.5	E
EnvPRC2	289 x 499	102 x 176	4 x 6.9	E
EnvPRC3	354 x 499	125 x 176	4.9 x 6.9	E
EnvPRC4	312 x 590	110 x 208	4.33 x 8.2	E
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

Media Size	Size in Points	Size in Millimeters	Size in Inches	Notes
EnvPRC9	649 x 918	229 x 324	9 x 12.75	E
EnvPRC10	918 x 1298	324 x 458	12.75 x 18	E
EnvYou4	298 x 666	105 x 235	4.13 x 9.25	E
Executive	522 x 756	184.2 x 266.7	7.25 x 10.5	
FanFoldGerman	612 x 864	215.9 x 304.8	8.5 x 12	
FanFoldGermanLegal	612 x 936	215.9 x 330	8.5 x 13	
FanFoldUS	1071 x 792	377.8 x 279.4	14.875 x 11	
Folio	595 x 935	210 x 330	8.27 x 13	
ISOB0	2835 x 4008	1000 x 1414	39.37 x 55.67	I
ISOB01	2004 x 2835	707 x 1000	27.83 x 39.37	I
ISOB2	1417 x 2004	500 x 707	19.68 x 27.83	I
ISOB3	1001 x 1417	353 x 500	13.90 x 19.68	I
ISOB4	709 x 1001	250 x 353	9.84 x 13.90	I
ISOB5	499 x 709	176 x 250	6.9 x 9.8	I
ISOB5Extra	569 x 782	201 x 276	7.9 x 10.8	
ISOB6	354 x 499	125 x 176	4.92 x 6.93	I
ISOB7	249 x 354	88 x 125	3.46 x 4.92	I
ISOB8	176 x 249	62 x 88	2.44 x 3.46	I
ISOB9	125 x 176	44 x 62	1.73 x 2.44	I
ISOB10	88 x 125	31 x 44	1.22 x 1.73	I
Ledger	1224 x 792	431.8 x 279.4	17 x 11	
Legal	612 x 1008	215.9 x 355.6	8.5 x 14	
LegalExtra	684 x 1080	241.3 x 381	9.5 x 15	
Letter	612 x 792	215.9 x 279.4	8.5 x 11	
LetterExtra	684 x 864	241.3 x 304.8	9.5 x 12	
LetterPlus	612 x 913	215.9 x 322.3	8.5 x 12.69	
Monarch	279 x 540	98.43 x 190.5	3.875 x 7.5	E
Postcard	284 x 419	100 x 148	3.94 x 5.83	
PRC16K	414 x 610	146 x 215	5.75 x 8.5	



## Appendix O 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 may not be the same for all of the devices in the cluster. (See “Locations of Physical Resources” on page 84 for more details on the use of location.)

Table O-1: Locations within Printers

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 <b>Media</b> specification.
<i>Bottom</i>	IO*	The bin that, when facing the device, can best be identified as ‘bottom’.
<i>Booklet</i>	O	The bin where the Device places booklets.
<i>BypassTray</i>	I	The input tray used to handle odd or special papers. May be used to specify the input tray that is used for inserts sheets that are not to be imaged.
<i>BypassTray-N</i>	I	The input tray used to handle odd or special papers. May be used to specify the input tray that is used for inserts sheets that are not to be imaged. N = ‘1’, ‘2’, ...
<i>Center</i>	—	The bin that, when facing the device, can best be identified as ‘center.’ <a href="#">Deprecated in JDF 1.2</a> — use Middle instead.
<i>Continuous</i>	IO	The location to handle continuous media, (i.e., continuously connected sheets.)
<i>Disc</i>	IO	The location to handle CD or DVD discs to be printed on.
<i>Disc-N</i>	IO	The location to handle CD or DVD discs to be printed on. N = ‘1’, ‘2’, ...
<i>Envelope</i>	IO	The location that is to contain envelopes.
<i>Envelope-N</i>	IO	The location that is to contain envelopes. N = ‘1’, ‘2’, ...
<i>FaceDown</i>	O	The bin that can best be identified as ‘face down’ with respect to the device.
<i>FaceUp</i>	O	The bin that can best be identified as ‘face up’ with respect to the device.
<i>FitMedia</i>	O	Requests the device to select a bin based on the size of the media.
<i>Front</i>	IO*	The location that, when facing the device, can best be identified as ‘front.’
<i>InsertTray</i>	I	The input tray that can best be identified as ‘insert tray.’ Used to specify the input tray that is used for inserts sheets (insert sheets are never imaged.)
<i>InsertTray-N</i>	I	The input tray that can best be identified as ‘insert tray-1’, ‘insert tray-2’, ... etc. Used to specify the input tray that is used for inserts sheets (insert sheets are never imaged.)
<i>LargeCapacity</i>	IO	The bin that can best be identified as the ‘large capacity’ bin (in terms of the number of sheets) with respect to the device.
<i>LargeCapacity-N</i>	IO	The location that can best be identified as the ‘large capacity-1’, ‘large-capacity-2’, ... etc., input tray (in terms of the number of sheets) with respect to the device.
<i>Left</i>	IO*	The bin that, when facing the device, can best be identified as ‘left.’
<i>MyMailbox</i>	O	The job will be output to the bin that is best identified as ‘MyMailbox’

Table O-1: Locations within Printers

<i>Mailbox-N</i>	O	The job will be output to the bin that is best identified as 'Mailbox-1', 'Mailbox-2'...etc.
<i>Middle</i>	IO*	The bin that, when facing the device, can best be identified as 'middle'.
<i>Rear</i>	IO*	The bin that, when facing the device, can best be identified as 'rear'.
<i>Right</i>	IO*	The bin that, when facing the device, can best be identified as 'right'.
<i>Roll</i>	IO	The location to handle roll fed media.
<i>Roll-N</i>	IO	The Nth location to handle the Nth roll fed media.
<i>Side</i>	IO*	The bin that, when facing the device, can best be identified as 'side'.
<i>Stacker-N</i>	O	The job will be output to the bin that is best identified as 'Stacker-1', 'Stacker-2'...etc.
<i>Top</i>	IO*	The bin that, when facing the device, can best be identified as 'top'.
<i>Tray</i>	IO	The location for a single tray device.
<i>Tray-N</i>	IO	The job will be output to the tray that is best identified as 'Tray-1', 'Tray-2' ... etc.

Following is a table that lists some common location names that are analogous to a location name in the above table. The location names listed in the table above should be used when possible.

<b>Name</b>	<b>Location Name to use instead</b>
Back	<i>Rear</i>
Cassette	<i>Tray-N</i>
Lower	<i>Bottom</i>
Main	<i>LargeCapacity</i>
Upper	<i>Top</i>

## Appendix P FileSpec Attribute Examples for MimeType and MimeTypeVersion Attributes

[New in JDF 1.2](#)

This appendix lists examples values for the following attributes of the **FileSpec** resource: *MimeType* and *MimeTypeVersion*. The preferred file name extension is also indicated for use in the **FileSpec/@URL** attribute. The tables below apply to the values of *PDLType* and *PDLVersion* defined in “Document Properties” on page 545 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 may be used as well. When these format standards are revised with new version numbers, they may be used and should follow the patterns established in the following tables.

Many *MimeTypeVersion* values are taken from the *Printer MIB* [RFC1759] by using the langTC (e.g., PS, PCL, PDF, 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 P-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 *MimeTypeVersion* values. A blank *MimeTypeVersion* table entry indicates that there is no recognized version number for the *MimeType*. Table P-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 which must be supplied

Table P-1: *MimeType* (MIME Media Types Registered with IANA), *MimeTypeVersion* combinations

MimeType	MimeTypeVersion	File Extension	Description [iana-mt] indicates IANA registration
application/mac-binhex40	HQX/4.0	.hqx	Macintosh BinHex 4.0 7-bit encoding <sup>a</sup> [RFC1741]
application/msword	MSWORD/5.0	.doc	Microsoft Word
application/msword	MSWORD/6.0	.doc	Microsoft Word
application/msword	MSWORD/2000	.doc	Microsoft Word
application/msword	MSWORD/XP	.doc	Microsoft Word
application/pdf	PDF/1.0	.pdf	Adobe Portable Document Format
application/pdf	PDF/1.1	.pdf	Adobe Portable Document Format
application/pdf	PDF/1.2	.pdf	Adobe Portable Document Format
application/pdf	PDF/1.3	.pdf	Adobe Portable Document Format [PDF1.3]
application/pdf	PDF/1.4	.pdf	Adobe Portable Document Format [PDF1.4]
application/pdf	PDF/1.5	.pdf	Adobe Portable Document Format [PDF1.5]
application/pdf	PDF/X-1a:2001	.pdf	Portable Document Format (PDF) PDF/X-1a [iso15930-1:2001]
application/pdf	PDF/X-2:2003	.pdf	Portable Document Format (PDF) PDF/X-2 [iso15930-5:2003]

Table P-1: MimeType (MIME Media Types Registered with IANA), MimeTypeVersion combinations

MimeType	MimeTypeVersion	File Extension	Description [iana-mt] indicates IANA registration
application/pdf	PDF/X-3:2002	.pdf	Portable Document Format (PDF) PDF/X-3 [iso15930-3:2002]
application/pdf	PDF/X-1a:2003	.pdf	Portable Document Format (PDF) PDF/X-3 [iso15930-4:2003]
application/pdf	PDF/X-3:2003	.pdf	Portable Document Format (PDF) PDF/X-3 [iso15930-6:2003]
application/postscript	PS/1	.ps	Adobe PostScript™ See [RFC2045] and [RFC2046]
application/postscript	PS/2	.ps	Adobe PostScript™ See [RFC2045] and [RFC2046]
application/postscript	PS/3	.ps	Adobe PostScript™ See [RFC2045] and [RFC2046]
application/vnd.cip4-jdf+xml	JDF 1.0	.jdf	CIP4 Job Definition Format (JDF) version 1.0, April 2001
application/vnd.cip4-jdf+xml	JDF 1.1	.jdf	CIP4 Job Definition Format (JDF) version 1.1, May 2002 and 1.1a, August 2002.
application/vnd.cip4-jdf+xml	JDF 1.2	.jdf	CIP4 Job Definition Format (JDF) version 1.2
application/vnd.cip4-jdf+xml	JMF 1.0	.jmf	CIP4 Job Definition Format (JDF) version 1.0, April 2001 (See Job Messaging Format)
application/vnd.cip4-jdf+xml	JMF 1.1	.jmf	CIP4 Job Definition Format (JDF) version 1.1 and 1.1a (See Job Messaging Format)
application/vnd.cip4-jdf+xml	JMF 1.2	.jmf	CIP4 Job Definition Format (JDF) version 1.2 (See Job Messaging Format)
application/vnd.cip3-ppf	PPF/1.0	.ppf	CIP3 Print Production Format (PPF) version 1.0, 1995 [PPF]
application/vnd.cip3-ppf	PPF/3.0	.ppf	CIP3 Print Production Format (PPF) version 3.0, 1998 [PPF]
application/vnd.hp-PCL	PCL/3	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.hp-PCL	PCL/4	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.hp-PCL	PCL/5	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.hp-PCL	PCL/5e	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.hp-PCL	PCL/6	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.hp-PCL	PCL/X	.pcl	Hewlett Packard Printer Control Language (PCL™)
application/vnd.podip-ppml+xml	PPML/1.5	.ppml	Personalized Print Markup Language [PPML]
application/vnd.podip-ppml+xml	PPML/2.0	.ppml	Personalized Print Markup Language [PPML]
application/vnd.podip-ppml+xml	PPML/2.1	.ppml	Personalized Print Markup Language [PPML]



Table P-1: *MimeType* (MIME Media Types Registered with IANA), *MimeTypeVersion* combinations

<b>MimeType</b>	<b>MimeTypeVersion</b>	<b>File Extension</b>	<b>Description [iana-mt] indicates IANA registration</b>
application/ vnd.Quark.QuarkX- Press	XPress/4.11	.qxd .qxt .qwd .qwt .qxl .qxb	QuarkXPress [Quark]
application/ vnd.Quark.QuarkX- Press	XPress/4.31	.qxd .qxt .qwd .qwt .qxl .qxb	QuarkXPress [Quark]
application/ vnd.Quark.QuarkX- Press	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 <b>FileSpec/@Compression</b> attribute for the contained file may use any <i>Compression</i> value, including "None", "Compress", "Gzip", and "ZLIB".
image/jpeg		.jpeg .jpg	JPEG See [RFC2045] and [RFC2046]. Note: image/jpeg is really an image format, not a file format. JFIF and EXIF are file formats that contain image/jpeg image format data, and some applications have their own formats that are similar to JFIF and EXIF but which are proprietary. None the less, the "image/jpeg" <i>MimeType</i> value is used to indentify these file types.
image/tiff <sup>b</sup>	tiff/6.0	.tiff .tif	Tag Image File Format [RFC3302]
multipart/related		.mjd	multipart/related with JDF as the first part [RFC2387]
multipart/related		.mjm	multipart/related with JMF as the first part [RFC2387]

- a. BinHex encoding converts an 8-bit file into a 7-bit format [RFC1741], similar to UUencoding. BinHex format preserves file attributes, as well as Macintosh resource forks, and includes CRC (Cyclic Redundancy Check) error-checking. This encoding method works on any type of file, including formatted word processing and spreadsheet files, graphics files, and even executable files (i.e. programs or applications). Note: BinHex is not to be confused with MacBinary encoding, which is an 8-bit format.
- b. Note: The image/tiff MIME *MediaType* is assumed to be TIFF Revision 6.0 as defined in detail by Adobe in [TIFF6]. TIFF/IT is a different MIME type.

Table P-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 P-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 which must be used.

Table P-2: *MimeType* (File Type) and *MimeTypeVersion* combinations

<i>MimeType</i>	<i>MimeTypeVersion</i>	File Extension	Description [iana-mt] indicates IANA registration
Base64		.mme	Base64 — A format for encoding arbitrary binary information for transmission by electronic mail. [RFC3548]
Compress			Compress — UNIX compression [RFC1977].
DCS	DCS/2.0	.eps	Document Color Separation (DCS), version 2.0. [DCS2.0]
Deflate			Deflate — The file is compressed using ZIP public domain compression format [RFC1951].
GZip		.gz	Gzip — GNU zip compression technology [RFC1952].
ICC Profile	ICC-Profile/2.1.0	.icc .icm	International Color Consortium (ICC) File Format for Color Profiles taken from the binary coded decimal Profile Header Profile Version Number field (bytes 8 through 11) [ICC.1]
ICC Profile	ICC-Profile/2.2.0	.icc .icm	International Color Consortium (ICC) File Format for Color Profiles taken from the binary coded decimal Profile Header Profile Version Number field (bytes 8 through 11) [ICC.1]
ICC Profile	ICC-Profile/2.4.0	.icc .icm	International Color Consortium (ICC) File Format for Color Profiles taken from the binary coded decimal Profile Header Profile Version Number field (bytes 8 through 11) [ICC.1]
ICC Profile	ICC-Profile/4.0.0	.icc .icm	International Color Consortium (ICC) File Format for Color Profiles taken from the binary coded decimal Profile Header Profile Version Number field (bytes 8 through 11) [ICC.1]
MacBinary		.bin	MacBinary — An encoding format that combines the two forks of a Mac file, together with the file information (Name, Creator Application, File Type, etc.) into a single binary data stream that is suitable for storage or transferring through non-Mac systems. [macbinary]
Tar		.tar	UNIX packaging format.
TIFF/IT	TIFF-IT/FP:1998	.fp	TIFF/IT <sup>a</sup> [iso12639:1998] — Full Page — baseline
TIFF/IT	TIFF-IT/CT:1998	.ct	TIFF/IT [iso12639:1998] — Continuous Tone picture data — baseline
TIFF/IT	TIFF-IT/LW:1998	.lw	TIFF/IT [iso12639:1998] — Continuous Line art — baseline
TIFF/IT	TIFF-IT/HC:1998	.hc	TIFF/IT [iso12639:1998] — High-resolution Continuous tone image data — baseline
TIFF/IT	TIFF-IT/MP:1998	.mp	TIFF/IT [iso12639:1998] — monochrome picture image data — baseline
TIFF/IT	TIFF-IT/BP:1998	.bp	TIFF/IT [iso12639:1998] — Binary Picture image data — baseline

Table P-2: MimeType (File Type) and MimeTypeVersion combinations

MimeType	MimeTypeVersion	File Extension	Description [iana-mt] indicates IANA registration
TIFF/IT	TIFF-IT/BL:1998	.bl	TIFF/IT [iso12639:1998] — Binary Line art image data — baseline
TIFF/IT	TIFF-IT/FP/P1:1998	.fp	TIFF/IT [iso12639:1998] — Full Page — profile 1
TIFF/IT	TIFF-IT/CT/P1:1998	.ct	TIFF/IT [iso12639:1998] — Continuous Tone picture data — profile 1
TIFF/IT	TIFF-IT/LW/P1:1998	.lw	TIFF/IT [iso12639:1998] — Color Line art data — profile 1
TIFF/IT	TIFF-IT/HC/P1:1998	.hc	TIFF/IT [iso12639:1998] — High-resolution Continuous tone image data — profile 1
TIFF/IT	TIFF-IT/MP/P1:1998	.mp	TIFF/IT [iso12639:1998] — monochrome picture image data — profile 1
TIFF/IT	TIFF-IT/BP/P1:1998	.bp	TIFF/IT [iso12639:1998] — Binary Picture image data — profile 1
TIFF/IT	TIFF-IT/BL/P1:1998	.bl	TIFF/IT [iso12639:1998] — Binary Line art image data — profile 1
TIFF/IT	TIFF-IT/FP:2003 <sup>b</sup>	.fp	TIFF/IT [iso12639-1:2003] — Full Page — baseline
TIFF/IT	TIFF-IT/CT:2003	.ct	TIFF/IT [iso12639-1:2003] — Continuous Tone picture data — baseline
TIFF/IT	TIFF-IT/LW:2003	.lw	TIFF/IT [iso12639-1:2003] — Color Line art data — baseline
TIFF/IT	TIFF-IT/HC:2003	.hc	TIFF/IT [iso12639-1:2003] — High-resolution Continuous tone image data — baseline
TIFF/IT	TIFF-IT/MP:2003	.mp	TIFF/IT [iso12639-1:2003] — monochrome picture image data — baseline
TIFF/IT	TIFF-IT/BP:2003	.bp	TIFF/IT [iso12639-1:2003] — Binary Picture image data — baseline
TIFF/IT	TIFF-IT/BL:2003	.bl	TIFF/IT [iso12639-1:2003] — Binary Line art image data — baseline
TIFF/IT	TIFF-IT/SD:2003	.sd	TIFF/IT [iso12639-1:2003]
TIFF/IT	TIFF-IT/FP/P1:2003	.fp	TIFF/IT [iso12639-1:2003] — Full Page — profile 1
TIFF/IT	TIFF-IT/CT/P1:2003	.ct	TIFF/IT [iso12639-1:2003] — Continuous Tone picture data — profile 1
TIFF/IT	TIFF-IT/LW/P1:2003	.lw	TIFF/IT [iso12639-1:2003] — Color Line art data — profile 1
TIFF/IT	TIFF-IT/HC/P1:2003	.hc	TIFF/IT [iso12639-1:2003] — High-resolution Continuous tone image data — profile 1
TIFF/IT	TIFF-IT/MP/P1:2003	.mp	TIFF/IT [iso12639-1:2003] — monochrome picture image data — profile 1
TIFF/IT	TIFF-IT/BP/P1:2003	.bp	TIFF/IT [iso12639-1:2003] — Binary Picture image data — profile 1
TIFF/IT	TIFF-IT/BL/P1:2003	.bl	TIFF/IT [iso12639-1:2003] — Binary Line art image data — profile 1 <sup>c</sup>
TIFF/IT	TIFF-IT/FP/P2:2003	.fp	TIFF/IT [iso12639-1:2003] — Full Page — profile 2

Table P-2: MimeType (File Type) and MimeTypeVersion combinations

MimeType	MimeTypeVersion	File Extension	Description [iana-mt] indicates IANA registration
TIFF/IT	TIFF-IT/CT/P2:2003	.ct	TIFF/IT [iso12639-1:2003] — Continuous Tone picture data — profile 2
TIFF/IT	TIFF-IT/LW/P2:2003	.lw	TIFF/IT [iso12639-1:2003] — Color Line art data — profile 2
TIFF/IT	TIFF-IT/HC/P2:2003	.hc	TIFF/IT [iso12639-1:2003] — High-resolution Continuous tone image data — profile 2
TIFF/IT	TIFF-IT/MP/P2:2003	.mp	TIFF/IT [iso12639-1:2003] — monochrome picture image data — profile 2
TIFF/IT	TIFF-IT/BP/P2:2003	.bp	TIFF/IT [iso12639-1:2003] — Binary Picture image data — profile 2
TIFF/IT	TIFF-IT/BL/P2:2003	.bl	TIFF/IT [iso12639-1:2003] — Binary Line art image data — profile 2
TIFF/IT	TIFF-IT/SD/P2:2003	.sd	TIFF/IT [iso12639-1:2003]
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. Uencoding is especially popular for sending e-mail attachments. [uuencode]
ZLIB			ZLIB — ZLIB compression [RFC1950]

- a. The file format TIFF/IT must not use the “application/tiff” *MimeType*. The “application/tiff” *MimeType* conforms to baseline TIFF 6.0 [RFC3302] which obsoletes [RFC2302], 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. [RFC3302] requires that an RFC be published in order to extend image/tiff with a parameter that would be needed in order to distinguish TIFF/IT from TIFF. There is no plan by the ISO committee that oversees TIFF/IT to register TIFF/IT with either a parameter to image/tiff or as new separate MIME type. Therefore, TIFF/IT will use the *FileType* attribute instead of the *MimeType* attribute.
- b. The revision of ISO 12639 TIFF/IT is being balloted as a Draft International Standard (DIS) and is expected to be published in the latter half of 2003.
- c. Note: There is no TIFF/IT P1 conformance level of SD in ISO 12639:2003.

# Appendix Q FileSpec mimeType, URL, and Compression attributes, and Container subelement

[New in JDF 1.2](#)

The purpose of this appendix is to give a series of use cases with examples for the use of the **FileSpec** attributes: *mimeType*, *URL*, *Compression*, and the **FileSpec Container** subelement. These use cases include container packaging files, such as tar, zip, and multipart/related files and container compression and encoding files, each of which require one or more *Container* subelements to link one **FileSpec** with its container **FileSpec**.

## Q.1 FileSpec attribute value examples

Table Q-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 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 359).

Table Q-1: Use Cases showing mimeType, URL, and Compression attribute values

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, un-encode, 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 Q-1: Use Cases showing MimeType, URL, and Compression attribute values

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 <sup>a</sup>	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.

## Q.2 Corresponding XML examples

The above use case examples are represented in XML as follows:

1 Single a.pdf PDF file, no compression:

```
<FileSpec MimeType="application/pdf" URL="ftp://www.any.com/share/a.pdf"/>
```

2 Single a.pdf PDF file, with Gzip compression:

```
<FileSpec Compression="Gzip" MimeType="application/pdf" URL="a.pdf">
  <Container>
    <FileSpec MimeType="Gzip" URL="ftp://www.any.com/a.gz"/>
  </Container>
</FileSpec>
```

3 Single a.pdf PDF file, no compression, but Base64 encoded:

```
<FileSpec Compression="Base64" MimeType="application/pdf" URL="a.pdf">
  <Container>
    <FileSpec MimeType="Base64" URL="ftp://www.any.com/a.mme"/>
  </Container>
</FileSpec>
```

4 Single PDF file, no compression, but BinHex encoded:

```
<FileSpec Compression="BinHex" MimeType="application/pdf" URL="a.pdf">
  <Container>
    <FileSpec MimeType="application/mac-binhex40" URL="ftp://www.any.com/a.hqx"/>
  </Container>
</FileSpec>
```

5 Single a.pdf PDF file, in b.zip ZIP file containing one or more files:

```
<FileSpec Compression="ZLIB" MimeType="application/pdf" URL="a.pdf">
  <Container>
    <FileSpec MimeType="application/zip" URL="ftp://www.any.com/b.zip"/>
  </Container>
</FileSpec>
```

6 Single a.pdf PDF file, in a b.zip with one or more files, and the b.zip packaging file itself is Base64 encoded as b.mme. To read, un-encode, then uncompress. To write, compress, then encode.

```
<FileSpec Compression="Deflate" MimeType="application/pdf" URL="a.pdf">
```

```

<Container>
  <FileSpec Compression="Base64" MimeType="application/zip" URL="b.zip">
    <Container>
      <FileSpec MimeType="Base64" URL="ftp://www.any.com/b.mme"/>
    </Container>
  </FileSpec>
</Container>
</FileSpec>

```

- 7 Single myFiles/myPicture.jpg file in myNestedZip.zip file with one or many files, but the myNestedZip.zip is itself zipped into c.zip

```

<FileSpec Compression="Deflate" MimeType="image/jpeg" URL="myFiles/myPicture.jpg">
  <Container>
    <FileSpec Compression="Deflate" MimeType="application/zip" URL="myNestedZip.zip">
      <Container>
        <FileSpec MimeType="application/zip" URL="ftp://www.any.com/c.zip"/>
      </Container>
    </FileSpec>
  </Container>
</FileSpec>

```

- 8 Single a.pdf PDF file, which is ZLIB compressed in a c.zip with one or many files which is contained in a tar file and compressed with ZLIB into a .tar.gz file.:

```

<FileSpec Compression="ZLIB" MimeType="application/pdf" URL="a.pdf">
  <Container>
    <FileSpec MimeType="application/zip" URL="c.zip">
      <Container>
        <FileSpec Compression="ZLIB" MimeType="Tar" URL="d.tar">
          <Container>
            <FileSpec MimeType="GZip" URL="ftp://www.any.com/d.tar.gz"/>
          </Container>
        </FileSpec>
      </Container>
    </FileSpec>
  </Container>
</FileSpec>

```

### Q.3 Additional examples showing partitioning of FileSpec

This section has additional examples of container files and various schemes of partitioning.

- 1 Package b.zip contains multiple pdf files a.pdf, b.pdf etc

```

<FileSpec ID="ID_002" MimeType="application/zip" URL="ftp://www.any.com/b.zip"/>
<FileSpec Compression="Deflate" ID="A_FILE" MimeType="application/pdf" URL="a.pdf">
  <Container>
    <FileSpecRef rRef="ID_002"/>
  </Container>
</FileSpec>
<FileSpec Compression="Deflate" ID="B_FILE" MimeType="application/pdf" URL="b.pdf">
  <Container>
    <FileSpecRef rRef="ID_002"/>
  </Container>
</FileSpec>

```

- 2 Package b.zip contains two pdf files a.pdf, b.pdf and a tiff, c.tiff used by a partitioned resource

```

<FileSpec ID="ID_003" MimeType="application/zip" URL="ftp://www.any.com/b.zip"/>
<FileSpec Compression="Deflate" ID="ALL_FILES" MimeType="application/pdf"
PartIDKeys="PartVersion">

```

```

<Container>
  <FileSpecRef rRef="ID_003"/>
</Container>
<FileSpec PartVersion="English" URL="a.pdf"/>
<FileSpec PartVersion="French" URL="b.pdf"/>
<FileSpec MimeType="application/tif" PartVersion="German" URL="c.tif"/>
</FileSpec>

```

### 3 Single a.pdf PDF file, in b.zip which is contained in c.tar file:

```

<FileSpec ID="ID_004_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>

<FileSpec ID="ID_004_ZIP" MimeType="application/zip" URL="b.zip">
  <Container>
    <FileSpecRef rRef="ID_004_TAR"/>
  </Container>
</FileSpec>

<FileSpec Compression="Deflate" ID="C_FILE" MimeType="application/pdf" URL="a.pdf">
  <Container>
    <FileSpecRef rRef="ID_004_ZIP"/>
  </Container>
</FileSpec>

```

### 4 Multiple files in several zip's contained in a tar file, various examples with and without partitioning, So the file layout looks like:

```

b.tar
  c.zip
    a.pdf
    b.pdf
  d.zip
    a.pdf
    b.pdf

```

#### Scheme 1 — No Partitioning

```

<FileSpec ID="ID_005_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>

<FileSpec ID="ID_005_ZIP_C" MimeType="application/zip" URL="c.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>
<FileSpec ID="ID_005_ZIP_D" MimeType="application/zip" URL="d.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>

<FileSpec Compression="Deflate" ID="A_ENGLISH_FILE" MimeType="application/pdf"
URL="a.pdf">
  <Container FileSpecRef="ID_005_ZIP_C"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="B_ENGLISH_FILE" MimeType="application/pdf"
URL="b.pdf">
  <Container FileSpecRef="ID_005_ZIP_C"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="A_GERMAN_FILE" MimeType="application/pdf"
URL="a.pdf">
  <Container FileSpecRef="ID_005_ZIP_D"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="B_GERMAN_FILE" MimeType="application/pdf"
URL="b.pdf">
  <Container FileSpecRef="ID_005_ZIP_D"/>
</FileSpec>

```

#### Scheme 2 — Intermediate container partitioned



```

<FileSpec ID="ID_005_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>

<FileSpec ID="ID_005_ZIPS" MimeType="application/zip" PartIDKeys="PartVersion">
  <Container FileSpecRef="ID_005_TAR"/>
  <FileSpec ID="EnglishFiles" PartVersion="English" URL="c.zip"/>
  <FileSpec ID="GermanFiles" PartVersion="German" URL="d.zip"/>
</FileSpec>

<FileSpec Compression="Deflate" ID="A_ENGLISH_FILE" MimeType="application/pdf"
URL="a.pdf">
  <Container FileSpecRef="EnglishFiles"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="B_ENGLISH_FILE" MimeType="application/pdf"
URL="b.pdf">
  <Container FileSpecRef="EnglishFiles"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="A_GERMAN_FILE" MimeType="application/pdf"
URL="a.pdf">
  <Container FileSpecRef="GermanFiles"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="B_GERMAN_FILE" MimeType="application/pdf"
URL="b.pdf">
  <Container FileSpecRef="GermanFiles"/>
</FileSpec>

```

**Scheme 3 — the pdf's partitioned**

```

<FileSpec ID="ID_005_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>

<FileSpec ID="ID_005_ZIP_C" MimeType="application/zip" URL="c.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>
<FileSpec ID="ID_005_ZIP_D" MimeType="application/zip" URL="d.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>

<FileSpec Compression="Deflate" ID="ALL_FILES" PartIDKeys="PartVersion DocIndex">
  <FileSpec ID="ENGLISH_FILES" PartVersion="English">
    <Container FileSpecRef="ID_005_ZIP_C"/>
    <FileSpec DocIndex="1" ID="A_ENGLISH_FILE" MimeType="application/pdf" URL="a.pdf"/>
    <FileSpec DocIndex="2" ID="B_ENGLISH_FILE" MimeType="application/pdf" URL="b.pdf"/>
  </FileSpec>
  <FileSpec ID="GERMAN_FILES" PartVersion="German">
    <Container FileSpecRef="ID_005_ZIP_D"/>
    <FileSpec DocIndex="1" ID="A_GERMAN_FILE" MimeType="application/pdf" URL="a.pdf"/>
    <FileSpec DocIndex="2" ID="B_GERMAN_FILE" MimeType="application/pdf" URL="b.pdf"/>
  </FileSpec>
</FileSpec>

```

**Scheme 3a — As above but the file layout is not reflected in the container structure, the files are intermingled**

```

<FileSpec ID="ID_005_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>

<FileSpec ID="ID_005_ZIP_C" MimeType="application/zip" URL=" c.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>
<FileSpec ID="ID_005_ZIP_D" MimeType="application/zip" URL=" d.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>

```

```
<FileSpec Compression="Deflate" ID="ALL_FILES" MimeType="application/pdf"
PartIDKeys="PartVersion DocIndex">
  <FileSpec ID="ENGLISH_FILES" PartVersion="English">
    <FileSpec DocIndex="1" ID="A_ENGLISH_FILE" URL="a.pdf">
      <Container FileSpecRef="ID_005_ZIP_C"/>
    </FileSpec>
    <FileSpec DocIndex="2" ID="B_ENGLISH_FILE" URL="a.pdf">
      <Container FileSpecRef="ID_005_ZIP_D"/>
    </FileSpec>
  </FileSpec>
  <FileSpec ID="GERMAN_FILES" PartVersion="German">
    <FileSpec DocIndex="1" ID="A_GERMAN_FILE" URL="b.pdf">
      <Container FileSpecRef="ID_005_ZIP_C"/>
    </FileSpec>
    <FileSpec DocIndex="2" ID="B_GERMAN_FILE" URL="b.pdf">
      <Container FileSpecRef="ID_005_ZIP_D"/>
    </FileSpec>
  </FileSpec>
</FileSpec>
```

**Scheme 4 — Both partitioned**

```
<FileSpec ID="ID_005_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>

<FileSpec ID="ID_005_ZIPS" MimeType="application/zip" PartIDKeys="PartVersion">
  <Container FileSpecRef="ID_005_TAR"/>
  <FileSpec ID="EnglishFiles" PartVersion="English" URL="c.zip"/>
  <FileSpec ID="GermanFiles" PartVersion="German" URL="d.zip"/>
</FileSpec>

<FileSpec Compression="Deflate" ID="ALL_FILES" PartIDKeys="PartVersion DocIndex">
  <FileSpec ID="ENGLISH_FILES" PartVersion="English">
    <Container FileSpecRef="EnglishFiles"/>
    <FileSpec DocIndex="1" ID="A_ENGLISH_FILE" MimeType="application/pdf" URL="a.pdf"/>
    <FileSpec DocIndex="2" ID="B_ENGLISH_FILE" MimeType="application/pdf" URL="b.pdf"/>
  </FileSpec>
  <FileSpec ID="GERMAN_FILES" PartVersion="German">
    <Container FileSpecRef="GermanFiles"/>
    <FileSpec DocIndex="1" ID="A_GERMAN_FILE" MimeType="application/pdf" URL="a.pdf"/>
    <FileSpec DocIndex="2" ID="B_GERMAN_FILE" MimeType="application/pdf" URL="b.pdf"/>
  </FileSpec>
</FileSpec>
```

- 5 Multiple PDF and TIFF files in several zip's contained in a tar file. Use all PDF files in c.zip, using the FileSpec/@FileFormat mechanism and just Pictures/TIFS/a.pdf in d.zip. File layout looks like:

```
b.tar
  c.zip
    a.pdf
    a.tif
    b.pdf
    b.tif
  d.zip
    PDFS/a.pdf
    PDFS/b.pdf
    Pictures/TIFS/a.pdf
    Pictures/TIFS/b.pdf
```

```
<FileSpec ID="ID_005_TAR" MimeType="Tar" URL="ftp://www.any.com/c.tar"/>
```

```

<FileSpec ID="ID_005_ZIP_C" MimeType="application/zip" URL="c.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>
<FileSpec ID="ID_005_ZIP_D" MimeType="application/zip" URL="d.zip">
  <Container FileSpecRef="ID_005_TAR"/>
</FileSpec>

<FileSpec Compression="Deflate" FileFormat="%s.pdf" FileTemplate="all" ID="PDF_FILES"
MimeType="application/pdf">
  <Container FileSpecRef="ID_005_ZIP_C"/>
</FileSpec>
<FileSpec Compression="Deflate" ID="Pictures" URL="Pictures/TIFS/a.pdf">
  <Container FileSpecRef="ID_005_ZIP_D"/>
</FileSpec>

```

## Q.4 Example of an Intent Job Ticket with a doubly nested ZIP packaging file

Here is a complete example of an intent job ticket using **ArtDeliveryIntent** with a doubly nested packaging file. The example shows a myPictures.jpg file that is contained in myNestedZip.zip file which is contained in myZip.zip file:

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="FileSpecProposal01" JobID="bookJob"
Status="Waiting" Type="Product" Version="1.2">
  <ResourcePool>
    <ArtDeliveryIntent ID="FileSpecProposal02" Status="Draft">
      <ArtDeliveryType="DigitalMedia">
        <RunList ID="FileSpecProposal05">
          <LayoutElement>
            <FileSpec Compression="Deflate" MimeType="image/jpeg" URL="myFiles/
myPicture.jpg">
              <Container FileSpecRef="ID_002"/>
            </FileSpec>
          </LayoutElement>
        </RunList>
      </ArtDelivery>
    </ArtDeliveryIntent>
    <Component Amount="100" Class="Quantity" ComponentType="FinalProduct"
DescriptiveName="FileSpec Test" ID="FileSpecProposal03" Status="Unavailable"/>
    <FileSpec ID="ID_001" MimeType="application/zip" URL="http://www.CIP4.org/
myZip.zip"/>
      <FileSpec Compression="Deflate" ID="ID_002" MimeType="application/zip"
URL="myNestedZip.zip">
        <Container FileSpecRef="ID_001"/>
      </FileSpec>
    </ResourcePool>
    <ResourceLinkPool>
      <ComponentLink Amount="100" Usage="Output" rRef="FileSpecProposal03"/>
      <ArtDeliveryIntentLink Usage="Input" rRef="FileSpecProposal02"/>
    </ResourceLinkPool>
  <JDF ID="FileSpecProposal04" Status="Waiting" Type="DigitalPrinting">
    <ResourceLinkPool>
      <RunListLink Usage="Input" rRef="FileSpecProposal05"/>
    </ResourceLinkPool>
  </JDF>
</JDF>

```



---

# Appendix R 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.

## R.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 [RFC2396] that specifies a Base URI to resolve each URI attribute in the relements of **RunList**. 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.<sup>1</sup> 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 *URI* attribute in the **RunList**.

If the **FileSpec/Container/FileSpec** element is supplied indicating that the **FileSpec** is contained in another file, the Base URI is the Absolute URI of where the JDF Consumer extracted the container file (whether or not *Directory* is specified). (See "FileSpec" on page 359.)

After determining the Base URI depending on the presence or absence of **RunList/@Directory** and **FileSpec/Container/FileSpec** element as described above, each *URL* attribute in a **RunList** relement (e.g., **LayoutElement/FileSpec/@URL** or **InsertSheet/Sheet/Media/QualityControlResult/FileSpec/@URL**) is used in combination with the Base URI to form the Resolved URI as follows according to one of the following mutually exclusive patterns.<sup>2</sup>

- 1 **RunList URL** starts with a scheme (token ending with ":", (e.g., "*file:*" or "*cid:*"): <sup>3</sup>

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 [RFC2396] 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 [RFC2396] 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 [RFC2396] and [RFC2396bis] for the detailed algorithm. See [FileURL] for examples.

---

## Appendix S AppOS and OSVersion Attributes

[New in JDF 1.2](#)

This appendix 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 may be used and should follow the patterns established in this the following table.

Table S-1: AppOS and OSVersion Examples

AppOS	OSVersion	Description
Linux	2.2	Linux operation system
Mac	10.2.4	Macintosh operation system
Solaris	4.0	Sun Solaris operation system
UNIX	BSD	Berkley UNIX
UNIX	V	System V UNIX
UNIX	V.1	System V UNIX
UNIX	V.2	System V UNIX
UNIX	V.3	System V UNIX
UNIX	PC	UNIX for the PC
Windows	95	Windows 95
Windows	98	Windows 98
Windows	NT	Windows NT
Windows	NT-5	Windows 2000
Windows	NT-5.1	XP [not yet registered by Microsoft with IANA]





## Appendix T References

Throughout this specification references to other documents are indicated by short symbolic names inside square brackets, (e.g., [ICC.1]). Implementers must 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.<sup>1</sup>

Table T-1: Complete References

Term	Definition
[CCIR601-2]	<p><i>CCIR Recommendation 601-2</i>  <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>  <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 <a href="http://www.npes.org/standards/Standards-Technical-OrderForm.pdf">http://www.npes.org/standards/Standards-Technical-OrderForm.pdf</a>.</p>
[CGATS.20-2002]	<p><i>CGATS.20-2002</i>  <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 <a href="http://www.npes.org/standards/Standards-Technical-OrderForm.pdf">http://www.npes.org/standards/Standards-Technical-OrderForm.pdf</a>.</p>
[ColorPS]	<p><i>Color Separation Conventions for PostScript Language Programs</i>  <i>Technical Note #5044</i></p> <p>Date: 24 May 1996            Produced by: Adobe Systems Inc.            Available at: <a href="http://partners.adobe.com/asn/developer/pdfs/tn/5044.ColorSep_Conv.pdf">http://partners.adobe.com/asn/developer/pdfs/tn/5044.ColorSep_Conv.pdf</a></p>
[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: <a href="http://www.npes.org/standards/Tools/DCS20Spec.pdf">http://www.npes.org/standards/Tools/DCS20Spec.pdf</a></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: <a href="http://partners.adobe.com/misc/search.html">http://partners.adobe.com/misc/search.html</a></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: <a href="http://www.cip4.org">http://www.cip4.org</a></p>

Table T-1: Complete References

Term	Definition
[FIRST]	<i>Flexographic Image Reproduction Specifications &amp; Tolerances (FIRST) Second Edition</i> Date: November 1999 Produced by: Flexography Technical Association Available at: <a href="http://www.fta-ffta.org">http://www.fta-ffta.org</a> .
[GRACoL]	<i>General Requirements for Applications in Commercial offset Lithography (GRACoL) Version 6.0</i> Date: June 2002 Produced by: IDEAlliance (formerly Graphic Communications Association) Available at: <a href="http://www.gracol.com">http://www.gracol.com</a> .
[iana-mt]	<i>IANA Registry of MIME Media Types</i> Available at: <a href="http://www.iana.org/assignments/media-types">http://www.iana.org/assignments/media-types</a>
[iana-os]	<i>IANA Registry of Operating System Names</i> Available at: <a href="http://www.iana.org/assignments/operating-system-names">http://www.iana.org/assignments/operating-system-names</a>
[ICC.1]	<i>Specification ICC.1:2001-12 File Format for Color Profiles, Version 4.0.0</i> Date: 2001 Produced by: International Color Consortium (ICC) Available at: <a href="http://www.color.org/ICC_Minor_Revision_for_Web.pdf">http://www.color.org/ICC_Minor_Revision_for_Web.pdf</a>
[IEEE754]	<i>IEEE 754-1985 Standard for Binary Floating-Point Arithmetic</i> Date: 1985 Produced by: IEEE Available at: <a href="http://grouper.ieee.org/groups/754/">http://grouper.ieee.org/groups/754/</a>
[IEEE1284]	<i>IEEE 1284-2000 IEEE Standard Signaling Method for a Bi-directional Parallel Peripheral Interface for Personal Computers</i> Date: 2000 Produced by: IEEE Available at: <a href="http://standards.ieee.org/catalog/olis/busarch.html">http://standards.ieee.org/catalog/olis/busarch.html</a>
[ifra]	<i>IfraTrack Specification Ifra Special Report 6.21.2, Version 2.0</i> Date: June 1998 Produced by: Ifra Available at: <a href="http://www.ifra.com/">http://www.ifra.com/</a>
[iso12639:1998]	<i>ISO 12639:1998 Graphic technology — Prepress digital data exchange — Tag image file format for image technology (TIFF/IT).</i> Date: 1998 Produced by: ISO Available at: <a href="http://www.iso.ch/iso/en/prods-services/ISOstore/store.html">http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</a>

- 
1. The references appendix is “Appendix T” in honor of Tom Hastings of Xerox whose diligent work in identifying and flushing out these reference has resulted in the very complete list that you see here. To many of the members of CIP4, this is really “Appendix Tom”!

Table T-1: Complete References

Term	Definition
[iso12639-1:2003]	<p><i>ISO 12639-1:2003</i>  <i>Graphic technology — Prepress digital data exchange — Tag image file format for image technology (TIFF/IT).</i></p> <p>Date: 2003            Produced by: ISO            Available at: <a href="http://www.iso.ch/iso/en/prods-services/ISOstore/store.html">http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</a></p>
[iso14977:1996]	<p><i>ISO 14977:1996(E)</i>  <i>Information technology -- Syntactic metalanguage -- Extended BNF</i></p> <p>Date: 1996            Produced by: ISO            Available at: <a href="http://www.iso.ch/iso/en/prods-services/ISOstore/store.html">http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</a></p>
[iso15930-1:2001]	<p><i>ISO 15930-1:2001</i>  <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: <a href="http://www.iso.ch/iso/en/prods-services/ISOstore/store.html">http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</a></p>
[iso15930-4:2003]	<p><i>ISO 15930-4:2003</i>  <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: <a href="http://www.iso.ch/iso/en/prods-services/ISOstore/store.html">http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</a></p>
[iso15930-5:2003]	<p><i>ISO 15930-2:2003</i>  <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: <a href="http://www.iso.ch/iso/en/prods-services/ISOstore/store.html">http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</a></p>
[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: <a href="http://www.iso.ch/iso/en/prods-services/ISOstore/store.html">http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</a></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: <a href="http://www.iso.ch/iso/en/prods-services/ISOstore/store.html">http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</a></p>

Table T-1: Complete References

Term	Definition
[iso12647-2]	<p><i>ISO 12647-2:1996</i>  <i>Graphic technology — Process control for the manufacture of half-tone colour separations, proof and production prints — Part 2: Offset lithographic processes</i></p> <p>Date: 1996            Produced by: ISO            Available at: <a href="http://www.iso.ch/iso/en/prods-services/ISOstore/store.html">http://www.iso.ch/iso/en/prods-services/ISOstore/store.html</a></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>
[JDF11a]	<p><i>Job Definition Format 1.1a *</i></p> <p>Date: 2002            Produced by: International Cooperation for Integration of Processes in Prepress, Press and Postpress (CIP4)            Available at: <a href="http://www.cip4.org">http://www.cip4.org</a></p>
[JDFMIME]	<p><i>The MIME application/vnd.cip4-jdf+xml Content-Type</i>  <i>Work in progress</i></p> <p>Date: 25 January 2003            Produced by: Hastings, T., and McDonald, I.            Available at: <a href="draft-mcdonald-cip4-jdf-mime-00.txt">draft-mcdonald-cip4-jdf-mime-00.txt</a></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: <a href="http://www.lazerware.com/formats/">http://www.lazerware.com/formats/</a></p>
[opentypefont]	<p><i>OpenType specification</i>  <i>v.1.4</i></p> <p>Date: 11 October 2002            Produced by: Microsoft Corporation            Available at: <a href="http://www.microsoft.com/typography/specs/">http://www.microsoft.com/typography/specs/</a></p>
[PDF1.3]	<p><i>PDF reference : Adobe portable document format version 1.3 / Adobe Systems Incorporated.</i>  <i>2nd Edition</i></p> <p>Date: 3 July 2000            Produced by: Addison-Wesley            Available at: <a href="http://partners.adobe.com/asn/developer/technotes/acrobatpdf.html">http://partners.adobe.com/asn/developer/technotes/acrobatpdf.html</a></p>
[PDF1.4]	<p><i>PDF reference : Adobe portable document format version 1.4 / Adobe Systems Incorporated.</i>  <i>3rd Edition</i></p> <p>Date: November 2001            Produced by: Addison-Wesley            Available at: <a href="http://partners.adobe.com/asn/developer/technotes/acrobatpdf.html">http://partners.adobe.com/asn/developer/technotes/acrobatpdf.html</a></p>

Table T-1: Complete References

Term	Definition
[PDF1.5]	<p><i>PDF reference : Adobe portable document format version 1.5 / Adobe Systems Incorporated. Version 1.5</i></p> <p>Date: 20 June 2003            Produced by: Addison-Wesley            Available at: <a href="http://partners.adobe.com/misc/search.html">http://partners.adobe.com/misc/search.html</a></p>
[PJTF]	<p><i>The Portable Job Ticket Format Version 1.1</i></p> <p>Date: 2 April 1999            Produced by: Adobe Systems Inc.            Available at: <a href="http://partners.adobe.com/asn/developer/pdfs/tn/5620.pjtf.pdf">http://partners.adobe.com/asn/developer/pdfs/tn/5620.pjtf.pdf</a>.</p>
[PPF]	<p><i>Print Production Format Version 3.0</i></p> <p>Date: 2 June 1998            Produced by: The International Cooperation for Integration of Prepress, Press, and Postpress            Available at: <a href="http://www.cip4.org/documents/technical_info/cip3v3_0.pdf">http://www.cip4.org/documents/technical_info/cip3v3_0.pdf</a>.</p>
[PPML]	<p><i>PPML Personal Print Markup Language (PPML) Version 2.1</i></p> <p>Produced by: Print On Demand Initiative (PODi)            Available at: <a href="http://www.podi.org">http://www.podi.org</a></p>
[PrintTalk]	<p><i>PrintTalk Implementation Version 1.1</i></p> <p>Produced by: PrintTalk Consortium            Available at: <a href="http://www.printtalk.org/">http://www.printtalk.org/</a>.</p>
[PS]	<p><i>PostScript Language Reference (Redbook) Third Edition</i></p> <p>Date: —            Produced by: Adobe Systems, Inc.            Available at: <a href="http://partners.adobe.com/asn/developer/pdfs/tn/PLRM.pdf">http://partners.adobe.com/asn/developer/pdfs/tn/PLRM.pdf</a></p>
[PWG]	<p><i>The Printer Working Group</i></p> <p>Date: —            Produced by: IEEE-ISTO            Available at: <a href="http://www.pwg.org">http://www.pwg.org</a></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: <a href="https://datatracker.ietf.org/public/pidtracker.cgi">https://datatracker.ietf.org/public/pidtracker.cgi</a></p>
[Quark]	<p>See <a href="http://www.quark.com">http://www.quark.com</a>.</p>
[RFC1738]	<p><i>RFC 1738 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 RFC Database search: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</p>

Table T-1: Complete References

Term	Definition
[RFC1741]	<p><i>RFC 1741</i>  <i>MIME Content Type for BinHex Encoded Files, by Faltstrom, P., Crocker, D., and Fair, E.</i></p> <p>Date: December 1994  Produced by: Internet Engineering Task Force (IETF), Network Working Group  Available at: All IETF (Internet Engineering Task Force) RFCs (Request for Comments) are available at RFC Database search: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</p>
[RFC1759]	<p><i>RFC 1759</i>  <i>Printer MIB, Version 2.0 by Smith, R., Wright, F., Hastings, T., Zilles, S., and Gyllenskog, J.</i></p> <p>Date: June 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: <a href="https://datatracker.ietf.org/public/pidtracker.cgi">https://datatracker.ietf.org/public/pidtracker.cgi</a>.</p>
[RFC1808]	<p><i>RFC 1808</i>  <i>Relative Uniform Resource Locators</i></p> <p>Date: June 1995  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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</p>
[RFC1950]	<p><i>RFC 1950</i>  <i>ZLIB Compressed Data Format Specification version 3.3, by P. Deutsch.</i></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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</p>
[RFC1951]	<p><i>RFC 1951</i>  <i>DEFLATE Compressed Data Format Specification version 1.3, by Deutsch, P.</i></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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</p>
[RFC1952]	<p><i>RFC 1952</i>  <i>GZIP file format specification version 4.3, by Deutsch, P.</i></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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</p>
[RFC1977]	<p><i>RFC 1977</i>  <i>PPP BSD Compression Protocol, by Schryver, V.</i></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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</p>

Table T-1: Complete References

Term	Definition
[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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</p>
[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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</p>
[RFC2302]	<p><i>RFC 2302</i>  <i>Tag Image File Format (TIFF) — image/tiff MIME Sub-type Registration</i>, by Parsons, G., Rafferty, J., and Zilles, S. (Obsoleted by RFC3302)</p> <p>Date: March 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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</p>
[RFC2396]	<p><i>RFC 2396</i>  <i>Uniform Resource Identifiers (URI): Generic Syntax</i>, by Berners-Lee, T., and Fielding, R., Masinter, L. (See also [RFC2396bis].)</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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</p>

Table T-1: Complete References

Term	Definition
[RFC2396bis]	<p><i>RFC 2396bis</i>  <i>Uniform Resource Identifiers (URI): Generic Syntax, Internet-Draft, &lt;work in progress&gt;, &lt;draft-fielding-uri-rfc2396bis-03.txt&gt;, by Berners-Lee, T., and Fielding, R., Masinter, L.</i></p> <p>Date: June 6, 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: <a href="https://datatracker.ietf.org/public/pidtracker.cgi">https://datatracker.ietf.org/public/pidtracker.cgi</a>.</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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</p>
[RFC2806]	<p><i>RFC 2806</i>  <i>URLs for Telephone Calls, by A. Vaha-Sipila</i></p> <p>Date: April 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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</p>
[RFC3302]	<p><i>RFC 3302</i>  <i>Tag Image File Format (TIFF) — image tiff MIME Sub-type Registration, by Parsons, G., and Rafferty, J. (Obsoletes RFC2302)</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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</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: <a href="http://www.rfc-editor.org/rfcsearch.html">http://www.rfc-editor.org/rfcsearch.html</a>.</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: <a href="http://www.gain.net/store/item.cfm?productid=488">http://www.gain.net/store/item.cfm?productid=488</a></p>
[TIFF6]	<p><i>TIFF Revision 6.0</i></p> <p>Date: June 1992  Produced by: Adobe Systems, Inc.  Available at: <a href="http://partners.adobe.com/asn/tech/tiff/specification.jsp">http://partners.adobe.com/asn/tech/tiff/specification.jsp</a></p>
[TIFFPS]	<p><i>Adobe Photoshop TIFF Technical Notes</i></p> <p>Date: March 2002  Produced by: Adobe Systems, Inc.  Available at: <a href="http://partners.adobe.com/asn/tech/tiff/specification.jsp">http://partners.adobe.com/asn/tech/tiff/specification.jsp</a></p>
[truetypefont]	<p><i>TrueType font file and TrueType Open specification</i></p> <p>Date: August 1995  Produced by: Microsoft Corporation  Available at: <a href="http://www.microsoft.com/typography/specs/">http://www.microsoft.com/typography/specs/</a></p>



Table T-1: Complete References

Term	Definition
[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: <a href="http://partners.adobe.com/asn/developer/pdfs/tn/T1_SPEC.PDF">http://partners.adobe.com/asn/developer/pdfs/tn/T1_SPEC.PDF</a></p>
[uuencode]	<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: <a href="http://www.opengroup.org/onlinepubs/007908799/xcu/uuencode.html">http://www.opengroup.org/onlinepubs/007908799/xcu/uuencode.html</a></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: <a href="http://www.upnp.org/standardizeddcp/printers.asp">http://www.upnp.org/standardizeddcp/printers.asp</a></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: <a href="http://www.pkware.com/products/enterprise/white_papers/appnote.html">http://www.pkware.com/products/enterprise/white_papers/appnote.html</a></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: <a href="http://www.w3.org/TR/REC-xml">http://www.w3.org/TR/REC-xml</a>.</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: <a href="http://www.w3.org/TR/REC-xml-names/">http://www.w3.org/TR/REC-xml-names/</a></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: <a href="http://www.w3.org/TR/xpath.html">http://www.w3.org/TR/xpath.html</a>.</p>
[XMLSchema]	<p><i>XML Schema Part 0+1+2: Primer, Structures and Datatypes *</i>  <i>Version (W3C Recommendation of 02 May 2001)</i></p> <p>Date: 02 May 2001  Produced by: World Wide Web Consortium (W3C) XML Schema working group  Available at: <a href="http://www.w3.org/TR/xmlschema-0/">http://www.w3.org/TR/xmlschema-0/</a>, <a href="http://www.w3.org/TR/xmlschema-1/">http://www.w3.org/TR/xmlschema-1/</a> and <a href="http://www.w3.org/TR/xmlschema-2/">http://www.w3.org/TR/xmlschema-2/</a>.</p>



## Appendix U 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:







<b>General</b>	<m i>: m = metric (millimeter is used), i = imperial (inch, where 1 inch = 25.4 mm)
<b>Ring Binding</b>	R<#holes><m i>-<variant> Example: R2m-DIN = RingBind, 2 hole, metric, DIN
<b>Plastic Comb</b>	P<pitch><m i>-<shape>-<#thumbcuts>t Example: P16:9m-round-0t = Plastic Comb, 9/16" pitch (16:9), round, no thumbcut
<b>Wire Comb</b>	W<pitch><m i>-<shape>-<#thumbcuts>t Example: W2:1i-square-1t = Wire Comb, 1/2" pitch (2:1), square, one thumbcut
<b>Coil/Spiral</b>	C<pitch><m i>-<shape>-<#thumbcuts>t Example: C9.5m-round-0t = Coil, 9.5 mm, round, no thumbcut

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
<b>RING BINDING (R...)</b>										
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 = twoHoleUS-Top 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
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		

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
<b>RING BINDING (R3...)</b>										
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" rng: 0.2-0.5" typ: 1/4", 9/32", 11/32", 3/8", 13/ 32", 1/2"	4.25"	0.25" + ½ diameter range: 6/16" - 1/2"	29.25 (≅ 13/32")	Left		Printer Finishing MIB
<b>4 Holes (R4...)</b>										
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
R4m-DIN-A4	DIN 4-hole for A4	4	●	5.5 ± 0.1 mm	80 ± 0.1 mm	7 or 11 ± 0.3 mm 7 mm for blocks of 15 mm or less	31.18 (≅ 11 mm)	Left	A4	DIN 5005:1991 DIN 821:1973
R4m-DIN-A5	DIN 4-Hole for A5	4	●	5.5 ± 0.1 mm	45-65-45 mm	7 or 11 ± 0.3 mm 7 mm for blocks of 15 mm or less	31.18 (≅ 11 mm)	Left	A5	DIN 5005:1991
R4m-swedish	Swedish 4-hole MIB: 11 = swedish4Hole	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		

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 (l)	Default Process Edge	Usage Notes	Source Standard
<b>RING BINDING (R6...)</b>										
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
R5i-US-a	US 5-hole, Variant A	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	MIB: 13 = fiveHoleUS 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 = norwegianHole	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	l	5.5 ± 0.1 mm	37.5-7.5-65-7.5-37.5 mm	7 or 11 ± 0.3 mm 7 mm for blocks of ≤ 15 mm thick	31.18 (≅ 11 mm)	Left	Only used with A5	DIN 5005:1991

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
<b>RING BINDING (R...)</b>										
<b>7 Holes (R7...)</b>										
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 = sevenHoleUS	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		
<b>11 Holes (R11...)</b>										
R11m-7h4s	7-hole (round) mixed with 4 slots (rectangular) MIB: 15 = mixed7H4S	11	H: ● S: ■	Holes: 5 - 8 mm Slots: 12 x 6 mm	7 holes/2slots Pattern: H-S-H-S-H-S-H-S-H	4.5 - 13 mm	31.18 (≅ 11 mm)	Left for A4 Top for A3		Printer Finishing MIB

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
<b>PLASTIC COMB BINDING (P...)</b>										
P16_9i-rect-0t	US spacing, no thumb-cut MIB: 9 = nineteenHoleUS	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		
<b>WIRE COMB BINDING (W...)</b>										
W2_li-round-0t	2:1, round, no thumbcut MIB: 8 = twentyTwoHoleUS	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 thumb-cut	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 thumb-cuts	A4: 34 A5: 24 Letter: 32		5/32 x 5/32" (4x4 mm)	1/3"	0.2"	14.40 (≅ 0.2")	Left		
<b>COIL/SPIRAL BINDING (C...)</b>										
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
<b>SPECIAL (S...)</b>										
										Reserved for future extensions



---

## Appendix V Examples

Note that these examples were generated using prototype tools and should 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.

### V.1 Brief Example

#### V.1.1 Before Processing

This is a simple example of a JDF that describes color conversion for one file.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="ColorTest" JobID="ColorJob"
Status="Waiting" Type="ColorSpaceConversion" Version="1.2">
  <!--Generated by the CIP4 C++ open source JDF Library version CIP4 JDFWriter 1.0.01
beta-->
  <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"
SourceProfile="File:///image.icc"/>
      <ColorSpaceConversionOp Operation="Convert" RenderingIntent="Perceptual"
SourceCS="RGB" SourceObjects="Text LineArt" SourceProfile="File:///text.icc"/>
    </ColorSpaceConversionParams>
    <ColorPool Class="Parameter" ID="Link0005" Status="Available">
      <Color CMYK="1 0 0 0" Name="Cyan"/>
      <Color CMYK="0 1 0 0" Name="Magenta"/>
      <Color CMYK="0 0 1 0" Name="Yellow"/>
      <Color CMYK="0 0 0 1" Name="Black"/>
      <Color CMYK="0.8 0.8 0 0" Name="Blue"/>
    </ColorPool>
    <ColorantControl Class="Parameter" ID="Link0006" ProcessColorModel="DeviceCMYK"
Status="Available">
      <ColorPoolRef rRef="Link0005"/>
    </ColorantControl>
    <RunList Class="Parameter" ID="Link0007" Pages="0~-1" Status="Unavailable">
      <LayoutElement>
        <FileSpec URL="File:///out/colortest.pdf"/>
      </LayoutElement>
    </RunList>
  </ResourcePool>
  <ResourceLinkPool>
    <RunListLink Usage="Input" rRef="Link0003"/>
    <ColorSpaceConversionParamsLink Usage="Input" rRef="Link0004"/>
    <ColorPoolLink Usage="Input" rRef="Link0005"/>
    <ColorantControlLink Usage="Input" rRef="Link0006"/>
    <RunListLink Usage="Output" rRef="Link0007"/>
  </ResourceLinkPool>
  <AuditPool>
    <Created Author="Rainer's JDFWriter 0.2000" TimeStamp="2000-11-01T10:26:11+01:00"/>
  </AuditPool>
</JDF>
```

## V.1.2 After Processing

This is a simple example of a JDF that describes color conversion for one file after the color conversion process has been executed.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="ColorTest " JobID="ColorJob"
Status="Completed" Type="ColorSpaceConversion" Version="1.2">
  <!--Generated by the CIP4 C++ open source JDF Library version CIP4 JDFWriter 1.0.01
beta-->
  <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"
SourceProfile="File:/// image.icc"/>
      <ColorSpaceConversionOp Operation="Convert" RenderingIntent="Perceptual"
SourceCS="RGB" SourceObjects="Text LineArt" SourceProfile="File:///text.icc"/>
    </ColorSpaceConversionParams>
    <ColorPool Class="Parameter" ID="Link0005" Status="Available">
      <Color CMYK="1 0 0 0" Name="Cyan"/>
      <Color CMYK="0 1 0 0" Name="Magenta"/>
      <Color CMYK="0 0 1 0" Name="Yellow"/>
      <Color CMYK="0 0 0 1" Name="Black"/>
      <Color CMYK="0.8 0.8 0 0" Name="Blue"/>
    </ColorPool>
    <ColorantControl Class="Parameter" ID="Link0006" ProcessColorModel="DeviceCMYK"
Status="Available">
      <ColorPoolRef rRef="Link0005"/>
    </ColorantControl>
    <RunList Class="Parameter" ID="Link0007" Pages="0~-1" Status="Available">
      <LayoutElement>
        <FileSpec URL="File:///out/colortest.pdf"/>
      </LayoutElement>
    </RunList>
  </ResourcePool>
  <ResourceLinkPool>
    <RunListLink Usage="Input" rRef="Link0003"/>
    <ColorSpaceConversionParamsLink Usage="Input" rRef="Link0004"/>
    <ColorPoolLink Usage="Input" rRef="Link0005"/>
    <ColorantControlLink Usage="Input" rRef="Link0006"/>
    <RunListLink Usage="Output" rRef="Link0007"/>
  </ResourceLinkPool>
  <AuditPool>
    <Created Author="Rainer's JDFWriter 0.2000" TimeStamp="2000-11-01T10:26:11+01:00"/>
    <Modified Author="EatJDF Complete: task=" TimeStamp="2000-11-01T10:26:57+01:00"/>
    <PhaseTime End="2000-11-01T10:26:57+01:00" Start="2000-11-01T10:26:57+01:00"
Status="Setup" TimeStamp="2000-11-01T10:26:57+01:00"/>
    <PhaseTime End="2000-11-01T10:26:57+01:00" Start="2000-11-01T10:26:57+01:00"
Status="InProgress" TimeStamp="2000-11-01T10:26:57+01:00"/>
    <PhaseTime End="2000-11-01T10:26:57+01:00" Start="2000-11-01T10:26:57+01:00"
Status="Cleanup" TimeStamp="2000-11-01T10:26:57+01:00"/>
    <ProcessRun End="2000-11-01T10:26:57+01:00" EndStatus="Completed" Start="2000-11-
01T10:26:57+01:00" TimeStamp="2000-11-01T10:26:57+01:00"/>
  </AuditPool>
</JDF>
```

## V.2 Product JDF

The following example describe a pair of college textbooks, one teachers edition and one students edition as product intent. Most intent resources are intentionally left empty.

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="bookTest" JobID="bookJob"
Status="Waiting" Type="Product" Version="1.2">
  <!--Generated by the CIP4 C++ open source JDF Library version CIP4 JDFWriter 1.0.01
beta-->
  <ResourcePool>
    <Component Amount="100" Class="Quantity" DescriptiveName="Teacher's Book"
ID="Link0003" Status="Unavailable"/>
    <Component Amount="2000" Class="Quantity" DescriptiveName="Cover" ID="Link0005"
Status="Unavailable">
      <!--This cover is reused by both-->
    </Component>
    <LayoutIntent Class="Intent" ID="Link0006" Status="Available">
      <Dimensions DataType="XYPairSpan" Preferred="612 792" Range="576 756~648 828"/>
    </LayoutIntent>
    <LayoutIntent Class="Intent" ID="Link0008" Status="Available">
      <Dimensions DataType="XYPairSpan" Preferred="612 792" Range="576 756~648 828"/>
      <Pages DataType="IntegerSpan" Preferred="240"/>
    </LayoutIntent>
    <Component Amount="1000" Class="Quantity" DescriptiveName="Student's Book"
ID="Link0011" Status="Unavailable">
      <!--Students Book Intent-->
    </Component>
    <LayoutIntent Class="Intent" ID="Link0014" Status="Available">
      <Dimensions DataType="XYPairSpan" Preferred="612 792" Range="576 756~648 828"/>
      <Pages DataType="IntegerSpan" Preferred="198"/>
    </LayoutIntent>
  </ResourcePool>
  <AuditPool>
    <Created Author="Rainer's JDFWriter 0.2000" TimeStamp="2000-11-01T12:46:56+01:00"/>
  </AuditPool>
  <JDF DescriptiveName="Teacher's Edition" ID="Link0002" JobPartID="0" Status="Waiting"
Type="Product">
    <ResourcePool>
      <Component Amount="100" Class="Quantity" DescriptiveName="Insert" ID="Link0009"
Status="Unavailable"/>
    </ResourcePool>
    <ResourceLinkPool>
      <ComponentLink Amount="100" Usage="Output" rRef="Link0003"/>
      <ComponentLink Amount="100" Usage="Input" rRef="Link0009"/>
      <ComponentLink Amount="100" Usage="Input" rRef="Link0005"/>
    </ResourceLinkPool>
    <JDF DescriptiveName="Teacher's Insert" ID="Link0007" JobPartID="2"
Status="Waiting" Type="Product">
      <ResourceLinkPool>
        <LayoutIntentLink Usage="Input" rRef="Link0008"/>
        <ComponentLink Amount="100" Usage="Output" rRef="Link0009"/>
      </ResourceLinkPool>
    </JDF>
  </JDF>
  <JDF DescriptiveName="Cover" ID="Link0004" JobPartID="1" Status="Waiting"
Type="Product">
    <ResourceLinkPool>
      <ComponentLink Amount="2000" Usage="Output" rRef="Link0005"/>
      <LayoutIntentLink Usage="Input" rRef="Link0006"/>
    </ResourceLinkPool>
```

```

</JDF>
<JDF DescriptiveName="Student's Edition" ID="Link0010" JobPartID="3" Status="Waiting"
Type="Product">
  <ResourcePool>
    <Component Amount="1000" Class="Quantity" DescriptiveName="Insert" ID="Link0013"
Status="Unavailable"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ComponentLink Amount="1000" Usage="Output" rRef="Link0011"/>
    <ComponentLink Amount="1000" Usage="Input" rRef="Link0013"/>
    <ComponentLink Amount="1000" Usage="Input" rRef="Link0005"/>
  </ResourceLinkPool>
  <JDF DescriptiveName="Student's Insert" ID="Link0012" JobPartID="4"
Status="Waiting" Type="Product">
    <ResourceLinkPool>
      <ComponentLink Amount="1000" Usage="Output" rRef="Link0013"/>
      <LayoutIntentLink Usage="Input" rRef="Link0014"/>
    </ResourceLinkPool>
  </JDF>
</JDF>
</JDF>

```

## V.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.

### V.3.1 Example 2 Component JDF before Spawning

The following JDF file describes a two-component brochure. The resources are not fleshed out.

```

<JDF ID="SpawnTest" Type="Product" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" Version="1.2" JobPartID="Part1">
  <!--Generated by the CIP4 C++ open source JDF Library version CIP4 JDFWriter 1.0.01
beta-->
  <AuditPool>
    <Created Author="CIP4 JDFWriter 1.0.01 beta" TimeStamp="2002-04-05T15:27:58+02:00"/
  >
</AuditPool>
<ResourcePool>
  <Component ID="r0043" Class="Quantity" Amount="10000" Status="Unavailable"/>
  <BindingIntent ID="r0044" Class="Intent" Status="Available"/>
  <ProductionIntent ID="r0045" Class="Intent" Status="Available">
    <PrintProcess Range="Gravure" DataType="EnumerationSpan"/>
  </ProductionIntent>
  <Component ID="r0047" Class="Quantity" Status="Unavailable"/>
  <Component ID="r0051" Class="Quantity" Status="Unavailable"/>
</ResourcePool>
<ResourceLinkPool>
  <ComponentLink rRef="r0043" Usage="Output"/>
  <BindingIntentLink rRef="r0044" Usage="Input"/>
  <ProductionIntentLink rRef="r0045" Usage="Input"/>
  <ComponentLink rRef="r0047" Usage="Input"/>
  <ComponentLink rRef="r0051" Usage="Input"/>
</ResourceLinkPool>
  <JDF ID="n0046" Type="Product" Status="Waiting" JobPartID="Part2"
DescriptiveName="Cover">
    <ResourceLinkPool>
      <ComponentLink rRef="r0047" Usage="Output"/>

```

```

    <LayoutIntentLink rRef="r0048" Usage="Input"/>
    <ColorIntentLink rRef="r0049" Usage="Input"/>
  </ResourceLinkPool>
</ResourcePool>
  <LayoutIntent ID="r0048" Class="Intent" Status="Available"/>
  <ColorIntent ID="r0049" Class="Intent" Status="Available"/>
</ResourcePool>
</JDF>
<JDF ID="n0050" Type="Product" Status="Waiting" JobPartID="Part3"
DescriptiveName="Insert">
  <ResourceLinkPool>
    <ComponentLink rRef="r0051" Usage="Output"/>
    <LayoutIntentLink rRef="r0052" Usage="Input"/>
    <ColorIntentLink rRef="r0053" Usage="Input"/>
  </ResourceLinkPool>
  <ResourcePool>
    <LayoutIntent ID="r0052" Class="Intent" Status="Available"/>
    <ColorIntent ID="r0053" Class="Intent" Status="Available"/>
  </ResourcePool>
</JDF>
</JDF>

```

### V.3.2 Example 2 Component JDF Parent after spawning the cover node

The following JDF is the parent JDF after spawning. The **Component** that describes the cover is marked as *SpawnedRW*, since it was copied into the spawned node and may 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="1.2" JobPartID="Part1">
  <!--Generated by the CIP4 C++ open source JDF Library version CIP4 JDFWriter 1.0.01
beta-->
  <AuditPool>
    <Created Author="CIP4 JDFWriter 1.0.01 beta" TimeStamp="2002-04-05T15:27:58+02:00"/
  >
    <Spawned URL="File:///spawn.jdf" jRef="n0046" TimeStamp="2002-04-05T15:34:43+02:00"
NewSpawnID="Sp0057" rRefsRWCopied="r0047"/>
  </AuditPool>
  <ResourcePool>
    <Component ID="r0043" Class="Quantity" Amount="10000" Status="Unavailable"/>
    <BindingIntent ID="r0044" Class="Intent" Status="Available"/>
    <ProductionIntent ID="r0045" Class="Intent" Status="Available">
      <PrintProcess Range="Gravure" DataType="EnumerationSpan"/>
    </ProductionIntent>
    <Component ID="r0047" Class="Quantity" Status="Unavailable" SpawnIDs="Sp0057"
SpawnStatus="SpawnedRW"/>
    <Component ID="r0051" Class="Quantity" Status="Unavailable"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ComponentLink rRef="r0043" Usage="Output"/>
    <BindingIntentLink rRef="r0044" Usage="Input"/>
    <ProductionIntentLink rRef="r0045" Usage="Input"/>
    <ComponentLink rRef="r0047" Usage="Input"/>
    <ComponentLink rRef="r0051" Usage="Input"/>
  </ResourceLinkPool>
  <JDF ID="n0046" Type="Product" Status="Spawned" JobPartID="Part2"
DescriptiveName="Cover">
    <ResourceLinkPool>
      <ComponentLink rRef="r0047" Usage="Output"/>
      <LayoutIntentLink rRef="r0048" Usage="Input"/>

```

```

    <ColorIntentLink rRef="r0049" Usage="Input"/>
  </ResourceLinkPool>
  <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>

```

### V.3.3 Example 2 Component JDF spawned node

The Component that represents the cover was copied into the spawned node, since it is the output resource. It is not locked, since it was spawned in RW mode. The existence of an **AncestorPool** denotes the node as spawned and defines the parent node

```

<JDF ID="n0046" Type="Product" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" SpawnID="Sp0057" Version="1.2" JobPartID="Part2"
DescriptiveName="Cover">
  <!--Generated by the CIP4 C++ open source JDF Library version CIP4 JDFWriter 1.0.01
beta-->
  <AuditPool>
    <Created Author="CIP4 JDFWriter 1.0.01 beta" TimeStamp="2002-04-05T15:34:43+02:00"/
  >
  </AuditPool>
  <ResourceLinkPool>
    <ComponentLink rRef="r0047" Usage="Output"/>
    <LayoutIntentLink rRef="r0048" Usage="Input"/>
    <ColorIntentLink rRef="r0049" Usage="Input"/>
  </ResourceLinkPool>
  <ResourcePool>
    <LayoutIntent ID="r0048" Class="Intent" Status="Available"/>
    <ColorIntent ID="r0049" Class="Intent" Status="Available"/>
    <Component ID="r0047" Class="Quantity" Status="Available" SpawnIDs="Sp0057"/>
  </ResourcePool>
  <AncestorPool>
    <Ancestor NodeID="SpawnTest" FileName="testjdf4.jdf"/>
  </AncestorPool>
</JDF>

```

### V.3.4 Example 2 Component JDF after merging

In this example, it is assumed that the cover output component was created by some processor that processed the spawned node. This resulted in the Component becoming available. The Component was also removed from the copy of the spawned node, since it would otherwise exist twice.

```

<JDF ID="SpawnTest" Type="Product" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" Version="1.2" JobPartID="Part1">

```

```

<!--Generated by the CIP4 C++ open source JDF Library version CIP4 JDFWriter 1.0.01
beta-->
<AuditPool>
  <Created Author="CIP4 JDFWriter 1.0.01 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" Status="Unavailable"/>
  <BindingIntent ID="r0044" Class="Intent" Status="Available"/>
  <ProductionIntent ID="r0045" Class="Intent" Status="Available">
    <PrintProcess Range="Gravure" DataType="EnumerationSpan"/>
  </ProductionIntent>
  <Component ID="r0047" Class="Quantity" Status="Available"/>
  <Component ID="r0051" Class="Quantity" Status="Unavailable"/>
</ResourcePool>
<ResourceLinkPool>
  <ComponentLink rRef="r0043" Usage="Output"/>
  <BindingIntentLink rRef="r0044" Usage="Input"/>
  <ProductionIntentLink rRef="r0045" Usage="Input"/>
  <ComponentLink rRef="r0047" Usage="Input"/>
  <ComponentLink rRef="r0051" Usage="Input"/>
</ResourceLinkPool>
<JDF ID="n0046" Type="Product" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" Version="1.2" JobPartID="Part2" DescriptiveName="Cover">
  <!--Generated by the CIP4 C++ open source JDF Library version CIP4 JDFWriter 1.0.01
beta-->
  <AuditPool>
    <Created Author="CIP4 JDFWriter 1.0.01 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>

```

### V.3.5 Example of a Partitioned ImageSetting Node before Spawning

The following example shows a simple ImageSetting node that is partitioned by Separation. The resources are not filled with data. The input resources are Available.

```
<JDF ID="n20020701190951" Type="ImageSetting" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" Version="1.2">
  <!--Generated by the CIP4 C++ open source JDF Library version CIP4 JDFWriter 1.1.01
beta-->
  <ResourcePool>
    <ImageSetterParams ID="r0052" Class="Parameter" Locked="false" Status="Available"/>
    <Media ID="r0053" Class="Consumable" Locked="false" Status="Available"
PartIDKeys="Separation">
      <Media Separation="Cyan"/>
      <Media Separation="Magenta"/>
      <Media Separation="Yellow"/>
      <Media Separation="Black"/>
    </Media>
    <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>
```

### V.3.6 The Spawned Cyan Partition of the ImageSetting Node

The following example shows the spawned Cyan partition of the ImageSetting node from the previous example.

```
<JDF ID="n20020701190951" Type="ImageSetting" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting" SpawnID="Sp0059" Version="1.2">
  <!--Generated by the CIP4 C++ open source JDF Library version CIP4 JDFWriter 1.1.01
beta-->
  <AuditPool/>
  <ResourcePool>
    <ImageSetterParams ID="r0052" Class="Parameter" Locked="true" Status="Available"/>
    <Media ID="r0053" Class="Consumable" Locked="true" Status="Available"/>
    <ExposedMedia ID="r0054" Class="Handling" Locked="true" Status="Unavailable"
PartIDKeys="Separation">
      <ExposedMedia Separation="Cyan"/>
    </ExposedMedia>
    <RunList ID="r0055" Class="Parameter" Locked="true" Status="Available"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ImageSetterParamsLink rRef="r0052" Usage="Input"/>
    <MediaLink rRef="r0053" Usage="Input">
      <Part Separation="Cyan"/>
    </MediaLink>
    <ExposedMediaLink rRef="r0054" Usage="Output">
      <Part Separation="Cyan"/>
    </ExposedMediaLink>
  </ResourceLinkPool>
</JDF>
```



```

    <RunListLink rRef="r0055" Usage="Input"/>
  </ResourceLinkPool>
  <AncestorPool>
    <Part Separation="Cyan"/>
    <Ancestor Type="ImageSetting" xmlns="http://www.CIP4.org/JDFSchema_1_1"
NodeID="n20020701190951" Status="Waiting" Version="1.2" FileName="testjdf5.jdf"/>
  </AncestorPool>
</JDF>

```

### V.3.7 The Root Partitioned ImageSetting Node after Spawning

```

<JDF ID="n20020701190951" Type="ImageSetting" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Pool" Version="1.2">
  <!--Generated by the CIP4 C++ open source JDF Library version CIP4 JDFWriter 1.1.01
beta-->
  <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>
    <ImageSetterParams ID="r0052" Class="Parameter" Locked="false" Status="Available"
SpawnIDs="Sp0059" SpawnStatus="SpawnedRO"/>
    <Media ID="r0053" Class="Consumable" Locked="false" Status="Available"
SpawnIDs="Sp0059" PartIDKeys="Separation">
      <Media Locked="true" Separation="Cyan" SpawnStatus="SpawnedRW"/>
      <Media Separation="Magenta"/>
      <Media Separation="Yellow"/>
      <Media Separation="Black"/>
    </Media>
    <ExposedMedia ID="r0054" Class="Handling" Locked="false" Status="Unavailable"
SpawnIDs="Sp0059" PartIDKeys="Separation">
      <MediaRef rRef="r0053"/>
      <ExposedMedia Locked="true" Separation="Cyan" SpawnStatus="SpawnedRW"/>
      <ExposedMedia Separation="Magenta"/>
      <ExposedMedia Separation="Yellow"/>
      <ExposedMedia Separation="Black"/>
    </ExposedMedia>
    <RunList ID="r0055" Class="Parameter" Locked="false" Status="Available"
SpawnIDs="Sp0059" SpawnStatus="SpawnedRO"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ImageSetterParamsLink rRef="r0052" Usage="Input"/>
    <MediaLink rRef="r0053" Usage="Input"/>
    <ExposedMediaLink rRef="r0054" Usage="Output"/>
    <RunListLink rRef="r0055" Usage="Input"/>
  </ResourceLinkPool>
  <StatusPool Status="Waiting">
    <PartStatus Status="Spawned">
      <Part Separation="Cyan"/>
    </PartStatus>
  </StatusPool>
</JDF>

```

### V.3.8 The Merged ImageSetting Node

The Node has now been executed and merged.

```
<?xml version='1.0' encoding='utf-8' ?>
```

## Appendix V Examples

```
<JDF ID="n20020701190951" Type="ImageSetting" xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Pool" Version="1.2">
  <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" PartIDKeys="Separation">
      <Media Separation="Cyan" Status="Unavailable"/>
      <Media Separation="Magenta"/>
      <Media Separation="Yellow"/>
      <Media Separation="Black"/>
    </Media>
    <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"/>
  </ResourcePool>
  <ResourceLinkPool>
    <ImageSetterParamsLink rRef="r0052" Usage="Input"/>
    <MediaLink rRef="r0053" Usage="Input"/>
    <ExposedMediaLink rRef="r0054" Usage="Output"/>
    <RunListLink rRef="r0055" Usage="Input"/>
  </ResourceLinkPool>
  <StatusPool Status="Waiting">
    <PartStatus Status="Completed">
      <Part Separation="Cyan"/>
    </PartStatus>
  </StatusPool>
</JDF>
```

## V.4 Conversion of PJTF to JDF

### V.4.1 PJTF input

The following code defines 4-up duplex impositioning of a 17 page pdf document in Adobe PJTF format:

```
%JTF-1.2
1 0 obj
<<
/A [ 3 0 R ]
/V 1.1
/Cn [ 2 0 R ]
>>
endobj
2 0 obj
```

```
<<
/Type /JobTicketContents
/D [ 6 0 R ]
/PL 8 0 R
>>
endobj
3 0 obj
<<
/D (D:19991111173640)
/JTM (Default JT Creator)
/C (JT created)
>>
endobj
4 0 obj
<<
/Type /Catalog
/JT 1 0 R
>>
endobj
5 0 obj
<<
/Producer (HD PDFWrite vs. 0.1)
>>
endobj
6 0 obj
<<
/Fi [ 7 0 R ]
>>
endobj
7 0 obj
<<
/Fi (panrt17a.pdf)
>>
endobj
8 0 obj
<<
/Si 9 0 R
>>
endobj
9 0 obj
<<
/S 10 0 R
>>
endobj
10 0 obj
[ 11 0 R ]
endobj
11 0 obj
<<
/MS
<<
/C1 (sheet of paper)
/Me 12 0 R
>>
/Fr 13 0 R
/B 18 0 R
>>
endobj
12 0 obj
```

## Appendix V Examples

```
<<
/Dm [ 842 1191 842 1191 ]
>>
endobj
13 0 obj
<<
/PO [ 14 0 R 15 0 R 16 0 R 17 0 R ]
>>
endobj
14 0 obj
<<
/CTM [ 0.45 0 0 0.45 21 624 ]
/O 0
/C1 [ 21 624 399 1159 ]
>>
endobj
15 0 obj
<<
/CTM [ 0.45 0 0 0.45 442 624 ]
/O 1
/C1 [ 442 624 820 1159 ]
>>
endobj
16 0 obj
<<
/CTM [ 0.45 0 0 0.45 21 29 ]
/O 2
/C1 [ 21 29 399 564 ]
>>
endobj
17 0 obj
<<
/CTM [ 0.45 0 0 0.45 442 29 ]
/O 3
/C1 [ 442 29 820 564 ]
>>
endobj
18 0 obj
<<
/PO [ 19 0 R 20 0 R 21 0 R 22 0 R ]
>>
endobj
19 0 obj
<<
/CTM [ 0.45 0 0 0.45 21 624 ]
/O 4
/C1 [ 21 624 399 1159 ]
>>
endobj
20 0 obj
<<
/CTM [ 0.45 0 0 0.45 442 624 ]
/O 5
/C1 [ 442 624 820 1159 ]
>>
endobj
21 0 obj
<<
/CTM [ 0.45 0 0 0.45 21 29 ]
```

```

/O 6
/Cl [ 21 29 399 564 ]
>>
endobj
22 0 obj
<<
/CTM [ 0.45 0 0 0.45 442 29 ]
/O 7
/Cl [ 442 29 820 564 ]
>>
endobj
xref
0 23
0000000000 65535 f
0000000009 00000 n
0000000071 00000 n
0000000146 00000 n
0000000233 00000 n
0000000283 00000 n
0000000338 00000 n
0000000377 00000 n
0000000419 00000 n
0000000453 00000 n
0000000487 00000 n
0000000516 00000 n
0000000608 00000 n
0000000660 00000 n
0000000722 00000 n
0000000810 00000 n
0000000900 00000 n
0000000985 00000 n
0000001072 00000 n
0000001134 00000 n
0000001222 00000 n
0000001312 00000 n
0000001397 00000 n
trailer
<<
/Root 4 0 R
/Info 5 0 R
/Size 23
>>
startxref
1484
%%EOF

```

## V.4.2 JDF output

This JDF file describes the Imposition process defined by the PJTF file.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="PJTFJob" JobID="Job"
Status="Waiting" Type="Impositioning" Version="1.2">
  <!--Generated by the CIP4 C++ open source JDF Library version CIP4 JDFWriter 1.0.01
beta-->
  <ResourcePool>
    <Layout Class="Parameter" ID="Link0002" Status="Available">
      <Signature ID="Cos9">
        <SheetRef rRef="Cos11"/>
      </Signature>

```

```

    </Layout>
    <Surface ID="Cos13" Side="Front">
      <ContentObject CTM="0.45 0 0 0.45 21 624" ClipBox="21 624 399 1159" ID="Cos14"
Ord="0"/>
      <ContentObject CTM="0.45 0 0 0.45 442 624" ClipBox="442 624 820 1159" ID="Cos15"
Ord="1"/>
      <ContentObject CTM="0.45 0 0 0.45 21 29" ClipBox="21 29 399 564" ID="Cos16"
Ord="2"/>
      <ContentObject CTM="0.45 0 0 0.45 442 29" ClipBox="442 29 820 564" ID="Cos17"
Ord="3"/>
    </Surface>
    <Surface ID="Cos18" Side="Back">
      <ContentObject CTM="0.45 0 0 0.45 21 624" ClipBox="21 624 399 1159" ID="Cos19"
Ord="4"/>
      <ContentObject CTM="0.45 0 0 0.45 442 624" ClipBox="442 624 820 1159" ID="Cos20"
Ord="5"/>
      <ContentObject CTM="0.45 0 0 0.45 21 29" ClipBox="21 29 399 564" ID="Cos21"
Ord="6"/>
      <ContentObject CTM="0.45 0 0 0.45 442 29" ClipBox="442 29 820 564" ID="Cos22"
Ord="7"/>
    </Surface>
    <Sheet ID="Cos11">
      <SurfaceRef rRef="Cos18"/>
      <SurfaceRef rRef="Cos13"/>
    </Sheet>
    <Media Dimensions="842 1191 842 1191" ID="Cos12"/>
    <RunList Class="Parameter" ID="Link0003" NPage="17" Pages="0~16"
Status="Available">
      <LayoutElement>
        <FileSpec URL="File:///panrt17a.pdf"/>
      </LayoutElement>
    </RunList>
    <RunList Class="Parameter" ID="Link0004" Status="Unavailable"/>
  </ResourcePool>
  <ResourceLinkPool>
    <RunListLink Usage="Input" rRef="Link0003"/>
    <LayoutLink Usage="Input" rRef="Link0002"/>
    <RunListLink Usage="Output" rRef="Link0004"/>
  </ResourceLinkPool>
  <AuditPool>
    <Created Author="PJTF2JDF" TimeStamp="2000-11-07T17:42:15+01:00"/>
  </AuditPool>
</JDF>

```

## V.5 Conversion of PPF to JDF

Simple example of a PPF.

```

%!PS-Adobe-3.0
%%CIP3-File Version 2.0

CIP3BeginSheet
(This example was manually created by Stefan Daun) CIP3Comment
/CIP3AdmJobName (8 pages with workturn and 5 color separations) def
/CIP3AdmSoftware (Text editor) def
/CIP3AdmCreationTime (Wed Feb 19 12:00:00 1997) def
/CIP3AdmArtist (Joerg Zedler) def
/CIP3AdmCopyright (Copyright by Fraunhofer-IGD, 1997) def

```

```
/CIP3AdmSheetName (E08P5C) def
/CIP3AdmSheetLay /Left def
/CIP3AdmPSExtent [ 40 inch 27 inch ] def
/CIP3TransferFilmCurveData [0.0 0.0 1.0 1.0 ] def
/CIP3TransferPlateCurveData [0.0 0.0 1.0 1.0 ] def
/CIP3AdmFilmTrf [0 1 -1 0 1944 0] def
/CIP3AdmPlateTrf [0 -1 1 0 0 2880] def
CIP3BeginFront
/CIP3AdmSeparationNames [ (Cyan) (Magenta) (Yellow) (Black) (PANTONE Green CV)] def
```

```
CIP3BeginPreviewImage
```

```
CIP3BeginSeparation
(First separation of Front) CIP3Comment
/CIP3PreviewImageWidth 2030 def
/CIP3PreviewImageHeight 1370 def
/CIP3PreviewImageBitsPerComp 8 def
/CIP3PreviewImageComponents 1 def
/CIP3PreviewImageMatrix [0 1370 -2030 0 1370 0] def
/CIP3PreviewImageResolution [50.75 50.75] def
/CIP3PreviewImageEncoding /Binary def
/CIP3PreviewImageCompression /RunLengthDecode def
/CIP3PreviewImageByteAlign 4 def
CIP3PreviewImage
... <image data>
CIP3EndSeparation
```

```
CIP3BeginSeparation
(Second separation of Front) CIP3Comment
/CIP3PreviewImageWidth 2030 def
/CIP3PreviewImageHeight 1370 def
/CIP3PreviewImageBitsPerComp 8 def
/CIP3PreviewImageComponents 1 def
/CIP3PreviewImageMatrix [0 1370 -2030 0 1370 0] def
/CIP3PreviewImageResolution [50.75 50.75] def
/CIP3PreviewImageEncoding /Binary def
/CIP3PreviewImageCompression /RunLengthDecode def
/CIP3PreviewImageByteAlign 4 def
CIP3PreviewImage
... <image data>
CIP3EndSeparation
```

```
CIP3BeginSeparation
(Fourth separation of Front) CIP3Comment
/CIP3PreviewImageWidth 2030 def
/CIP3PreviewImageHeight 1370 def
/CIP3PreviewImageBitsPerComp 8 def
/CIP3PreviewImageComponents 1 def
/CIP3PreviewImageMatrix [0 1370 -2030 0 1370 0] def
/CIP3PreviewImageResolution [50.75 50.75] def
/CIP3PreviewImageEncoding /Binary def
/CIP3PreviewImageCompression /RunLengthDecode def
/CIP3PreviewImageByteAlign 4 def
CIP3PreviewImage
... <image data>
CIP3EndSeparation
```

```
CIP3BeginSeparation
(Fifth separation of Front) CIP3Comment
```

## Appendix V Examples

```
/CIP3PreviewImageWidth 2030 def
/CIP3PreviewImageHeight 1370 def
/CIP3PreviewImageBitsPerComp 8 def
/CIP3PreviewImageComponents 1 def
/CIP3PreviewImageMatrix [0 1370 -2030 0 1370 0] def
/CIP3PreviewImageResolution [50.75 50.75] def
/CIP3PreviewImageEncoding /Binary def
/CIP3PreviewImageCompression /RunLengthDecode def
/CIP3PreviewImageByteAlign 4 def
CIP3PreviewImage

CIP3BeginSeparation
(Second separation of Front) CIP3Comment
/CIP3PreviewImageWidth 2030 def
/CIP3PreviewImageHeight 1370 def
/CIP3PreviewImageBitsPerComp 8 def
/CIP3PreviewImageComponents 1 def
/CIP3PreviewImageMatrix [0 1370 -2030 0 1370 0] def
/CIP3PreviewImageResolution [50.75 50.75] def
/CIP3PreviewImageEncoding /Binary def
/CIP3PreviewImageCompression /RunLengthDecode def
/CIP3PreviewImageByteAlign 4 def
CIP3PreviewImage
... <image data>
CIP3EndSeparation
CIP3EndSeparation
CIP3EndPreviewImage

CIP3BeginRegisterMarks
20 inch 0 0 /cross&circle CIP3PlaceRegisterMark
CIP3EndRegisterMarks

CIP3BeginColorControl
/C100 << /CIE-L* 62 /CIE-a* -31 /CIE-b* -48 /Diameter 4.7 mm /Light /D65 /
Observer 2 /Tolerance 5 /Type /CIELAB >> def
/M100 << /CIE-L* 48 /CIE-a* 83 /CIE-b* -3 /Diameter 4.7 mm /Light /D65 /
Observer 2 /Tolerance 5 /Type /CIELAB >> def
/Y100 << /CIE-L* 94 /CIE-a* -14 /CIE-b* 100 /Diameter 4.7 mm /Light /D65 /
Observer 2 /Tolerance 5 /Type /CIELAB >> def
/K100 << /CIE-L* 0 /CIE-a* 0 /CIE-b* 0 /Diameter 4.7 mm /Light /D65 /Observer
2 /Tolerance 5 /Type /CIELAB >> def
0 0 0 360 18
[
[ 14.77 0 C100 ]
[ 41.85 0 Y100 ]
[ 68.92 0 M100 ]
[ 177.23 0 K100 ]

] /PrepsColorBar CIP3PlaceColorControlStrip
CIP3EndColorControl

CIP3BeginCutData
CIP3BeginCutBlock
/CIP3BlockTrf [1 0 0 1 44 mm 45.9 mm] def
/CIP3BlockSize [ 420 mm 594 mm] def
/CIP3BlockType /CutBlock def
/CIP3BlockName (Front Sides) def
/CIP3BlockFoldingProcedure /F08-07_li_2x2_1 def
CIP3EndCutBlock
```



```

CIP3BeginCutBlock
/CIP3BlockTrf [1 0 0 1 552 mm 45.9 mm] def
/CIP3BlockSize [ 420 mm 594 mm] def
/CIP3BlockType /CutBlock def
/CIP3BlockName (Back Sides) def
/CIP3BlockFoldingProcedure /F08-07_li_2x2_1 def
400 400 /RightHorizontalCutMark CIP3PlaceCutMark
CIP3EndCutBlock
100 200 /TopVerticalCutMark CIP3PlaceCutMark
CIP3EndCutData
CIP3BeginFoldProcedures
/F08-07_li_2x2_1 <<
  /CIP3FoldDescription (F8-7)
  /CIP3FoldSheetIn [210 mm 297 mm]
  /CIP3FoldProc
  [
    297.638 /Front /Up Fold
    420.945 /Left /Up Fold
  ]
  >> def
CIP3EndFoldProcedures
CIP3EndFront
CIP3EndSheet
%%CIP3EndOfFile

```

The translated JDF:

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="PPFJDF" JobID="MyJob"
Status="Waiting" Type="Product" Version="1.2">
  <!--Generated by the CIP4 C++ open source JDF Library version CIP4 JDFWriter 1.0.01
beta-->
  <JDF ID="n1152" Status="Waiting" Type="InkZoneCalculation">
    <ResourceLinkPool>
      <LayoutLink Usage="Input" rRef="r1106"/>
      <PreviewLink Usage="Input" rRef="r1116"/>
      <TransferCurvePoolLink Usage="Input" rRef="r1111"/>
      <InkZoneCalculationParamsLink Usage="Input" rRef="r1118"/>
      <InkZoneProfileLink Usage="Output" rRef="r1119"/>
    </ResourceLinkPool>
    <ResourcePool>
      <Layout Class="Parameter" ID="r1106" Status="Available">
        <Signature Name="HDM">
          <SheetRef rRef="r1107"/>
        </Signature>
      </Layout>
      <Sheet Class="Parameter" ID="r1107" Name="E08P5C" Status="Unavailable"
SurfaceContentsBox="0 0 2880 1944">
        <SurfaceRef rRef="r1112"/>
      </Sheet>
      <Surface Class="Parameter" ID="r1112" Side="Front" Status="Unavailable">
        <MarkObject CTM="1 0 0 1 0 0" Type="Mark">
          <ColorControlStripRef rRef="r1114"/>
        </MarkObject>
        <MarkObject CTM="1 0 0 1 0 0">
          <RegisterMarkRef rRef="r1115"/>
        </MarkObject>
        <MarkObject CTM="1 0 0 1 0 0" Type="Mark">
          <CutMarkRef rRef="r1130"/>
        </MarkObject>

```

## Appendix V Examples

```
<MarkObject CTM="1 0 0 1 0 0" Type="Mark">
  <CutMarkRef rRef="r1134"/>
</MarkObject>
</Surface>
<ColorControlStrip Center="0 0" Class="Parameter" ID="r1114" Rotation="0"
Size="360 18" Status="Unavailable">
  <CIELABMeasuringField CIE_Lab="62 -31 -48" Center="14.77 0"
Diameter="13.3228346457" Observer="2" Tolerance="5">
    <ColorMeasurementConditionsRef rRef="RCMC"/>
  </CIELABMeasuringField>
  <CIELABMeasuringField CIE_Lab="94 -14 100" Center="41.85 0"
Diameter="13.3228346457" Tolerance="5">
    <ColorMeasurementConditionsRef rRef="RCMC"/>
  </CIELABMeasuringField>
  <CIELABMeasuringField CIE_Lab="48 83 -3" Center="68.92 0"
Diameter="13.3228346457" Tolerance="5">
    <ColorMeasurementConditionsRef rRef="RCMC"/>
  </CIELABMeasuringField>
  <CIELABMeasuringField CIE_Lab="0 0 0" Center="177.23 0" Diameter="13.3228346457"
Tolerance="5">
    <ColorMeasurementConditionsRef rRef="RCMC"/>
  </CIELABMeasuringField>
</ColorControlStrip>
<ColorMeasurementConditions ID="RCMC" Illumination="D65" Observer="2"/>
<RegisterMark Center="1440 0" Class="Parameter" ID="r1115"
MarkType="cross&circle" Rotation="0" Status="Unavailable"/>
<CutMark Class="Parameter" ID="r1130" MarkType="TopVerticalCutMark" Position="100
200" Status="Available"/>
<CutMark Blocks="Back_Sides" Class="Parameter" ID="r1134"
MarkType="RightHorizontalCutMark" Position="400 400" Status="Available"/>
<Preview Class="Parameter" ID="r1116" PartIDKeys="SheetName Side Separation"
PreviewType="Separation" Status="Available">
  <Preview SheetName="E08P5C">
    <Preview Side="Front">
      <Preview Separation="Cyan" URL="File:///Bild0000.png"/>
      <Preview Separation="Magenta" URL="File:///Bild0001.png"/>
      <Preview Separation="Yellow" URL="File:///Bild0002.png"/>
      <Preview Separation="Black" URL="File:///Bild0003.png"/>
      <Preview Separation="PANTONE Green CV" URL="File:///Bild0004.png"/>
    </Preview>
  </Preview>
</Preview>
<TransferCurvePool Class="Parameter" ID="r1111" Status="Available">
  <TransferCurveSet CTM="0 1 -1 0 1944 0" Name="Film">
    <TransferCurve Curve="0 0 1 1"/>
  </TransferCurveSet>
  <TransferCurveSet CTM="1 0 0 1 0 0" Name="Press">
    <TransferCurve Curve="0 0 1 1"/>
  </TransferCurveSet>
  <TransferCurveSet CTM="0 -1 1 0 0 2880" Name="Plate"/>
  <TransferCurveSet CTM="1 0 0 1 0 0" Name="Paper"/>
</TransferCurvePool>
<InkZoneCalculationParams Class="Parameter" ID="r1118" Status="Available"/>
<InkZoneProfile Class="Parameter" ID="r1119" Status="Unavailable"/>
</ResourcePool>
</JDF>
<JDF ID="n1153" Status="Waiting" Type="ConventionalPrinting">
  <ResourceLinkPool>
    <LayoutLink Usage="Input" rRef="r1106"/>
  </ResourceLinkPool>
</JDF>
```

```

    <ColorantControlLink Usage="Input" rRef="r1113"/>
    <InkZoneProfileLink Usage="Input" rRef="r1119"/>
    <ComponentLink ProcessUsage="Good" Usage="Output" rRef="r1125"/>
    <MediaLink Usage="Input" rRef="r1108"/>
    <ConventionalPrintingParamsLink Usage="Input" rRef="r1126"/>
    <InkLink Usage="Input" rRef="r1127"/>
    <ExposedMediaLink Usage="Input" rRef="r1123"/>
  </ResourceLinkPool>
  <ResourcePool>
    <ColorantControl Class="Parameter" ID="r1113" PartIDKeys="SheetName Side"
    ProcessColorModel="DeviceCMYK" Status="Available">
      <ColorantControl SheetName="E08P5C">
        <ColorantControl Side="Front">
          <ColorantParams>
            <SeparationSpec Name="PANTONE Green CV"/>
          </ColorantParams>
          <ColorantOrder>
            <SeparationSpec Name="Cyan"/>
            <SeparationSpec Name="Magenta"/>
            <SeparationSpec Name="Yellow"/>
            <SeparationSpec Name="Black"/>
            <SeparationSpec Name="PANTONE Green CV"/>
          </ColorantOrder>
        </ColorantControl>
      </ColorantControl>
    </ColorantControl>
    <Component Class="Quantity" ID="r1125" PartIDKeys="SheetName"
    Status="Unavailable">
      <Component ComponentType="Sheet" SheetName="E08P5C">
        <SheetRef rRef="r1107"/>
      </Component>
    </Component>
    <Media Class="Consumable" ID="r1108" MediaType="Paper" PartIDKeys="SheetName
    Side" Status="Available">
      <Media Dimension="2880 1944" SheetName="E08P5C">
        <Media Dimension="2880 1944" Side="Front"/>
      </Media>
    </Media>
    <ConventionalPrintingParams Class="Parameter" ID="r1126" PartIDKeys="SheetName
    Side" Status="Available">
      <ConventionalPrintingParams SheetLay="Left" SheetName="E08P5C">
        <ConventionalPrintingParams Side="Front"/>
      </ConventionalPrintingParams>
    </ConventionalPrintingParams>
    <Ink Class="Consumable" ID="r1127" Status="Draft"/>
    <ExposedMedia Class="Handling" ID="r1123" Status="Unavailable">
      <MediaRef rRef="r1110"/>
    </ExposedMedia>
    <Media Class="Consumable" ID="r1110" MediaType="Plate" PartIDKeys="SheetName
    Side" Status="Available">
      <Media Dimension="2880 1944" SheetName="E08P5C">
        <Media Dimension="2880 1944" Side="Front"/>
      </Media>
    </Media>
  </ResourcePool>
</JDF>

<JDF ID="n1154" Status="Waiting" Type="Cutting">
  <ResourceLinkPool>

```

## Appendix V Examples

```

    <ComponentLink Usage="Input" rRef="r1125">
      <Part SheetName="E08P5C"/>
    </ComponentLink>
    <CuttingParamsLink Usage="Input" rRef="r1129"/>
    <ComponentLink Usage="Output" rRef="r1131"/>
  </ResourceLinkPool>
  <ResourcePool>
    <CuttingParams Class="Parameter" ID="r1129" Status="Available">
      <CutMarkRef rRef="r1130"/>
      <CutBlockRef rRef="r1132"/>
      <CutBlockRef rRef="r1133"/>
      <CutMarkRef rRef="r1134"/>
    </CuttingParams>
    <CutBlock BlockName="Front_Sides" BlockSize="1190.55118111 1683.77952756"
BlockTrf="1 0 0 1 124.724409449 130.110236221" BlockType="CutBlock" Class="Parameter"
ID="r1132" Status="Available"/>
    <CutBlock BlockName="Back_Sides" BlockSize="1190.55118111 1683.77952756"
BlockTrf="1 0 0 1 1564.72440945 130.110236221" BlockType="CutBlock" Class="Parameter"
ID="r1133" Status="Available"/>
    <Component Class="Quantity" ID="r1131" PartIDKeys="BlockName"
Status="Unavailable">
      <Component BlockName="Front_Sides" ComponentType="Block" SourceSheet="E08P5C">
        <SheetRef rRef="r1107"/>
      </Component>
      <Component BlockName="Back_Sides" ComponentType="Block" SourceSheet="E08P5C">
        <SheetRef rRef="r1107"/>
      </Component>
    </Component>
  </ResourcePool>
</JDF>
<JDF ID="n1155" Status="Waiting" Type="ImageSetting">
  <ResourceLinkPool>
    <ImageSetterParamsLink Usage="Input" rRef="r1121"/>
    <MediaLink Usage="Input" rRef="r1110"/>
    <RunListLink Usage="Input" rRef="r1122"/>
    <ExposedMediaLink Usage="Output" rRef="r1123"/>
  </ResourceLinkPool>
  <ResourcePool>
    <ImageSetterParams Class="Parameter" ID="r1121" Status="Available"/>
    <RunList Class="Parameter" ID="r1122" Status="Available"/>
  </ResourcePool>
</JDF>
<JDF ID="n1158" Status="Waiting" Type="Folding">
  <ResourceLinkPool>
    <FoldingParamsLink Usage="Input" rRef="r1136"/>
    <ComponentLink Usage="Input" rRef="r1131">
      <Part BlockName="Front_Sides"/>
    </ComponentLink>
    <ComponentLink Usage="Output" rRef="r1138"/>
  </ResourceLinkPool>
  <ResourcePool>
    <FoldingParams Class="Parameter" DescriptionType="FoldProc" ID="r1136"
Status="Available">
      <Fold From="Front" To="Up" Travel="297.638"/>
      <Fold From="Left" To="Up" Travel="420.945"/>
    </FoldingParams>
    <Component Class="Quantity" ComponentType="Block" DescriptiveName="Front_Sides"
ID="r1138" Status="Unavailable"/>
  </ResourcePool>

```

```

</JDF>
<JDF ID="n1159" Status="Waiting" Type="Folding">
  <ResourceLinkPool>
    <FoldingParamsLink Usage="Input" rRef="r1140"/>
    <ComponentLink Usage="Input" rRef="r1131">
      <Part BlockName="Back_Sides"/>
    </ComponentLink>
    <ComponentLink Usage="Output" rRef="r1142"/>
  </ResourceLinkPool>
  <ResourcePool>
    <FoldingParams Class="Parameter" DescriptionType="FoldProc" ID="r1140"
Status="Available">
      <Fold From="Front" To="Up" Travel="297.638"/>
      <Fold From="Left" To="Up" Travel="420.945"/>
    </FoldingParams>
    <Component Class="Quantity" ComponentType="Block" DescriptiveName="Back_Sides"
ID="r1142" Status="Unavailable"/>
  </ResourcePool>
</JDF>
</JDF>

```

## V.6 RunList

The following example shows the various separation types, all mixed into one big RunList. Both in-line and ResourceRef versions of **LayoutElement** are used.

```

<ResourcePool>
  <Runlist Class="Parameter" ID="Link0003" NPage="10" PartIDKeys="Run Separation"
Status="Available">
    <Comment>Preseparated Runs in multiple files
    All LayoutElements are inline resources
    </Comment>
    <RunList FirstPage="0" NPage="1" Run="1">
      <RunList Separation="Cyan">
        <LayoutElement Status="Unavailable">
          <FileSpec URL="File:///Cyan.pdf"/>
        </LayoutElement>
      </RunList>
      <RunList Separation="Magenta">
        <LayoutElement Status="Unavailable">
          <FileSpec URL="File:///Magenta.pdf"/>
        </LayoutElement>
      </RunList>
      <RunList Separation="Yellow">
        <LayoutElement Status="Unavailable">
          <FileSpec URL="File:///Yellow.pdf"/>
        </LayoutElement>
      </RunList>
      <RunList Separation="Black">
        <LayoutElement Status="Unavailable">
          <FileSpec URL="File:///Black.pdf"/>
        </LayoutElement>
      </RunList>
      <RunList Separation="SpotGreen">
        <LayoutElement Status="Unavailable">
          <FileSpec URL="File:///Green.pdf"/>
        </LayoutElement>
      </RunList>
    </RunList>
    <RunList NPage="2" Run="2" SkipPage="4">

```

```

<Comment>
Preseparated Runs in one file CMYKGCMYKG
LayoutElements are inter-resource links
</Comment>
<RunList FirstPage="0" Separation="Cyan">
  <LayoutElementRef rRef="Link0004"/>
</RunList>
<RunList FirstPage="1" Separation="Magenta">
  <LayoutElementRef rRef="Link0004"/>
</RunList>
<RunList 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 CCMYYKGG
    </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:///PreSepCCMYYKGG.pdf"/>
</LayoutElement>
</ResourcePool>

```

## V.7 Messages

### V.7.1 Simple KnownMessages

The following simple example shows a KnownMessages Query and the Response sent by a fairly dumb controller:

Query:

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFCClient" TimeStamp="2000-11-07T13:15:56+01:00" Version="1.2">
  <Query ID="Q0001" Type="KnownMessages">
    <KnownMsgQuParams ListCommands="true" ListQueries="true" ListSignals="false"/>
  </Query>
</JMF>

```

Response:

```

<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="JMFCClient #2" TimeStamp="2000-11-07T13:15:56+01:00" Version="1.2">
  <Response ID="R0001" Type="KnownMessages" refID="Q0001">
    <KnownMessages>
      <MessageService Query="true" Type="KnownMessages"/>
    </KnownMessages>
  </Response>
</JMF>

```

```

    <MessageService Persistent="true" Query="true" Type="Status"/>
    <MessageService Command="true" Type="StopPersistentChannel"/>
  </KnownMessages>
</Response>
</JMF>

```

## V.7.2 Simple persistent channel

The following query requests a persistent channel for Status messages. An update is requested whenever an attribute changes.

```

<JMF xmlns="http://www.CIP4.org/JDFSchemas_1_1" SenderID="JMFCClient" TimeStamp="2000-
11-07T16:02:09+01:00" Version="1.2">
  <Query ID="Q0011" Type="Status">
    <Subscription URL="http://123.123.123.123/message/recipient">
      <ObservationTarget Attributes="*" />
    </Subscription>
    <StatusQuParams JobDetails="brief" />
  </Query>
</JMF>

```

The following four examples are a set of typical, simple responses that are emitted whenever *DeviceStatus* changes.

This is the **Response** that is sent immediately within the same HTTP connection as the **Query**.

```

<JMF xmlns="http://www.CIP4.org/JDFSchemas_1_1" SenderID="JMFCClient #2" TimeStamp="2000-
11-07T16:02:19+01:00" Version="1.2">
  <Response ID="R0013" Type="Status" refID="Q0011">
    <DeviceInfo DeviceStatus="Idle" />
  </Response>
</JMF>

```

This is an intermediate **Signal** that was emitted when *DeviceStatus* changed.

```

<JMF xmlns="http://www.CIP4.org/JDFSchemas_1_1" SenderID="JMFCClient #2" TimeStamp="2000-
11-07T17:02:19+01:00" Version="1.2">
  <Signal ID="Q0015" Type="Status" refID="Q0011">
    <DeviceInfo DeviceStatus="Setup" />
  </Signal>
</JMF>

```

This is an intermediate **Signal** that was emitted when *DeviceStatus* changed.

```

<JMF xmlns="http://www.CIP4.org/JDFSchemas_1_1" SenderID="JMFCClient #2" TimeStamp="2000-
11-07T17:08:19+01:00" Version="1.2">
  <Signal ID="Q0017" Type="Status" refID="Q0011">
    <DeviceInfo DeviceStatus="Running" />
  </Signal>
</JMF>

```

This is the last **Signal** of the persistent channel.

```

<JMF xmlns="http://www.CIP4.org/JDFSchemas_1_1" SenderID="JMFCClient #2" TimeStamp="2000-
11-07T19:02:19+01:00" Version="1.2">
  <Signal ID="Q0017" Type="Status" refID="Q0011" LastRepeat="true">
    <DeviceInfo DeviceStatus="Idle" />
  </Signal>
</JMF>

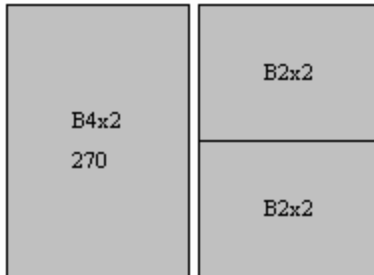
```



## V.8 Stripping

### V.8.1 Using Position

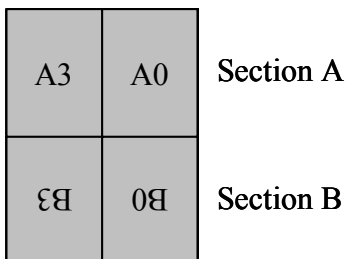
The following example illustrates the more advanced use of the *Position* object. Note that the two B2x2 signatures are filled independently.



```
<BinderySignature Class="Parameter" ID="B4x2" NumberUp="4 2" Status="Available"/>
<BinderySignature Class="Parameter" ID="B2x2" NumberUp="2 2" Status="Available"/>
<StrippingParams Class="Parameter" ID="L1" PartIDKeys="SheetName
BinderySignatureName" Status="Available" WorkStyle="WorkAndBack">
  <StrippingParams SheetName="Sheet1">
    <StrippingParams BinderySignatureName="B4x2">
      <BinderySignatureRef rRef="B4x2"/>
      <Position RelativeBox="0 0 0.5 1" Rotate="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>
```

### V.8.2 Multiple BinderySignatures

The following example illustrates how two identical **BinderySignatures** that represent the same sections are placed onto a surface. It also shows how **StripCellParams** are overwritten for various sections.



```
<BinderySignature Class="Parameter" ID="B4x2" NumberUp="4 2" Status="Available"/>
<BinderySignature Class="Parameter" ID="B2x2" NumberUp="2 2" Status="Available"/>
<StrippingParams Class="Parameter" ID="L1" PartUsage="Implicit" Status="Available"
PartIDKeys="SheetName BinderySignatureName CellIndex" WorkStyle="WorkAndBack">
  <StrippingParams JobID="Customer Job 1" SheetName="Sheet1">
    <StrippingParams BinderySignatureName="B4x2">
```

```

    <BinderySignatureRef rRef="B4x2"/>
    <StripCellParams BleedFace="42" BleedSpine="0" MillingDepth="21"/>
    <Position RelativeBox="0 0 0.5 1" Rotate="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>

```

### V.8.3 Multisection BinderySignatures

The following example illustrates the imposition of a job containing 80 pages using *ComeAndGo*. Five sheets need to be produced each containing two sections.

B5	B4	B1	B8
A4	A5	A8	A1

```

<BinderySignature Class="Parameter" ID="ComeAndGo" NumberUp="4 2" Status="Available">
  <SignatureCell BackPages="1 6 5 2" FrontPages="3 4 7 0" Orientation="Up"
SectionIndex="0"/>
  <SignatureCell BackPages="6 1 2 5" FrontPages="4 3 0 7" Orientation="Down"
SectionIndex="1"/>
</BinderySignature>
<StrippingParams Class="Parameter" ID="L1" PartIDKeys="SheetName" Status="Available"
WorkStyle="WorkAndBack">
  <BinderySignatureRef rRef="ComeAndGo"/>
  <StrippingParams SectionList="0 9" SheetName="Sheet1"/>
  <StrippingParams SectionList="1 8" SheetName="Sheet2"/>
  <StrippingParams SectionList="2 7" SheetName="Sheet3"/>
  <StrippingParams SectionList="3 6" SheetName="Sheet4"/>
  <StrippingParams SectionList="4 5" SheetName="Sheet5"/>
</StrippingParams>

```

### V.8.4 Multiple job parts in one imposition

The following example illustrates partitioning by *SectionIndex*. We reuse the *ComeAndGo* **BinderySignature** from the previous example, but map the **BinderySignature** to sections of different job parts.

```

<StrippingParams Class="Parameter" ID="L1" JobID="MyJob" PartIDKeys="SheetName
SectionIndex" Status="Available" WorkStyle="WorkAndBack">
  <BinderySignatureRef rRef="ComeAndGo"/>
  <StrippingParams SheetName="Sheet1">
    <StrippingParams AssemblyID="Book1" SectionIndex="0" SectionList="0"/>
    <StrippingParams AssemblyID="Book2" SectionIndex="1" SectionList="9"/>
  </StrippingParams>
  <StrippingParams SheetName="Sheet2">

```

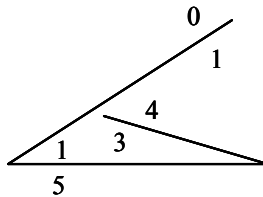
```

    <StrippingParams AssemblyID="Book1" SectionIndex="0" SectionList="1"/>
    <StrippingParams AssemblyID="Book2" SectionIndex="1" SectionList="8"/>
</StrippingParams>
<StrippingParams SheetName="Sheet3">
    <StrippingParams AssemblyID="Book1" SectionIndex="0" SectionList="2"/>
    <StrippingParams AssemblyID="Book2" SectionIndex="1" SectionList="7"/>
</StrippingParams>
<StrippingParams SheetName="Sheet4">
    <StrippingParams AssemblyID="Book1" SectionIndex="0" SectionList="3"/>
    <StrippingParams AssemblyID="Book2" SectionIndex="1" SectionList="6"/>
</StrippingParams>
<StrippingParams SheetName="Sheet5">
    <StrippingParams AssemblyID="Book1" SectionIndex="0" SectionList="4"/>
    <StrippingParams AssemblyID="Book2" SectionIndex="1" SectionList="5"/>
</StrippingParams>
</StrippingParams>

```

### V.8.5 FoldOuts

The following example illustrates the use of foldouts. The same foldout is placed twice on a press sheet.



4	5	0
4	5	0

```

<BinderySignature Class="Parameter" ID="foldout" NumberUp="2 1" Status="Available">
    <SignatureCell BackFacePages="3" BackPages="2" FrontFacePages="4" FrontPages="5"
Orientation="Up"/>
    <SignatureCell BackPages="1" FrontPages="0" Orientation="Up"/>
</BinderySignature>
<StrippingParams SheetName="Cover" WorkStyle="WorkAndBack">
    <BinderySignatureRef rRef="foldout"/>
    <Position RelativeBox="0 0 1 0.5"/>
    <Position RelativeBox="0 0.5 1 1"/>
</StrippingParams>

```

### V.8.6 Multiple Web Layout

The following example illustrates a regular double-web layout. A double-web **BinderySignature** is used in two signatures. This results in four sheets.

91	51	8	£7
31	0	7	24

**Web1**

81	£1	01	17
29	2	5	26

**Web2**

```
<BinderySignature Class="Parameter" ID="B001" NumberUp="4 2" PartIDKeys="WebName"
Status="Available">
  <BinderySignature WebName="Web1">
    <SignatureCell BackPages="22 9 14 17" FrontPages="31 0 7 24" Orientation="Up"/>
    <SignatureCell BackPages="25 6 1 30" FrontPages="16 15 8 23" Orientation="Down"/>
  </BinderySignature>
  <BinderySignature WebName="Web2">
    <SignatureCell BackPages="20 11 12 19" FrontPages="29 2 5 26" Orientation="Up"/>
    <SignatureCell BackPages="27 4 3 28" FrontPages="18 13 10 21" Orientation="Down"/>
  </BinderySignature>
</BinderySignature>
<StrippingParams Class="Parameter" ID="MultiWeb1" PartIDKeys="SignatureName SheetName"
Status="Available" WorkStyle="WorkAndBack">
  <StrippingParams SignatureName="Signature1">
    <StrippingParams SheetName="Sheet1">
      <BinderySignatureRef rRef="B001">
        <Part WebName="Web1"/>
      </BinderySignatureRef>
    </StrippingParams>
    <StrippingParams SheetName="Sheet2">
      <BinderySignatureRef rRef="B001">
        <Part WebName="Web2"/>
      </BinderySignatureRef>
    </StrippingParams>
  </StrippingParams>
  <StrippingParams SignatureName="Signature2">
    <StrippingParams SheetName="Sheet3">
      <BinderySignatureRef rRef="B001">
        <Part WebName="Web1"/>
      </BinderySignatureRef>
    </StrippingParams>
    <StrippingParams SheetName="Sheet4">
      <BinderySignatureRef rRef="B001">
        <Part WebName="Web2"/>
      </BinderySignatureRef>
    </StrippingParams>
  </StrippingParams>
</StrippingParams>
```

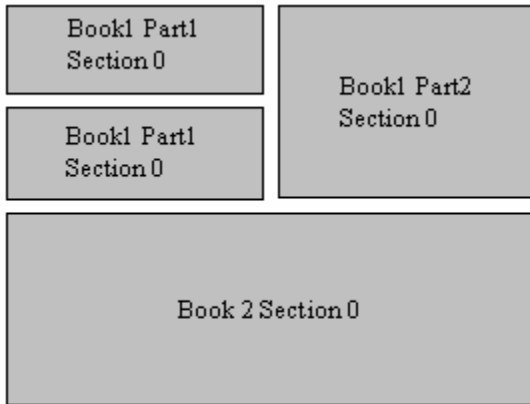
```

    </StrippingParams>
  </StrippingParams>
</StrippingParams>

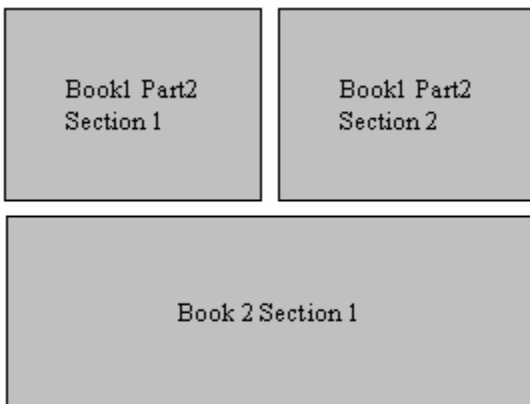
```

### V.8.7 Stripping Process

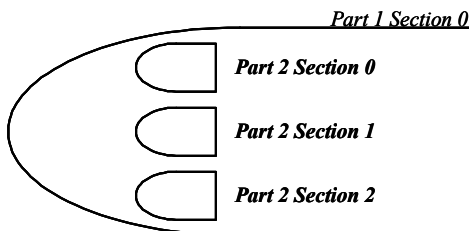
The next sample illustrates the **Stripping** process and its **StrippingParams** and **Assembly** resources. Sheet1:



Sheet2:



#### Assembly 1



```

<JDF ID="n001" Type="Stripping" Version="1.2">
  <ResourcePool>
    <BinderySignature Class="Parameter" FoldCatalog="F4-1" ID="F4-1"
      Status="Available"/>

```

## Appendix V Examples

```
<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 AssemblyID="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 AssemblyID="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 AssemblyID="Part2" BinderySignatureName="F8-7_1" JobID="Book1"
SectionList="1">
      <BinderySignatureRef rRef="F8-7"/>
      <Position RelativeBox="0 0.5 0.5 1"/>
    </StrippingParams>
    <StrippingParams AssemblyID="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>
</AssemblySection>
</Assembly>
<Assembly Class="Parameter" ID="A1" JobID="Book1" Order="List" Status="Available">
  <AssemblySection AssemblyID="Part1" Order="Gathering">
    <AssemblySection AssemblyID="Part2"/>
    <AssemblySection AssemblyID="Part2"/>
    <AssemblySection AssemblyID="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>
```

## V.9 DigitalDelivery Examples

**Example 1:** Instruct the digital delivery device to compress the files delivered in gzip compression.

**Before the delivery:**

```
<ResourcePool>
  <RunList ID="SourceFilesLink" Class="Parameter" Status="Available">
    <LayoutElement>
      <FileSpec URL="File:///e:/ToSend/xxx.pdf"/>
    </LayoutElement>
  </RunList>

  <RunList ID="TargetFilesLink" Class="Parameter" Status="Unavailable">
    <LayoutElement>
      <FileSpec Compression="Gzip"/>
    </LayoutElement>
  </RunList>
</ResourcePool>

...
<ResourceLinkPool>
  <RunListLink rRef="SourceFilesLink" Usage="Input"/>
  <RunListLink rRef="TargetFilesLink" Usage="Output"/>
</ResourceLinkPool>
```

Since the input **RunList** resource is without *Compression* and the output **RunList** resource is with *Compression* — it will instruct the digital delivery device to compress the files delivered.

**After the delivery:**

```
<RunList ID="TargetFilesLink" Class="Parameter" Status="Available">
  <LayoutElement>
    <FileSpec Compression="Gzip" URL="File:///FileServer1/ComingJobs/job702555.gz"/>
  </LayoutElement>
</RunList>
```

### Full example of ArtDeliveryIntent translated to Delivery and DigitalDelivery processes

Please note that elements and attributes which are candidate to be part of JDF 1.2 are used in this example. They are prefixed by the “add12” namespace. It was validated with CheckJDF except the proposed changes that conflict with current schema. 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" Version="1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <NodeInfo JobPriority="100"/>
  <CustomerInfo CustomerJobName="Job title ...">
    <Contact ContactTypes="Customer">
      <Address City="Alta" PostalCode="369300" Region="AV" Street="123 Gibrish Street"/>
    </Contact>
  </CustomerInfo>
  <Person FamilyName="Spencer" FirstName="Ron"/>
  <ComChannel ChannelType="Phone" ChannelUsage="DayTime" Locator="+44 019-1234-
4567"/>
</JDF>
```

## Appendix V Examples

```
<ComChannel ChannelType="Fax" ChannelUsage="Business DayTime NightTime"
Locator="+44 019-1234-4567"/>
</Contact>
</CustomerInfo>
<ResourcePool>
  <ArtDeliveryIntent Class="Intent" ID="Link002" ReturnList="DigitalMedia"
Status="Available">
    <ArtHandling DataType="EnumerationSpan" Range="Return ReturnWithProduct"/>
    <ReturnMethod DataType="NameSpan" Preferred="FedEx"/>
    <ArtDelivery ArtDeliveryType="DigitalNetwork">
      <Contact ContactTypes="Delivery">
        <ComChannel ChannelType="WWW" ChannelTypeDetails="Form" Locator="http://
www.server.com/uploader.aspx"/>
      </Contact>
    </RunList>
    <LayoutElement>
      <FileSpec URL="file:///D:/WINNT/Profiles/23423/Desktop/test.pdf"/>
    </LayoutElement>
  </RunList>
</ArtDelivery>
</ArtDeliveryIntent>
<Contact Class="Parameter" ContactTypes="Delivery" ID="Shipping001"
Status="Available">
  <Address City="Alta" PostalCode="369300" Region="AV" Street="123 Gibrish Street"/
>
  <Person FamilyName="Jones" FirstName="Bill"/>
  <ComChannel ChannelType="Phone" ChannelTypeDetails="Mobile" Locator="+44 078-
1234-4567"/>
</Contact>
<Component Amount="500" Class="Quantity" ComponentType="FinalProduct"
ID="ItemFinal" Status="Unavailable"/>
</ResourcePool>
<ResourceLinkPool>
  <ArtDeliveryIntentLink Usage="Input" rRef="Link002"/>
  <ComponentLink Amount="500" Usage="Output" rRef="ItemFinal"/>
</ResourceLinkPool>
<JDF ID="J171373" Status="Completed" Type="DigitalDelivery">
  <CustomerInfo CustomerJobName="Job title ...">
    <CustomerMessage Language="FR" MessageEvents="Delivered" ShowList="StartTime
EndTime Error">
      <ComChannel ChannelType="Email" Locator="sender@Email.com"/>
      <ComChannel ChannelType="InstantMessaging" ChannelTypeDetails="Yahoo!"
Locator="bill_jones"/>
    </CustomerMessage>
  </CustomerInfo>
  <ResourcePool>
    <RunList Class="Parameter" ID="FileListLink1" Status="Available">
      <LayoutElement>
        <FileSpec URL="file:///D:/WINNT/Profiles/23423/Desktop/test.pdf"/>
      </LayoutElement>
    </RunList>
    <DigitalDeliveryParams Class="Parameter" DigitalDeliveryDirection="Push"
DigitalDeliveryProtocol="FTP" ID="DestinationLink" Method="WebServer"
Status="Available">
      <Contact ContactTypes="Delivery">
        <ComChannel ChannelType="WWW" ChannelTypeDetails="Form" Locator="http://
www.server.com/uploader.aspx"/>
      </Contact>
      <Contact ContactTypes="Sender">
```



```

    <ComChannel ChannelType="Email" Locator="sender@Email.com"/>
  </Contact>
</DigitalDeliveryParams>
<RunList Class="Parameter" ID="FileListLink2" Status="Available">
  <Disposition MinDuration="P30D"/>
  <LayoutElement>
    <FileSpec Compression="Deflate" URL="test.pdf">
      <Container>
        <FileSpec MimeType="application/zip" URL="file://network_share/
uploaded%20files/test.zip"/>
      </Container>
    </FileSpec>
  </LayoutElement>
</RunList>
</ResourcePool>
<ResourceLinkPool>
  <DigitalDeliveryParamsLink Usage="Input" rRef="DestinationLink"/>
  <RunListLink Usage="Input" rRef="FileListLink1"/>
  <RunListLink Usage="Output" rRef="FileListLink2"/>
</ResourceLinkPool>
<AuditPool>
  <PhaseTime DescriptiveName="Upload of Job 171373 to Server" End="2003-01-
08T12:27:56Z" Start="2003-01- 08T12:27:40Z" Status="InProgress" TimeStamp="2003-01-
08T12:27:56Z"/>
  <Created Author="Server uploader 1.51" TimeStamp="2003-01-08T12:27:40Z"/>
  <ProcessRun End="2003-01-08T12:27:56Z" EndStatus="Completed" Start="2003-01-
08T12:27:40Z" TimeStamp="2003-01-08T12:27:56Z"/>
</AuditPool>
</JDF>

<JDF DescriptiveName="The Return of product and digital media with intermediate
materials" ID="X00000" Status="Waiting" Type="Delivery">
  <ResourceLinkPool>
    <ComponentLink Usage="Output" rRef="Item001"/>
    <DigitalMediaLink Usage="Output" rRef="Item002"/>
    <DeliveryParamsLink Usage="Input" rRef="Delivery001"/>
  </ResourceLinkPool>
  <ResourcePool>
    <RunList Class="Parameter" ID="FileListLink0" PartIDKeys="Run"
Status="Available">
      <RunList Run="1">
        <LayoutElement>
          <FileSpec URL="./ForReturn/Intermediate/test.pdf"/>
        </LayoutElement>
      </RunList>
      <RunList Run="2">
        <LayoutElement>
          <FileSpec URL="./ForReturn/Final/test.pdf"/>
        </LayoutElement>
      </RunList>
    </RunList>
    <Component Amount="500" Class="Quantity" ComponentType="FinalProduct"
ID="Item001" ProductID="AG5678" Status="Available" Unit="1"/>
    <DigitalMedia Amount="1" Capacity="700" Class="Handling" ID="Item002"
MediaLabel="TempResults" MediaType="CD" Status="Available">
      <RunListRef rRef="FileListLink0"/>
    </DigitalMedia>
    <DeliveryParams Class="Parameter" ID="Delivery001" Status="Available">
      <Drop Method="FedEx" ServiceLevel="Ground" TrackingID="1234567890Z">

```

```

    <ContactRef rRef="Shipping001"/>
    <DropItem Amount="500" Unit="1">
      <ComponentRef rRef="Item001"/>
    </DropItem>
    <DropItem Amount="1">
      <DigitalMediaRef rRef="Item002"/>
    </DropItem>
  </Drop>
</DeliveryParams>
</ResourcePool>
</JDF>
</JDF>

```

### Full example of digital delivery through central server

Please note that elements and attributes which are candidate to be part of JDF 1.2 are used in this example. They are prefixed by the “add12” namespace. It was validated with CheckJDF except the proposed changes that conflict with current schema. The following example describes:

- 1 Upload of files to server by FTP protocol
- 2 Request for 10 days storage on server
- 3 Request to Mac Binary encode the files on server
- 4 Send to multiple destinations: 1 is email address and 1 is registered address in a private directory
- 5 Download of files by HTTP protocol
- 6 Decode from Mac Binary when downloading to target
- 7 Download by 1 destination out of the 2.

```

<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" DescriptiveName="Digital Delivery
through central server - example with process group" ID="ID000" Status="InProgress"
Type="ProcessGroup" Version="1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <NodeInfo JobPriority="60"/>
  <ResourcePool>
    <Contact Class="Parameter" ContactTypes="Delivery" ID="DestLink"
PartIDKeys="Location" Status="Available">
      <Contact Location="Dest1">
        <ComChannel ChannelType="Email" Locator="Reciever1@hotmail.com"/>
      </Contact>
      <Contact Location="Dest2">
        <ComChannel ChannelType="PrivateDirectory" ChannelTypeDetails="VioAddress"
Locator="Best Workgroup@Best Company"/>
      </Contact>
    </Contact>
    <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>

```

```

</ResourcePool>

  <JDF DescriptiveName="Upload Job to Server" ID="ID001" JobID="J702555"
  Status="Completed" Type="DigitalDelivery">
    <CustomerInfo CustomerJobName="World atlas maps #2">
      <CustomerMessage Language="FR" MessageEvents="Delivered Accepted">
        <ComChannel ChannelType="Email" Locator="sender@email.com"/>
      </CustomerMessage>
    </CustomerInfo>
    <ResourcePool>
      <RunList Class="Parameter" Directory="file:///c:/MyDir/JobForSend"
      ID="SourceFileListLink0" PartIDKeys="Run" Status="Available">
        <RunList Run="1">
          <LayoutElement>
            <FileSpec FileSize="240066" URL="./Atlas/Europe.bmp"/>
          </LayoutElement>
        </RunList>
        <RunList Run="2">
          <LayoutElement>
            <FileSpec FileSize="33947" URL="./Atlas/America.jpg"/>
          </LayoutElement>
        </RunList>
      </RunList>
      <Contact Class="Parameter" ContactTypes="Sender" ID="SendLink"
      Status="Available">
        <ComChannel ChannelType="Email" Locator="sender@email.com"/>
      </Contact>
      <DigitalDeliveryParams Class="Parameter" DigitalDeliveryDirection="Push"
      DigitalDeliveryProtocol="FTP" ID="DestinationLink0" Method="Vio" PartIDKeys="Location"
      Status="Available">
        <Comment Name="Instruction">Please take these maps and add them to the rest
        ...</Comment>
        <DigitalDeliveryParams Location="SenderToDest1">
          <ContactRef rRef="SendLink"/>
          <ContactRef rRef="DestLink">
            <Part Location="Dest1"/>
          </ContactRef>
        </DigitalDeliveryParams>
        <DigitalDeliveryParams Location="SenderToDest2">
          <ContactRef rRef="SendLink"/>
          <ContactRef rRef="DestLink">
            <Part Location="Dest2"/>
          </ContactRef>
        </DigitalDeliveryParams>
      </DigitalDeliveryParams>
    </ResourcePool>
    <ResourceLinkPool>
      <DigitalDeliveryParamsLink Usage="Input" rRef="DestinationLink0"/>
      <RunListLink Usage="Input" rRef="SourceFileListLink0"/>
      <RunListLink Usage="Output" rRef="TempFileListLink"/>
    </ResourceLinkPool>
    <AuditPool>
      <ProcessRun DescriptiveName="Upload of Job 702555 to Vio Server" End="2002-07-
      21T10:47:11Z" EndStatus="Completed" Start="2002-07-21T10:45:52Z" TimeStamp="2002-07-
      21T10:47:11Z"/>
      <Created Author="Vio Server 4.3" TimeStamp="2002-07-21T10:45:52Z"/>
    </AuditPool>
  </JDF>

```

## Appendix V Examples

```
<JDF DescriptiveName="Download Job from Server to destination" ID="ID002"
JobID="J702555" Status="Pool" Type="DigitalDelivery">
  <ResourcePool>
    <RunList Class="Parameter" Directory="File:///e:/My%20Download"
ID="TargetFileListLink1" PartIDKeys="Run" Status="Available">
      <RunList Run="1">
        <LayoutElement>
          <FileSpec FileSize="240066" URL="./Atlas/Europe.bmp"/>
        </LayoutElement>
      </RunList>
      <RunList Run="2">
        <LayoutElement>
          <FileSpec FileSize="33947" URL="./Atlas/America.jpg"/>
        </LayoutElement>
      </RunList>
    </RunList>
    <DigitalDeliveryParams Class="Parameter" DigitalDeliveryDirection="Pull"
DigitalDeliveryProtocol="HTTP" ID="DestinationLink1" Method="Vio" PartIDKeys="Location"
Status="Available">
      <DigitalDeliveryParams Location="ToDest1">
        <ContactRef rRef="DestLink">
          <Part Location="Dest1"/>
        </ContactRef>
      </DigitalDeliveryParams>
      <DigitalDeliveryParams Location="ToDest2">
        <ContactRef rRef="DestLink">
          <Part Location="Dest2"/>
        </ContactRef>
      </DigitalDeliveryParams>
    </DigitalDeliveryParams>
  </ResourcePool>
  <ResourceLinkPool>
    <DigitalDeliveryParamsLink Usage="Input" rRef="DestinationLink1"/>
    <RunListLink Usage="Input" rRef="TempFileListLink"/>
    <RunListLink Usage="Output" rRef="TargetFileListLink1"/>
  </ResourceLinkPool>
  <StatusPool Status="InProgress">
    <PartStatus Status="Completed">
      <Part Location="ToDest2"/>
    </PartStatus>
  </StatusPool>
  <AuditPool>
    <Created Author="Vio Server 4.3" TimeStamp="2002-07-21T10:48:57Z"/>
    <ProcessRun DescriptiveName="HTTP Download of Job 702555 by Best Workgroup@Best
Company" End="2002-07-21T10:50:11Z" EndStatus="Completed" Start="2002-07-21T10:48:57Z"
TimeStamp="2002-07-21T10:50:11Z">
      <Part Location="ToDest2"/>
    </ProcessRun>
  </AuditPool>
</JDF>
</JDF>
```

---

## Appendix W New, Deprecated, Modified, Illegal, & Removed Items

A word of caution: The following tables are provided for your convenience, but are not a comprehensive list of all changes made in the production of JDF 1.2. Some changes that are global or comprehensive to one or more sections of this document are explained in the main section of the document. Changes in JDF 1.2 are extensive and even experienced JDF developers, who may even be active in one or more CIP4 Working Group, are strongly advised to read JDF in its entirety, and are advised not to rely solely on this appendix to identify all changes to JDF in JDF 1.2.

Note that many editorial and administrative changes were made to version JDF 1.2 prior to the application of changes that originated in committee work. The bulk of changes consists of clarifications of ambiguities such as:

- Placing all “e.g.” and “i.e.” comments into parenthesis
- Ensuring that the use of terms such as “finished page,” “reader page,” “folios,” “leaf,” “sheet,” and so forth are consistent throughout the document.
- Eliminating all occurrences of “system specified” enumerations and changing annotation of default attribute values as defined in Section 1.3.4, Specification of Cardinality and “Intent Resources” on page 237.
- Clarification of examples and default values throughout Section 7, Resources to be consistent with Encoding.
- All processes and resources deprecated in their entirety were removed from the Resources and Processes chapters and put into “Deprecated Processes, Resources, and JMF Messaging Elements” on page 743.
- All number, NumberList, NumberRange, and NumberRangeList data types were changed to double, DoubleList, DoubleRange, and DoubleRangeList data types throughout the document.

The above changes and other editorial changes are not listed below and are too numerous to list.

## W.1 Compatibility Warnings

Location	Section Title	Comments
page 204	Section 6.4.19, Preflight	<b>Compatibility Warning.</b> Preflight has been modified substantially in JDF 1.2 and is no longer compatible with previous versions of JDF.
page 123	Section 4.4.5, Case 5: Spawning and Merging of Independent Jobs	<b>Compatibility Warning.</b> Note that Spawning and Merging of Independent Jobs is under development and subject to major changes in a future release of this specification.
page 267	Section 7.1.11, LayoutIntent	In <i>FinishedDimensions</i> attribute: <b>Compatibility Warning.</b> In JDF 1.1 height and width were erroneously switched in the description.
page 267	Section 7.1.11, LayoutIntent	In <i>NumberUp</i> attribute: <b>Compatibility Warning.</b> In JDF 1.0 and 1.1 rows and columns were erroneously switched in the description.
page 267	Section 7.1.11, LayoutIntent	In <i>Pages</i> attribute: <b>Compatibility Warning.</b> 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.
page 311	Section 7.2.29, ColorSpaceConversionParams	In <i>RenderingIntent</i> attribute: <b>Compatibility Warning.</b> The default has changed in JDF 1.2. from Perceptual.
page 408	Section 7.2.99, LayoutPreparationParams	In <i>NumberUp</i> attribute: <b>Compatibility Warning.</b> In JDF 1.1 rows and columns were erroneously switched in the description.
page 447	Section 7.2.130, PStoPDFConversionParams	In <i>LockDistillerParams</i> : <b>Compatibility warning.</b> In JDF 1.1A and previous versions, the definition of <i>LockDistillerParams</i> was accidentally inverted. It is now consistent with the PostScript <i>SetDistillerParams</i> operator.

## W.2 New Items

Location	Section Title	Comments
Table 1-3 on page 9	Introduction	Addition of DateTimeRange, DateTimeRangeList, DurationRange, and DurationRangeList to JDF data types.
page 371	Section 7.2.71, FormatConversionParams	Addition of TIFFFormatParams, ImageCompressionParams, and ColorPool, including Structure of the TIFFFormatParams Element, Structure of the TIFFTag Subelement, and Structure of the TIFFEmbeddedFile Subelement tables.
page 3	Section 1.3.3, Call-Outs	Added section that details use of flags identifying new, modified, and deprecated items.

Location	Section Title	Comments
page 7	Section 1.4, Glossary of Terminology	Added definitions for folio, form, leaf, sheet, signature, and surface, along with a callout explaining the changes to “page” types — please note that changes have been made throughout JDF 1.2 that reflect these changes in terminology.
page 323	Section 7.2.33, Component	Example Use of Condition Attribute section added.
page 576	Section A.3.3.2, NamedColor	Added Cyan and Magenta as named colors.
page 222	Section 6.6.16, Feeding	New process with new input and output resources.
page 85	“Implicit and Explicit PartUsage in Partitioned Resources” on page 85	New sub-section with examples.
page 97	Section 3.9.1.6, Deleted	New section and element with attributes.
page 196	Section 6.4.1, AssetListCreation	New process (with new resources).
page 210	Section 6.4.29, Stripping	New process (with new resources).
page 283	Section 7.2.6, Assembly	New resource with new attributes and elements.
page 285	Section 7.2.9, BinderySignature	New resource with new attributes and elements.
page 346	Section 7.2.54, DigitalMedia	New resource with new attributes and elements.
page 311	Section 7.2.29, ColorSpaceConversionParams	Added Effect of color space conversion operations on color spaces and Mapping of SourceCS enumeration values to color spaces in the most common input file formats tables.
page 356	Section 7.2.64, FeedingParams	New Resource with new attributes and elements.
page 479	Section 7.2.157, StrippingParams	New Resource with new attributes and elements.
page 452	Section 7.2.131, QualityControlParams	New resource.
page 452	Section 7.2.132, QualityControlResult	New resource.
page 344	Section 7.2.51, DeviceMark	New resource — elevated from status as subelement of the <b>Surface</b> resource with significant modifications and new attributes.
page 575	Section A.3.3, Defined JDF enumeration Data Types	Added regExp, DateTimeRange, DateTimeRangeList, XYRelation and DurationRangeList data types.
page 635	Section P, FileSpec Attribute Examples for MimeType and MimeTypeVersion Attributes	New Appendix.
page 228	Section 6.6.31, PrintRolling	New process with input and output resources.
page 446	Section 7.2.128, PrintRollingParams	New resource with new attributes.

Location	Section Title	Comments
page 457	Section 7.2.138, RollStand	New resource with new attributes.
page 651	Section S, AppOS and OSVersion Attributes	New Appendix.
page 615	Section G, StatusDetails Supported Strings	Addition of multiple StatusDetails values, including addition of tables specifying PostPress Device specific StatusDetails and Printing Device specific StatusDetails.
page 617	Section H, ModuleType Supported Strings	Addition of multiple ModuleType values, including addition of tables specifying ModuleType definition Gathering / Collecting and ModuleType definition for DigitalPrinting.
page 217	Section 6.6.4, Bundling	New process with new input and output resources.
page 290	Section 7.2.14, BundlingParams	New resource with new attributes and elements.
page 345	Section 7.2.53, DigitalDeliveryParams	New resource with new attributes and elements.
page 199	Section 6.4.8, DigitalDelivery	New process with new input and output resources.
page 560	Section 8.3, JDF Packaging	New to JDF 1.2
page 101	Section 3.11, JDF Versioning	New to JDF 1.2
page 9	Section 1.5, Data Structures	Added DateTimeRange, DateTimeRangeList, DurationRangeList, regExp, XPath, and XYRelation data types.
page 35	Section 3.1.1, Generic Contents of JDF Elements	Added SettingsPolicy attribute to generic contents of elements table.
page 44	Section 3.1.4.1, Use of the Types attribute in ProcessGroup nodes	New section.
page 45	Section 3.1.4.3, ResourceLink Structure in ProcessGroup nodes	New section that includes an JDF 1.1 example.
page 3	Section 1.3.2, XPath Notation Used in this Specification	New in JDF 1.2
page 72	Section 3.8.1.2, Specifying Amount for a partially completed process	New in JDF 1.2
page 75	Section 3.8.2.2, Relating PartIDKeys and Partitions	New in JDF 1.2
page 76	Section 3.8.2.2.1, Incomplete Partitions	New in JDF 1.2
page 76	Section 3.8.2.2.2, Multiple Keys per Partitioned Leaf or Node	New in JDF 1.2
page 76	Section 3.8.2.2.3, Degenerate Partitions	New in JDF 1.2
page 77	Section 3.8.2.3, Partitioning of Resource sub-Elements	New in JDF 1.2 — A new section created from slightly modified JDF 1.1 content.



Location	Section Title	Comments
page 75	Section 3.8.2.1, Amount in Partitionable Resources	New in JDF 1.2 — A new section created from slightly modified JDF 1.1 content.
page 78	Section 3.8.2.4, Additional Attributes for use with partitioned Resources	New in JDF 1.2 — A new section created from slightly modified JDF 1.1 content.
page 625	“Color Adjustment Attribute Description and Usage” on page 625	New Appendix.
page 633	“Input Tray and Output Bin Names” on page 633	New Appendix.
page 629	“New, Deprecated, Modified, Illegal, & Removed Items” on page 705.	New Appendix.
page 112	Section 4.3.2, Partial Processing of Nodes with Partitioned Resources	New section.
page 195	Section 6.3, Product Intent Descriptions	New section that summarizes existing content.
page 347	Section 7.2.55.1, Coordinate systems in DigitalPrinting	New section.
page 559	Section 8.2.1.1, JMF Transport Using The File Protocol	New section.
page 561	Section 8.3.2.1, MIME Fields	New section. with subsections.
page 562	Section 8.3.2.2, Example Packaging of Individual JDF/JMF files in MIME	New section..
page 562	Section 8.3.2.2, Example Packaging of Individual JDF/JMF files in MIME	New section..
page 562	Section 8.3.2.3, CID URL Scheme	New section..

Location	Section Title	Comments
page 563	Section , 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 [RFC2396] does not permit in URLs; any such character (such as "+" or "&") must be hex-encoded using the %hh escape mechanism in the URL (see [RFC2396]). Therefore, matching the URL with the CID, must take account of the escaped equivalencies. Case-insensitive matching must be used.	New section..
page 579	Section B.1, Using xsi:type	New section with subsections.
page 622	Section J.1.6, Event	New section..
page 177	“Contents of the RemoveQueueEntry message” on page 177	New section.
page 538	Section 7.4, Concept of the Preflight Process	New section.
page 649	Resolving RunList/ @Directory and FileSpec/ @URL URI references	New Appendix..
page 641	FileSpec MimeType, URL, and Compression attributes, and Container subelement	New Appendix..

### W.3 Deprecated Items

Location	Table Info	Comments
“Delivery” on page 193	Input Resources	Deprecated <b>Resource</b> .
“Proofing” on page 207	All	The <b>Proofing</b> process is deprecated in JDF/1.2. Instead, use a combined process to produce the hard proof, (e.g., one that includes the <b>ImageSetting</b> , <b>ConventionalPrinting</b> , or <b>DigitalPrinting</b> process.) Then input the hard proof to a separate <b>Approval</b> process
“SoftProofing” on page 210	All	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 <b>Approval</b> process that approves the soft proof.
Section 7.2.71, FormatConversionParams	Resource Structure	Both occurrences of <b>FileSpec</b> were deprecated.
“ScreeningParams” on page 465	Structure of ScreenSelector Subelement	Sample string values for <i>ScreeningFamily</i> were deprecated.
“TrappingDetails” on page 494	Resource Structure	Deprecated <i>TrappingType</i> .
“TrimmingParams” on page 499	Resource Structure	Deprecated <i>TrimmingType</i> .
“Interpreting” on page 203	Output Resources	Deprecated <b>InterpretedPDLData</b> .
“Rendering” on page 208	Output Resources	Deprecated <b>InterpretedPDLData</b> .
“ConventionalPrinting” on page 213	Output Resources	Deprecated <b>Component</b> ? (Waste) resource.
“Collecting” on page 219	Input Resources	Deprecated <b>IdentificationField</b> resource.
“FileSpec” on page 359	Structure of FileAlias Subelement	In JDF/1.2 and beyond, use <i>Disposition</i> in <b>FileSpec</b> reelement.
“FileSpec” on page 359	Structure of FileAlias Subelement	In JDF/1.2 and beyond, use <i>MimeType</i> in <b>FileSpec</b> reelement.
“FileSpec” on page 359	Structure of FileAlias Subelement	In JDF/1.2 and beyond, use <i>URL</i> in <b>FileSpec</b> reelement.
“Gathering” on page 224	Input Resources	Deprecated <b>IdentificationField</b> resource.
“Inserting” on page 225	Input Resources	Deprecated <b>IdentificationField</b> resource.
“PSToPDFConversionParams” on page 447	Resource Structure	<i>ImageMemory</i> attribute was deprecated because it is an internal application setting and not a parameter setting.
“SpinePreparationParams” on page 470	Resource Structure	Deprecated <i>FlexValue</i> and <i>PullOutValue</i> . (In JDF 1.2 and beyond, these are defined in “QualityControlParams” on page 452.)
“Defined JDF enumeration Data Types” on page 575	n/a	Deprecated TimeRange data type.
“Structure of the DeviceCap Subelement” on page 502	Structure of the DeviceCap Subelement	Deprecated <i>OptionalCombinedType</i> , <i>Type</i> , and <i>TypeOrder</i> attributes.

Location	Table Info	Comments
“Structure of the Performance Subelement” on page 524	Structure of the Performance Subelement	Deprecated <i>Name</i> attribute.
“Structure of the DevCaps Subelement” on page 505	Structure of the DevCaps Subelement	Deprecated <i>Types</i> ? attribute. Replaced by <i>TypeIndex</i> in JDF 1.2 and beyond.
“Structure of the Abstract State Subelement” on page 508	Structure of the Abstract State Subelement	Deprecated the <i>Span</i> ? attribute. Replaced with <i>ListType</i> = “ <i>Span</i> ” in JDF 1.2 and beyond.
“Structure of the IntegerState Subelement” on page 513	Structure of the IntegerState Subelement	Deprecated <i>AllowedValueMax</i> , <i>AllowedValueMin</i> , <i>PresentValueMax</i> , and <i>PresentValueMin</i> attributes.
See “Structure of the MatrixState Subelement” on page 514.	Structure of the Value element	Deprecated <i>PresentValue</i> attribute.
See “Structure of the NumberState Subelement” on page 516.	Structure of the NumberState Subelement	Deprecated <i>AllowedValueMax</i> , <i>PresentValueMax</i> , <i>PresentValueMin</i> , and <i>AllowedValueMin</i> attributes.
See “Structure of the ShapeState Subelement” on page 518.	Structure of the ShapeState Subelement	Deprecated <i>AllowedValueMax</i> , <i>AllowedValueMin</i> , <i>PresentValueMin</i> , and <i>PresentValueMax</i> attributes.
See “Structure of the StringState Subelement” on page 519.	Structure of Value element	Deprecated <i>PresentValue</i> attribute. In JDF 1.2 and beyond, use <i>ValueUsage</i> .
See “Structure of the XYPairState Subelement” on page 520.	Structure of the XYPairState Subelement	Deprecated <i>AllowedValueMax</i> , <i>AllowedValueMin</i> , <i>PresentValueMin</i> , and <i>PresentValueMax</i> attributes.
See “Controller Registration and Communication Messages” on page 142.	Table 5-16, “Process registration and communication messages,” on page 142	Deprecated KnownJDFServices element.
See “KnownJDFServices” on page 146.	<i>Multiple</i>	Deprecated element/section.
“Resource Links” on page 61	Table 3-19, “Contents of the abstract ResourceLink element,” on page 64	Deprecated <i>rSubRef</i> attribute — In JDF 1.2 and beyond, resource links should only reference resources that are present in a <b>ResourcePool</b> .
“Inter-Resource Linking Using ResourceRef” on page 68	Table 3-24, “Contents of the abstract ResourceRef element,” on page 68	Deprecated <i>ID</i> — In JDF 1.2 and beyond, <b>ResourceRef</b> and <b>ResourceLink</b> elements should only reference resources that are present in a <b>ResourcePool</b> . Therefore elements that are defined locally within a resource should not be referenced and should not contain an ID.
“Inter-Resource Linking Using ResourceRef” on page 68	Table 3-25, “Contents of the abstract ResourceElement,” on page 69	Deprecated <i>rSubRef</i> attribute — In JDF 1.2 and beyond, resource links should only reference resources that are present in a <b>ResourcePool</b> .

Location	Table Info	Comments
“Rendering” on page 208	Input Resources	Deprecated InterpretedPDLData resource. In JDF 1.2 and beyond, a <b>RunList</b> with <b>InterpretedPDLData</b> sub-elements describes the input content data for <b>Rendering</b> .
“DigitalPrinting” on page 214	Output Resources	Deprecated the <b>Component</b> (Waste) resource from this process. In JDF 1.2 and beyond, waste is tracked by partitioning the output using the <i>Condition PartIDKey</i> .
“Structure of Abstract Span Subelement” on page 238	n/a	Deprecated <i>Priority</i> attribute — replaced by <i>SettingsPolicy</i> in JDF 1.2 and beyond.
“MediaIntent” on page 270	Resource Structure	Deprecated <i>Dimensions</i> attribute. In JDF 1.2 and beyond the specifics of <i>BuyerSupplied</i> media should be specified using a <b>Media</b> resource. The dimensions of the finished product are specified with <b>LayoutIntent/@Dimensions</b> or <b>LayoutIntent/@FinishedDimensions</b> .
“MediaIntent” on page 270	Resource Structure	Deprecated <i>MediaUnit</i> attribute. Intent attributes pertain to finished product, not the raw media format. If <i>BuyerSupplied</i> = “true” then the <b>Media</b> resource can be used to provide this attribute.
“MediaIntent” on page 270	Resource Structure	Deprecated <i>Recycled</i> attribute. In JDF 1.2 and beyond, use <i>RecycledPercentage</i> .
“MediaIntent” on page 270	Resource Structure	Deprecated <i>USWeight</i> attribute. In JDF 1.2 and beyond use <i>Weight</i> .
“ApprovalParams” on page 282	Structure of ApprovalPerson Subelement	Deprecated <i>Obligated</i> attribute. In JDF 1.2 and beyond, use <i>ApprovalRole</i> .
“CoilBindingParams” on page 296	Resource Structure	Deprecated <i>Shift</i> attribute. In JDF 1.2 and beyond, use the value implied by <b>HoleMakingParams/@HoleType</b> .
“FileSpec” on page 359	Resource Structure	Deprecated <i>Disposition</i> attribute — in JDF 1.2 and beyond, retention of assets is specified in the <b>Disposition</b> element
“FoldingParams” on page 366	Resource Structure	Deprecated <i>DescriptionType</i> attribute — In JDF 1.2 and beyond, the <i>FoldCatalog</i> defines the topology of the folding scheme. The specifics of each individual <b>Fold</b> may be described using <b>Fold</b> elements. If both <i>FoldCatalog</i> and <b>Fold</b> are specified, <b>Fold</b> takes precedence
“HoleMakingParams” on page 380	Resource Structure	Deprecated <i>HoleReferenceEdge</i> attribute — <i>HoleReferenceEdge</i> has been replaced with an explicit <i>Transformation</i> or <i>Orientation</i> of the input <b>Component</b> . If both <i>Transformation</i> or <i>Orientation</i> and <i>HoleReferenceEdge</i> are specified, the result is the matrix product of both transformations. <i>Transformation</i> or <i>Orientation</i> must be applied first.
“Media” on page 417	Resource Structure	Deprecated <i>ColorName</i> attribute — in JDF 1.2 and beyond, use <i>MediaColorName</i> and <i>MediaColorNameDetails</i> .

Location	Table Info	Comments
“PlasticCombBindingParams” on page 433	Resource Structure	Deprecated <i>Type</i> attribute — In JDF 1.2 and beyond, use the value implied by <b>HoleMakingParams/@HoleType</b> .
“Preview” on page 442	Resource Structure	The <i>PreviewType</i> attribute was deprecated and replaced with <i>PreviewUsage</i> —
“ProofingParams” on page 447	n/a	Deprecated resource — in JDF 1.2 and beyond, proofing is handled as a combined process.
“RunList” on page 458	Resource Structure	Deprecated <i>NDoc</i> attribute — in JDF 1.2 and beyond, only <i>Docs</i> is supported.
“RunList” on page 458	Resource Structure	Deprecated <i>NSet</i> attribute — in JDF 1.2 and beyond, only <i>Sets</i> is supported.
“ScreeningParams” on page 465	Resource Structure	Deprecated <i>AbortJobWhenScreenMatchingFails</i> attribute — Use <b>SettingsPolicy</b> in JDF 1.2 and beyond.
“StitchingParams” on page 475	Resource Structure	Deprecated <i>ReferenceEdge</i> attribute. <i>ReferenceEdge</i> has been replaced in JDF 1.2 and beyond with an explicit <i>Transformation</i> or <i>Orientation</i> of the input <b>Component</b> . 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.
“StitchingParams” on page 475	Resource Structure	Deprecated the <i>StitchFromFront</i> attribute. <i>StitchFromFront</i> has been replaced with an explicit <i>Transformation</i> or <i>Orientation</i> of the input <b>Component</b> .
“StripBindingParams” on page 479	Resource Structure	Deprecated the <i>Distance</i> attribute. In JDF 1.2 and beyond, use the value implied by <b>HoleMakingParams/@HoleType</b> .
“WireCombBindingParams” on page 501	Resource Structure	Deprecated the <i>Distance</i> attribute. In JDF 1.2 and beyond, use the value implied by <b>HoleMakingParams/@HoleType</b> .
“Color” on page 297	Resource Structure	Deprecated <i>UsePDLAlternateCS</i> attribute — In JDF 1.2 and beyond, use <i>MappingSelection</i> .
See “QueueEntryStatus” on page 185.	n/a	Deprecated <b>QueueEntryStatus</b> message as well as Structure of the <b>QueueEntryDefList</b> Element.
See “Contents of the SubmissionMethods element” on page 186.	—	Deprecated <i>File</i> attribute — in JDF 1.2 and beyond, include “ <i>file</i> ” in <i>URLSchemes</i> .
See “Contents of the SubmissionMethods element” on page 186.	—	Deprecated <i>HttpGet</i> attribute — In JDF 1.2 and beyond, include “ <i>http</i> ” in <i>URLSchemes</i> .
See “Contents of the SubmissionMethods element” on page 186.	—	Deprecated MIME attribute.
“PreflightAnalysis” on page 434	—	This resource was deprecated as a result of a major revision to the <b>Preflight</b> process and its associated resources.

---

Location	Table Info	Comments
“PreflightInventory” on page 434	—	This resource was deprecated as a result of a major revision to the <b>Preflight</b> process and its associated resources.
“PreflightProfile” on page 436	—	This resource was deprecated as a result of a major revision to the <b>Preflight</b> process and its associated resources.

## W.4 Modified Items

Location	Table Info	Comments
page 2	Table 1-1	Addition of, and updating of, multiple references to supporting standards, specifications, and other documents. Organized references in a new table and added short-reference nomenclature for use in cross-references within the JDF document.
“JDF Preface and User Overview” on page xxix	n/a	Moved Preface to follow TOC and List of Figures
“Legal Notice” on page i	n/a	Updated dates of copyright. Corrected “Job Definition Format”.
page 383	Resource Structure	Addition of DCTParams, CCITTFaxParams, JPEG2000Params, LZWParams, and FlateParams elements and element structure tables subordinated to Section 7.2.83, ImageCompressionParams.
“ColorIntent” on page 255	Structure of the ColorUsed Subelement	Changed data type of SpearationSpec from element to refelement.
“NumberingIntent” on page 275	Structure of the NumberIntent Subelement	Changed data type of SpearationSpec from element to refelement.
“ColorantAlias” on page 303	n/a	<b>ColorantAlias</b> was elevated from a subelement of <b>ColorantControl</b> to a top level resource in JDF 1.2.
Section 3.9.1, Audit Elements	Table 3-31 on page 91 plus body copy.	Changes include deletion of informaiton on prospective extensions and movement of a extension examples into Audit type table.
Section 7.2.91, InterpretedPDLData on page 399	n/a	In JDF 1.2 and beyond this is not a resource, but rather a subelement of a <b>RunList</b> .
“FileSpec” on page 359	n/a	All JDF 1.2 users are advised to take a close look at the FileSpec resource, as it contains numerous changes and is referenced to new appendicies.
“Device Capability Definitions” on page 502	n/a	This section has been modified significantly throughout.
See “Examples of Device Capabilities” on page 532.	n/a	Updated device capabilities examples.
See “Events” on page 142.	n/a	Modified introduction to section/usage.
See “Data Structures” on page 9.	JDF Data Types	Changed path data type to PDFPath.
See “Data Structures” on page 9.	JDF Data Types	Change definition of string to omit tabs as well as line feed characters.
See “Coordinate Systems of Resources and Processes” on page 23.	Matrices and names used to describe the orientation of a Component	Modified several transformation action descriptions.
See “Coordinate Systems of Resources and Processes” on page 23.	n/a	Deleted Section 2.5.3.1, Coordinate Systems in Combined processes, as well as some language and a figure dealing with cultural issues regarding left-to-right vs. right-to-left reading and orientation.



Location	Table Info	Comments
See “Generic Contents of JDF Elements” on page 35.	Table 3-2	Changed data type of Path attribute to PDFPath and changed description.
See “Inter-Resource Linking Using ResourceRef” on page 68.	n/a	New examples and several changes to section text.
Section 4.3.6, Approval, Quality Control, and Verification	n/a	Section was completely rewritten.
Section 4.8, Capability and Constraint Definitions	n/a	Section was completely rewritten.
Section 6.4.2, ColorCorrection	n/a	Section introduction was modified.
Section 7.2.24, ColorantControl	n/a	Section introduction was modified.
Section 7.2.26, ColorCorrectionParams	n/a	Section introduction was modified.
“ColorSpaceConversionOp” on page 313	n/a	Promoted to resource status from sub-element of <b>ColorSpaceConversionParams</b> .
“Building a System Around JDF” on page 559	n/a	Multiple and significant changes throughout chapter.
“Postpress Processes Structure” on page 233	n/a	Updated section to reflect added and deprecated postpress processes.
“InsertingParams” on page 394	n/a	Added illustrations and clarifications of <i>Location</i> .

## W.5 Clarified Items

Location	Table Info	Comments
“Combined Process Nodes” on page 46	n/a	Clarified the order of the Types attribute.
page 57	Table 3-14	Clarification of <i>NoOp</i> attribute.
page 58	Table 3-15	Clarification of <i>Brand</i> attribute.
“Extending NMTOKEN Lists” on page 100	n/a	Clarified the processing of unknown NMTOKEN extensions.
“Node and Resource IDs” on page 125	n/a	Clarified pureID.
“ConventionalPrinting” on page 213	n/a	Clarified that Proof Component may be from a DigitalPrinting process.
“ConventionalPrinting” on page 213	Input Resources	Clarified that Proof Component may be from a DigitalPrinting process.
“DigitalPrinting” on page 214	n/a	Added four more example processes that may be combined with <b>DigitalPrinting: Approval, ColorCorrection, ColorSpaceConversion, and ImageReplacement</b> . Clarified that Proof <b>Component</b> may be from a <b>ConventionalPrinting</b> process.
“DigitalPrinting” on page 214	Input Resources	Clarified <b>Component</b> (Proof)
“Stitching” on page 231	n/a/	Clarified that components containing staples of different characteristics like shape, width, and so on are defined by a combined process, and provided an example.
“BindingIntent” on page 247		Removed duplicate <i>StripBind</i> value in <i>BindingType</i> , deprecated <b>BookCase</b> element reference in <b>BindingIntent/BindList/BindItem</b> element to agree with JDF/1.1 deprecation of <b>BindingIntent/BindList/BindItem/BookCase</b> subelement
“HoleMakingIntent” on page 265	n/a	Clarified that the <b>HoleMakingIntent</b> applies equally to pre-drilled and drilling/punching.
“LayoutIntent” on page 267	Resource Structure	Clarified <i>Dimensions, PageVariance, and FinishedDimensions</i> attributes.
“LayoutIntent” on page 267	Resource Structure	Clarified <b>Layout</b> element usage.
“MediaIntent” on page 270	General + Resource Structure	General clarifications in introductory text, plus clarification of <i>Dimensions</i> and <i>Weight</i> attributes.
“Process Resource Template” on page 280	n/a	Add clarification, including introduction of <b>ColorSpaceConversionOp</b> resource.
“Color” on page 297	Resource Structure	Clarified <i>Name</i> attribute.
“ColorantControl” on page 303	Resource Structure	Clarified <i>DeviceN</i> NMTOKEN definition in <i>ProcessColorModel</i> .
“ColorCorrectionParams” on page 307	Resource Structure	Clarified <i>ColorManagementSystem</i> definition.
“ColorSpaceConversionParams” on page 311	n/a	Clarified Structure of <b>ColorSpaceConversionOp</b> Subelement.
“FileSpec” on page 359	Resource Structure	Clarified <i>AppVersion</i> by adding examples. Clarified <i>URL</i> with references.

Location	Table Info	Comments
“Ink” on page 391	Resource Structure	Clarified <i>ColorName</i> ? and <i>Family</i> ?.
“Media” on page 417	Resource Structure	Clarified definition of <i>Brightness</i> .
“Media” on page 417	Resource Structure	Clarified definition of <i>Grade</i> .
“Media” on page 417	Resource Structure	Clarified <i>Recycled</i> .
“PSToPDFConversionParams” on page 447	Structure of Advanced-Params Subelement	In JDF 1.1A and previous versions, the definition of <i>LockDistillerParams</i> was accidentally inverted. It is now consistent with the postscript setdistillerparams operator.
“RenderingParams” on page 455	Resource Structure	Clarified <i>ColorantDepth</i> .
“RunList” on page 458	Resource Structure	Clarified <i>EndOfDocument</i> , <i>NDoc</i> , <i>Pages</i> , <i>RunTag</i> , <i>SetNames</i> , <i>Sets</i> , and <i>EndOfSet</i> attributes.
“ScreeningParams” on page 465	Structure of ScreenSelector Subelement	Clarified <i>Angle</i> , <i>SpotFunction</i> , and <i>DotSize</i> .
“Sheet” on page 469	Resource Structure	<i>Name</i> must be unique within a given <b>Layout</b> .
Section 7.2.153, <i>StitchingParams</i>	n/a	Clarified section introduction.
“TrappingDetails” on page 494	Resources Structure +	Clarified section introduction, Resource Properties, and <i>IgnoreFileParams</i> and <i>Trapping</i> .
“Notes About Encoding” on page 565	XML Schema Data Types	Clarified definition of ID and language data types.
Section D.3.1, Administration Data	Converting administration data	Clarified <b>CIP3AdminInkInfo</b> and <b>IP3AdminInkColors</b> .
“ColorSpaceConversionParams” on page 311	Structure of ColorSpaceConversionOp Subelement section.	Clarified <b>FileSpec</b> .
“Structure of the DevCaps Subelement” on page 505	Structure of the DevCaps Subelement	Clarified <i>DevCap</i> + element.
“Structure of the Abstract State Subelement” on page 508	Structure of the Abstract State Subelement.	Clarified <i>HasDefault</i> = “ <i>true</i> ” attribute.
“MediaIntent” on page 270	Resource Structure	Clarified <i>Grade</i> attribute.

Location	Table Info	Comments
“AutomatedOverPrintParams” on page 285	n/a	Clarified application in section introduction.
“AutomatedOverPrintParams” on page 285	Resource Structure	Clarified <i>OverPrintBlackLineArt</i> and <i>OverPrintBlackText</i> attributes.
“ColorantControl” on page 303	Resource Structure	Clarified <i>ColorantOrder</i> , <i>ColorantParams</i> , <i>DeviceColorantOrder</i> , and <i>ColorantAlias</i> elements.
“Disjointing” on page 349	Resource Structure	Clarified description of <i>OffsetDirection</i> = “None”.
“FoldingParams” on page 366	Resource Structure	Clarified <i>FoldCatalog</i> attribute. Clarified the usage of <i>Fold</i> element.
“ScreeningParams” on page 465	Resource Structure	Clarified <i>AngleMap</i> attribute and extended example in Structure of <i>ScreenSelector</i> Subelement.
“ColorIntent” on page 255	Resource Structure	Clarified <b>ColorPool</b> reelement.
“Color” on page 297	n/a	Clarified introduction to section.
“Color” on page 297	Resource Structure	Clarified <b>FileSpec</b> reelement and <i>DeviceNColor</i> attribute.
“Color” on page 297	Resource Structure	Clarified <b>ColorPool</b> reelement.
“RunList” on page 458	Resource Structure	Clarified <i>Directory</i> attribute.
“Device” on page 342	Resource Structure	Clarified <i>Directory</i> attribute.
“FileSpec” on page 359	Resource Structure	Clarified <i>FileFormat</i> attribute.

## W.6 New/Modified Attributes and Elements

### W.6.1 Structure of JDF Nodes and Jobs

Location	Name	Data Type	Comment
Table 3-2 on page 36	<i>Name</i> = "Description"	NMTOKEN	Added JobDescription, OperatorText, and Template Description NMTOKEN values.
Table 3-27 on page 78	<i>PartIDKeys</i> ?	enumerations	Added <i>BundleItemIndex</i> , <i>Condition</i> , <i>SetDocIndex</i> , and <i>SetRunIndex</i> to list of NMTOKEN values.
Table 3-28 on page 79	<i>BundleItemIndex</i> ?	IntegerRangeList	Added <i>BundleItemIndex</i> attribute.
Table 3-4 on page 38	<i>Types</i> ?	NMTOKENS	Modified interpretation in description.
Table 3-5 on page 48	<i>Part</i> *	element	Modified interpretation in description.
Table 3-7 on page 49	<i>CustomerMessage</i> *	element	New element and subelement table defining subordinate attributes and elements.
Table 3-13 on page 53	<i>PipeProtocol</i> ?	NMTOKEN	Added <i>PipeProtocol</i> attribute.
Table 3-15 on page 58	<i>AmountProduced</i> ?	number	Added <i>AmountProduced</i> , <i>PipePause</i> , <i>PipeResume</i> , <i>RemotePipeEndPause</i> , and <i>RemotePipeEndResume</i> attributes.
Table 3-19 on page 64	<i>PipePartIDKeys</i> ?	enumerations	Modified interpretation of default in description.
Table 3-27 on page 78	—	—	Added <i>PipePartIDKeys</i> , <i>SetDocIndex</i> , <i>SetRunIndex</i> , and <i>ItemNames</i> attributes. Modified definitions of <i>DocIndex</i> , <i>DocRunIndex</i> , <i>RunIndex</i> , <i>RunTags</i> , <i>SetIndex</i> , and <i>SheetIndex</i> .
Table 3-34 on page 93	—	—	Addition of <i>BillingType</i> , <i>BillingTypeDetails</i> , and <i>Duration</i> attributes.
Table 3-35 on page 94	—	—	Addition of <i>Duration</i> attribute. Modified <i>ModuleIndex</i> attribute.

Location	Name	Data Type	Comment
Table 3-37 on page 96	—	—	Added <i>TemplateID</i> and <i>TemplateVersion</i> attributes.
Table 3-4 on page 38	<i>Category</i> ?	NMTOKEN	New attribute.
Table 3-4 on page 38	<i>ICSVersions</i> ?	NMTOKEN	New attribute.
Table 3-4 on page 38	<i>MaxVersion</i> ?	string	New attribute.
Table 3-4 on page 38	<i>RelatedJobID</i> ?	string	New attribute.
Table 3-4 on page 38	<i>RelatedJobPartID</i> ?	string	New attribute.
Table 3-4 on page 38	<i>StatusDetails</i> ?	string	New attribute.
Table 3-4 on page 38	<i>TemplateID</i> ?	string	New attribute.
Table 3-4 on page 38	<i>TemplateVersion</i> ?	string	New attribute.
Table 3-4 on page 38	<i>xsi:type</i> ?	NMTOKEN	New attribute.
Table 3-6 on page 48	<i>MaxVersion</i> ?	string	New attribute.
Table 3-6 on page 48	<i>StatusDetails</i> ?	string	New attribute.
Table 3-9 on page 51	<i>CostType</i> ?	enumeration	New attribute.
Table 3-9 on page 51	<i>WorkType</i> ?	enumeration	New attribute.
Table 3-9 on page 51	<i>WorkTypeDetails</i> ?	string	New attribute.
Table 3-10 on page 52	<i>StatusDetails</i> ?	string	New attribute.
Table 3-11 on page 53	<i>StatusDetails</i> ?	string	New attribute.
Table 3-11 on page 53	Part	element	Changed description.
Table 3-13 on page 53	<i>AgentName</i> ?	string	New attribute.
Table 3-13 on page 53	<i>AgentVersion</i> ?	string	New attribute.
Table 3-13 on page 53	<i>Author</i> ?	string	New attribute.
Table 3-19 on page 64	<i>Usage</i>	enumeration	Added enumeration.
Table 3-19 on page 64	<i>AmountPool</i> ?	element	Modified description.
Table 3-23 on page 67	<i>ActualAmount</i> ?	double	New attribute.
Table 3-28 on page 79	<i>BinderySignatureName</i> ?	NMTOKEN	New attribute.
Table 3-28 on page 79	<i>CellIndex</i> ?	IntegerRangeList	New attribute.
Table 3-28 on page 79	<i>Condition</i> ?	NMTOKEN	New attribute.
Table 3-28 on page 79	<i>PreflightRule</i> ?	string	New attribute.
Table 3-28 on page 79	<i>PreviewType</i> ?	enumeration	Added enumeration and changed description.
Table 3-28 on page 79	<i>SectionIndex</i> ?	IntegerRangeList	New attribute.
Table 3-28 on page 79	<i>Separation</i> ?	string	Modified description.
Table 3-31 on page 91	<i>Author</i> ?	string	Modified description.
Table 3-31 on page 91	<i>AgentName</i> ?	string	New attribute.
Table 3-31 on page 91	<i>AgentVersion</i> ?	string	New attribute.
Table 3-34 on page 93	<i>CostType</i> ?	enumeration	New attribute.
Table 3-34 on page 93	<i>WorkType</i> ?	enumeration	New attribute.
Table 3-34 on page 93	<i>WorkTypeDetails</i> ?	string	New attribute.
Table 3-4 on page 38	<i>NamedFeatures</i> ?	NMTOKEN	New attribute.

## W.6.2 JDF Messaging with the Job Messaging Format

Location	Name	Data Type	Comment
Table 5-1, "Contents of the JMF element," on page 130	<i>ResponseURL ?</i> <i>Unidirectional</i>	URL	New attribute.
Table 5-1, "Contents of the JMF element," on page 130	<i>Version ?</i>	string	New description
Table 5-11, "Contents of the Command message element," on page 136	<i>AcknowledgeFormat ?</i> <i>Unidirectional</i>	string	New attribute.
Table 5-11, "Contents of the Command message element," on page 136	<i>AcknowledgeTemplate ?</i> <i>Unidirectional</i>	string	New attribute.
Table 5-11, "Contents of the Command message element," on page 136	<i>AcknowledgeURL ?</i> <i>Bidirectional</i>	URL	Changed description.
Table 5-13, "Contents of the Subscription element," on page 139	<i>Format ?</i> <i>Unidirectional</i>	string	New attribute.
Table 5-13, "Contents of the Subscription element," on page 139	<i>Template ?</i> <i>Unidirectional</i>	string	New attribute.
Table 5-13, "Contents of the Subscription element," on page 139	<i>URL ?</i> <i>Unidirectional</i>	URL	Changed description.
Table 5-18, "Contents of the NotificationFilter element," on page 143	<i>QueueEntryID ?</i>	string	New attribute.
Table 5-18, "Contents of the NotificationFilter element," on page 143	<i>SignalTypes =</i> <i>"Notification"</i>	NMTOKENS	New attribute.
Table 5-18, "Contents of the NotificationFilter element," on page 143	<i>Part *</i>	element	New element.
Table 5-21, "Contents of the JDFController element," on page 144	<i>ControllerID ?</i>	string	New attribute.
Table 5-29, "Contents of the MsgFilter element," on page 147	<i>JobID ?</i>	string	New attribute.
Table 5-29, "Contents of the MsgFilter element," on page 147	<i>JobPartID ?</i>	string	New attribute.
Table 5-29, "Contents of the MsgFilter element," on page 147	<i>QueueEntryID ?</i>	string	New attribute.
Table 5-29, "Contents of the MsgFilter element," on page 147	<i>Part *</i>	element	New element.
Table 5-31, "Contents of the StopPersChParams element," on page 148	<i>QueueEntryID ?</i>	string	New attribute.
Table 5-31, "Contents of the StopPersChParams element," on page 148	<i>Part *</i>	element	New element.
Table 5-32, "Status and progress messages," on page 149	<i>FlushResources,</i> <i>ResourcePull,</i> <i>ShutDown,</i> <i>and</i> <i>WakeUp.</i>	element(s)	New types of status and progress messages.
See "Contents of the FlushResourceParams element" on page 150.	— <i>Multiple</i> —	— <i>Various</i> —	New element with element and attribute definitions.

Location	Name	Data Type	Comment
See “Contents of the Occupation message” on page 154.	<i>QueueEntryID</i> ?	string	New attribute.
See “Contents of the Occupation message” on page 154.	<b>Part *</b>	element	New element.
See “Contents of the ResourceQuParams element” on page 155.	<i>QueueEntryID</i> ?	string	New attribute.
See “Contents of the ResourceQuParams element” on page 155.	<b>Part *</b>	element	New element.
See “Contents of the ResourceCmdParams element” on page 157.	<i>QueueEntryID</i> ?	string	New attribute.
See “Contents of the ResourceCmdParams element” on page 157.	<i>Status</i> ?	enumeration	New attribute.
See “Contents of the ResourceCmdParams element” on page 157.	<b>Part *</b>	element	New element.
See “NewJDF” on page 150.	— <i>Multiple</i> —	— <i>Various</i> —	New element with element and attribute definitions.
See “NodeInfo” on page 152.	— <i>Multiple</i> —	— <i>Various</i> —	New element with element and attribute definitions.
See “ResourcePull” on page 160.	— <i>Multiple</i> —	— <i>Various</i> —	New element with element and attribute definitions.
See “Shutdown” on page 162.	— <i>Multiple</i> —	— <i>Various</i> —	New element with element and attribute definitions.
See “WakeUp” on page 169.	— <i>Multiple</i> —	— <i>Various</i> —	New element with element and attribute definitions.
See “Contents of the StatusQuParams element” on page 163.	<i>QueueEntryID</i> ?	string	New attribute.
See “Contents of the StatusQuParams element” on page 163.	<b>Part *</b>	element	New element.
See “Contents of the DeviceInfo element” on page 164.	<i>DeviceCondition</i> ?	enumeration	New attribute.
See “Contents of the JobPhase element” on page 165.	<i>CostType</i> ?	enumeration	New attribute.
See “Contents of the TrackFilter element” on page 168.	<i>QueueEntryID</i> ?	string	New attribute.
See “Contents of the TrackFilter element” on page 168.	<b>Part *</b>	element	New element.
See “Contents of the TrackResult element” on page 169.	<i>QueueEntryID</i> ?	string	New attribute.
See “Contents of the TrackResult element” on page 169.	<b>Part *</b>	element	New element.
See “Contents of the ResourceInfo element” on page 158.	<i>ActualAmount</i> ?	double	New attribute.



Location	Name	Data Type	Comment
See “Contents of the JobPhase element” on page 165.	<i>WorkType ?</i>	enumeration	New attribute.
See “Contents of the JobPhase element” on page 165.	<i>WorkTypeDetails ?</i>	string	New attribute.
See “Contents of the Queue element” on page 187.	<i>QueueEntry *</i>	element	Changed description.
See “Contents of the QueueEntry element” on page 188.	<i>DeviceID ?</i>	string	New attribute.
See “Contents of the QueueEntry element” on page 188.	<i>EndTime ?</i>	dateTime	New attribute.
See “Contents of the QueueEntry element” on page 188.	<i>Status</i>	enumeration	Added several enumerations.
See “Contents of the QueueEntry element” on page 188.	<i>JobPhase ?</i>	element	New element.
See “Contents of the QueueFilter Element” on page 189.	— Several —	— Various —	New element with multiple attributes and elements described.
See “Contents of the AbortQueueEntry message” on page 176.	<i>QueueFilter ?</i>	element	New element.
See “Contents of the HoldQueueEntry message” on page 176.	<i>QueueFilter ?</i>	element	New element.
See “Contents of the RemoveQueueEntry message” on page 177.	<i>QueueFilter ?</i>	element	New element.
See “Contents of the ResubmitQueueEntry message” on page 178.	<i>QueueFilter ?</i>	element	New element.
See “Contents of the ResumeQueueEntry message” on page 178.	<i>QueueFilter ?</i>	element	New element.
See “Contents of the SetQueueEntry message” on page 179.	<i>QueueFilter ?</i>	element	New element.
See “Contents of the SetQueueEntryPriority message” on page 180.	<i>QueueFilter ?</i>	element	New element.
See “Contents of the SubmitQueueEntry message” on page 180.	<i>QueueFilter ?</i>	element	New element.
See “Contents of the QueueSubmissionParams element” on page 181.	<i>ReturnURL ?</i>	URL	Changed description.
See “Contents of the QueueSubmissionParams element” on page 181.	<i>URL</i>	URL	Changed description.
See “Contents of the QueueSubmissionParams element” on page 181.	<i>WatchURL ?</i>	URL	Changed description.
See “Contents of the QueueSubmissionParams element” on page 181.	<i>Disposition ?</i>	element	New element.

Location	Name	Data Type	Comment
See “Contents of the CloseQueue message” on page 183.	QueueFilter ?	element	New element.
See “Contents of the FlushQueue Command message” on page 184.	QueueFilter ?	element	New element.
See “Contents of the FlushQueue Command message” on page 184.	FlushQueueParams ?	element	New element.
See “Contents of the HoldQueue message” on page 185.	QueueFilter ?	element	New element.
See “Contents of the OpenQueue message” on page 185.	QueueFilter ?	element	New element.
See “Contents of the QueueStatus message” on page 185.	QueueFilter ?	element	New element.
See “Contents of the ResumeQueue message” on page 186.	QueueFilter ?	element	New element.
See “Contents of the SubmissionMethods message” on page 186.	<i>JDFInline</i> = “false”	boolean	New attribute.
See “Contents of the SubmissionMethods message” on page 186.	<i>Packaging</i> ?	enumerations	New attribute.
See “Contents of the SubmissionMethods message” on page 186.	<i>URLSchemes</i> ?	NMTOKENS	New attribute.
See “Contents of the Queue element” on page 187.	<i>QueueSize</i> ?	integer	New attribute.
See “Contents of the QueueEntry element” on page 188.	Part *	element	New element.

### W.6.3 Processes

Location	Name	Comment
“Interpreting” on page 203	<b>RunList ?</b>	Added resource.
“LayoutPreparation” on page 204	<b>RunList ?</b> (Document)	Made optional.
“Rendering” on page 208	<b>RunList ?</b>	Added resource.
“ConventionalPrinting” on page 213	<b>Component</b>	Modified.
“Trimming” on page 232	<b>Component</b>	Modified description.
“Delivery” on page 193	Resource	Modified cardinality and description.
“QualityControl” on page 194	— Various —	New element with multiple attributes and elements described.
“Verification” on page 195	Resource	New element.
“ImageSetting” on page 201	ColorantControl	New input resource
“DigitalPrinting” on page 214	<b>Component</b> (Good)	Changed description.
“Preflight” on page 204	— Various —	Major modification of entire preflight section.

### W.6.4 Resources

Location	Name	Data Type	Comment
“ArtDeliveryIntent” on page 242	<i>Method ?</i>	NameSpan	Added multiple NameSpan values.
“ArtDeliveryIntent” on page 242	<i>Method ?</i>	NameSpan	Added multiple NameSpan values in ArtDelivery Elements.
“ArtDeliveryIntent” on page 242	— <i>several</i> —	— <i>various</i> —	Modified description by including <b>DigitMedia</b> .
“ArtDeliveryIntent” on page 242	<i>ServiceLevel ?</i>	stringSpan	New to <b>ArtDeliveryIntent</b> and its subelements.
“BindingIntent” on page 247	<i>BindingType</i>	Enumeration-Span	Added <i>CornerStitch</i> enumeration value.
“BindingIntent” on page 247	<b>HoleMakingParams ?</b>	refelement	Added to subelements CoilBinding, PlasticCombBinding, RingBinding, StripBinding, and WireCombBinding.
“DeliveryIntent” on page 258	<i>BuyerAccount ?</i>	string	New attribute in Structure of DeliveryIntent Elements, DropIntent.
“DeliveryIntent” on page 258	<i>SurplusHandling ?</i>	Enumeration-Span	New to <b>DeliveryIntent</b> and its subelements..
“HoleMakingIntent” on page 265	<b>HoleMakingParams ?</b>	refelement	New.
“LayoutElement” on page 406	<i>ElementType</i> = “Unknown”	enumeration	Added <i>MultiSet</i> enumeration.

Location	Name	Data Type	Comment
“LayoutElement” on page 406	<b>ImageCompressionParams ?</b>	refelement	New.
“LayoutElement” on page 406	<b>ScreeningParams ?</b>	refelement	New.
“LayoutElement” on page 406	<b>QualityControlResult *</b>	refelement	New.
“LayoutIntent” on page 267	<i>FinishedGrainDirection ?</i>	enumeration-Span	New attribute in Resource Structure table.
“LayoutIntent” on page 267	<i>NumberUp = “11”</i>	XYPair	In JDF 1.0 and 1.1 rows and columns were erroneously switched in the description, whihc was fixed in JDF 1.2.
“LayoutPreparationParams” on page 408	<i>NumberUp ?</i>	XYPair	In JDF 1.0 and 1.1 rows and columns were erroneously switched in the description, whihc was fixed in JDF 1.2.
“Media” on page 417	<b>QualityControlResult *</b>	refelement	New.
“MediaIntent” on page 270	<i>FrontCoatings = “None”</i>	Enumeration-Span	Added <i>Coated</i> and <i>InkJet</i> enumerations.
“ProofingIntent” on page 277	<b>SeparationSpec *</b>	refelement	Changed data type to refelement.
“ScreeningIntent” on page 279	— <i>Several</i> —	— <i>Various</i> —	New Resource with new attributes.
“Bundle” on page 289	<i>ItemName ?</i>	NMTOKEN	New attribute.
“ByteMap” on page 291	<b>ColorPool ?</b>	refelement	New in Resource Structure.
“Color” on page 297	<i>MappingSelection ?</i>	enumeration	New attribute.
“Color” on page 297	<i>RawName ?</i>	hexBinary	New attribute.
“Color” on page 297	<b>PrintConditionColor *</b>	element	New sub-element.
“ColorantControl” on page 303	<b>ColorantAlias *</b>	refelement	Modified data type from element to refelement.
“ColorantControl” on page 303	<b>DeviceNSpace *</b>	refelement	Modified data type from element to refelement.
“ColorantControl” on page 303	<b>SeparationSpec *</b>	refelement	Modified data type from element to refelement in Structure of ColorantOrder, ColorantParams, and DeviceColorantOrder Elements table.
“ColorantControl” on page 303	<b>PDLResourceAlias and SeparationSpec +</b>	refelement	Modified data type from element to refelement in Structure of ColorSpaceSubstitute Sub-element table.
“ColorSpaceConversionParams” on page 311	<b>SourceCS</b>	enumeration	Many new, modified, and clarified enumerations.
“ColorSpaceConversionParams” on page 311	<b>DeviceNSpace ?</b>	refelement	New in the Structure of ColorSpaceConversionOp Sub-element section.

Location	Name	Data Type	Comment
“ColorSpaceConversionParams” on page 311	<b>FileSpec ?</b>	refelement	New in the Structure of ColorSpaceConversionOp Subelement section.
“DeviceNSpace” on page 344	— <i>Several</i> —	— <i>Various</i> —	<b>DeviceNSpace</b> was elevated to a resource in JDF 1.2. Includes changes to subordinate attributes and refelements.
“ComChannel” on page 321	<i>ChannelType</i>	enumeration	Added enumeration values.
“ComChannel” on page 321	<i>ChannelTypeDetails ?</i>	NMTOKEN	New.
“ComChannel” on page 321	<i>ChannelUsage ?</i>	NMTOKEN	New
“ComChannel” on page 321	<i>Locator</i>	string	
“Component” on page 323	<i>ComponentType</i>	enumerations	Added and clarified enumeration values.
“Component” on page 323	<i>Condition</i> = “ <i>Good</i> ”	NMTOKEN	New.
“Component” on page 323	<b>QualityControlResult *</b>	refelement	New.
“Contact” on page 327	<i>ContactTypes</i>	NMTOKENS	Added <i>Approver</i> NMTOKEN.
“ConventionalPrintingParams” on page 328	<i>PrintingType</i>	enumeration	Added <i>ContinuousFed</i> enumeration.
“CoverApplicationParams” on page 331	<i>CoverOffset ?</i>	XYPair	Changed to optional.
“CreasingParams” on page 332	<i>Depth ?</i>	number	New.
“DeliveryParams” on page 338	<i>ServiceLevel ?</i>	string	New.
“DeliveryParams” on page 338	<i>ServiceLevel ?</i>	string	New in Structure of the Drop Subelement.
“DeliveryParams” on page 338	<i>Pickup ?</i>	string	Deprecated in Drop and replaced with <i>Transfer</i>
“DeliveryParams” on page 338	<i>TrackingID ?</i>	string	New in Structure of the Drop Subelement.
“DeliveryParams” on page 338	— <i>several</i> —	— <i>various</i> —	All attributes and elements in Structure of the DropItem Subelement table are new or modified.
“Employee” on page 353	<i>Roles ?</i>	NMTOKEN	New.
“ExposedMedia” on page 355	<b>QualityControlResult *</b>	refelement	New.
“FileSpec” on page 359	<i>AppOS</i> = “ <i>Unknown</i> ”	string	Changed data type and added reference.
“FileSpec” on page 359	<i>Compression</i> = “ <i>None</i> ”	NMTOKEN	Changed data type and added description.
“FileSpec” on page 359	<i>Encoding</i> = “ <i>None</i> ”	NMTOKEN	New.
“FileSpec” on page 359	<i>FilePath ?</i>	string	New.
“FileSpec” on page 359	<i>FileTargetDeviceModel ?</i>	string	New.
“FileSpec” on page 359	<i>MimeType ?</i>	string	Heavily modified description.

Location	Name	Data Type	Comment
“FileSpec” on page 359	<i>OSVersion ?</i>	string	Modified description including addition of examples.
“FileSpec” on page 359	<i>RawAlias ?</i>	hexBinary	New.
“FileSpec” on page 359	<b>FileSpec ?</b>	refelement	Added (new) to Structure of FileAlias Subelement.
“FormatConversionParams” on page 371	— <i>Several</i> —	— <i>Various</i> —	New components of the TIFFFormatParams element. Note: TTIFFFormatParams includes many new elements and attributes. Subelements such as TIFFTag are further detailed in subelement tables that includes many other new attributes.
“Fold” on page 366	<i>Travel ?</i>	double	Made optional; description modified.
“Fold” on page 366	<i>RelativeTravel ?</i>	double	New.
“HoleMakingParams” on page 380	<i>HoleCount ?</i>	integerList	New.
“ImageCompressionParams” on page 383	<i>ImageFilter ?</i>	NMTOKEN	A component of imageCompression Subelement — addition of enumerations.
“ImageCompressionParams” on page 383	— <i>Several</i> —	element	Addition of CCITTFaxParams, DCTParams, FlateParams, and LZWParams elements to Structure of ImageCompression Subelement table plus subelement tables for each detailing their attributes.
“ImageReplacementParams” on page 388	<i>ImagePreScanStrategy ?</i>	NMTOKEN	New attribute.
“ImageSetterParams” on page 390	<i>ManualFeed = “false”</i>	boolean	New.
“ImageSetterParams” on page 390	<i>Sides = “OneSidedFront”</i>	enumeration	New.
“ImageSetterParams” on page 390	<i>SourceWorkStyle = “Simplex”</i>	enumeration	New.
“InsertSheet” on page 396	<i>SheetFormat ?</i>	NMTOKEN	Added <i>Duplicate</i> value.
“InsertSheet” on page 396	<i>SheetUsage</i>	enumeration	Added <i>InterleavedBefore</i> value.
“JobField” on page 403	<i>ShowList</i>	NMTOKENS	Added multiple NMTOKEN values.
“LayoutElement” on page 406	<b>SeparationSpec *</b>	refelement	Changed data type to refelement.

Location	Name	Data Type	Comment
“Media” on page 417	<i>BackGlossValue ?</i>	number	New attribute.
“Media” on page 417	<i>CIETint ?</i>	double	New attribute.
“Media” on page 417	<i>CIEMWhiteness ?</i>	double	New attribute.
“Media” on page 417	<i>FrontCoatings = “None”</i>	enumeration	Added <i>Coated</i> and <i>Inkjet</i> enumerations.
“Media” on page 417	<i>FrontGlossValue ?</i>	number	New attribute.
“Media” on page 417	<i>LabColorValue ?</i>	LabColor	New attribute.
“Media” on page 417	<i>MediaColorNameDetails ?</i>	string	New attribute.
“Media” on page 417	<i>MediaType = “Unknown”</i>	enumeration	Added <i>Disc</i> and <i>Other</i> enumerations.
“Media” on page 417	<i>MediaUnit = “Sheet”</i>	enumeration	Added <i>Continuous</i> enumeration.
“Media” on page 417	<i>Opacity = “Opaque”</i>	enumeration	Added <i>Translucent</i> enumeration.
“Media” on page 417	<i>RecycledPercentage ?</i>	double	New attribute.
“Media” on page 417	<i>Texture ?</i>	NMTOKEN	Added <i>Uncalendar</i> value.
“ObjectResolution” on page 424	<i>AntiAliasing ?</i>	NMTOKEN	New attribute.
“PDFToPSConversionParams” on page 428	<i>OutputType = “PostScript”</i>	enumeration	Changed to optional with default value.
“PerforatingParams” on page 432	<i>Depth ?</i>	number	New.
“Preview” on page 442	<i>PreviewFileType = “PNG”</i>	enumeration	New.
“Preview” on page 442	<i>PreviewType</i>	enumeration	Added <i>SeparationRaw</i> enumeration.
“PreviewGenerationParams” on page 444	<i>PreviewUsage = “Separation”</i>	enumeration	Made <i>Separation</i> the default.
“PrintCondition” on page 445	new Resource		
“RegisterMark” on page 453	<i>SeparationSpec *</i>	refelement	Changed data type to refelement.
“RunList” on page 458	<i>EndOfBundleItem ?</i>	boolean	New attribute.
“RunList” on page 458	<i>ByteMap ?</i>	refelement	Redefined in descriptor.
“RunList” on page 458	<i>InterpretedPDLData ?</i>	refelement	New.
“RunList” on page 458	<i>LayoutElement ?</i>	refelement	Redefined in descriptor.
“ScavengerArea” on page 465	<i>SeparationSpec *</i>	refelement	Changed data type.
“ScreeningParams” on page 465	<i>ScreeningType = “AM”</i>	enumeration	Added enumerations and clarified old enumerations.
“SpinePreparationParams” on page 470	<i>MillingDepth ?</i>	double	Attribute changed to optional.
Section 7.2.153, <i>StitchingParams</i>	<i>NumberOfStitches ?</i>	integer	Changed interpretation of values.
Section 7.2.159, <i>ThreadSealingParams</i>	<i>ThreadPositions ?</i>	NumberList	Attribute changed to optional.

Location	Name	Data Type	Comment
Section 7.2.159, ThreadSealingParams	<i>ThreadLength</i> ?	double	Attribute changed to optional.
Section 7.2.159, ThreadSealingParams	<i>ThreadStitchWidth</i> ?	double	Attribute changed to optional.
Section 7.2.160, ThreadSewingParams	<i>GlueLineRefSheets</i> ?	IntegerList	Attribute changed to optional.
Section 7.2.160, ThreadSewingParams	<i>NumberOfNeedles</i> ?	integer	Attribute changed to optional.
“TrappingDetails” on page 494	<i>SeparationSpec</i> *	refelement	Changed data type.
“Structure of the DeviceCap Subelement” on page 502	<i>ExecutionPolicy</i> = "AllFound"	enumeration	New attribute.
“Structure of the DeviceCap Subelement” on page 502	<i>Lang</i> ?	languages	New attribute.
“Structure of the DeviceCap Subelement” on page 502	<i>TypeExpression</i> ?	regExp	New attribute.
“Structure of the DeviceCap Subelement” on page 502	<i>Types</i> ?	NMTOKEN	Modified description.
“Structure of the DeviceCap Subelement” on page 502	<i>DisplayGroupPool</i> ?	element	New element.
“Structure of the DeviceCap Subelement” on page 502	<i>ActionPool</i> ?	element	New element.
“Structure of the DeviceCap Subelement” on page 502	<i>FeaturePool</i> ?	element	New element.
“Structure of the DeviceCap Subelement” on page 502	<i>MacroPool</i> ?	element	New element.
“Structure of the DeviceCap Subelement” on page 502	<i>TestPool</i> ?	element	New element.
“Structure of the Performance Subelement” on page 524	<i>DevCapsRef</i> ?	IDREF	New attribute.
“Structure of the DevCaps Subelement” on page 505	<i>Availability</i> = "Installed"	enumeration	New attribute.
“Structure of the DevCaps Subelement” on page 505	<i>Context</i> = "Resource"	enumeration	New attribute.
“Structure of the DevCaps Subelement” on page 505	ID	ID	New attribute.
“Structure of the DevCaps Subelement” on page 505	<i>LinkUsage</i> = "Both"	enumeration	New attribute.
“Structure of the DevCaps Subelement” on page 505	<i>Name</i>	NMTOKEN	Modified definition.
“Structure of the DevCaps Subelement” on page 505	<i>Required</i> ?	boolean	New attribute.
“Structure of the DevCaps Subelement” on page 505	<i>TypeOccurrenceNum</i> ?	IntegerRange-List	New attribute.
“Structure of the DevCaps Subelement” on page 505	<i>Loc</i> *	element	New element.



Location	Name	Data Type	Comment
“Structure of the DevCap Subelement” on page 507	<i>Availability ?</i>	enumeration	New attribute.
“Structure of the DevCap Subelement” on page 507	Loc *	element	New element.
See “Structure of the FeaturePool Subelement” on page 521.	n/a	various	New subelement to DeviceCap with elements and attributes.
See “Structure of the MacroPool Subelement” on page 522.	n/a	various	New subelement to DeviceCap with elements and attributes.
See “Structure of the DisplayGroupPool Subelement” on page 520.	n/a	various	New subelement to DeviceCap with elements and attributes.
See “Structure of the Performance Subelement” on page 524.	n/a	various	New subelement to DeviceCap with elements and attributes.
“Structure of the Abstract State Subelement” on page 508	<i>Availability ?</i>	enumeration	New attribute.
“Structure of the Abstract State Subelement” on page 508	<i>ID</i>	ID	New attribute.
“Structure of the Abstract State Subelement” on page 508	<i>MaxOccurs = "1"</i>	integer	New attribute.
“Structure of the Abstract State Subelement” on page 508	<i>MinOccurs = "1"</i>	integer	New attribute.
“Structure of the Abstract State Subelement” on page 508	<i>Required ?</i>	boolean	New attribute.
“Structure of the Abstract State Subelement” on page 508	<i>ListType = "SingleValue"</i>	enumeration	New attribute.
“Structure of the Abstract State Subelement” on page 508	<i>ActionRefs ?</i>	IDREFS	New attribute.
“Structure of the Abstract State Subelement” on page 508	<i>Editable ?</i>	boolean	New attribute.
“Structure of the Abstract State Subelement” on page 508	<i>MacroRefs ?</i>	IDREFS	New attribute.
“Structure of the Abstract State Subelement” on page 508	<i>DependentMacroRef ?</i>	IDREF	New attribute.
“Structure of the Abstract State Subelement” on page 508	<i>UserDisplay = "Display"</i>	enumeration	New attribute.
“Structure of the Abstract State Subelement” on page 508	Loc *	element	New attribute.
“Structure of the Abstract State Subelement” on page 508	DateTimeState, DurationState, PDFPathState, and RectangleState	element	New State element types.
“Structure of the BooleanState Subelement” on page 511	ValueLoc *	element	New element.

Location	Name	Data Type	Comment
“Structure of the IntegerState Subelement” on page 513	<i>AllowedValueList ?</i>	IntegerRange-List	Modified data type.
“Structure of the IntegerState Subelement” on page 513	<i>AllowedValueMod ?</i>	XYPair	New attribute.
“Structure of the IntegerState Subelement” on page 513	<i>PresentValueList ?</i>	IntegerRange-List	Modified data type.
“Structure of the IntegerState Subelement” on page 513	<i>UnitType ?</i>	NMTOKEN	New attribute.
“Structure of the IntegerState Subelement” on page 513	<i>ValueLoc *</i>	element	New element.
See “Structure of the Loc Subelement” on page 507.	n/a	– various –	LOC element definition with new attributes — used throughout DeviceCaps subordinate elements.
See “Structure of the BooleanState Subelement” on page 511.	<i>ValueLoc</i>	– various –	<i>ValueLoc</i> element definition with new attributes — used throughout State element definitions.
See “Structure of the MatrixState Subelement” on page 514.	<i>AllowedRotateMod ?</i>	double	New attribute.
See “Structure of the MatrixState Subelement” on page 514.	<i>AllowedShift ?</i>	NumberList	New attribute.
See “Structure of the MatrixState Subelement” on page 514.	<i>AllowedTransforms ?</i>	enumerations	New attribute.
See “Structure of the MatrixState Subelement” on page 514.	<i>PresentRotateMod ?</i>	double	New attribute.
See “Structure of the MatrixState Subelement” on page 514.	<i>PresentShift ?</i>	NumberList	New attribute.
See “Structure of the MatrixState Subelement” on page 514.	<i>PresentTransforms ?</i>	enumerations	New attribute.
See “Structure of the MatrixState Subelement” on page 514.	<i>ValueUsage ?</i>	enumeration	New attribute in Structure of the Value Element table.
See “Structure of the MatrixState Subelement” on page 514.	<i>Loc *</i>	element	New element in Structure of the Value Element table.
See “Structure of the NameState Subelement” on page 515.	<i>AllowedRegExp ?</i>	regExp	New attribute.
See “Structure of the NameState Subelement” on page 515.	<i>PresentRegExp ?</i>	regExp	New attribute.

Location	Name	Data Type	Comment
See “Structure of the NameState Subelement” on page 515.	ValueLoc *	element	New element.
See “Structure of the NumberState Subelement” on page 516.	AllowedValueList ?	Number-RangeList	Changed data type.
See “Structure of the NumberState Subelement” on page 516.	AllowedValueMod ?	XYPair	New attribute.
See “Structure of the NumberState Subelement” on page 516.	PresentValueList ?	Number-RangeList	Changed data type.
See “Structure of the NumberState Subelement” on page 516.	UnitType ?	NMTOKEN	New attribute.
See “Structure of the NumberState Subelement” on page 516.	ValueLoc *	element	New element.
See “Structure of the PDFPathState Subelement” on page 517.	– Several –	– Various –	Definitions of elements and attributes of new element.
See “Structure of the RectangleState Subelement” on page 517.	– Several –	– Various –	Definitions of elements and attributes of new element.
See “Structure of the ShapeState Subelement” on page 518.	AllowedValueList ?	ShapeRange-List	Changed data type.
See “Structure of the ShapeState Subelement” on page 518.	AllowedX ?	Number-RangeList	New attribute.
See “Structure of the ShapeState Subelement” on page 518.	AllowedY ?	Number-RangeList	New attribute.
See “Structure of the ShapeState Subelement” on page 518.	AllowedZ ?	Number-RangeList	New attribute.
See “Structure of the ShapeState Subelement” on page 518.	PresentValueList ?	ShapeRange-List	Changed data type.
See “Structure of the ShapeState Subelement” on page 518.	PresentX ?	Number-RangeList	New attribute.
See “Structure of the ShapeState Subelement” on page 518.	PresentY ?	Number-RangeList	New attribute.
See “Structure of the ShapeState Subelement” on page 518.	PresentZ ?	Number-RangeList	New attribute.

Location	Name	Data Type	Comment
See “Structure of the ShapeState Subelement” on page 518.	ValueLoc *	element	New element.
See “Structure of the StringState Subelement” on page 519.	AllowedLength ?	IntegerRange	New attribute.
See “Structure of the StringState Subelement” on page 519.	AllowedRegExp ?	regExp	New attribute.
See “Structure of the StringState Subelement” on page 519.	PresentLength ?	IntegerRange	New attribute.
See “Structure of the StringState Subelement” on page 519.	PresentRegExp ?	regExp	New attribute.
See “Structure of the StringState Subelement” on page 519.	ValueUsage ?	enumeration	New attribute in Structure of the Value element table.
See “Structure of the StringState Subelement” on page 519.	Loc *	element	New element in Structure of the Value element table.
See “Structure of the XYPairState Subelement” on page 520.	AllowedValueList ?	XYPair-RangeList	Changed data type.
See “Structure of the XYPairState Subelement” on page 520.	PresentValueList ?	XYPair-RangeList	Changed data type.
See “Structure of the XYPairState Subelement” on page 520.	AllowedXYRelation ?	XYRelation	New attribute.
See “Structure of the XYPairState Subelement” on page 520.	PresentXYRelation ?	XYRelation	New attribute.
See “Structure of the XYPairState Subelement” on page 520.	UnitType ?	NMTOKEN	New attribute.
See “Structure of the XYPairState Subelement” on page 520.	ValueLoc *	element	New element.
See “Structure of the ActionPool Subelement” on page 504.	– Several –	– Various –	Definitions of elements and attributes of new element.
See “Structure of the TestPool Subelement” on page 524.	– Several –	– Various –	Definitions of elements and attributes of new element, as well as many new definitions of attributes and elements that are subordinate to TestPool.

Location	Name	Data Type	Comment
“Structure of ArtDelivery Elements” on page 244	<i>ArtDeliveryType</i>	NMTOKEN	Added <i>DigitalFile</i> value
“Structure of ArtDelivery Elements” on page 244	<b>DigitalMedia</b> ?	refelement	New.
“Structure of the SideStitching Subelement.” on page 254	<i>StitchNumber</i> ?	IntegerSpan	New attribute.
“ColorIntent” on page 255	<i>ColorStandard</i> ?	NameSpan	Change description.
“ColorIntent” on page 255	<b>SeparationSpec</b> *	refelement	Added color values to <b>SperationsSpec</b> element.
“DeliveryIntent” on page 258	<i>DeliveryCharge</i> = “Buyer”	Enumeration-Span	Added enumeration.
“HoleMakingIntent” on page 265	<i>Extent</i> ?	XYPair	New attribute.
“LayoutIntent” on page 267	<i>Pages</i> = “1”	IntegerSpan	Modified description and removed default value.
“LayoutIntent” on page 267	<i>RotatePolicy</i> = “NoRotate”	enumeration	New attribute.
“LayoutIntent” on page 267	<i>Sides</i> ?	enumeration	Added enumeration.
“LayoutIntent” on page 267	<i>SizePolicy</i> ?	Enumeration-Span	New attribute.
“MediaIntent” on page 270	<i>Brightness</i> ?	NumberSpan	New description.
“MediaIntent” on page 270	<i>GrainDirection</i> ?	Enumeration-Span	New attribute.
“MediaIntent” on page 270	<i>MediaColorDetails</i> ?	StringSpan	New attribute.
“MediaIntent” on page 270	<i>MediaType</i> = “Paper”	Enumeration-Span	Added new enumerations.
“MediaIntent” on page 270	<i>Opacity</i> = “Opaque”	Enumeration-Span	Added new enumeration and changed description.
“MediaIntent” on page 270	<i>OpacityLevel</i> ?	NumberSpan	New attribute.
“MediaIntent” on page 270	<i>RecycledPercentage</i> ?	NumberSpan	New attribute.
“ProofingIntent” on page 277	<i>ImageStrategy</i> ?	Enumeration-Span	New attribute in Structure of the ProofItem Element.
“ProofingIntent” on page 277	<b>ApprovalParams</b> ?	refelement	New reelement in Structure of the ProofItem Element.
“ShapeCuttingIntent” on page 279	<i>CutPath</i> ?	PDFPath	Changed data type in Structure of ShapeCut Subelement.
“ApprovalParams” on page 282	<i>MinApprovals</i> = “1”	integer	New attribute.
“ApprovalParams” on page 282	<i>ApprovalRole</i> = “Obligated”	enumeration	New attribute.
“ApprovalSuccess” on page 283	<b>Contact</b> *	refelement	New reelement.
“CoilBindingParams” on page 296	<b>HoleMakingParams</b> ?	refelement	New reelement.
“Color” on page 297	<i>ColorBookEntry</i> ?	string	New description.
“Color” on page 297	<i>ColorType</i> ?	enumeration	Changed description.

Location	Name	Data Type	Comment
“Color” on page 297	<i>MediaType ?</i>	string	Added enumeration.
“ColorCorrectionParams” on page 307	<i>AdjustCyanRed ?</i>	double	New attribute in Structure of ColorCorrectionOp Subelement.
“ColorCorrectionParams” on page 307	<i>AdjustMagentaGreen ?</i>	double	New attribute in Structure of ColorCorrectionOp Subelement.
“ColorCorrectionParams” on page 307	<i>AdjustYellowBlue ?</i>	double	New attribute in Structure of ColorCorrectionOp Subelement.
“ColorCorrectionParams” on page 307	<i>AdjustContrast ?</i>	double	New attribute in Structure of ColorCorrectionOp Subelement.
“ColorCorrectionParams” on page 307	<i>AdjustHue ?</i>	double	New attribute in Structure of ColorCorrectionOp Subelement.
“ColorCorrectionParams” on page 307	<i>AdjustLightness ?</i>	double	New attribute in Structure of ColorCorrectionOp Subelement.
“ColorCorrectionParams” on page 307	<i>AdjustSaturation ?</i>	double	New attribute in Structure of ColorCorrectionOp Subelement.
“ColorCorrectionParams” on page 307	<b>FileSpec ?</b>	refelement	New refelement in Structure of ColorCorrectionOp Subelement. (Two variants.)
“ColorSpaceConversionParams” on page 311	<i>ICCProfileUsage = “UsePDL”</i>	enumeration	New attribute.
“ColorSpaceConversionOp” on page 313	<i>Operation ?</i>	enumeration	Changed cardinality and description.
“ColorSpaceConversionOp” on page 313	<i>RenderingIntent = “ColorSpaceDependent”</i>	enumeration	Changed default and added an enumeration.
“ColorSpaceConversionOp” on page 313	<i>RGBGray2Black = “false”</i>	boolean	New description.
“ColorSpaceConversionOp” on page 313	<i>RGBGray2BlackThreshold = “1”</i>	double	New attribute.
“ColorSpaceConversionOp” on page 313	<i>SourceRenderingIntent ?</i>	enumeration	New attribute.
“ConventionalPrintingParams” on page 328	<b>ApprovalParams ?</b>	refelement	New refelement.
“CreasingParams” on page 332	<i>Travel ?</i>	double	New attribute.
“CreasingParams” on page 332	<i>RelativeTravel ?</i>	double	New attribute.
“CreasingParams” on page 332	<i>RelativeStartPosition ?</i>	XYPair	New attribute.
“CreasingParams” on page 332	<i>RelativeWorkingPath ?</i>	XYPair	New attribute.
“CreasingParams” on page 332	<i>StartPosition ?</i>	XYPair	Changed cardinality and description.
“CuttingParams” on page 335	<i>RelativeStartPosition ?</i>	XYPair	New attribute.

Location	Name	Data Type	Comment
“CuttingParams” on page 335	<i>RelativeWorkingPath</i> ?	XYPair	New attribute.
“CuttingParams” on page 335	<i>StartPosition</i> ?	XYPair	Changed cardinality and description.
“CuttingParams” on page 335	<i>WorkingPath</i>	XYPair	Changed cardinality and description.
“Device” on page 342	<i>JDFErrorURL</i> ?	URL	New attribute.
“Device” on page 342	<i>JDFInputURL</i> ?	URL	New attribute.
“Device” on page 342	<i>JDFOutputURL</i> ?	URL	New attribute.
“Device” on page 342	<i>KnownLocalizations</i> ?	languages	New attribute.
“DigitalPrintingParams” on page 347	<i>DirectProofAmount</i> = “0”	integer	New attribute.
“DigitalPrintingParams” on page 347	<i>OutputBin</i> ?	NMTOKEN	Suggested values modified and moved and linked to new appendix.
“DigitalPrintingParams” on page 347	<i>NonPrintableMarginBottom</i> ?	double	New attribute.
“DigitalPrintingParams” on page 347	<i>NonPrintableMarginLeft</i> ?	double	New attribute.
“DigitalPrintingParams” on page 347	<i>NonPrintableMarginRight</i> ?	double	New attribute.
“DigitalPrintingParams” on page 347	<i>NonPrintableMarginTop</i> ?	double	New attribute.
“DigitalPrintingParams” on page 347	<b>ApprovalParams</b> ?	refelement	New refelement.
“ElementColorParams” on page 351	— Several —	— Various —	New resource with multiple new attributes and elements.
“PageList” on page 425	— Several —	— Various —	New resource with multiple new attributes and elements.
“ExposedMedia” on page 355	<i>ProofName</i> ?	string	New attribute.
“Disposition” on page 350	— Several —	— Various —	New resource with multiple new attributes and elements.
“FileSpec” on page 359	<b>Disposition</b> ?	refelement	New refelement.
“GlueLine” on page 376	<i>RelativeStartPosition</i> ?	XYPair	New attribute.
“GlueLine” on page 376	<i>RelativeWorkingPath</i> ?	XYPair	New attribute.
“ImageCompressionParams” on page 383	<i>ImageAutoFilterStrategy</i> = “JPEG”	NMTOKEN	New attribute.
“ImageCompressionParams” on page 383	<i>JPXQuality</i> ?	integer	New attribute.
“ImageSetterParams” on page 390	<b>FitPolicy</b> ?	refelement	
“InkZoneCalculationParams” on page 392	<i>ZoneWidth</i> ?	double	Changed cardinality.
“InkZoneCalculationParams” on page 392	<i>Zones</i> ?	integer	Changed cardinality.
“InsertSheet” on page 396	<i>IncludeInBundleItem</i> ?	enumeration	New attribute.

Location	Name	Data Type	Comment
“LayoutElement” on page 406	<i>IsBlank</i> ?	boolean	New attribute.
“LayoutElement” on page 406	<i>HasBleeds</i> = “false”	boolean	Changed description and default.
“LayoutElement” on page 406	<i>IsPrintable</i> = “true”	boolean	Changed description and default.
“LayoutElement” on page 406	<i>IsTrapped</i> = “false”	boolean	Changed description and default.
“LayoutElement” on page 406	<i>PageListIndex</i> ?	IntegerRange-List	New attribute.
“LayoutElement” on page 406	<i>SourceBleedBox</i> ?	rectangle	Changed description and default.
“LayoutElement” on page 406	<i>SourceClipBox</i> ?	rectangle	Changed description and default.
“LayoutElement” on page 406	<i>SourceTrimBox</i> ?	rectangle	Changed description and default.
“LayoutElement” on page 406	<i>Template</i> = “false”	boolean	Changed description and default.
“LayoutElement” on page 406	<b>ColorPool</b> ?	refelement	New refelement.
“LayoutElement” on page 406	<b>Dependencies</b> ?	element	New element with subelement table defining attributes and elements also added to section.
“LayoutElement” on page 406	<b>FileSpec</b> ?	refelement	Changed cardinality and description.
“LayoutPreparationParams” on page 408	<i>HorizontalCreep</i> ?	IntegerList	Changed description.
“LayoutPreparationParams” on page 408	<i>PositionX</i> = “None”	enumeration	Added enumeration in Structure of the ImageShift Subelement.
“LayoutPreparationParams” on page 408	<i>PositionY</i> = “None”	enumeration	Added enumeration in Structure of the ImageShift Subelement.
“PerforatingParams” on page 432	<i>WorkingPath</i>	XYPair	Changed description.
“PlasticCombBindingParams” on page 433	<b>HoleMakingParams</b> ?	refelement	New refelement.
“Preview” on page 442	<i>PreviewUsage</i> = “Separation”	enumeration	New attribute — replacing <i>PreviewType</i> .
“PSToPDFConversionParams” on page 447	<b>PDFXParams</b> ?	element	New element ... with structure table added defining multiple elements and attributes.
“PSToPDFConversionParams” on page 447	<i>AllowPSXObjects</i> = “true”	boolean	New attribute in Structure of AdvancedParams Subelement.



Location	Name	Data Type	Comment
“PSToPDFConversionParams” on page 447	<i>AllowTransparency</i> = “false”	boolean	New attribute in Structure of AdvancedParams Subelement.
“PSToPDFConversionParams” on page 447	<i>EmbedJobOptions</i> = “false”	boolean	New attribute in Structure of AdvancedParams Subelement.
“PSToPDFConversionParams” on page 447	<i>PassThroughJPEGImages</i> = “false”	boolean	New attribute in Structure of AdvancedParams Subelement.
“PSToPDFConversionParams” on page 447	<i>SidelineEPS</i> = “false”	boolean	New attribute in Structure of ThinPDFParams Subelement.
“RingBindingParams” on page 456	<b>HoleMakingParams</b> ?	refelement	New refelement.
“RunList” on page 458	<i>ComponentGranularity</i> = “Document”	enumeration	New attribute.
“RunList” on page 458	<i>PageListIndex</i> ?	IntegerRange-List	New attribute.
“ScreeningParams” on page 465	<i>Frequency</i> ?	double	Changed attribute description in Structure of ScreenSelector Subelement.
“ScreeningParams” on page 465	<i>SourceFrequency</i> ?	DoubleRange	Changed attribute description and data type in Structure of ScreenSelector Subelement.
“StripBindingParams” on page 479	<b>HoleMakingParams</b> ?	refelement	New refelement.
“TransferCurvePool” on page 493	<i>Name</i>	NMTOKEN	Modified description of attribute in Structure of TransferCurveSet Subelement.
“TrappingParams” on page 495	<i>ImageTrapWidth</i> ?	double or XYPair	New attribute.
“TrappingParams” on page 495	<i>StepLimit</i> ?	double	Modified description of attribute.
“TrappingParams” on page 495	<i>TrapWidth</i> ?	double	Modified description of attribute.
“TrappingParams” on page 495	<i>TrapPlacement</i> = “Normal”	enumeration	New attribute.
“WireCombBindingParams” on page 501	<b>HoleMakingParams</b> ?	refelement	New refelement.
“Color” on page 297	<i>CMYK</i> ?	CMYKColor	Modified description of attribute.
“PreflightParams” on page 434	— Several —	— Various —	New resource with multiple new attributes and elements.
“PreflightReport” on page 437	— Several —	— Various —	New resource with multiple new attributes and elements.

Location	Name	Data Type	Comment
“PreflightReportRulePool” on page 440	— Several —	— Various —	New resource with multiple new attributes and elements.
“FileSpec” on page 359	<i>MimeTypeVersion ?</i>	string	New attribute.
“FileSpec” on page 359	<i>OverwritePolicy ?</i>	enumeration	New attribute.
“FileSpec” on page 359	<i>PageOrder ?</i>	enumeration	Added enumeration.
“FileSpec” on page 359	<i>Container ?</i>	element	New element with element structure table defining its attributes.
“LayoutElement” on page 406	<i>ClipPath ?</i>	PDFPath	Changed description.
“LayoutElement” on page 406	<b>ElementColorParams ?</b>	refelement	New refelement.
“Media” on page 417	<i>MediaTypeDetails ?</i>	NMTOKEN	Deleted and added NMTOKEN values.
“Preview” on page 442	<i>URL</i>	URL	Changed description.
“StackingParams” on page 473	<i>StandardAmount ?</i>	integer	Made optional.
“StitchingParams” on page 475	<i>StitchType ?</i>	enumeration	Changed description.
“PreviewGenerationParams” on page 444	<b>Device</b>	refelement	New refelement.
“InsertingParams” on page 394	<i>FinishedPage ?</i>	integer	New attribute.

---

# Appendix X    **Deprecated Processes, Resources, and JMF Messaging Elements**

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 withing 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.

## **X.1    Deprecated Processes**

### **X.1.1    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 6.6.48.5, Packaging Processes.

#### **Input Resources**

<b>Name</b>	<b>Description</b>
<b>PackingParams</b>	Necessary information about the packing process.
<b>PhysicalResource</b>	All kinds of physical resources can be packed.

#### **Output Resources**

<b>Name</b>	<b>Description</b>
<b>PhysicalResource</b>	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.

### **X.1.2    FilmToPlateCopying**

[Deprecated in JDF 1.1](#)

**FilmToPlateCopying** has been replaced by the more generic **ContactCopying**.

**FilmToPlateCopying** is the process of making an analog copy of a film onto a printing plate.

#### **Input Resources**

<b>Name</b>	<b>Description</b>
<b>DevelopingParams</b> ?	Controls the physical and chemical specifics of the media development process.
<b>ExposedMedia</b>	The film or films to be copied onto the plate.
<b>Media</b>	The unexposed plate.
<b>PlateCopyParams</b>	The settings of the exposure task.

#### **Output Resources**

<b>Name</b>	<b>Description</b>
<b>ExposedMedia</b>	The resulting exposed plate.

### X.1.3 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:	<b>Preflight</b>

#### Resource Structure

Name	Data Type	Description
ColorsResultsPool ?	element	A pool of <b>PreflightDetail</b> and <b>PreflightInstance</b> subelements that provides analysis about color.
DocumentResultsPool ?	element	A pool of <b>PreflightDetail</b> and <b>PreflightInstance</b> subelements that provides analysis about documents.
FontsResultsPool ?	element	A pool of <b>PreflightDetail</b> and <b>PreflightInstance</b> subelements that provides analysis about fonts.
FileTypeResultsPool ?	element	A pool of <b>PreflightDetail</b> and <b>PreflightInstance</b> subelements that provides analysis about file types.
ImagesResultsPool ?	element	A pool of <b>PreflightDetail</b> and <b>PreflightInstance</b> subelements that provides analysis about images.
PagesResultsPool ?	element	A pool of <b>PreflightDetail</b> and <b>PreflightInstance</b> subelements that provides analysis about finished pages.

#### Structure of PreflightDetail Subelement

**PreflightDetail** subelements are used to describe one property within the **PreflightAnalysis** category in which they occur. This subelement is also used by **PreflightInventory** resource.

Name	Data Type	Description
<i>PageRefs</i>	IntegerRangeList	Identifies the set of pages in a <b>RunList</b> 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.

Name	Data Type	Description
<i>Status ?</i>	enumeration	<p>Possible values are:</p> <p><i>Error</i> – Value violates the <b>ConstraintValue</b> specified in the associated <b>PreflightConstraint</b> element. The constraint was flagged as an Error in the profile.</p> <p><i>Warning</i> – Value violates the <b>ConstraintValue</b> specified in the associated <b>PreflightConstraint</b> element. The constraint was flagged as a Warning in the profile.</p> <p><i>Ignore</i> – The constraint is ignored, and no <b>PreflightDetail</b> or <b>PreflightInstanceDetail</b> elements are created for this constraint.</p> <p><i>IgnoreValue</i> – No comparison was made against a <b>ConstraintValue</b>. In other words, either the <i>Status</i> for the <b>PreflightConstraint</b> was <i>Ignore</i> or <i>IgnoreValue</i>, or this <b>PreflightDetail</b> is part of a <b>PreflightInventory</b> hierarchy.</p>
<i>Value ?</i>	element	Identifies the value of the property. The semantics are PDL-specific.

### Structure of PreflightInstance Subelement

**PreflightInstance** subelements are used to collect **PreflightInstanceDetail** elements for one instance of some object which occurs in the PDL files referenced by a run list. For example, there might be one **PreflightInstance** element for each font that occurs in the pages of a run list. This subelement is also used by **PreflightInventory** resources.

Name	Data Type	Description
<i>Identifier ?</i>	string	Identifies the instance this element collects <b>PreflightInstanceDetail</b> elements.
<i>PageRefs</i> <a href="#">Modified in JDF 1.1</a>	IntegerRange-List	Identifies the set of finished pages in a <b>RunList</b> on which the instance occurs.
<b>PreflightInstanceDetail</b> * <a href="#">Modified in JDF 1.1</a>	element	A pool of <b>PreflightInstanceDetail</b> elements that describe the properties for this instance

### Structure of PreflightInstanceDetail Subelement

**PreflightInstanceDetail** subelements describe one property of one instance of some object type that occurs in a PDL file. For example, several **PreflightInstanceDetail** elements might describe the properties of a single font. This subelement is also used by **PreflightInventory** resources.

Name	Data Type	Description
<i>Status ?</i>	enumeration	<p>Specifies the results of the comparison between the value of the property for this instance with the <b>ConstraintValue</b> for the associated <b>PreflightConstraint</b> element. Possible values are:</p> <p><i>Error</i> – Value violates the <b>ConstraintValue</b> specified. The constraint was flagged as an Error in the profile.</p> <p><i>Warning</i> – Value violates the <b>ConstraintValue</b> specified. The constraint was flagged as a Warning in the profile.</p> <p><i>IgnoreValue</i> – No comparison was made against a <b>ConstraintValue</b>. In other words, either the <i>Status</i> for the <b>Constraint</b> was <i>Ignore</i> or <i>IgnoreValue</i>, or this <b>PreflightInstanceDetail</b> is part of a <b>PreflightInventory</b> hierarchy.</p>
<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.

### X.1.4 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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	—
<b>Input of processes:</b>	<b>Preflight</b>
<b>Output of processes:</b>	<b>Preflight</b>

#### Resource Structure

Name	Data Type	Description
ColorsResultsPool ?	element	A pool of <b>PreflightDetail</b> and <b>PreflightInstance</b> subelements that provides a color inventory.
DocumentResultsPool ?	element	A pool of <b>PreflightDetail</b> and <b>PreflightInstance</b> subelements that provides a document inventory.
FontsResultsPool ?	element	A pool of <b>PreflightDetail</b> and <b>PreflightInstance</b> subelements that provides a font inventory.
FileTypeResultsPool ?	element	A <b>PreflightDetail</b> and <b>PreflightInstance</b> subelement that provides a file-type inventory.
ImagesResultsPool ?	element	A pool of <b>PreflightDetail</b> and <b>PreflightInstance</b> subelements that provides an image inventory.
PagesResultsPool ?	element	A pool of <b>PreflightDetail</b> and <b>PreflightInstance</b> subelements that provides a finished page inventory.

### X.1.5 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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	—
<b>Input of processes:</b>	<b>Preflight</b>

Output of processes: —

**Resource Structure**

Name	Data Type	Description
ColorsConstraintsPool ?	element	A pool of <b>PreflightConstraint</b> subelements. Each element in this pool identifies a specific constraint concerning colors against which to test the file
DocumentConstraintsPool ?	element	A pool of <b>PreflightConstraint</b> subelements. Each element in this pool identifies a specific constraint concerning documents against which to test the file
FontsConstraintsPool ?	element	A pool of <b>PreflightConstraint</b> subelements. Each element in this pool identifies a specific constraint concerning fonts against which to test the file
FileTypeConstraintsPool ?	element	A <b>Preflight</b> 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 <b>PreflightConstraint</b> subelements. Each element in this pool identifies a specific constraint concerning images against which to test the file
PagesConstraintsPool ?	element	A pool of <b>PreflightConstraint</b> subelements. Each element in this pool identifies a specific constraint concerning finished pages against which to test the file

**Structure of PreflightConstraint Subelement**

Name	Data Type	Description
<i>AttemptFixupErrors</i> = "false"	boolean	If <i>true</i> , the device performing preflight should attempt to fix errors that are identified during preflight. Errors that are corrected are not given a <i>Status</i> attribute. Default = "false"
<i>AttemptFixupWarnings</i> = "false"	boolean	If <i>true</i> , the device performing preflight should attempt to fix warnings that are identified during preflight. Warnings that are corrected are not given a <i>Status</i> attribute. Default = "false"
<i>Constraint</i> ?	string	Describes the specific file characteristic to be checked.
<i>Status</i>	enumeration	Possible values are: <i>Error</i> – Values that violate the <i>ConstraintValue</i> specified are flagged as Errors in <i>PreflightDetail</i> and <i>PreflightInstanceDetail</i> elements. <i>Warning</i> – Values that violate the <i>ConstraintValue</i> specified are flagged as Warnings in <i>PreflightDetail</i> and <i>PreflightInstanceDetail</i> elements. <i>Ignore</i> – The constraint is ignored, and no <i>PreflightDetail</i> or <i>PreflightInstanceDetail</i> elements are created for this constraint. <i>IgnoreValue</i> – No comparison is made against the <i>ConstraintValue</i> .
<i>ConstraintValue</i> ?	element	Provides a value against which to test occurrences of the characteristic in the file. Note: The semantics of the <i>ConstraintValue</i> element depend on the PDL characteristic in question.

**X.1.6 Proofing**[Deprecated in JDF 1.2](#)

The **Proofing** process is deprecated in JDF/1.2. Instead, use a combined process to produce the hard proof, (e.g., one that includes the **ImageSetting**, **ConventionalPrinting**, or **DigitalPrinting** process.) Then input the hard proof to a separate **Approval** process.

The **Proofing** process results in the creation of a physical proof, represented by an **ExposedMedia** resource. Proofs can be used to check an imposition or the expected colors for a job. The inputs of this process are a **RunList**, which identifies the pages to proof; the **ProofingParams** resource, which describes the type of proof to be created; and a **Media** resource to describe the physical media that will be used.

### Input Resources

Name	Description
<b>ColorantControl</b> ? <a href="#">Modified in JDF1.1A</a>	Identifies the color model used by the job.
<b>ColorSpaceConversionParams</b> ?	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 <b>RunList</b> have already been operated on by a previous colorspace conversion process.
<b>Layout</b> ?	Required if an imposition proof is desired.
<b>Media</b>	This resource characterizes the output media for the proof.
<b>ProofingParams</b>	This resource provides the parameters needed to produce the desired proof.
<b>RunList</b> (Document)	Identifies the pages to be proofed. When the <b>Layout</b> resource is present in the <b>ProofingParams</b> resource, <i>Ord</i> values from ContentObject subelements refer to pages in this <b>RunList</b> .
<b>RunList</b> ? (Marks)	Structured list of incoming marks. These are typically printers marks, (e.g., fold, cut or punch marks, or color bars.) When the <b>Layout</b> resource is present in the <b>ProofingParams</b> resource, <i>Ord</i> values from MarkObject subelements refer to pages in this <b>RunList</b> .

### Output Resources

Name	Description
<b>ExposedMedia</b>	The resulting physical proof.

#### X.1.7 SoftProofing

[Deprecated in JDF 1.2](#)

The SoftProofing process is deprecated in JDF/1.2. Instead, use a combined process to produce the soft proof in which the last process is the **Approval** process that approves the soft proof.

**SoftProofing** is the process of reviewing final-form output on a monitor rather than in paper form. The inputs are a **RunList**, which identifies the pages to proof; the **ProofingParams** resource, which describes the type of proof to be created.

Within the **ProofingParams** resource, the proof device parameter specifies the characterization the monitor on which the proof will be viewed. This processor must create and perform a transformation from the final target device to the proof device colors before displaying the document contents.

The soft proofing parameters allow sufficient control to determine whether any images are displayed in the proof. If so, the ability to select low resolution proxies or full resolution images is provided. The mechanism for approving proofs requires the generation of a PDF file containing the proofing parameters and a digital signature noting the acceptance of them. The approval PDF file need not contain any graphical data.



Like all other color manipulation supported in JDF, the color conversion controls are based on the use of ICC profiles. While the assumed characterization of input data can take many forms, each can internally be represented as an ICC Profile. In order to perform the transformations, input profiles must be paired with the identified final target device profile to create the transformation.

### Input Resources

Name	Description
<b>ColorantControl</b> ? <a href="#">Modified in JDF1.1A</a>	Identifies the color model used by the job.
<b>ColorSpaceConversionParams</b> ?	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 <b>RunList</b> have already been operated on by a previous colorspace conversion process.
<b>Layout</b> ?	Required if an imposition proof is desired.
<b>ProofingParams</b>	Provides the parameters needed to produce the desired proof.
<b>RunList</b> (Document)	Identifies the pages to be proofed. When the <b>Layout</b> resource is present in the <b>ProofingParams</b> resource, <i>Ord</i> values from <b>ContentObject</b> subelements refer to pages in this <b>RunList</b> .
<b>RunList</b> ? (Marks)	Structured list of incoming marks. These are typically printer's marks, e.g., fold marks, cut marks, punch marks, or color bars. When the <b>Layout</b> resource is present in the <b>ProofingParams</b> resource, <i>Ord</i> values from <b>MarkObject</b> subelements refer to pages in this <b>RunList</b> .

### Output Resources

None.

## X.1.8 IDPrinting

[Deprecated in JDF 1.1](#)

The IDPrinting process was deprecated in JDF/1.1. Instead, implementations should use the **DigitalPrinting** process combined with other processes, thus improving interoperability by reducing one of the combinations of processes. Also the **IDPrinting** process defined a number of resources and subelements which are deprecated since they duplicate other resources.

**IDPrinting**, which stands for Integrated Digital Printing, is a specific form of digital printing. It combines functionality that might be represented by the **Interpreting**, **Rendering**, **Screening**, and **DigitalPrinting** processes in a single process. In addition, devices which support **IDPrinting** frequently provide some degree of finishing capabilities, such as collating and stapling, as well as some automated layout capabilities, such as N-up and duplex printing.

Controls for **IDPrinting** are provided in the **IDPrintingParams** resource. These controls are intended to be somewhat limited in their scope. If greater control over various aspects of the printing process is required, **IDPrinting** should not be used. Ultimately, the controls specified for **IDPrinting** can be used to generate an Internet Printing Protocol (IPP) job. See JDF/1.0 Appendix F for a mapping between JDF IDPrinting and IPP. **IDPrinting** may be combined with other processes, such as **Trapping** or **ColorSpaceConversion**.

### Input Resources

Name	Description
<b>ColorantControl</b> ?	The <b>ColorantControl</b> resources that define the ordering and usage of inks in print modules.
<b>Component</b> ? (Cover)	A finished cover may be combined with the pages that will be output by this process.

<b>Component</b> ? (Input)	Various components can be used in <b>IDPrinting</b> instead of <b>Media</b> . Examples include waste, precut <b>Media</b> , or a set of preprinted sheets or webs.
<b>Component</b> ? (Proof)	A Proof component is used if a proof was produced during an earlier <b>ConventionalPrinting</b> process.
<b>ExposedMedia</b> ?	A <b>Proof</b> is useful for comparisons (completeness, color accuracy) with the print out of the <b>IDPrinting</b> process.
<b>FontPolicy</b> ?	Describes the behavior of the font machinery in absence of requested fonts.
<b>Ink</b> ?	Ink or toner and information about it is needed for <b>IDPrinting</b> .
<b>InterpretingParams</b> *	A set of resources that specify how the device should interpret the PDL files which are referenced by the <b>RunList</b> for the process. Note that <b>InterpretingParams</b> is an abstract resource. Instances are PDL-specific.
<b>IDPrintingParams</b> ?	Specific parameters to set up the machinery.
<b>Media</b> ?	The physical <b>Media</b> and information about the <b>Media</b> , such as thickness, type, and size, are used to set up paper travel in the press. This has to be present if no preprinted <b>Component</b> (input) resource is present. Note: Printing a job on more than one web or sheet at the same time is parallel processing.
<b>RenderingParams</b> ?	This resource describes the format of the <b>ByteMaps</b> to be created.
<b>RunList</b>	The set of pages to be printed.
<b>ScreeningParams</b> ?	Parameters specifying which halftone mechanism is to be applied and with what specific controls.
<b>TransferFunctionControl</b> ?	Controls whether the device performs transfer functions and what values are used when doing so.

### Output Resources

Name	Description
<b>Component</b> (Good)	Components are produced for other printing processes or postpress processes. Note that the <i>Amount</i> attribute of the ResourceLink to this resource indicates the number of copies which will be produced.
<b>Component</b> ? (Waste)	Produced waste, may be used by other processes.

### X.1.9 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.

### X.1.10 Dividing

[Deprecated in JDF 1.1](#)

**Dividing** has been replaced by **Cutting**. In-line finishing of web presses often includes equipment for cutting the ribbon(s) in cross direction. This operation can be described with the **Dividing** process. **Dividing** in cross direction

is likely to happen after former folding, which is a **LongitudinalRibbonOperations** process. It may affect one or more ribbons at the same time that are all part of one **Component**.

### Input Resources

Name	Description
<b>Component</b>	The <b>Dividing</b> process consumes one <b>Component</b> : the web(s) or ribbon(s) entering the crosscutting machinery. The substrate might have been treated with <b>LongitudinalRibbonOperations</b> and may be folded with a former fold.
<b>DividingParams</b>	Specific parameters to set up the machinery.

### Output Resources

Name	Description
<b>Component</b>	One <b>Component</b> is produced: either the divided web or ribbon.

## X.1.11 LongitudinalRibbonOperations

[Deprecated in JDF 1.1](#)

In-line finishing within web printing presses can include folding, perforating, or applying a line of glue on the ribbon while it is traveling in longitudinal direction. In version 1.1 of JDF and beyond, in-line finishing is described using the “standard” finishing processes, (e.g., **Creasing**, **Cutting**, **Folding** or in a combined node with **ConventionalPrinting**).

### Input Resources

Name	Description
<b>Component</b>	The <b>Component</b> can consist of more than one web or ribbon that has been combined with the <b>Gathering</b> process.
<b>LongitudinalRibbonOperationParams</b>	Specific parameters to set up the machinery tools for the <b>LongitudinalRibbonOperations</b> process.

### Output Resources

Name	Description
<b>Component</b> +	A ribbon is produced that is used in other postpress processes. If the <b>LongitudinalRibbonOperations</b> process was slitting, more than one <b>Component</b> is produced.

## X.1.12 SaddleStitching

[Deprecated in JDF 1.1](#)

In **SaddleStitching**, signatures are collected so that all sections have a common spine, and then stitched with staples through the spine. **SaddleStitching** has been replaced by **Stitching** in JDF 1.1.

### Input Resources

Name	Description
<b>Component</b>	The only required <b>Component</b> is the collected pile.
<b>SaddleStitchingParams</b>	Specific parameters to set up the machinery.

### Output Resources

Name	Description
<b>Component</b>	The stitched-together components.

## X.1.13 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 6.6.48.2, *HoleMaking*). Then the entire book is sewn at once with a *ThreadMaterial* such as *Cotton* or *Polyester*. If the book block is rather thick, a *Stitching* process using wire might be performed before *SideSewing*.

### Input Resources

Name	Description
<b>Component</b>	The only required <b>Component</b> is the gathered sheets.
<b>SideSewingParams</b>	Specific parameters to set up the machinery.

### Output Resources

Name	Description
<b>Component</b>	The <b>Component</b> is produced.

## X.2 Deprecated Resources

### X.2.1 BindingIntent Deprecated Subelements

Note: **BindingIntent** is still a valid resource. The following sections from within **BindingIntent** were deprecated and were deemed large enough to warrant moving them to this section.

#### Structure of the AdhesiveBinding Subelement.

[Deprecated in JDF 1.1](#)

Name	Data Type	Description
<i>Scoring?</i>	EnumerationSpan	Scoring option for <b>AdhesiveBinding</b> . Possible values are: <i>TwiceScored</i> <i>QuadScored</i> <i>None</i> Values are based on viewing the cover in its flat pre-binding state.
<i>SpineGlue?</i>	EnumerationSpan	Glue type used to define <b>AdhesiveBinding</b> procedures. Possible values are: <i>ColdGlue</i> <i>Hotmelt</i> <i>PUR</i> – Polyurethane Rubber
<i>TapeBinding?</i>	OptionSpan	If " <i>true</i> ", a cloth tape which has been pre-glued with hot-melt adhesive is used in <b>AdhesiveBinding</b> the unmilled block, (e.g., FastBack or DocuTech binding).

#### Structure of the BookCase Subelement.

[Deprecated in JDF 1.1](#)

This subelements contains details of the book case for hard-cover book binding. The actual binding parameters are set in the appropriate **AdhesiveBinding**, **ThreadSewing**, or **ThreadSealing** elements.

Name	Data Type	Description
<i>HeadBands?</i>	OptionSpan	The following <b>CaseBinding</b> choice specifies the use of headbands on a case bound book. If " <i>true</i> ", headbands are inserted both top and bottom.

<i>Shape ?</i>	EnumerationSpan	Indicates the shape of the “back” or spine of a case bound book. Possible values are: <i>RoundedBack</i> <i>SquareBack</i>
<i>Thickness ?</i>	NumberSpan	Specifies thickness of board which is wrapped as front and back covers of a case bound book, in points.

### X.2.2 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.

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:	<b>CutMark, CuttingParams, Layout, LayoutPreparationParams, Sheet, Surface, TrimmingParams</b>
Example Partition:	<i>Option</i>
Input of processes:	Any Product Node
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>Dimensions</i>	XYPairSpan	Specifies the height and width of the product component in points. Note: Height and width are ambiguously specified in JDF 1.0.
<i>Pages ?</i>	IntegerSpan	Specifies the number of pages of the product component.
<i>Type = "Folded"</i>	enumeration	Specifies whether the product component referred to is flat or finished. Possible values are: <i>Folded</i> = Size of the product after folding. The default value <i>Flat</i> = Size of the unfolded sheet. Note that this describes the size of a sheet that is folded to create a product, not the size of the sheet in the press.

### X.2.3 AdhesiveBindingParams

#### [Deprecated in JDF 1.1](#)

This resource describes the details of the following four subprocesses of the **AdhesiveBinding** process:

- Back preparation
- Multiple glue applications
- Spine taping
- Cover application

These subprocesses are identified as instances of the abstract **ABOperation** element. Although a workflow may exist that groups these processes according to its own capabilities, it is likely that they will be performed in the order presented. A description of each follows the table containing the contents of the **AdhesiveBindingParams** resource.

#### Resource Properties

Resource class: Parameter  
 Resource referenced by: —  
 Example Partition: —  
 Input of processes: **AdhesiveBinding**  
 Output of processes: —

**Resource Structure**

Name	Data Type	Description
<i>FlexValue ?</i>	double	Flex quality parameter given in [N/cm].
<i>PullOutValue ?</i>	double	Pull out quality parameter given in [N/cm].
ABOperation +	Element	An abstract element which is a placeholder for an operation (SpinePreparation, GlueApplication, SpineTaping, and CoverApplication). Each ABOperation element describes the parameters of one single operation of the complete <b>AdhesiveBinding</b> process.

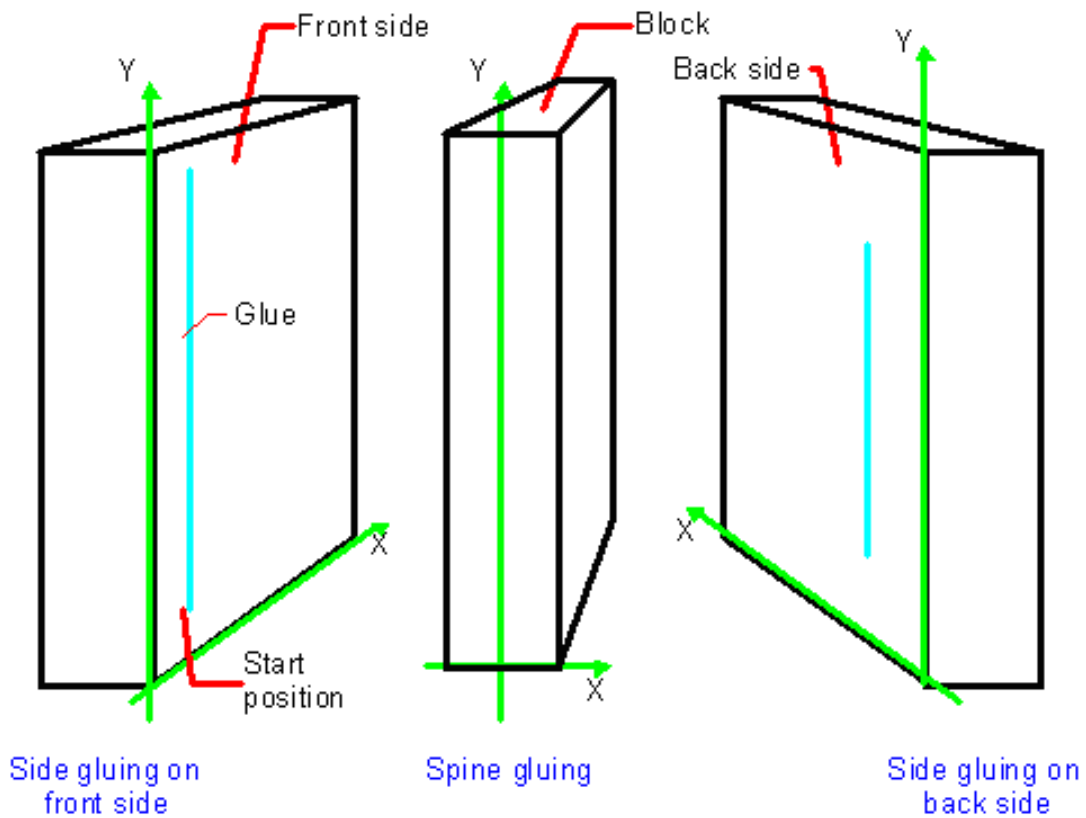


Figure X.1: Parameters and coordinate system for glue application

## X.2.4 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:	<b>Dividing</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>DividePositions</i>	DoubleList	Array containing the cross cut positions in y-direction (direction of web traveling).

## X.2.5 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:	<b>IDPrinting</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>AttributesNaturalLang = "US English"</i>	language	Language selected for communicating attributes. Default = "US English"
<i>IDPAttributeFidelity = "false"</i>	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 = "50"</i>	integer	The scheduling priority for the job where 100 is the highest and 1 is the lowest. Amongst the jobs that can be printed, all higher priority jobs must be printed before any lower priority ones. Default = 50
<i>IPPVersion ?</i>	XYPair	A pair of numbers indicating the version of the IPP protocol to use when communicating to IPP devices. The X value is the major version number.

Name	Data Type	Description
<i>OutputBin ?</i>	NMTO-KEN	<p>Specifies the bin to which the finished document should be output. Possible values are:</p> <p><i>Top</i> – The bin that, when facing the device, can best be identified as "top".</p> <p><i>Middle</i> – The bin that, when facing the device, can best be identified as "middle".</p> <p><i>Bottom</i> – The bin that, when facing the device, can best be identified as "bottom".</p> <p><i>Side</i> – The bin that, when facing the device, can best be identified as "side".</p> <p><i>Left</i> – The bin that, when facing the device, can best be identified as "left".</p> <p><i>Right</i> – The bin that, when facing the device, can best be identified as "right".</p> <p><i>Center</i> – The bin that, when facing the device, can best be identified as "center".</p> <p><i>Rear</i> – The bin that, when facing the device, can best be identified as "rear".</p> <p><i>FaceUp</i> – The bin that can best be identified as "face up" with respect to the device.</p> <p><i>FaceDown</i> – The bin that can best be identified as "face down" with respect to the device.</p> <p><i>FitMedia</i> – Requests the device to select a bin based on the size of the media.</p> <p><i>LargeCapacity</i> – The bin that can best be identified as the "large capacity" bin (in terms of the number of sheets) with respect to the device.</p> <p><i>Mailbox-N</i> – The job will be output to the bin that is best identified as "Mailbox-1", "Mailbox-2" ...etc.</p> <p><i>Stacker-N</i> – The job will be output to the bin that is best identified as "Stacker-1", "Stacker-2" ...etc.</p> <p><i>Tray-N</i> – The job will be output to the tray that is best identified as "Tray-1", "Tray-2" ... etc.</p>
<i>PageDelivery ?</i>	enumeration	<p>Indicates how pages are to be delivered to the output bin or finisher. Possible values are:</p> <p><i>SameOrderFaceUp</i> – Order as defined by the RunList, with the "front" sides of the media up.</p> <p><i>SameOrderFaceDown</i> – Order as defined by the RunList, with the "front" sides of the media up.</p> <p><i>ReverseOrderFaceUp</i> – Order reversed, as defined by the RunList, with the "front" sides of the media up.</p> <p><i>ReverseOrderFaceDown</i> – Order reversed, as defined by the RunList, with the "front" sides of the media down.</p>
<i>PrintQuality ?</i>	enumeration	<p>Indicates how pages are to be delivered to the output bin or finisher. Possible values are:</p> <p><i>High</i> – Highest quality available on the printer.</p> <p><i>Normal</i> – The default quality provided by the printer.</p> <p><i>Draft</i> – Lowest quality available on the printer.</p>
<i>SheetCollate ?</i>	boolean	<p>Determines whether the sequencing of the leaves in the output of the job.</p> <p>If <i>true</i>, sheets for each copy of the document are sequenced together, followed by the sheets for the next copy.</p> <p>If <i>false</i>, all copies of the first sheet are sequenced, followed by the second and subsequent sheet.</p> <p><i>SheetCollate</i> describes the order of the final sheet, but does not prescribe the order in which they are produced.</p>
<i>Cover *</i>	element	0, 1 or 2 <i>Cover</i> elements. The default instance is that there is no cover.



Name	Data Type	Description
IDPFinishing ?	refelement	This element provides the details of how media for each instance document should 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 <b>MediaIntent</b> element. This element is ignored if a <b>MediaSource</b> resource is present and can be honored for the <b>IDPrinting</b> process. If <b>MediaSource</b> 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.
<b>MediaSource</b> ?	refelement	Describes the source and physical orientation of the media to be used.

### Structure of the Cover Subelement

[Deprecated in JDF 1.1](#)

This element describes the cover requested for the job. Covers may be applied to the whole job, or to each instance document in the job. Note that front and back covers may be specified.

Name	Data Type	Description
<i>BackSide</i> = "false"	boolean	The next page from the RunList is imaged onto the back of this cover. This would be the inside of a <i>Front</i> cover and outside of a <i>Back</i> cover. Default = "false"
<i>CoverType</i> = "Front"	enumeration	Specifies whether this <b>Cover</b> element specifies the front or back cover. <i>Front</i> – The front cover. <i>Back</i> – The back cover. Default = "Front"
<i>FrontSide</i> = "false"	boolean	The next page from the RunList is imaged onto the front of this cover. This would be the outside of a <i>Front</i> cover and inside of a <i>Back</i> cover. Default = "false"
IDPFinishing ?	refelement	An IDPFinishing element that describes the finishing options for the cover.
IDPLayout ?	element	This element provides the details of how page contents will be imaged onto the cover.
MediaIntent ?	refelement	A <b>MediaIntent</b> element. This element describes the media to be used for the job. This element is ignored if a <b>MediaSource</b> resource is present and can be honored for the <b>IDPrinting</b> process. If <b>MediaSource</b> 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.
<b>MediaSource</b> ?	refelement	Describes the source and physical orientation of the media to be used.

### Properties of the IDPFinishing Subelement

[Deprecated in JDF 1.1](#)

IDPFinishing elements describe finishing operations that should 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 pro-

vides controls for a finishing operation. Additional subelements are expected to be defined over time. Also, more detail will be added to the currently defined elements.

Name	Data Type	Description
<i>Finishings</i> ?	IntegerList	<p>A set of finishing operations to apply to the job. The operations are encoded as an enumeration. Possible values are:</p> <p>3 – (none) Perform no finishing</p> <p>4 – (staple) Bind the document(s) with one or more staples. The exact number and placement of the staples is site-defined.</p> <p>5 – (punch) This value indicates that holes are required in the finished document. The exact number and placement of the holes is site-defined. The punch specification may be satisfied (in a site- and implementation-specific manner) either by drilling/punching, or by substituting predrilled media.</p> <p>6 – (cover) This value is specified when it is desired to select a non-printed (or preprinted) cover for the document. This does not supplant the specification of a printed cover (on cover stock medium) by the document itself.</p> <p>7 – (bind) This value indicates that a binding is to be applied to the document; the type and placement of the binding is site-defined.</p> <p>8 – (saddle-stitch) Bind the document(s) with one or more staples (wire stitches) along the middle fold. The exact number and placement of the staples and the middle fold is implementation and/or site-defined.</p> <p>9 – (edge-stitch) Bind the document(s) with one or more staples (wire stitches) along one edge. The exact number and placement of the staples is implementation and/or site-defined.</p> <p>10 – (fold) Fold the document(s) with one or more folds. The exact number and orientations of the folds is implementation and/or site-defined.</p> <p>11 – (trim) Trim the document(s) on one or more edges. The exact number of edges and the amount to be trimmed is implementation and/or site-defined.</p> <p>12 – (bale) Bale the document(s). The type of baling is implementation and/or site-defined.</p> <p>13 – (booklet-maker) Deliver the document(s) to the signature booklet maker. This value is a short cut for specifying a job that is to be folded, trimmed and then saddle-stitched.</p> <p>14 – (jog-offset) Shift each copy of an output document from the previous copy by a small amount which is device dependent. This value has no effect on the “job-sheet.” This value should not have an effect if each copy of the job consists of one sheet.</p> <p>50 – (bind-left) Bind the document(s) along the left edge. The type of the binding is site-defined.</p> <p>51 – (bind-top) Bind the document(s) along the top edge. The type of the binding is site-defined.</p> <p>52 – (bind-right) Bind the document(s) along the right edge. The type of the binding is site-defined.</p> <p>53 – (bind-bottom) Bind the document(s) along the bottom edge. The type of the binding is site-defined.</p>
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.

Name	Data Type	Description
IDPStitching ?	refelement	Provides details of how to stitch the set of pages (or document). When this element is present, <i>Finishings</i> is ignored.
IDPTrimming ?	refelement	Provides details of how to trim the set of pages (or document). When this element is present, <i>Finishings</i> is ignored.

### Structure of IDPFolding Subelement

[Deprecated in JDF 1.1](#)

This element describes the folding requested for a set of pages in the document.

Name	Data Type	Description
<b>FoldingParams ?</b>	Refelement	Describes the details of how to fold the media.

### Structure of IDPHoleMaking Subelement

[Deprecated in JDF 1.1](#)

This element describes the hole making requested for a set of pages in the document.

Name	Data Type	Description
<b>HoleMakingParams ?</b>	refelement	Describes the details of the holes to be punched into the Media.

### Structure of the IDPLayout Subelement

[Deprecated in JDF 1.1](#)

Name	Data Type	Description
<i>Border</i> = "0"	number	A real number that indicates the width of a border, in points, which will be drawn around the page images on the media. Default = "0", (i.e., no border will be drawn).
<i>FinishedPageOrientation</i> = "Portrait"	enumeration	Indicates the desired orientation of the finished page. This value is used with <i>PresentationDirection</i> to determine how pages will be imaged onto the media. Possible values are: <i>Portrait</i> – The short edges of the media are the top and bottom. <i>Landscape</i> – The long edges of the media are the top and bottom. Default = "Portrait".
<i>ForceFrontSide</i> ?	Number-RangeList	A set of numbers which identify a set of finished pages in the <b>RunList</b> that should always be imaged on the front side of a piece of media.
<i>ImageShift</i> ?	element	Element which describes how page images should be placed onto the media. When <i>NumberUp</i> is present and is not "1,1", <i>NumberUp</i> is applied before the <i>ImageShift</i> , and all contents for each surface are shifted the same amount.
<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. Possible 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>
<i>Sides</i> = “OneSided”	enumeration	<p>Indicates how pages should be imposed onto sides of the medium. Possible 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>

### Structure of IDPStitching Subelement

[Deprecated in JDF 1.1](#)

This element describes the stitching requested for a set of pages in the document.

Name	Data Type	Description
<i>StitchingPosition</i> ?	enumeration	Specifies the location for stitching. All locations are interpreted as if the document were a portrait document. Ignored if <i>StitchingParams</i> is present. Possible values are: <i>None</i> – The document is not to be stitched. <i>TopLeft</i> – Bind the document with one or more staples in the top left corner. <i>BottomLeft</i> – Bind the document with one or more staples in the Bottom left corner. <i>TopRight</i> – Bind the document with one or more staples in the top right corner. <i>BottomRight</i> – Bind the document with one or more staples in the bottom right corner. <i>LeftEdge</i> – Bind the document with one or more staples across the left edge. <i>TopEdge</i> – Bind the document with one or more staples across the top edge. <i>RightEdge</i> – Bind the document with one or more staples across the right edge. <i>BottomEdge</i> – Bind the document with one or more staples across the bottom edge. <i>DualLeftEdge</i> – Bind the document with two staples across the left edge. <i>DualTopEdge</i> – Bind the document with two staples across the top edge. <i>DualRightEdge</i> – Bind the document with two staples across the right edge. <i>DualBottomEdge</i> – Bind the document with two staples across the bottom edge.
<i>StitchingReferenceEdge</i> ?	enumeration	The edge of the output media relative to which the stapling or stitching must be applied. If <i>StitchingParams</i> is present, <i>StitchingReferenceEdge</i> defines the <i>BindingEdge</i> . Possible values are: <i>Bottom</i> – The bottom edge coincides with the x-axis of the coordinate system. <i>Top</i> – The top edge is opposite and parallel to the bottom edge. <i>Left</i> – The left edge coincides with the y-axis of the coordinate system. <i>Right</i> – The right edge is opposite and parallel to the left edge.
<i>StitchingParams</i> ?	refelement	A <i>StitchingParams</i> element which provides detailed control of the stitching. <i>StitchingReferenceEdge</i> must be present if <i>StitchingParams</i> is provided.

### Structure of IDPTrimming Subelement

[Deprecated in JDF 1.1](#)

This element describes the trimming requested for a set of pages in the document.

Name	Data Type	Description
<i>TrimmingParams</i> ?	Refelement	Describes the details of how to trim the media.

### Structure of the ImageShift Subelement

[Deprecated in JDF 1.1](#)

**ImageShift** elements describe how finished page contents will be imaged onto media. All attributes refer to positioning along the “X” or “Y” axis. The “X” dimension is the first number of the *Media Dimension* attribute; “Y” is the second number.

Name	Data Type	Description
<i>PositionX</i> = “None”	enumeration	Indicates how finished page images should 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 should be positioned vertically on the surface. Shifts are applied after positioning. Values are: <i>Bottom</i> – Position the bottom edge of the page images so they is coincident with the bottom edge of the printable area of the surface. <i>Center</i> – Center the page images horizontally on the surface without regard to limitations of the printable area. <i>None</i> – Place the page images wherever the print data specifies (the default). <i>Top</i> – Position the top edge of the page images so they is coincident with the top edge of the printable area of the surface.
<i>ShiftX</i> ?	integer	The image is to be shifted along the x axis on both sides of the media.
<i>ShiftY</i> ?	integer	The image is to be shifted along the y axis on both sides of the media.
<i>ShiftXSide1</i> ?	integer	The image is to be shifted along the x axis on the front side of the media.
<i>ShiftXSide2</i> ?	integer	The image is to be shifted along the x axis on the back side of the media.
<i>ShiftYSide1</i> ?	integer	The image is to be shifted along the y axis on the front side of the media.
<i>ShiftYSide2</i> ?	integer	The image is to be shifted along the y axis on the back side of the media.

### Structure of the JobSheet Subelement

#### [Deprecated in JDF 1.1](#)

This element describes a job sheet which may be produced along with the job. Job sheets include separators, sheets, and error sheets. The information provided on the sheet depends on the type of sheet. In addition, any sheet type may

include an optional message as a comment subelement for the sheet element. Such a message comment must have a *Name* attribute with the value ‘SheetMessage’.

Name	Data Type	Description
<i>SheetFormat</i> = "Standard"	NMTOKEN	Identifies the format of the JobSheet. The default is "Standard", but site-specific values may be defined.
<i>SheetOccurrence</i>	enumeration	Indicates when the sheet is to be produced and inserted into the set of output pages. Possible values are: <i>Always</i> – Valid for <i>ErrorSheet</i> or <i>AccountingSheet</i> . The sheet is always produced at the end of the job. <i>End</i> – Valid for <i>JobSheet</i> or <i>SeparatorSheet</i> . The sheet is produced at the end of the job (for <i>JobSheet</i> ) or at the end of each copy of each instance document (for <i>SeparatorSheet</i> ). <i>OnError</i> – Valid for <i>ErrorSheet</i> . The sheet is produced at the end of the job when an error or warning occurs. <i>Slip</i> – Valid for <i>SeparatorSheet</i> . The sheet is produced between each copy of each instance document. <i>Start</i> – Valid for <i>JobSheet</i> or <i>SeparatorSheet</i> . The sheet is produced at the start of the job (for <i>JobSheet</i> ) or at the start of each copy of each instance document (for <i>SeparatorSheet</i> ). <i>Both</i> – Valid for <i>JobSheet</i> or <i>SeparatorSheet</i> . The sheet is produced at the beginning and end of the job (for <i>JobSheets</i> ) or at the beginning and end of each copy of each instance document (for <i>SeparatorSheets</i> ). <i>None</i> – Valid for any <i>SheetType</i> .
<i>SheetType</i>	enumeration	Identifies the type of sheet. Possible values are: <i>AccountingSheet</i> – A sheet that reports accounting information for the job. <i>ErrorSheet</i> – A sheet that reports errors for the job. <i>JobSheet</i> – A sheet that delimits the job. <i>SeparatorSheet</i> – A sheet that delimits one copy (set) of the job.
IDPFinishing ?	refelement	An IDPFinishing element that describes the finishing options for the job sheet.
IDPLayout ?	element	This element provides the details of how page contents will be imaged onto the job sheet.
MediaIntent ?	refelement	A MediaIntent element. This element describes the media to be used for the job sheets. This element is ignored if a <b>MediaSource</b> resource is present and can be honored. If <b>MediaSource</b> 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.
<b>MediaSource</b> ?	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 3.8.2, Description of Partitionable Resources. Overrides may apply to a set of instance documents, set of copies of instance documents, or to a set of finished pages, output surfaces, sheets of media in a personalized printing job, or header or trailer insert sheets added by a RunList. Note: If more than one override refers to the same content, the lowest level override takes precedence. The following list defines partitioning precedence, from lowest to highest, (i.e., the lower entries in the list take precedence):

Job level partitioning (*lowest priority*):

*PartVersion, Run, SheetName, Side, RunTags*

Page level partitioning:

*RunIndex*

*SheetIndex*

Instance document level partitioning (*highest priority*):

*DocCopies*

*DocIndex*

*DocSheetIndex*

*DocRunIndex*

Note: It is strongly discouraged to mix page-level partitions and instance document-level partitions. **Cover** elements in **IDPrintingParams** are counted when calculating *DocSheetIndex* or *DocRunIndex*.

### Example of a partitioned IDPrinting Node

The following example shows how partitioning can be used to describe a fairly complex example. Three color models (**ColorantControl** partitions) are applied to a set of sheets using the *DocSheetIndex* key;

- 1 *DeviceN:DocSheetIndex* = "0" defines the cover;
- 2 *DeviceCMYK DocSheetIndex* = "1" defines the first sheet (non cover);
- 3 *DeviceGray:DocSheetIndex* = "2~1" defines all other sheets;

The cover is selected from a different input tray using the *Location* key. The same key is used to describe the **Media** in each tray.

```
<?xml version='1.0' encoding='utf-8' ?>
<JDF ID="HDM20010402140111" Type="IDPrinting" JobID="HDM20010402140111"
Status="Waiting" Version="1.2">
  <ResourcePool>
    <Media ID="Link0003" Class="Consumable" Locked="false" Status="Available"
Dimension="700 900" MediaType="Paper" PartIDKeys="Location">
      <Media Weight="90" Location="Tray 1"/>
      <Media Weight="120" Location="Tray 2"/>
    </Media>
    <RunList ID="Link0004" Class="Parameter" Locked="false" Status="Available"
PartIDKeys="Run">
      <RunList Run="Run0005" Pages="0">
        <LayoutElement>
          <FileSpec URL="Cover.pdf"/>
        </LayoutElement>
      </RunList>
      <RunList Run="Run0006" Pages="0~7">
        <LayoutElement>
          <FileSpec URL="File2.pdf"/>
        </LayoutElement>
      </RunList>
    </RunList>
    <IDPrintingParams ID="Link0008" Class="Parameter" Locked="false"
Status="Available">
      <IDPLayout NumberUp="2 2"/>
      <MediaSource MediaLocation="Tray 1">
        <MediaRef rRef="Link0003"/>
      </MediaSource>
      <Cover CoverType="Front" FrontSide="true">
        <IDPLayout NumberUp="1 1"/>
        <MediaSource MediaLocation="Tray 2">
          <MediaRef rRef="Link0003"/>
        </MediaSource>
      </Cover>
    </IDPrintingParams>
  </ResourcePool>
</JDF>
```



```

    <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>

```

## X.2.6 LongitudinalRibbonOperationParams

[Deprecated in JDF 1.1.](#)

This resource provides the parameters of the **LongitudinalRibbonOperation** process. It is defined as a list of abstract *LROperation* elements.

### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	<i>RibbonName, SheetName, SignatureName, WebName</i>
Input of processes:	<b>LongitudinalRibbonOperations</b>
Output of processes:	—

### Resource Structure

Name	Data Type	Description
<i>LROperation</i> +	element	Abstract element which is a placeholder for a longitudinal ribbon operation.

### Structure of LongitudinalRibbonOperationParams Elements

#### LROperation

[Deprecated in JDF 1.1.](#)

LROperation is an abstract element that describes the **LongitudinalRibbonOperation** process. The defined instances (subclasses) of LROperation are LongFold, LongGlue, LongPerforate, and LongSlit. All instances of LROperation have the following common contents.

Name	Data Type	Description
<i>WorkingList</i> = "0 1000000000"	NumberList	List of lengths of the <i>Operation</i> to be performed in point. Entries with an odd position (first, third, etc.) in the list define an offset where the tool is inactive. Entries with an even position define a working length where the tool is on. The start position is the leading edge of the plate.  If the sum of all entries is higher than the circumference of the press cylinder, the values exceeding the circumference are cropped. Counting always restarts at the leading edge. Default = "0 1000000000", (i.e., always on).
<i>XOffset</i>	double	Position of the tool for longitudinal action along the cylinder axis.

#### LongFold

[Deprecated in JDF 1.1.](#)

LongFold is derived from the abstract element LROperation and describes a longitudinal fold operation and has no further contents in addition to those of LROperation.

**LongGlue**[Deprecated in JDF 1.1.](#)

LongGlue is derived from the abstract element LROperation and describes a longitudinal gluing operation and has the following contents in addition to those of LROperation.

Name	Data Type	Description
<i>GlueBrand ?</i>	string	Glue brand. Use only when <i>Operation = Glue</i> .
<i>GlueType ?</i>	Enumeration	If <i>Operation = Glue</i> , the following values can be used: <i>ColdGlue</i> <i>Hotmelt</i> <i>PUR</i> – Polyurethane
<i>LineWidth ?</i>	double	Width of the <i>Operation</i> line.
<i>MeltingTemperature ?</i>	integer	Required temperature for melting the glue (in degrees centigrade). Use only when <i>GlueType = Hotmelt</i> and <i>Operation = Glue</i> .

**LongPerforate**[Deprecated in JDF 1.1.](#)

LongPerforate is derived from the abstract element LROperation and describes a longitudinal gluing operation and has the following contents in addition to those of LROperation.

Name	Data Type	Description
<i>TeethPerDimension ?</i>	integer	If <i>Operation = Perforate</i> , the number of teeth in a given perforation extent is defined in teeth/point. <i>MicroPerforation</i> is defined by specifying a large number of teeth (n>1000).

**LongSlit**[Deprecated in JDF 1.1.](#)

LongSlit is derived from the abstract element LROperation and describes a longitudinal cut operation and has no further contents in addition to those of LROperation.

**X.2.7 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:	<b>DigitalPrintingParams, IDPrintingParams, InsertSheet, Layout, Sheet, Tile</b>
Example Partition:	—
Input of processes:	<b>DigitalPrinting, IDPrinting</b>
Output of processes:	—

## Resource Structure

Name	Data Type	Description
<i>LeadingEdge</i> ?	number	Specifies the size, in points, of the edge of the media that represents the scanline direction. If this attribute is absent, the scanline direction is assumed to be along the x-axis of the <i>Dimension</i> parameter for the <b>Media</b> .
<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 3.8.2.6, Locations of Physical Resources) there should be a match between one <i>Location</i> partition key and this <i>MediaLocation</i> value.
<i>ManualFeed</i> = "false"	boolean	Indicates whether the media will be fed manually. Default = "false"
<i>SheetLay</i> ? <a href="#">New in JDF 1.1</a>	enumeration	Lay of input media. Reference edge of where paper is placed in feeder. Possible values are: <i>Left</i> <i>Right</i> <i>Center</i>
<b>Component</b> ? <a href="#">New in JDF 1.1</a>	refelement	A <b>Component</b> resource which identifies the preprinted media to be used. Only one of <b>Component</b> or <b>Media</b> should be specified.
<b>Media</b> ?	refelement	A <b>Media</b> resource which identifies the media to be used. Only one of <b>Component</b> or <b>Media</b> should be specified.

### X.2.8 PackingParams

[Deprecated in JDF 1.1](#)

The PackingParams resource has been deprecated in version 1.1 and beyond. It is replaced by the individual resources used by the processes defined in Section 6.6.48.4, Numbering and Section 6.6.48.5, 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:	<b>Packing</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>BoxedQuantity</i> ?	integer	How many units of <i>product</i> in a box.
<i>BoxShape</i> ?	shape	Describes the length, width and height of the box in points.
<i>CartonQuantity</i> ?	integer	How many units of <i>product</i> in a carton.
<i>CartonShape</i> ?	shape	Describes the length, width and height of the carton in points, (e.g., 288 544 1012).

Name	Data Type	Description
<i>CartonMaxWeight</i> ?	double	Maximum weight of an individual carton in kilograms.
<i>CartonStrength</i> ?	double	Strength of the carton in Newtons per square meter.
<i>PalletQuantity</i> ?	integer	Number of <i>product</i> per pallet
<i>PalletSize</i> ?	XYPair	Describes the length and width of the pallet in points, (e.g., 3500 3500).
<i>PalletMaxHeight</i> ?	double	Maximum height of a loaded pallet in points.
<i>PalletMaxWeight</i> ?	double	Maximum weight of a loaded pallet in kilograms.
<i>PalletType</i> ?	enumeration	Type of pallet used. Examples include: <i>2Way</i> – Two-way entry <i>4Way</i> – Four-way entry <i>Euro</i> – Standard 1*1 m Euro pallet
<i>PalletWrapping</i> = "None"	enumeration	Wrapping of the completed pallet. Examples include: <i>StretchWrap</i> <i>Banding</i> <i>None</i> – The default.
<i>WrappedQuantity</i> ?	integer	Number of units of <i>product</i> per wrapped package.
<i>WrappingMaterial</i> = "None"	name	Examples include: <i>RubberBand</i> <i>ShrinkWrap</i> <i>PaperBand</i> <i>Polyethylene</i> <i>None</i> – The default.

### X.2.9 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:	<b>FilmToPlateCopying</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>Cycle</i> ?	integer	Number of exposure light units to be used. The amount depends on the subject to be exposed.
<i>Diffusion</i> ?	enumeration	The diffusion foil setting. Possible values are: <i>On</i> <i>Off</i>
<i>Vacuum</i> ?	double	Amount of vacuum pressure to be used. Measured in bars.

### X.2.10 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

<b>Resource class:</b>	Parameter
<b>Resource referenced by:</b>	—
<b>Example Partition:</b>	<i>DocIndex, RunIndex, RunTags, SheetName, Side, SignatureName</i>
<b>Input of processes:</b>	<b>Proofing, SoftProofing</b>
<b>Output of processes:</b>	—

### Resource Structure

Name	Data Type	Description
<i>ColorType</i> ?	enumeration	Color quality of the proof. Possible values are: <i>Monochrome</i> – Black and white. <i>BasicColor</i> – Color does not match precisely. This implies the absence of a color matching system. <i>MatchedColor</i> – Color is matched to the output of the press using a color matching system.
<i>DisplayTraps</i> = “false”	boolean	If <i>true</i> , the trap networks are shown in the proof. Default = “false”
<i>HalfTone</i> = “false”	boolean	Specifies whether the proof should emulate halftone screens. Default = “false”
<i>ImageViewingStrategy</i> = “NoImages”	string	Identifies which images will be displayed during the <b>SoftProofing</b> process. Possible values are: <i>NoImages</i> – Default value. <i>OmitReference</i> – Displays only images actually embedded in the file. <i>UseProxies</i> – Displays images embedded in the file and proxy versions of referenced data. <i>UseReplacements</i> – Displays embedded images plus the full resolution version of referenced images.
<i>ManualFeed</i> = “false” <a href="#">New in JDF 1.1</a>	boolean	Indicates whether the media will be fed manually. Default = “false”
<i>ProofRenderingIntent</i> = “Perceptual” <a href="#">New in JDF 1.1</a>	enumeration	Identifies the rendering intents associated with the proof. Possible ICC-defined rendering intent values are: <i>Saturation</i> <i>Perceptual</i> – The default. <i>RelativeColorimetric</i> <i>AbsoluteColorimetric</i>
<i>ProofType</i> = “None”	enumeration	Describes the type of the proof. Possible values are: <i>None</i> – Default value. Not a proof or the type is unknown. <i>Page</i> – Page proof <i>Imposition</i> – Imposition proof.
<i>Resolution</i> ?	XYPair	Resolution of the output.
<i>FileSpec</i> ?	reference	A <b>FileSpec</b> resource pointing to an ICC profile that describes the proofer device. The <i>ResourceUsage</i> attribute of the <i>FileSpec</i> must be “ <i>ProoferProfile</i> ”.
<b>Media</b> ?	reference	Describes the media to be used.

### X.2.11 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:	<b>SaddleStitching</b>
Output of processes:	—

#### Resource Structure

Name	Data Type	Description
<i>NumberOfStitches</i>	integer	The number of stitches that will be made.
<i>StitchPositions ?</i>	NumberList	Array containing the stitch positions along the saddle. The center of the stitch must be specified, and the number of entries must match the number given in the <i>NumberOfStitches</i> attribute.
<i>StapleShape ?</i>	enumeration	Shape of staples. Possible values are: <i>Crown</i> <i>Overlap</i> <i>Butted</i> <i>ClinchOut</i> <i>Eyelet</i> These values are displayed in Figure X.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.

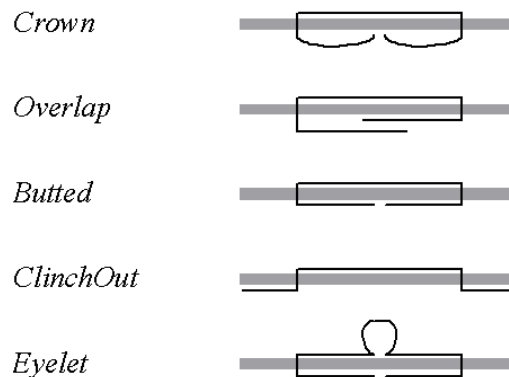


Figure X.2: Staple shapes

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.

## X.2.12 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).

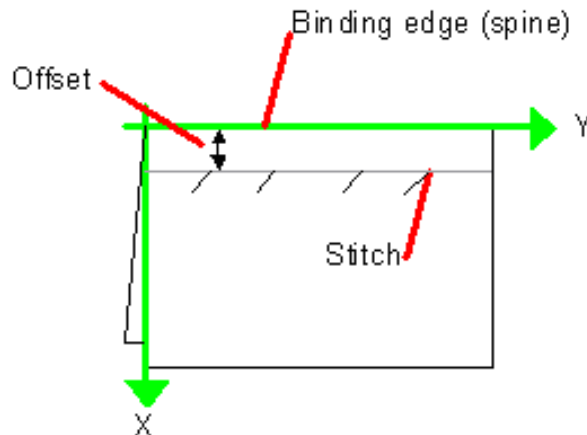


Figure X.3: Parameters and coordinate system used for side sewing

### Resource Properties

Resource class:	Parameter
Resource referenced by:	—
Example Partition:	—
Input of processes:	<b>SideSewing</b>
Output of processes:	—

### Resource Structure

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. Possible values are: <i>Normal</i> <i>Staggered</i> <i>CombinedStaggered</i>

Name	Data Type	Description
<i>ThreadMaterial</i> ?	enumeration	Specifies the thread material to be used. Possible values are: <i>Cotton</i> <i>Nylon</i> <i>Polyester</i>
<i>ThreadThickness</i> ?	double	The thickness of the thread to be used.
<i>ThreadBrand</i> ?	string	The brand of thread to be used.

## X.3 JMF Messaging Elements

### X.3.1 KnownJDFServices

[Deprecated in JDF 1.2](#)

In JDF 1.2 and beyond, KnownJDFServices has been replaced with KnownDevices and DeviceDetails = "Capabilities".

Table X-1: Contents of the KnownJDFServices message

Object Type	Element name	Description
QueryTypeObj	—	—
ResponseTypeObj	JDFService *	Processes that the controller or device can execute.

The KnownJDFServices query returns a list of services that are defined in the JDF specification, such as **ConventionalPrinting**, **RIPing**, or **EndSheetGluing**. It allows a controller to publish the services that the devices it controls are capable of providing. The response is a list of JDFService elements, one for each supported process type.

#### JDFService

JDFService elements define the node types that can be processed by the controller. A JDF processor should be capable of processing *Combined* nodes of any of the individual JDFService elements that are specified. It is therefore not necessary to define every permutation of allowed combinations. It need not be able to process individual nodes with a type defined in the *Types* attribute of a *Combined* JDFService element.

Table X-2: Contents of the JDFService element

Name	Data Type	Description
<i>CombinedMethod</i> ? <a href="#">New in JDF 1.1</a>	enumeration	Specifies how the processes specified in <i>Types</i> may be specified. One of: <i>Combined</i> – The list of processes in <i>Types</i> must be specified as a <i>Combined</i> process. <i>ProcessGroup</i> – The list of processes in <i>Types</i> must be specified as a <i>ProcessGroup</i> of individual processes. <i>CombinedProcessGroup</i> – The list of processes in <i>Types</i> may be specified either as a <i>Combined</i> process or as a <i>ProcessGroup</i> of individual processes. <i>None</i> – No support for <i>Combined</i> or <i>ProcessGroup</i> . Only the individual process type defined in <i>Types</i> is supported. The default.
<i>Type</i>	NMTOKEN	JDF <i>Type</i> attribute of the supported process. Extension types may be specified by stating the namespace in the value.



Table X-2: Contents of the JDFService element

<a href="#">TypeOrder ?</a> <a href="#">New in JDF 1.1</a>	enumeration	Ordering restriction for combined nodes. <i>Fixed</i> – The order of process types specified in the <i>Types</i> attribute is ordered and each type can be specified only once, e.g., Cutting, Folding; order does matter. The default. <i>Unordered</i> – The order of process types specified in the <i>Types</i> attribute is unordered and each type can be specified only once, e.g., DigitalPrinting, Screening, Trapping; order does not matter. <i>Unrestricted</i> – The order of process types specified in the <i>Types</i> attribute is unordered and each type can be specified multiply, e.g., Cutting, Folding, where the device can do both processes, in any order and multiple times.
<a href="#">Types ?</a>	NMTOKENS	If <i>Type</i> = <i>Combined</i> , or <i>Type</i> = <i>ProcessGroup</i> this attribute represents the list of combined processes. If any of the services are in a namespace other than JDF, the namespace prefix should be included in this list. For details, see Section 3.1.5, Combined Process Nodes.

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

```
<Response ID="M1" refID="Q1" Type="KnownJDFServices">
  <JDFService Type="Rendering" />
  <JDFService Type="Folding" />
  <JDFService Type="Combined" Types="Gathering Stitching"/>
  <JDFService Type="AnyCompaniesNamespace:MyFolding" />
  ...
</Response>
```

### X.3.2 QueueEntryStatus

[Deprecated in JDF 1.2](#)

In JDF 1.2 and beyond, use QueueStatus with an appropriate QueueFilter instead of QueueEntryStatus.

Table X-3: Contents of the QueueEntryStatus message

Object Type	Element name	Description
<a href="#">QueryTypeObj</a> <a href="#">Modified in JDF 1.1A</a>	QueueEntryDefList	Defines the addressed queue entries. Note that this element was QueueEntryDef * prior to JDF1.1A.
<a href="#">ResponseTypeObj</a>	QueueEntry *	Describes the status of the queried queue entries.
For the definition of the elements above see Section 5.6.5, Queue-Handling Elements.		

The QueueEntryStatus message returns queue entry descriptions. The QueueEntryDef elements specify the queue entries to be queried. If no QueueEntryDef element is specified, the query returns a list of QueueEntry elements, one for each entry in the queue. If no QueueEntryDef is specified and the query defines a persistent channel, a Signal is emitted for any entry whose status changes. This includes changes as a result of modifications of the queue status, such as hold or resume.

#### Structure of the QueueEntryDefList Element

[New in JDF 1.1A](#)

[Deprecated in JDF 1.2](#)

The QueryTypeObj of QueueEntryStatus has been modified from QueueEntryDef \* to QueueEntryDefList because of a type collision in the XML Schema. QueueEntryDef had been used both as a QueryTypeObj and as a CommandTypeObj.

*Table X-4: Contents of the QueueEntryDefList element*

Name	Data Type	Description
QueueEntryDef *	element	Defines the addressed queue entries.

## Appendix Y Table of Tables

Table 1-1 Basic References .....	2
Table 1-2 Conformance Terminology .....	7
Table 1-3 JDF data types .....	9
Table 1-4 Units used in JDF .....	12
Table 2-1 Information contained in JDF nodes, arranged numerically .....	18
Table 2-2 Information contained in JDF nodes, arranged by group .....	18
Table 2-3 Matrices and Orientation values used to describe the orientation of a Component ...	24
Table 3-1 Generic Contents of elements .....	35
Table 3-2 Contents of the Comment element .....	36
Table 3-3 Definition of “Scope” Terms used in Table 3-4 on page 38 .....	37
Table 3-4 Contents of a JDF node .....	38
Table 3-5 Contents of the AncestorPool element .....	48
Table 3-6 Attributes of the Ancestor element .....	48
Table 3-7 Contents of the CustomerInfo element .....	49
Table 3-8 Contents of the CustomerMessage element .....	50
Table 3-9 Contents of the NodeInfo element .....	51
Table 3-10 Contents of the StatusPool element .....	52
Table 3-11 Contents of the PartStatus element .....	53
Table 3-12 Contents of the ResourcePool element .....	53
Table 3-13 Contents of the abstract Resource element .....	53
Table 3-14 Additional contents of the abstract parameter Resource elements .....	57
Table 3-15 Additional contents of the abstract physical Resource elements .....	58
Table 3-16 Contents of the Location element .....	58
Table 3-17 Contents of the abstract ResourceUpdate Element .....	60
Table 3-18 Contents of the ResourceLinkPool element .....	63
Table 3-19 Contents of the abstract ResourceLink element .....	64
Table 3-20 Contents of the AmountPool element .....	65
Table 3-21 General contents of the PartAmount element .....	66
Table 3-22 Contents of the abstract ImplementationLink or PartAmount element .....	66
Table 3-23 Additional contents of the abstract physical ResourceLink and PartAmount element ..	67
Table 3-24 Contents of the abstract ResourceRef element .....	68
Table 3-25 Contents of the abstract ResourceElement .....	69
Table 3-26 Example of Actual Amount and Amount Handling .....	73
Table 3-27 Contents of the Partitionable Resource Element .....	78
Table 3-28 Contents of the Part element .....	79
Table 3-29 PartUsage example uses .....	86
Table 3-30 Contents of the AuditPool element .....	89
Table 3-31 Contents of the abstract Audit type .....	91
Table 3-32 Contents of the ProcessRun element .....	91
Table 3-33 Contents of the Notification element .....	92
Table 3-34 Contents of the PhaseTime element .....	93
Table 3-35 Contents of the ModulePhase element .....	94
Table 3-36 Contents of the ResourceAudit element .....	95
Table 3-37 Contents of the Created element .....	96
Table 3-38 Contents of the Deleted Element .....	97
Table 3-39 Contents of the Modified element .....	97
Table 3-40 Contents of the Spawned element .....	97

Table 3-41 Contents of the Merged element.....	98
Table 3-42 Example from TrappingParams .....	100
Table 4-1 Business Objects as defined by PrintTalk .....	106
Table 4-2 Examples of resource and process states in the case of simple process routing.....	112
Table 4-3 Actions generated when a dynamic-pipe buffer passes various levels .....	116
Table 5-1 Contents of the JMF element.....	130
Table 5-2 Contents of the abstract Message element.....	130
Table 5-3 Contents of the Query message element.....	132
Table 5-4 Contents of the Response message element .....	133
Table 5-5 Contents of the Signal message element .....	134
Table 5-6 Table 5-6 Contents of the Trigger element.....	134
Table 5-7 Contents of the ChangedPath element.....	135
Table 5-8 Contents of the ChangedAttribute element.....	135
Table 5-9 Contents of the Added element.....	136
Table 5-10 Contents of the Removed element.....	136
Table 5-11 Contents of the Command message element .....	136
Table 5-12 Contents of the Acknowledge message element.....	138
Table 5-13 Contents of the Subscription element .....	139
Table 5-14 Contents of the ObservationTarget element.....	139
Table 5-15 Messaging table template .....	141
Table 5-16 Process registration and communication messages.....	142
Table 5-17 Contents of the Events message .....	142
Table 5-18 Contents of the NotificationFilter element .....	143
Table 5-19 Contents of the NotificationDef element.....	144
Table 5-20 Contents of the KnownControllers message.....	144
Table 5-21 Contents of the JDFController element.....	144
Table 5-22 Contents of the KnownDevices message .....	145
Table 5-23 Contents of the DeviceFilter element.....	145
Table 5-24 Contents of the DeviceList element .....	146
Table 5-25 Contents of the KnownMessages message.....	146
Table 5-26 Contents of the KnownMsgQuParams element.....	146
Table 5-27 Contents of the MessageService element .....	146
Table 5-28 Contents of the RepeatMessages message .....	147
Table 5-29 Contents of the MsgFilter element .....	147
Table 5-30 Contents of the StopPersistentChannel message.....	148
Table 5-31 Contents of the StopPersChParams element .....	148
Table 5-32 Status and progress messages.....	149
Table 5.33 Contents of the FlushResources message .....	150
Table 5.34 Contents of the FlushResourceParams element.....	150
Table 5-35 Contents of the NewJDF query message .....	150
Table 5-36 Contents of the NewJDFQuParams element.....	151
Table 5-37 Contents of the NewJDF Command Message.....	151
Table 5-38 Contents of the NewJDFCmdParams element .....	151
Table 5-39 Contents of the IDInfo element .....	151
Table 5-40 Contents of the NodeInfo query message.....	152
Table 5-41 Contents of the NodeInfoQuParams element .....	152
Table 5-42 Contents of the NodeInfo Command Message.....	152
Table 5-43 Contents of the NodeInfoCmdParams element .....	153
Table 5-44 Contents of the NodeInfoResp element.....	153
Table 5-45 Contents of the Occupation message .....	154

---

Table 5-46 Contents of the EmployeeDef element .....	154
Table 5-47 Contents of the Occupation element.....	154
Table 5-48 Contents of the Resource query message.....	155
Table 5-49 Contents of the ResourceQuParams element .....	155
Table 5-50 Contents of the Resource command message .....	156
Table 5-51 Contents of the ResourceCmdParams element.....	157
Table 5-52 Contents of the ResourceInfo element.....	158
Table 5.53 Contents of the ResourcePull message .....	160
Table 5.54 Contents of the ResourcePullParams element .....	160
Table 5.55 Contents of the ShutDown message.....	162
Table 5.56 Contents of the ShutdownCmdParams element .....	162
Table 5-57 Contents of the Status message .....	162
Table 5-58 Contents of the StatusQuParams element.....	163
Table 5-59 Contents of the DeviceInfo element .....	164
Table 5-60 Contents of the JobPhase element .....	165
Table 5-61 Contents of the ModuleStatus element .....	167
Table 5-62 Contents of the Track message .....	168
Table 5-63 Contents of the TrackFilter element .....	168
Table 5-64 Contents of the TrackResult element.....	169
Table 5.65 Contents of the WakeUp message.....	169
Table 5.66 Contents of the WakeUpCmdParams element.....	169
Table 5-67 Dynamic pipe messages .....	170
Table 5-68 Contents of the PipeClose message.....	170
Table 5-69 Contents of the PipePull message .....	170
Table 5-70 Contents of the PipeParams element .....	171
Table 5-71 Contents of the PipePush message.....	172
Table 5-72 Contents of the PipePause message.....	173
Table 5-73 QueueEntry handling messages .....	174
Table 5-74 Status transitions for QueueEntry handling messages .....	174
Table 5-75 Contents of the AbortQueueEntry message .....	176
Table 5-76 Contents of the HoldQueueEntry message.....	176
Table 5-77 Contents of the RemoveQueueEntry message.....	177
Table 5-78 Contents of the RequestQueueEntry message.....	177
Table 5-79 Contents of the ResubmitQueueEntry message.....	178
Table 5-80 Contents of the ResubmissionParams element.....	178
Table 5-81 Contents of the ResumeQueueEntry message.....	178
Table 5-82 Contents of the ReturnQueueEntry message .....	178
Table 5-83 Contents of the ReturnQueueEntryParams element.....	179
Table 5-84 Contents of the SetQueueEntry message.....	179
Table 5-85 Contents of the QueueEntryPosParams element .....	179
Table 5-86 Contents of the SetQueueEntryPriority message .....	180
Table 5-87 Contents of the QueueEntryPriParams element.....	180
Table 5-88 Contents of the SubmitQueueEntry message.....	180
Table 5-89 Contents of the QueueSubmissionParams element .....	181
Table 5-90 Contents of the SuspendQueueEntry message.....	182
Table 5-91 Global queue-handling commands .....	182
Table 5-92 Definition of the Queue Status Attribute values.....	183
Table 5-93 Contents of the CloseQueue message .....	183
Table 5-94 Contents of the FlushQueue Command message.....	184
Table 5-95 Contents of the FlushQueue Query message.....	184

---

Table 5-96 Contents of the HoldQueue message .....	185
Table 5-97 Contents of the OpenQueue message .....	185
Table 5-98 Contents of the QueueStatus message .....	185
Table 5-99 Contents of the ResumeQueue message .....	186
Table 5-100 Contents of the SubmissionMethods message .....	186
Table 5-101 Contents of the SubmissionMethods element.....	186
Table 5-102 Contents of the Queue element .....	187
Table 5-103 Contents of the QueueEntry element.....	188
Table 5-104 Contents of the QueueEntryDef element .....	189
Table 5-105 Contents of the QueueFilter Element .....	189
Table 7-1 ChannelTypeDetails predefined values for different ChannelType values .....	322
Table 7-2 Terms and definitions for components .....	324
Table 7-3 Cut Mark Types .....	335
Table 7-4 Predefined variables used in FileTemplate .....	364
Table 7-5 Parameters in Stacking .....	473
Table 7-6 Example 1 of Ord in PlacedObjects .....	486
Table 7-7 Example 2 of Ord in PlacedObjects .....	486
Table 7-8 Document Object Classes.....	539
Table 7-9 Properties Implemented by Class .....	539
Table 7-10 Mapping between property types (in the preflight spec) and Evaluations .....	540
Table 7-11 Allowed Box Types .....	542
Table A-1 JDFJMFVersion enumeration Values .....	575
Table A-2 Named Colors.....	576
Table A-3 Page Orientation.....	576
Table A-4 Side enumeration Values.....	577
Table A-5 WorkStyle enumeration Values .....	577
Table A-6 XYRelation enumeration Values.....	577
Table D-1 Conversion of PPF Data Types .....	598
Table D-2 JDF Representation of a product definition step .....	598
Table D-3 Converting a PPF Component.....	599
Table D-4 Converting the PPF EndSheetGluing operation to JDF .....	600
Table D-5 Converting the PPF AdhesiveBinding operation to JDF .....	600
Table D-6 Converting the PPF AdhesiveBinding suboperation Lining .....	601
Table D-7 Converting the PPF AdhesiveBinding suboperation CoverApplication .....	601
Table D-8 Converting the PPF GluingIn operation to JDF .....	601
Table D-9 Converting the PPF Folding operation to JDF.....	602
Table D-10 Converting the PPF Folding suboperation of type Fold.....	602
Table D-11 Converting the PPF Folding suboperation of type Lime .....	603
Table D-12 Converting the PPF Folding suboperation of all other types .....	603
Table D-13 Converting administration data .....	605
Table D-14 PPF preview representation as PNG.....	606
Table D-15 Converting the parameter of the CIP3PlaceRegisterMark command.....	607
Table D-16 Converting PPF color-measuring data .....	608
Table D-17 Converting PPF density-measuring data.....	608
Table D-18 Converting the parameter of the CIP3PlaceColorControlStrip command.....	608
Table D-19 Converting the Cutting Data structure .....	609
Table D-20 Converting the parameter of the CIP3PlaceCutMark command .....	609
Table E-1 IFRA object states .....	611
Table G-1 StatusDetails and Status mapping for generic devices .....	615
Table G-2 Printing Device specific StatusDetails .....	615

---

Table G-3 PostPress Device specific StatusDetails .....	616
Table H-1 ModuleType definition for conventional printing devices .....	617
Table H-2 ModuleType definition Gathering / Collecting .....	617
Table H-3 ModuleType definition for DigitalPrinting .....	617
Table I-1 Return codes for JMF.....	619
Table J-1 Contents of the Barcode element.....	621
Table J-2 Contents of the FCNKey element.....	621
Table J-3 Contents of the SystemTimeSet element.....	621
Table J-4 Contents of the CounterReset element.....	621
Table J-5 Contents of the Error element, derived from NotificationDetails .....	622
Table J-6 Contents of the Event element, derived from NotificationDetails .....	622
Table O-1 Locations within Printers .....	633
Table P-1 MimeType (MIME Media Types Registered with IANA), MimeTypeVersion combinations 635	
Table P-2 MimeType (File Type) and MimeTypeVersion combinations .....	638
Table Q-1 Use Cases showing MimeType, URL, and Compression attribute values .....	641
Table S-1 AppOS and OSVersion Examples .....	651
Table T-1 Complete References .....	653
Table X-1 Contents of the KnownJDFServices message .....	772
Table X-2 Contents of the JDFService element.....	772
Table X-3 Contents of the QueueEntryStatus message.....	773
Table X-4 Contents of the QueueEntryDefList element.....	774





## Appendix Z Terminology Usage

This document contains many terms specific to its interpretation and intent. Many of the terms are described in relation to various processes, components, and values throughout the document. The more prominent terms are listed below to make it easier for the casual user to locate precise definitions and usage

Table Z.1: Terminology Usage

Term	Term Type	Glossary of Terminology (Sect. 1.4)	Data Structures (Sect. 1.5)	Job Components (Sect. 2.1.1)	Workflow Components (Sect. 2.1.2)	Relationships (Sect. 2.1.1.4)	Other
Acknowledge	message						Section 5.2.1.5
Activation	enumeration						Table 3.3, Table 5.38
Agent(s)	consumer	X			Section 2.1.2.3		
Ancestor	element					X	
AncestorPool	element						Sect. 3.3 Table 3.4
Attribute(s)	attribute	X		Section 2.1.1.3			Sect. 3.1.2
AuditPool	elements						Sect. 3.10
Big job		X					
boolean	data type		X				Table A.1
Branch	node					X	
Child	element					X	
Class	data type	X					
CMYK color	data type		X				A.2.1
Command	message						Section 5.2.1.4
Controllers	consumer	X			Section 2.1.2.4		
Coordinate systems							Section 2.5
Customer	node						Section 3.4
Date	data type		X				Table A.1
DateTime	data type		X				Table A.1
Default	value	X					Sect. 1.4.2.1
Deprecated		X					
Descendant	element					X	
Devices	consumer	X			Section 2.1.2.2		

Table Z.1: Terminology Usage

Term	Term Type	Glossary of Terminology (Sect. 1.4)	Data Structures (Sect. 1.5)	Job Components (Sect. 2.1.1)	Workflow Components (Sect. 2.1.2)	Relationships (Sect. 2.1.1.4)	Other
Document set		X					
Double	data type		X				Table A.1
DoubleList	data type		X				A.2.11
DoubleRange	data type		X				A.2.12
DoubleRangeList	data type		X				A.2.13
Duration	data type		X				Table A.1
DurationRange	data type		X				A.2.2
Element(s)	job component	X	X	Section 2.1.1.2			
<i>Enumeration(s)</i>	data type		X				
Finished page	job component	X					
gYearMonth	data type		X				Table A.1
ID/IDREF(s)			X				Table A.1
IfraTrack modeling							App. E
Instance document	job component	X					
Integer	data type		X				Table A.1
IntegerList	data type		X				A.2.3
IntegerRange	data type		X				A.2.4
IntegerRangeList	data type		X				A.2.5
intent resources							3.2.1, 7.1.1.1
IPP mapping							App. F
iterative processing							2.3
JDF consumer		X					
JMF		X					Chapt. 5

Table Z.1: Terminology Usage

Term	Term Type	Glossary of Terminology (Sect. 1.4)	Data Structures (Sect. 1.5)	Job Components (Sect. 2.1.1)	Workflow Components (Sect. 2.1.2)	Relationships (Sect. 2.1.1.4)	Other
Job(s)	job component	X		Section 2.1.1.1			
Job part	node	X					
LabColor	data type		X				A.2.6
Language	data type		X				Table A.1
Leaf	element					X	
Links	job components	X		Section 2.1.1.5			A.3.1
Machines	job components	X			Section 2.1.2.1		
Matrix	data type		X				A.2.7
<b>Merging</b>	<b>process</b>						Section 4.4
MIME File Packaging							A.4.1
MIS		X			Sesction 2.1.2.5		
NamedColor	data type		X				A.2.8
NameRange	data type		X				A.2.9
NameRangeList	data type		X				A.2.10
NMTO-KEN(S)	data type		X				Table A.1
Node(s)	element	X		Section 2.1.1.1			Table 3.3
Parent	element					X	
Partitioned resource	<b>resource</b>	X					
Path	data type		X				A.2.14
PDL		X					
PJTF conversion							App. C
PNG format							A.4.3
PPF conversion							App. D
Process	consumer	X					
Process nodes							Section 3.2 Chapter 6
Product intent nodes	node						Section 3.2.1

Table Z.1: Terminology Usage

Term	Term Type	Glossary of Terminology (Sect. 1.4)	Data Structures (Sect. 1.5)	Job Components (Sect. 2.1.1)	Workflow Components (Sect. 2.1.2)	Relationships (Sect. 2.1.1.4)	Other
Query	message						Section 5.2.1.1
Queue	consumer	X					
Reader page	value	X					
Rectangle	data type		X				A.2.15
Refelement	data type		X				
Relationships	job components			Section 2.1.1.4			
Resource(s)	job component	X					
Response	message						Section 5.2.1.2
Root	element					X	
Shape	data type		X				
ShapeRange	data type		X				A.2.16
ShapeRangeList	data type		X				A.2.17
Sibling	element					X	
Signal	message						Section 5.2.1.3
Small job		X					
<b>Spawning process</b>							Section 4.4
sRGBcolor	data type		X				A.2.18
String	data type		X				Table A.1
Support	value	X					
System interaction	job components				Section 2.1.2.6		
Tag	value	X					
Telem	data type		X				
Text	data type		X				
TimeRange	data type		X				A.2.19
Transfer-Function	data type		X				A.2.20
URI	data type		X				Table A.1
URL	data type		X				Table A.1
Work center		X					
Workflow components	job components				Section 2.1.2		

Table Z.1: Terminology Usage

Term	Term Type	Glossary of Terminology (Sect. 1.4)	Data Structures (Sect. 1.5)	Job Components (Sect. 2.1.1)	Workflow Components (Sect. 2.1.2)	Relationships (Sect. 2.1.1.4)	Other
XYPair	data type		X				A.2.21
XYPair-Range	data type		X				A.2.22
XYPair/RangeList	data type		X				A.2.23