Technical guide

PRISMAdirect

Technical reference manual

océ

A CANON COMPANY

Canon

Production Systems - Cutsheet

Edition 2017-11

Software version All versions

GB

Contents

Chapter 1
Introduction

This manual

Release information

This manual is valid for All versions of PRISMAdirect.

For whom is this Technical guide intended?

The PRISMAdirect Technical reference manual (TRM) is not the user manual of PRISMAdirect.

The TRM provides the information about the behaviour, configuration and programming of the PRISMAdirect.

The target group for the TRM are the Administrators and Application Developers of PRISMAdirect.

Abbreviations

- TRM: Technical Reference Manual

Copyright and Trademarks

Copyright

Language

Original instructions that are in British English.

Trademarks

Additional information

[WROX] – “Professional XML”; Wrox Press, August 2000; ISBN 1-861003-11-0.

[XMLSDK] – “Microsoft XML 3.0 - XML Developer’s Guide”; part of MS XML SDK 3.0; Microsoft Corp.; 2000.

[MSDN] – http://msdn.microsoft.com/xml

[JDF1.3] – “JDF Specification”; CIP 4; version 1.3.

Downloads

http://downloads.oce.com/ProductDownloads/Index/542 on page

Chapter 2
PRISMAdirect

Introduction

Contents

This document describes the PRISMAdirect Open Interface v1.0 regarding Orders, Jobs and Definitions.

Chapter 2 describes the position of the Open Interface in the system.

Chapter 3 describes the import and export functionality.

Chapter 4 describes the media attributes.

Chapter 5 describes the XML format used in the Open Interface.

Chapter 6 presents the functionality offered by the JDF/JMF Interface and describes the tables used for JDF to PRISMAdirect ticket and job status mapping.

Concepts

To understand this document a number of concepts must be understood:

- **Hot Folder**: Directory on a file-system that is monitored by a dedicated application. The application will respond to changes in that directory.
- **NT Service**: a program that is running continuosly in the background, even if no user is logged in.
- **GUID & UUID**: Global Unique Identifier resp. Universal Unique Identifier. Large number that is guarantied unique.
- **Order processing workspace**: a part (workspace) of the PRISMAdirect web application that allows order managers and operators to manage orders and their jobs and perform print-related tasks.
- **Product & order editor**: a part (workspace) of the PRISMAdirect web application allowing the management of the Current Definition (see below).
- **JobNumber**: shown in the Order processing workspace > Order and Job views as the number of the job. Note that this number is not unique.
- **OrderNumber**: shown in the Order processing workspace > Order view as the number of the job. Note that this number is not unique.
- **JobID**: identifer of a Job. A GUID is used to guarantee that this number is always unique.
- **OrderID**: identifer of an Order. A GUID is used to guarantee that this number is always unique.
- **DefinitionID**: identifier of a Definition. A GUID is used to guarantee that this number is always unique.
- **Job**: is a combination of one or more documents (PDF file(s)) and a set of Job attributes. The set of attributes is collectively called the jobticket. The jobticket must be interpreted as the print job description containing print/production information related to that Job (e.g. copies, binding) as specified by the creator (end-user).
- **Order**: is a combination of one or more Jobs and a set of Order attributes. The set of attributes is collectively called the orderticket. The orderticket must be interpreted as order description containing specific details (e.g. owner, delivery information) as declared by the creator (end-user).
- **Product**: this is the blueprint of a Job, a collection of job type, Item arrangement in Views, default values and value constraints. It is defined in Product & order editor.
- **Item**: an attribute belonging to a Job or an Order, e.g. a field in a jobticket. An item has some related concepts in the context of the PRISMAdirect XML Schema. These are explained below.
- **ItemDefinition**: defines the datatype, default value and possible values of an item. In other words, an ItemDefinition is a type specifier for an item.The set of ItemDefinition elements is part of the Definition element.
- **ItemValue**: the instance of an item in a job. It represents the actual value of an item in the context of that job.
- **ItemViewLeaf & ItemViewGroup**: represent the position of an item in a graphical tree-like presentation of a job ticket. An ItemViewGroup represents an item in the "tree" which only purpose is to group several other items. Whereas an ItemViewLeaf represents a leaf of the "tree". ItemViewLeaf and ItemViewGroup elements are part of a View-element.
- **View**: defines the presentation of the job or order. Part of a Definition element. It contains the set of ItemViewLeaf and ItemViewGroup elements used.
- **Definition**: Structured description of the jobticket/orderticket used in the system and views on the ticket. It contains the "declaration" of all ItemDefinition elements that belong to a Job or Order and View declarations. Attachement elements describe files that belong to a Job (source documents).
- **Current Definition**: identifies the Definition that is "current", i.e. the Definition that is used to create Orders and their contained Jobs and to present them in the Order processing workspace. The Current Definition may be set to a different Definition over time. This may lead to a mismatch between new Jobs and Jobs that were created using an earlier

- **Media**: “a media is a physical substrate that can be processed by a printing system. It is something you can hold or lay down somewhere.”
- **Note**: This means a media is a sheet or a roll of a certain material and shape and should not be confused with just the size of a sheet, a side of a sheet or the size of an image to be printed.
- **Media attributes**: Main attributes that define a media. These are the attributes that together uniquely identify a media inside the system and allow a user to select that media instead of another one. Examples: media size, media type, etc
- **Media Catalog**: A media catalog is an ordered administration of media descriptions to be used in an Océ product. The order of the media in the catalog may be used by these Océ products as a default ranking mechanism when multiple media from the catalog match a media request.

This document assumes that the reader is familiar with XML.

PRISMAdirect context

Functionality overview

The Open Interface of PRISMAdirect provides functionality for:

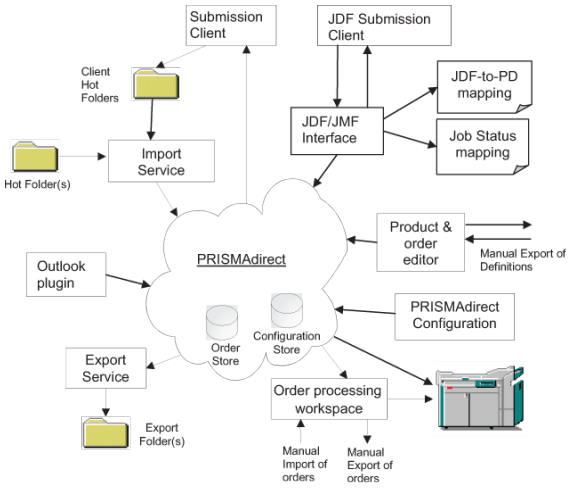
- Importing orders or jobs into the system and exporting orders from the system
- Submitting jobs from a JDF enabled external application and provide status feedback

With PRISMAdirect, jobs cannot exist without being associated to an order, and an order can be seen as a container having at least one or more jobs, a container defining common things like delivery or owner information, etc.

However, depending on the situation, we can speak of order (together with contained jobs) import or job import. Whenever job import is in discussion (e.g. JDF submission), a single job order is actually created with either a default orderticket or one created based on import mappings.

Export on the other hand will always export orders together with their jobs (tickets plus attachments) in a packet ZIP file.

Figure 1 shows how this functionality is embedded in the overall PRISMAdirect system. The base of the system is the PRISMAdirect, which contains orders and a configuration store. On top of this core, all applications are built.



Import & Export

This functionality is available in two modes: manual and automatic.

Office users submit orders/jobs by means of a Submission Client. These orders are loaded into the order store by the Import Service. The Order processing workspace provides the Print room 5 Order managers/Operators with a list of waiting orders/jobs and various handling possibilities. These include printing and manual export of the order. Also, from the Order processing workspace, orders can be imported manually.

Office users can also use the WebDriver/WebShop to submit orders. Because the WebShop has no relation with the Open Interface it is not discussed any further here.

The Product & order editor is used to update or create a new Current Definition used in the system. This includes creation and deletion of fields ('items') and adding or removing them in the client's or operator's view (a.k.a. Web shop or Print room views). The complete description of the jobticket and orderticket fields, and the views on them, is called a Definition. The Definition can be exported manually with the Product & order editor. Exported Definitions can be imported by any PRISMAdirect system with the same or newer version.

As can be seen in Figure 1, the Import Service imports orders or jobs from 'Hot Folders' into the system. This is called automatic import. The Export Service can perform an automatic export of orders and Definitions to files in the export folders.

The last system component shown in Figure 1 is the PRISMAdirect Configuration application. This application is used to configure the import & export services, edit UPP job ticket mappings, etc.

JDF/JMF Interface

The functionality permits external applications to submit JDF jobs to PRISMAdirect and get status feedback for the submmitted jobs. This functionality is partially supported by the JDF Hot Folder in the Import Service – without job status feedback and with a much smaller JDF support of the PD ticket.

NOTE
It is recommended that new applications use the new JDF/JMF interface instead of the old Hot Folder mechanism. The Hot Folder will be supported for a limited time and will become deprecated.
The JDF job contains only the JDF ticket and the Data Files. As Figure 1 shows the system maps the JDF ticket to the ODW ticket by the use of a JDF-to-PD mapping file. The job status exposed by PRISMAdirect can also be mapped to values specific for the external application. This is done via another mapping file. Both mapping files can be changed by the user to fit the external application's specifics.

Chapter 3

Import / export services

Import

Manual

Orders

Order processing

workspace Orders can be imported manually. Currently, only orders exported from systems with the same PRISMAdirect version are supported. When a directory is selected the application will show all names of ZIP files in the directory.

Files The ZIP file has a well defined structure and strict file names. It may contain one or more folders, each one containing an order together with all its related content. Such a (order) folder contains the following:

- An orderticket file (XML file) and a JDF ticket file (also XML), together defining order metadata
- For each job inside the order:
  - A jobticket file (XML file) and if exported a JDF ticket file (also XML) together defining job metadata
  - The job attachments and current/original merged PDF if available at export time. In previous releases these where PDF files, but as of ODW 3.1 (Open Interface version 1.4) this can be any type of file. However the accepted mimetype per attachment has limitations

File contents The XML files not only contain the user's intent, but also reference(s) to the related document files. If the definition used to create the imported Order or Job is different than the Current definition (different set of items), the Order/Job it will be updated to the Current definition with the remark that old ItemValue fields are not deleted.

The order and its jobs must be compliant with the rules specified in the Definition it was last been edited. If not, the import action will fail.

None of the imported files will be changed.

If an order is already present in the system, it cannot be imported again. The present order has to be deleted first. Note that orders are not identified by their OrderNumber, as shown in the Order processing workspace, but through their QEntryId. The QEntryId can be found in the XML file and is the only unique reference of an order.

Definitions

Product & order editor In the Product & order editor Definitions can be imported manualy. When a directory is selected, the application will show all names of ZIP files in the directory.

The ZIP file should contain a single Definition with format. The ZIP will also contain other files describing the Product informations, WebShops defined in the system and their settings, etc, which are not in the scope of this document.

At import the Definition is read into the Product & order editor, but not saved yet into the system. The user has to publish after importing. When the Definition is published it is also made current.

The Definition must contain a set of obliged items. Of these items, some of the properties must have the default value.

Automatic

The Import service has a “Pause” option. This will stop watching configured Hot folders. If it is on, clients can still submit jobs to the Hot folders but they will not be picked up until “Pause” is disabled.

A Hot folder can be configured to work in one of five modes. These modes are described in the following paragraphs.

In each case, a web shop must be also specified as in PRISMAdirect all orders must belong to a web shop, thus, any orders created via automatic import must be associated to a web shop.

Possible types of Import

Normal orders: this is the type of hot folder that accepts jobs exported from the current version of PRISMAdirect.

NOTE
Depending on the type of export selected, besides the PRISMAdirect ticket file and the corresponding attachments, there could be a JDF ticket and additional PDF files also. If preserve history is enabled, the JDF ticket is imported as is, but if the preserve history is disabled, the JDF ticket is not imported and is created from scratch (losing history). The JDF ticket is however not part of the Open Interface document.

JDF ticket orders: if a Hot folder is declared this way, it will accept JDF job tickets. For each JDF job ticket found in the folder, the import service will attempt to create a Job and an Order in the system using information from the JDF ticket. Based on a configurable mapping table the system transfers settings from the JDF ticket to the PRISMAdirect jobticket and orderticket. PRISMAdirect ships three default mapping tables that can be chosen at user’s convenience (of course, those can be customized and chosen to be used for import):

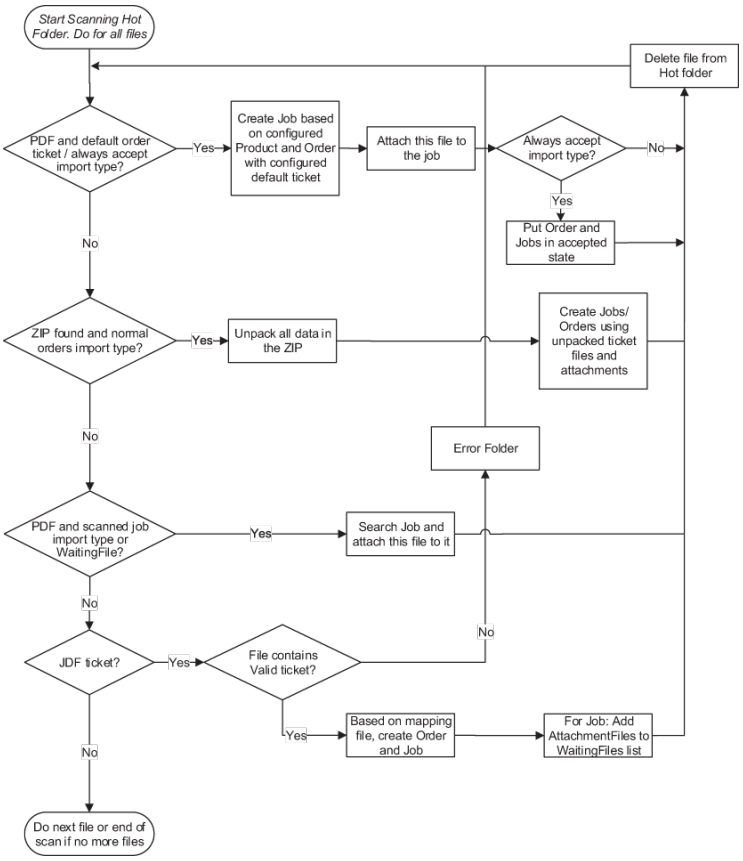
- Standard JDF to PD
- DSF (a customized JDF) to PD
- uniFLOW (slightly modified JDF) to PD

PDF only job with default order ticket: with this type of job the PDF placed in the hot folder is attached to a default jobticket filled with the default values from the definition. The product used to create the job can be chosen as an import setting.

The orderticket used to create the order can be also configured.

Always accept orders and jobs: this is the same as PDF only job with default order ticket type of import, but in this case both the

Scanned job: this type of hot folder accepts PDF files that come from a scanner. The name must satisfy a certain template that includes a prefix, a job number and the date and time of creation. The import service searches for the Job with the corresponding number. If it finds it and if that Job contains a placeholder attachment, it will attach the given PDF to that Job. The Job must be created previous to the scan operation.



Other issues

Validation Before importing a Job, Order or Definition it is validated.

Old Definitions

If a job is imported that was last edited with a non-current definition, the job is upgraded to the current definition.

Importing from older versions

Orders/jobs from PRISMAdirect 1.1.1 and older cannot be imported into PRISMAdirect 1.2 and higher.

The definition of PRISMAdirect 1.1.1 and higher can be imported.

Import checks

Before importing a job or Definition the import service performs a number of checks to ensure the integrity of the system. Of course all XML files must comply with the XML format.

Jobs/Orders

First of all, a Job or Order must comply with the XML schema.

Secondly, ItemValues must comply with the defining type. These checks are additional to the checks described in § 7.2.19.

- **Number**: The integer value must lie between the minimum and maximum.
- **Text (String)**: The string must contain no more characters than the Maximum length. If the Multi line attributes is set to "0", the string cannot contain carrige returns or line feeds.
- **Yes/No (Boolean)**: valid Boolean according to § 7.2.19.
- **Choice (Enum)**: An option must exist with this value. Checked against the Option name. In the XML this is called the EnumKey.
- **Date**: properly formatted date.
- **Group**: empty string ("").

Definitions

Changing Definitions When changing the Definition outside the Product & order editor it is important to change the DefinitionID too. Otherwise the system will assume it is the same Definition, causing all kinds of strange problems.

Language The language of the PRISMAdirect installation must be available in the Definition.

Views The following Views must be present for the OrderDefinition and each ProductDefinition in the Definition:

- - WebShop
- - PrintShop
- - WebShopPrintTicket
- - PrintShopPrintTicket

Each View's ViewKey must be of the form <PRODUCT_GUID>.<VIEW_NAME>

OrderItemDefinitions If importing the Definition, there are no strict requirements as the system has a 'self-healing' mechanism which will automatically add mandatory ItemDefinitions (R&D, Locked) to a Definition if they don't exist.

However, if manually editing the Current definition, the following items must exist in order to avoid corruption:

- EmailWhenBudgetApprovedOrRejected
- OrderHasBeenAccounted
- PaymentDate
- LastName
- WebShopId
- NumberOfJobsInside
- CommunicationState
- ProcessingTokenID
- FinalCost
- UserId
- PaymentTransactionInternalSettings
- ContactAddress
- SubmittersDescription
- IsOrderCostApproved
- FirstName
- PaymentTransactionId

- Account
- WsFriendlyCommunicationState
- IsJobCostPossiblyOutdated
- EmailWhenOrderRejected
- PaymentProvider
- EmailWhenOrderAccepted
- ManualLabor
- WebShopName
- ProcessingUserName
- OcFriendlyCommunicationState
- CreationDate
- IsOrderPaid
- OrderName
- BillingAddress
- EmailWhenOrderReady
- BudgetApproversRemarks
- OrderJobQEntryIDs
- EmailAddress
- IsOrderLocked
- RequestCostQuotation
- EmailMessage
- EstimatedCost
- Company
- Date
- OrderNumber
- JobType
- FinalizedDate

ProductItemDefinitions If importing the Definition, there are no strict requirements as the system has a 'self-healing' mechanism which will automatically add mandatory ItemDefinitions (R&D, Locked) to a Definition if they don't exist.

However, if manually editing the Current definition, the following items must exist in order to avoid corruption:

- EmailWhenBudgetApprovedOrRejected
- ProofPrint
- CoverMediaWeight
- CaseName
- OrderQEntryID
- Pages
- CopyrightAdministration
- Punching
- CoverMediaColor
- WebShopId
- PublishingDepartment
- Role
- NumberOfJobsInside
- DocumentMediaWeight
- BindingMethod
- JobName
- ChargeBackID1
- Author
- CommunicationState
- OperatorNotes
- IsStationeryJob

- NrOfSourcePages
- JobSizeEstimation
- HasStaticAttachment
- ProcessingTokenID
- FinalCost
- JobInfo
- FinishingTime
- Plexity
- JobHasBeenPrinted
- EstimatedColorPages
- StationeryHeader
- Title
- OrientationAndBindingEdge
- PrepareTime
- CoverMediaSize
- JobSource
- ProductName
- CaseType
- DocumentProgrammed
- IntentRemarks
- MergeRequired
- ProcessingServiceUserNameOnBehalfOf
- CoverPlace
- IsJobCostPossiblyOutdated
- EstimatedBWPages
- IsJobPaid
- JobNumber
- JobReady
- PrintInColor
- ProcessingServiceTokenIDOnBehalfOf
- IpAddress
- PDFAvailable
- Copies
- AssignedTo
- ManualLabor
- DocumentMediaColor
- Folding
- DocumentMediaType
- WebShopName
- IsVDPJob
- ProcessingUserName
- NumberOfCopiesPerSet
- StationeryDescription
- CreationDate
- Media
- JmfEndpointName
- JobHasConflicts
- Quantity
- ReadyDate
- PrinterName
- InternalJobState
- NewCoverMedia
- JobContainsPreparedAttachment
- RequestSoftProof

- DocumentMediaSize
- Category
- EstimatedCost
- EmailMessage
- NumberOfBussinessCardsPerSet
- IgnoreAutomationDocumentPreparationErrors
- IsJobCostApproved
- PrintingDuration
- Date
- CoverMediaType
- PreviousEstimatedCost
- DocumentType
- OrderNumber

Removed items The following Item Definitions were removed from the internal definition of previous PRISMAdirect versions.The system does not use them anymore internally but they are still supported for backwards-compatibility reasons:

CoverMedia
DocumentMedia
FinishingType
PrintOnBothSides
CoverAt
Orientation
PrintOnColor
Drilling

The values of these items are mapped automatically (using a mapping table) to the values of corresponding new Item Definitions:

Old Item Definition	New Item Definition
CoverMedia	NewCoverMedia
DocumentMedia	Media
FinishingType	BindingMethod
PrintOnBothSides	Plexity
CoverAt	CoverPlace
Orientation	OrientationAndBindingEdge
PrintOnColor	PrintInColor
Drilling	Punching

Export

Manual

Orders

In the Order processing workspace, one or more orders can be exported to ZIP file. Upon initializing export, after the preparation is complete, the data will be sent as ZIP response to the browser. Depending on its configuration, the browser will ask for a save location or automatically save the file to its 'Downloads' location.

Files The ZIP file has a well defined structure and strict file names. It may contain one or more folders, each one containing an order together with all its related content. Such a (order) folder contains the following:

- An orderticket file (XML file) and a JDF ticket file (also XML), together defining order metadata
- For each job inside the order:
  - A jobticket file (XML file) and if exported a JDF ticket file (also XML) together defining job metadata
  - The job attachments and current/original merged PDF if available

File names When an order is exported, filenames for the order and its jobs will be generated by the system. If possible a combination of a prefix and the jobnumber will be used, e.g. "Job-1890.xml". If a file with that name already exists in the chosen folder, a new name is generated. In the example shown above, the system would try "Job-1890.1.xml", "Job-1890.2.xml", etc.

The names of the attachement files can be found in the XML files.

Compatibility Mode Backwards compatibility is out of discussion as previous PRISMAdirect versions didn't support any type of import at all. However, those exported ZIP files will probably support import in later versions of PRISMAdirect8.

Definitions

In the Product & order editor the Definition can be exported to a ZIP containing XML files. These files contain no Orders, only one Definition.

Automatic

In the PRISMAdirect Configuration the target directories for Orders can be chosen. Multiple target directories can be specified. It is also possible to configure export to delete the Orders after they have been exported.

Each Order will be individually exported into one ZIP file and the name of this file will be based on the order QEntryID.

Finally, it is possible to pause the export robot. If this checkbox is checked, automatic export is disabled.

Orders If the export service is configured to export Orders (so not paused), all Orders that comply to a certain query are exported to a certain directory. Each directory has its own SQL-like query that can be created using the PRISMAdirect Configuration build-in tool. This query determines which Orders are exported to that directory. If an Order matches multiple queries, the Export Service will export the Order to each of the matching directories. If one of these export-folders is configured

PRISMAdirect export types

At export, either manual or automatic, there is an option to “Export original files and job ticket only”. This means that the following files are omitted:

- JDF Ticket – used to store the history of the Job
- Converted Files – as a result, after import, if an operation is performed that will invalidate the Merge result, the convert operation must be run for all the attachments that are not in PDF format.

Logging

Two types of logging are used; the EventLog and log-files. The import and export services use the EventLog to log errors that effect the behaviour of the service itself. The log-files are used to log the actions done on specific orders, jobs or files10.

Automatic

Only automatic import and export actions are logged to log files. However, if a file cannot be used because it is open in another application, a new file will be created.

Not only all successful imports are logged, but also the failures. When possible the cause of the failure is also logged.

Format

Each action is written to a line in the log file, formatted as shown below:

***<Date+Time>”|”<LoggerName>”|”<Action Type>”|”<Action Description><end of line>

- Date+Time: Always logged as “YYYY-MM-DD HH:MM:SS” read from the system clock.
- Logger Name: Shows the source of the logged message.
- Action Type: Either “Error”, “Warning” or “Information”.
- Action Description: Free text.

Manual

Manual imports and exports are not logged in the log-files. Error situations however are logged in the EventLog by the respective application.

Chapter 4

Media catalogue management

Media Catalog management

PRISMAdirect defines two types of media, found in two ItemDefinitions: NewCoverMedia and Media. NewCoverMedia contains the full media specification for document covers. Media contains the full media specification for document itself.

Format The media is identified by a fixed set of media attributes, formatted as below:

<Name>”,”<Size>”,”<Tab>”,”<Insert>”,”<CycleLength>”,”<Weight>”,”<Type>”,

”<Color>”,”<PunchPattern>”,”<SingleSided>”,”<PrePrinted>

- Name – free (case sensitive) string, recommended use is the commercial name or ordering name
- Size – Indicates the the media width and height. MediaWidth is always smaller than or equal to MediaHeight. For roll media, MediaWidth indicates the width of the roll and MediaHeight equals 0 (zero)
- Tab – Describes wheter the media has tab edges attached to the main body.
- Insert – If this attribute has values “Insert”, no images may be printed on the media. For inserts “ImageableSides” is ignored.
- CycleLength – Indicates the set size of cyclic media, allows for advanced error handling, and automatic tab spooling.
- Weight – Indicates the media weight.
- Type – Recommended use is the material of which the media is made
- Color – Recommended use is to indicate the colour appearance of the media.
- PunchPattern – Used to indicate how many holes are prepunched in the sheets of the media.
- SingleSided – Flag used to describe if the media is single sided or not
- PrePrinted – Flag used to describe if the media is preprinted or not

After installation PRISMAdirect comes with a fixed set of media.

A new media can be created using PRISMAdirect and then imported in PRISMAdirect using Product & order editor. New media can also be created from the Product & order editor by a combination of the above mentioned media attributes inherited from PRISMAdirect.

Chapter 5
XML format

Intentions This chapter describes the XML format used in the Open Interface. This chapter is derived from [XMLSCHEMA], which is an interal R&D document.

Overview

Figure 3 shows a simplified view of the root of the Schema. The root element is the Oce element. From the DocWorks element the schema is devided into three subtrees. The first part shows the Job subtree, which provides a presentation of a Job. The second part shows the Order subtree. The third part shows the Definitions subtree. The Job element consist of Attachement, VdpRelevantData and ItemValues elements. The first refers to digital source files/documents belonging to a job. The second stores properties related to VDP jobs The last represents the properties of the job (number of required copies, required finishing options, name of document, name of job submitter, email address of job submitter, etc). A Job has a related Definition. This Definition specifies, among others, what Attachement and ItemValue elements must be included in a Job element.

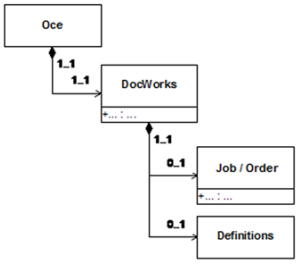


Figure 3: Simplified root of the Schema

Figure 4 shows a simplified view of a Definition element. Definition elements are more complex than Job elements. A Definition contains several other elements like ItemDefinition elements, View elements and Language elements. Each of these element types are stored in container elements: ProductItemDefinitions, OrderItemDefinitions, Views, Languages, etc.

An ItemDefintion specifies datatype, default value and possible values of an ItemValue. The collection of ItemDefinition elements contained in the ProductItemDefinitions and OrderItemDefinitions containers specifies all items that must be present in a Job or Order.

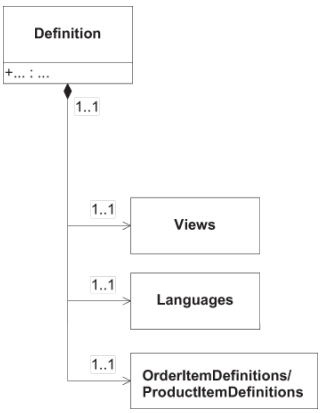


Figure 4: Simplified Definition element

A Definition does not only define the ItemValue elements of a Job or Order, it also defines their presentation and behavioral aspects. These presentation and behavioral aspects are covered by

View elements. A View is used to generate a graphical presentation (user interface) of a Job or Order.

There are 4 structural types of Views:
- ProductTreeview: defines the presentation of the Product
- OrderTreeView: defines the presentation of the Order
- ProductDefaultGroup: a predefined blueprint group with Product ItemDefinitions
- OrderDefaultGroup: a predefined blueprint group with Order ItemDefinitions

The DefaultGroups must only be seen as templates of ItemDefinitions groups and are used to more easily define the Product (defining a Job type) in the Product & order editor.

The relation between Jobs and Products should be clarified: Products are Job blueprints of different fixed major types (Flyer, Business cards, Generic) that also capture a collection of settings. A Job is created in the WebShop or Order processing workspace based on a product that must be chosen, thus, a Job can be of different types.

Views are mostly used to define Products and the Order (which is a single blueprint), and for each Product and the Order there are 4 views that must be present in the Definition:
- WebShop: defines the client view and it is used to generate the user interface of a Job or Order in the PRISMAdirect WebShop application
- PrintShop: defines how a Job or Order will be presented to a DPC operator in the PRISMAdirect Order processing workspace
- WebShopPrintTicket: defines the layout of the print ticket for the client
- PrintShopPrintTicket: defines the layout of the print ticket for the operator

Besides defining which items will be presented in the user interface and what their ordering and grouping will be, a view also defines some item specific attributes:
- a Boolean that specifies whether an item is read-only or editable,
- a Boolean that specifies whether entering a value for an item is required
- a Boolean that specifies whether the last entered value should be stored into the system for that user and prefilled when the user orders that Product again

Note that as a consequence of the View concept, a Job or Order presented in the WebShop application may differ from the presentation a DPC operator will get in the Order processing workspace. The client may not see all Job/Order items that are visible for the operator or vice versa and the ordering and grouping of items may differ. Besides, the item attributes as specified above may differ. For example, an item that is editable for the client may be read-only for the DPC operator.

In order to provide multi-language support in PRISMAdirect applications that present Jobs/Orders to end-users, all string elements in Definition elements that might be presented in a user-interface are available in all supported languages, stored in Translations elements.

All languages supported by a particular Defintion are specified in its Languages element. Language dependent strings (i.e. elements of type Translations) must be provided for all languages for which there is a Language element present in the Definition.

Element descriptions

This paragraph describes the elements used in the PRISMAdirect schema.

Unless stated otherwise, elements can contain no other elements and attributes than the ones specified here.

All elements have a “Version” attribute of type Int used for deserialization purposes. Theoretically, elements could evolve independently in the evolution of Definition structure and this attribute can be used to independently interpret each element.

However, to avoid redundancy, “Version” will be omitted from the following descriptions.

Oce element

The Oce element is the root element of the XML file. It contains one DocWorks element. The schema doesn’t allow other elements to be added to the Oce element.

Element	**Oce**	
Attributes	-	
Parent element	-	
Child elements	**DocWorks** **<other elements>**	**Exactly one.**

DocWorks element

The DocWorks element is the beginning of the DocWorks specific part of the XML file. The DocWorks element contains a Job, Order and/or a Definitions element. It can also contain neither, although that wouldn't make sense. The Definitions element contains zero one or two definitions of a ticket-template and the views on the ticket. The Job and Order elements can appear only once.

The SchemaVersion attribute is now obsolete and used to contain the version number of the Newton schema to which the XML-file complies. This attribute must be of the type integer (MS XML: int) and the value "12" should be used.

The SystemVersion attribute is now obsolete and used to describe the version number of the 'things'-list11. This attribute must be of the type int and "10" should be used.

The Version attribute has been introduced and is now used to internally identify the definition version:

- "5" for PRISMAdirect 1.1 and 1.1.1

- "6" for PRISMAdirect 1.2 and higher

Element	**DocWorks**	
Attributes	**SchemaVersion** **SystemVersion** **Version**	**Int.** **Int.** **Int.**
Parent element	**Oce**	**-**
Child elements	**Job** **Order** **Definitions**	**Zero or one.** **Zero or one.** **Zero or one.**

Job Element

The Job element is located in the DocWorks element. It contains a single job, consisting of a jobticket, eventual VDP settings and references to a number of files. The files can contain anything. The Attachments child element contains the references to the files. The ItemValues child element contains the jobticket.

The QEntryID attribute is the job's unique identifier (GUID). See § 7.3.3 for more info.

JobID is another unique identifier but it is not used anymore. For compatibility still, it should be uniquely set according to the syntax described in the schema.

The ProductType attribute defines the type of job attachments and CardID defines the product type.

ProductDefinition is the GUID of the Product itself.

The CreationLanguageKey attribute contains the language at which the client submitted the job. The possible values for the attribute a listed in § 7.3.2.

The CreationDefinitionID is obsolete and it is not used anymore. It should be set to empty GUID: "00000000-0000-0000-0000-000000000000".

The LastUsedDefinitionID attribute contains the ID of the Definition with which the job was last saved.

The CreationViewID attribute is not used anymore and it should always be set to "PrintShop".

The OrderID links the Job to a certain Order in the system as in PRISMAdirect Jobs cannot exist without Orders.

Element	**Job**	
Attributes	**JobID** **CreationLanguageKey** **LastUsedDefinitionID** **CreationDefinitionID** **CreationViewID** **ProductType** **ProductDefinition** **CardID** **OrderID** **QEntryID**	**UUID** **Enumeration** **UUID** **UUID** **Enumeration** **Enumeration** **UUID** **Enumeration** **UUID** **UUID**
Parent element	**DocWorks**	-
Child elements	**Attachments** **VdpRelevantData** **ItemValues**	**Exactly one.** **Exactly one.** **Exactly one.**

Order Element

The Order element is also placed as a child of DocWorks element. It contains a single order, consisting of a list of Jobs and the orderticket.

The QEntryID attribute uniquely identifies the Order (GUID). OrderID is another unique indentifier that should be set according to the schema but the QEntryID is the one that counts.

Like the Job, contains the same CreationLanguageKey, CreationDefinitionID and LastUsedDefinitionID with the same purpose. See § 7.2.3 for details.

Element	**Order**	
Attributes	**OrderID** **CreationLanguageKey** **LastUsedDefinitionID** **CreationDefinitionID** **QEntryID**	**UUID** **Enumeration** **UUID** **UUID** **UUID**
UUIDParent element	**DocWorks**	**-**
Child elements	**Jobs** **ItemValues**	**Exactly one.** **Exactly one.**

Jobs element (Order JobLinks)

The Jobs element placed under an Order element is only a container for Job element links to the Jobs belonging to the Order. It doesn't have any specific attributes.

Element	**Jobs**	
Attributes	**-**	
Parent element	**Order**	**-**
Child elements	**Job (Order JobLink)**	**One or more.**

Job element (Order JobLinks)

Each Job in the Jobs list is only a link.

The JobID attribute holds the ID of the Job linked to the Order and can be used to locate the corresponding Job entity that contains all data.

The JobPath attribute should be left empty.

Elemtent	**Job**	
Attributes	**JobID** **JobPath**	**UUID** **String**
Parent element	**Jobs (Order JobLinks)**	-
Child elements	-	

Attachments element

The Attachments element is located in the Job element. It contains a number of Attachment elements. The Attachments element has no attributes.

Element	**Attachments**	
Attributes	**-**	
Parent element	**Job**	**-**
Child elements	**Attachment**	**One or more.**

Attachment element

The Attachment element is located in the Attachments element. Each Attachment element contains a single reference to a file.

The AttachmentKey attribute contains the identifier of the attachment.

The MimeType specifies the mimetype of the attachment.

The FilePath attribute must be set to ".\" because the attachment files must be placed in the same Hot Folder as the XML file. The FileName attribute must contain the name of the file, including extension. E.g. "Test.pdf".

The DocumentName attribute contains the original name of the document that the client 'printed' to PRISMAdirect.

The FileIncluded attribute indicates whether or not the file in the Attachment is needed for the job to be complete. The import service only imports attachment files where FileIncluded is set to "1".

The FilePresent attribute is a Boolean that indicates that the attachment has a file connected to it. This attribute must have the value "0".

AttachmentType attribute contains the type of the attachment (no longer held by the JobType ticket item) and it can hold one of the value: atDigital (digital documents), atOther (other digital attachments), or atPaper (paper original submissions)

Converted attribute indicates if a certain attachment was converted on not. If true, the Convert element must be present as child element containing more information.

ConvertedFileName attribute indicates the URI/path of the file that was converted (For example . \file.pdf or http://server:port/path/file.pdf)

JDFId attribute is used for linking the Attachment with the Resource in the JDF ticket. This should not be changed unless 3'rd party tools also change the runlist id form the JDF ticket.

FileLastWriteTime stores the timestamp ticks at which the attachment was last written.

BWPages and ColorPages attribute caches color detection information for the attachment.

IsOperationRunning and APSCancellationToken are PRISMAdirect 'internal' attributes that help during attachment processing. They should be set as: IsOperationRunning="0" and APSCancellationToken=""

Element	**Attachment**	
Attributes	**AttachmentKey** **MimeType** **FilePath** **FileName** **DocumentName** **FilePresent** **FileIncluded** **AttachmentType** **Converted** **ConvertedFileName** **JDFId** **FileLastWriteTime** **BWPages** **ColorPages** **IsOperationRunning** **APSCancellationToken**	**String** **String** **String** **String** **String** **Boolean** **Boolean** **String** **Boolean** **String** **String** **Long** **Int** **Int** **Boolean** **String**

Parent element	**Attachments**	**-**
Child elements	**Preflight** **Convert** **AutoDocumentPreparation**	**-** **-** **-.**

Preflight element

The Preflight element is an optional element located in the Attachment element. It contains the result of the preflight operation on the referenced Attachment.

The IsPreflighted child element indicates if the current attachment was preflighted or not. It can be “0” or “1”.

If the attachment was indeed preflighted the PreflightResult child element indicates the state of the operation: Pending, OK, Error, etc.

The PreflightedFileName child element contains the name of the file resulted from the Preflight process.

The PdfReport child element contains the name of a PDF file that contains the report of the Preflight operation.

The XMLReport child element points to an XML file that contains the report of the Preflight operation.

Element	**Preflight**	
Attributes	-	
Parent element	**Attachment**	-
Child elements	**ISPreflighted** **PreflightResult** **PreflightedFileName** **PdfReport** **XmlReport**	**Boolean** **Enumeration** **String** **String** **String**

Convert element

This element is marked as optional in the schema, but it should be present if Converted flag is “1” at Attachment element level. It contains detailed conversion info in case of converted attachments.

The IsPreflighted child element indicates if the current attachment was preflighted or not. It can be “0” or “1”.

If the attachment was preflighted the PreflightResult child element indicates the state of the operation: Pending, OK, Error, etc.

The ConvertedFileName child element contains the name of the file resulted from the conversion process.

Element	**Convert**	
Attributes	**-**	
Parent element	**Attachment**	**-**
Child elements	**Converted** **PreflightResult** **ConvertedFileName**	**Boolean** **Enumeration** **String**

AutoDocumentPreparation element

This element is present if preparation templates had been applied to the Job.

The Result element stores the current status of the operation: InProgress, Pending, Error, OK, etc. Check the actual schema for all possible values.

Element	**AutoDocumentPreparation**	
Attributes	-	
Parent element	**Attachment**	-
Child elements	**Result** **AutoDocumentPreparationOperations**	**Enumeration** **Exactly one**

AutodocumentPreparationOperations element

This is simply a list corresponding to all templates that had been applied to the Job.

It has no specific attributes.

Element	**AutodocumentPreparationOperations**	
Attributes	**-**	
Parent element	**AutoDocumentPreparation**	**-**
Child element	**AutoDocumentPreparationOperation**	**One or more**

AutoDocumentPreparationOperation element

This element holds information about a template applied to the Job.

The TemplateName element stores the template name.

Result stores the current status of the operation: InProgress, Pending, Error, OK, etc. Check the actual schema for all possible values.

ResultMessage element stores the resulting message of template application. This can be a success message or error with detailed information. It is actually a dictionary stored in JSON format of Lang/Value pairs that holds messages for all 18 languages supported by PRISMA products. This is for localization purposes as the message is seen in the Order processing workspace.

Element	**AutoDocumentPreparationOperation**	
Attributes	**-**	
Parent element	**AutoDocumentPreparationOperations**	**-**
Child element	**TemplateName** **Result** **ResultMessage**	**String** **Enumeration** **String**

VdpRelevantData element

PRISMAdirect is VDP-enabled and supports fixed or user-submitted VDP master document and fixed or manual data source. This is tightly coupled with PRISMAprepare VDP functionality.

This element stores information relevant only for VDP-type jobs. It has various inner elements that store VDP settings.

VdpManualInputFields fields holds an encoded XML describing master document fields. In case of manual data source input, it will hold also the input values. VdpDataSourceInfo will store a text description of the fields in the master used only to provide nice information to the user.

VdpDelimiter and VdpSeparator are used in case of CSV (Text) data source do define the instance/field separators.

VDPPreviewGeneratedForCurrentJobState, VDPColorDetectionDoneForCurrentJobState, VDPPreviewGeneratedForCurrentJobStatePagesThreshold, VDPComposedDocumentTotalPages and IsPreviewTruncated are internal fields and should be set to “False” or “0” depending on the type.

VdpTable and VdpTableList are used only in case of an Excel/Access data source type. VdpTableList contains the name of the available tables in the data source while VdpTable stores the selected one. This value has to be present in the ones in VdpTableList.

VdpDataSourceType is an Integer from 0-4 representing the index of the possible data source types:

- - Undefined for non-VDP Jobs
- - Excel for XLS(X) data sources
- - Text for CSV data source
- - Access for MDB data source
- - Raw for manual input data source

VdpHasImages is a flag marking if the master document has image frames.

Element	**VdpRelevantData**	
Attributes	**-**	
Parent element	**Job**	
Child elements	**VdpManualInputFields** **VdpDelimiter** **VdpSeparator** **VdpDataSourceInfo** **VDPPreviewGeneratedForCurrentJobState** **VDPColorDetectionDoneForCurrentJobState** **VDPPreviewGeneratedForCurrentJobStatePagesThreshold** **VDPComposedDocumentTotalPages** **VdpTable** **VdpDataSourceType** **VdpHasImages** **VdpTableList** **IsPreviewTruncated**	**String(XML)** **String** **String** **String** **Boolean** **Boolean** **Integer** **Integer** **String** **Integer** **Boolean** **Zero or one** **Boolean**

VdpTableList element

This element is a container list of VdpTable elements. It holds the table names for Excel/Access VDP data source types.

It has no specific attributes.

Element	**VdpTableList**	
Attributes	**-**	
Parent element	**VdpRelevantData**	**-**
Child elements	**VdpTable**	**Zero or more**

VdpTable element

The element only stores a single value, a table name in the data source in case of Excel/Access data source types.

It has no specific attributes.

Element	**VdpTable**	**Holds the actual value**
Attributes	-	
Parent element	**VdpTableList**	-
Child elements	-	

ItemValues element

The ItemValues element is located in the Job or Order elements. It contains zero or more ItemValue elements.

Element	**ItemValues**	
Attributes	-	
Parent element	**Job** **Order**	- -
Child elements	**ItemValue**	**One or more.**

ItemValue element

The ItemValue element is located in the ItemValues element. It contains the value for a single jobticket or orderticket attribute. The value provided when the job was created is stored in the CreationValue element. The current value (may be the same) is stored in the LastUsedValue element. The ItemKey attribute indicates the item the value is related to. An ItemDefinition with the same ItemKey can exist in the Definition element (§ 7.2.22) specified for this job.

Element	ItemValue	
Attributes	ItemKey	string
Parent element	Itemvalues	-
Child elements	CreationValue LastUsedValue	Exactly one. Exactly one.

CreationValue element

The CreationValue element is located in the ItemValue element. It contains the value provided for the ItemValue when the job was created. That value is stored in the element's value.

The format in which the value is stored depends of the DataType attribute of the related ItemDefinition15:

- **dtNumber**: integer numeric value converted to a string.
- **dtString or dtLookup**: the string (value) itself.
- **dtBoolean**: Either ".TRUE" or ".FALSE" (respectively true (1) and false (0)).
- **dtDate**: a date formatted as "YYYYMMDDHHMMSS_TZ".
- **dtGroup**: must be an empty string ("").
- **dtEnum**: the EnumKey of the chosen option.

The DataType attribute also has some extensions with types of items used for the finishing process of a job. The value for these types is the same as for the **dtEnum** DataType.

The element has a single relevant attribute, Null, marking that there is no value at all for this item. There is a distinction between Null and an empty ("") or 0 value – which can still be considered values.

Element	**CreationValue**	**Element value is the value (string).**
Attributes	**Null**	**Boolean**
Parent element	**ItemValue**	-
Child elements	-	

LastUsedValue

The LastUsedValue element is located in the ItemValue element. It contains the current value for the ItemValue. That value is stored in the element's value.

The format in which the value is stored is the same as used for the CreationValue

Element	**LastUsedValue**	**Element value is the ItemValue's value (string).**
Attributes	**Null**	**Boolean**
Parent element	**ItemValue**	**-**
Child elements	**-**	

Definitions element

The Definitions element is located in the DocWorks element. It contains a set of Definition elements. Usually this will be a limited number of Definitions; zero to two.

Element	**Definitions**	
Attributes	**-**	
Parent element	**DocWorks**	**-**
Child elements	**Definition**	**Zero or one.**

Definition element

The Definition element is located in the Definitions element. It represents a single Definition.

The DefinitionID attribute uniquely identifies the Definition.

The Languages child element contains a list of languages supported in this definition. The AttachmentDefinitions element contains a list of attachment specifications. An attachment is a file that is connected to a job.

Element	**Definition**	
Attributes	**DefinitionID**	**UUID**
Parent element	**Definitions**	**-**
Child elements	**DefinitionText** **Languages** **Views** **OrderItemDefinitions** **ProductItemDefinitions** **OrderDefinition** **ProductDefinitions** **Icons**	**Exactly one.** **Exactly one.** **Exactly one.** **Exactly one.** **Exactly one.**

DefinitionText element

The DefinitionText element16 is located in the Definition element. It contains zero or more Translation elements. The DefinitionText element contains the localised name of the definition. For each translated string, one Translation child element is present.

The StringID and TooltipID attributes are for internal use only. If Export sets those attributes the values must be left unchanged. Otherwise the attributes must be set to “0”.

Element	DefinitionText	
Attributes	StringID TooltipID	**Int** **Int**
Parent element	Definition	**-**
Child elements	Translation	**One or more.**

Languages element

The Languages element is located in a few elements. It contains a number of Language elements, each specifying a language that is supported in the Definition.

Element	**Languages**	
Attributes	-	
Parent element	**Definition** **ProductDefinition** **OrderDefinition**	- - -
Child elements	**Language**	**One or more.**

Language element

The Language element is located in the Languages element. Each Language element contains a specification of a language that is supported in the Definition.

The LanguageKey attribute specifies the language.

Element	**Language**	
Attributes	**LanguageKey**	**Enumeration**
Parent element	**Languages**	-
Child elements	-	

Views element

The Views element is located in the Definition element. It contains a set of View elements, each describing a 'look' on the jobticket or orderticket.

Element	Views	
Attributes	-	
Parent element	Definition	**-**
Child elements	View	**Zero or more**.

View element

The View element is located in the Views element. A View element describes a ‘look’ on the jobticket/orderticket. Such a View consists of a tree-like list of the items to show. An item has several properties controlling its behaviour.

The ViewKey attribute identifies the view. The ViewKey must be unique within the Views element and is subjected to the rules. The maximum length of the ViewKey is 60 characters.

The Type attribute is an enumeration determining the type of the view. ViewType can have one of the following values (case sensitive):

- ProductTreeview
- OrderTreeView
- ProductDefaultGroup
- OrderDefaultGroup

The Visible attribute flags this view as visible or not. It is not currently used, it should be set to “1”.

The ViewText child element contains the localised name of the view. The Description child element contains possible details about the view (where it is displayed, what category of items it groups). They are both of the same type as a DefinitionText element.

The TreeDefinition child element is the ‘root’ of the ‘tree’ of items a jobticket consists of.

Element	**View**	
Attributes	**ViewKey** **Type** **Visible**	**String** **String** **Boolean**
Parent element	**Views**	**-**
Child elements	**TreeDefinition** **ViewText** **Description**	**Exactly one.** **Exactly one.** **Exactly one.**

Translation element

The Translation element is used in various other elements. The element contains a single translated string. The LanguageKey attribute contains the string's language and is subjected to the rules.

For each translated string, a corresponding Language element must be defined. The translated string is stored in the element's value. An empty string is also allowed. The maximum length of the translated string is 255 characters.

The Tooltip attribute contains the translated string that will be displayed in the tooltip for the element.

Element	**Translation**	**Element value is the translated string.**
Attributes	**LanguageKey** **Tooltip**	**Enumeration** **String**
Parent element	**Various**	
Child elements	-	

TreeDefinition element

The TreeDefinition element is located in the View element. It represents the ‘root’ of a ‘tree’ of jobticket/orderticket items17.

The ItemViewGroup child element contains a group of items (branch). In this case, that group is located in the root of the tree.

Element	**TreeDefinition**	
Attributes	**-**	
Parent element	**View**	**-**
Child elements	**ItemViewGroup**	**Zero or more.**

ItemViewGroup element

The ItemViewGroup element is either located in the TreeDefinition element, or in the ItemViewGroup element. The ItemViewGroup element represents a group of items that is organised in one branch.

The ItemKey attribute forms the link between the ItemDefinition and this element. An ItemViewGroup element must only be used if DataType of the related ItemDefinition is “dtGroup”.

The IsExpanded is not used anymore and should be set to “1”.

The Visible attribute flags this view as visible or not. Not used for now, should be set to “1”.

Element	**ItemViewGroup**	
Attributes	**ItemKey** **IsExpanded** **Visible**	**String** **Boolean** **Boolean**
Parent element	**TreeDefinition** **ItemViewGroup**	**-** **-**
Child elements	**ItemViewGroup** **ItemViewLeaf**	**Zero or more.** **Zero or more.**

ItemViewLeaf element

The ItemViewLeaf element is located in the ItemViewGroup element and describes the view on a single item.

The ItemKey attribute forms the link between the ItemDefinition and this element. An ItemViewLeaf element must only be used when the DataType of the related ItemDefinition is “dtNumber”, “dtString”, “dtEnum”, “dtBoolean” or “dtDate”.

The ReadOnly attribute specifies if the user can change the item. The Required attribute specifies if the user must provide a value for the item. The Stored attribute specifies if the value typed by the user should be used as the new default during the next session. An attribute value of “1” sets these properties, “0” resets them.

Element	**ItemViewLeaf**	
Attributes	**ItemKey** **ReadOnly** **Required** **Stored**	**String** **Boolean** **Boolean** **Boolean**
Parent element	**ItemViewGroup**	-
Child elements	-	

OrderItemDefinitions element

The OrderItemDefinitions element is located in the Definition element. It contains a set of declarations of orderticket items.

Element	**OrderItemDefinitions**	
Attributes	**-**	
Parent element	**Definition**	**-**
Child elements	**ItemDefinition**	**Zero or more**.

ProductItemDefinitions element

Similar to OrderItemDefinitions element, it is located in the Definition element. It contains a set of declarations of jobticket items.

Element	ProductItemDefinitions	
Attributes	-	
Parent element	Definition	-
Child elements	ItemDefinition	Zero or more.

ItemDefinition element

The ItemDefinition element is located in the OrderItemDefinitions or ProductItemDefinitions elements. Each ItemDefinition element represents a declaration for a single jobticket or orderticket item.

The ItemKey attribute identifies the item. Within the ItemDefinitions element the ItemKey must be unique and is subjected to the rules. If a certain item exists in multiple definitions

, multiple ItemDefinition elements will exists with identical ItemKeys. The maximum length of the ItemKey is 60 characters.

The ItemKey can have three forms depending on the case:

- ItemName: for system wide leaf items

- SpecializedGroupName.GroupName for default groups declarations

- Product/OrderID.ViewName.GroupName for groups created inside ProductDefinition/ OrderDefinition Views. Those may be predefined, created from default groups or custom

The DataType attribute determines the item type. This enumeration can have one of the following values:

 dtGroup: special type of Item. In the user interface this item can ‘contain’ other items, e.g. because it is a branch in a tree.

 dtNumber: numeric item. Integer numbers only.

 dtString: text item.

 dtBoolean: item can have one of two fixed values.

 dtEnum: item can have a one of a limited set of values.

 dtDate: date and time.

 dtLookup: a more special type of item definition that allows defining an ODBC data source and a query in order to dynamically retrieve a list of items. In this regard, ie behaves like a dynamic-value dtEnum.

PRISMAdirect supports a special type of items called Visual Ticket Items. These items are based on a dtEnum item and present choices strictly related to visual settings of a job. Because they are specialized and more complex items, new DataTypes are added for them:

 dtBinding

 dtOrientation

 dtPlexity

 dtCoverPlace

 dtMedia

 dtCover

 dtPrintInColor

 dtPunching

 dtFolding

The Locked attribute specifies if the ItemDefinition can be removed in the Product & order editor. If “0” the item can be removed, if “1” the item cannot. In [DEFAULTTICKET] a number of items are specified where this attribute must be set (attribute value = “1”). For all other items it must be set to “0”.

The RD_Only attribute specifies when the item is visible in the Product & order editor. If the attribute is “0” the item is visible, if “1” it is not. In [DEFAULTTICKET] a number of items are specified where this attribute must be set (attribute value = “1”). For all other attributes it must be set to “0”. Those items al mainly system internal items that should never appear anywhere.

The Log attribute specifies if the item should be logged in the joblog. If the attribute is “0” the item’s value is not logged, if it is “1” it is logged.

The RDHybrid_Only attribute marks the ItemDefinition as having an intermediary state between Locked and RD_Only: It can be seen in Product & order editor in order to be placed in products views, email templates, etc, but cannot be edited at all.

The ItemDefVersion identifies the structural version of the ItemDefinition and it is the same as Version.

The AllowAddition, AllowOther and BaseChoiceItemDefVersion are used in case of dtEnum types (also called “choice types”). AllowAddition is a flag marking if addition of new values from Product & order editor is allowed to this item. AllowOther controls if the “Other” option (value) is allowed. Finally, BaseChoiceItemDefVersion is internally used and should be set to Definition Version (“6” in the current version).

The ItemShortText element contains a localised string containing the name of the item. This string can be used in e.g. a joblist. The ItemTreeText element contains the localised text that should be used when drawing the ticket as described in the TreeDefinition. The difference between the ItemShortText and ItemTreeText lies in the use of the text, which determines the length of the string.

The xxxMetaData elements contain data-type specific metadata for the item. The name of the element depends on the DataType of the item definition.

Element	ItemDefinition	
Attributes	ItemKey DataType Locked RD_Only RDHybrid_Only Log ItemDefVersion AllowAddition AllowOther BaseChoiceItemDefVersion	String Enumeration Boolean Boolean Boolean Boolean Int Boolean Boolean Int
Parent element	xxxItemDefinitions	-
Child elements	ItemShortText ItemTreeText xxxMetaData	Exactly one. Exactly one. Exactly one.

ItemShortText element

The ItemShortText element contains the localised name of the item. This text is e.g. used in a joblist in the Order processing workspace. The StringID and TooltipID attributes are for internal use in PRISMAdirect. If Export sets these attributes the value must be left unchanged. Otherwise the attributes must be set to “0”.

Element	**ItemShortText**	
Attributes	**StringID** **TooltipID**	**Int** **Int**
Parent element	**ItemDefinition**	**-**
Child elements	**Translation**	**One or more.**

ItemTreeText element

The ItemTreeText element contains the localised name of the item. This text is e.g. used in the tree-view of the ticket specified by the TreeDefinition. The StringID and TooltipID attributes are for internal use in PRISMAdirect. If Export sets these attributes the value must be left unchanged. Otherwise the attributes must be set to “0”.

Element	**ItemTreeText**	
Attributes	**StringID** **TooltipID**	**Int** **Int**
Parent element	**ItemDefinition**	**-**
Child elements	**Translation**	**One or more.**

xxxMetaData element

The xxxMetaData element is an abstract19 element located under the ItemDefinition element.

Being an abstract element it does not actually exist. Instead exactly one of its child elements must be used. Which one depends on the DataType attribute of the parent ItemDefinition element.

Element	xxxMetaData	
Attributes	-	
Parent element	ItemDefinition	-
Child elements	GroupMetaData NumberMetaData StringMetaData BooleanMetaData EnumMetaData DateMetaData	Exactly one on the mentioned child elements.

xxxDefault element

The xxxDefault element is an abstract20 element located under the xxxMetaData element.

Being an abstract element it does not actually exist. Instead exactly one of its child elements must be used. Which one depends on the DataType attribute of the related ItemDefinition element.

Element	xxxDefault	
Attributes	-	
Parent element	xxxMetaData	-
Child elements	NumberDefault StringDefault BooleanDefault EnumDefault DateDefaultOffset	Exactly one of the elements. Which one depends on the item's datatype. -

GroupMetaData element

The GroupMetaData element is located in the xxxMetaData element. It contains the metadata for a group. Currently this element is empty.

Element	GroupMetaData	
Attributes	-	
Parent element	xxxMetaData	-
Child elements	-	

NumberMetaData element

The NumberMetaData element is located in the xxxMetaData element. It contains metadata of numeric items.

The Minimum attribute contains the minimum value of the item. The Maximum attribute contains the maximum. The NumberDefault child element contains the default value of the item.

The Decimals attribute contains the maximum number of decimals that the item can have.

Element	NumberMetaData	
Attributes	Minimum Maximum Decimals	Int Int Int
Parent element	xxxMetaData	-
Child elements	NumberDefault	Exactly one.

NumberDefault element

The NumberDefault element contains the default value for a numeric item. It is located in the NumberMetaData and the xxxDefault elements.

The element has a single relevant attribute, Null, marking that there is no value at all for this item. There is a distinction between Null and an empty ("") or 0 value – which can still be considered values.

Element	NumberDefault	Element value is the default value (Int).
Attributes	Null	Boolean
Parent element	NumberMetaData xxxDefault	-
Child elements	-	

StringMetaData element

The StringMetaData element is located in the xxxMetaData element. It contains the metadata of text (string) items.

The MaxLength attribute specifies the maximum number of characters in the string. The MultiLine attribute specifies if the string can contain multiple lines of text or not. The StringDefault child element contains the default value of the item.

Element	StringMetaData	
Attributes	MaxLength MultiLine	Int Boolean
Parent element	xxxMetaData	-
Child elements	StringDefault	Exactly one.

Stingdefault element

The StringDefault element contains the default value for string items. It is located in the StringMetaData and the xxxDefault element.

The element has a single relevant attribute, Null, marking that there is no value at all for this item. There is a distinction between Null and an empty ("") or 0 value – which can still be considered values.

Element	**StringDefault**	**Element value is the default value (string)**
Attributes	**Null**	**Boolean**
Parent element	**StringMetaData** **xxxDefault**	- -
Child elements	-	

BooleanMetaData element

The BooleanMetaData element contains the metadata of Boolean items. Boolean items can either be true or false.

The BooleanDefault child element contains the default value of the item. The ItemTrueText contains the localised caption of the value if that is True. The ItemFalseText contains the localised caption of the value if that is False. The ItemUndefinedText contains the localised caption when the value is not defined.

Element	BooleanMetaData	
Attributes	-	
Parent element	xxxMetaData	-
Child elements	BooleanDefault ItemTrueText ItemFalseText ItemUndefinedText	Exactly one. Exactly one. Exactly one. Exactly one.

BooleanDefault element

The BooleanDefault element contains the default value for a Boolean item. The default is stored in the element value and can either be true (“1”) or false (“0”). The BooleanDefault element is used in the BooleanMetaData and the xxxDefault element.

The element has a single relevant attribute, Null, marking that there is no value at all for this item. There is a distinction between Null and an empty (“”) or 0 value – which can still be considered values.

Element	BooleanDefault	Element value is the default (boolean).
Attributes	Null	Boolean
Parent element	BooleanMetaData xxxDefault	- -
Child elements	-	

ItemTrueText element

The ItemTrueText element contains the localised caption of a Boolean value for True-values.

The StringID and TooltipID attributes are for internal use only. If Export sets these attributes the value must be left unchanged. Otherwise the attributes must be set to “0”.

The Translation child elements contain the localised value-caption.

Element	**ItemTrueText**	
Attributes	**StringID** **TooltipID**	**Int** **Int**
Parent element	**BooleanMetaData**	**-**
Child elements	**Translation**	**Zero or more.**

ItemFalseText element

The ItemFalseText element contains the localised caption of a Boolean **value** for False-values.

The StringID and TooltipID attributes are for internal use only. If Export sets these attributes the value must be left unchanged. Otherwise the attributes must be set to “0”.

The Translation child elements contain the localised value-caption.

Element	**ItemFalseText**	
Attributes	**StringID** **TooltipID**	**Int** **Int**
Parent element	**BooleanMetaData**	**-**
Child elements	**Translation**	**Zero or more.**

ItemUndefinedText element

The ItemUndefinedText element contains the localised caption of a Boolean value for Undefined-values.

The StringID and TooltipID attributes are for internal use in only. If Export sets these attributes the value must be left unchanged. Otherwise the attributes must be set to “0”.

The Translation child elements contain the localised value-caption.

Element	**ItemUndefinedText**	
Attributes	**StringID** **oltipID**	**Int** **Int**
Parent element	**ToBooleanMetaData**	**-**
Child elements	**Translation**	**Zero or more.**

EnumMetaData element

The EnumMetaData element is located in the xxxMetaData element. It contains the metadata for an enumerated item.

The EnumValues element contains the possible enumerations. The EnumDefault element contains the default value.

Element	**EnumMetaData**	
Attributes	**-**	
Parent element	**xxxMetaData**	**-**
Child elements	**EnumValues** **EnumDefault**	**Exactly one.**. **Exactly one.**

EnumValues element

The EnumValues element contains the possible enumerations for an enumerated item. The possible enumerations are stored in an EnumValue element.

Element	**EnumValues**	
Attributes	-	
Parent element	**EnumMetaData**	-
Child elements	**EnumValue**	**One or more.**

EnumValue element

The EnumValue element contains a single enumeration for an enumerated item. It contains both the language independent key of the EnumValue, as well as the localised caption of the value.

The EnumKey attribute contains the identifier of the value. This identifier must be unique within the EnumValues element and is subjected to the rules. The maximum length of the EnumKey is 60 characters.

The StringID and TooltipID attributes are for internal use only. If Export sets these attributes the value must be left unchanged. Otherwise the attributes must be set to “0”.

The Locked attribute specifies if this EnumValue can be removed in the Product & order editor. A number of EnumValue elements must have the locked attribute set (=”1”). This are the EnumValues (’options’) listed in [DEFAULTTICKET] where the item itself is locked. In all other situations the Locked attribute must be set to “0”.

The DefaultIconIndex, HighlightIconIndex and LowAttentionIconIndex are optional attributes that configure the icons used for an enumeration value on the user interface for different states. However, for now this is not yet fully supported and DefaultIconIndex will be used in all cases. The value of this attribute represents the index in the Icons list defined in another section of the Definition. See Icons element for information on that section.

The Translation element contains the localised caption of the value.

Element	**EnumValue**	
Attributes	**EnumKey** **StringID** **TooltipID** **Locked** **DefaultIconIndex** **HighlightIconIndex** **LowAttentionIconIndex**	**String** **Int** **Int** **Boolean** **Int** **Int** **Int**
Parent element	**EnumValues**	**-**
Child elements	**Translation**	**Zero or more.**

EnumDefault element

The EnumDefault element is used in the EnumMetaData and the xxxDefault elements. It contains the default value for an enumerated item. More specific, it contains the EnumKey of the default value.

The element has a single relevant attribute, Null, marking that there is no value at all for this item. There is a distinction between Null and an empty (“”) or 0 value – which can still be considered values.

Element	**EnumDefault**	**Element value is the default (string)**
Attributes	**Null**	**Boolean**
Parent element	**EnumMetaData** **xxxDefault**	- -
Child elements	-	

DateMetaData element

The DateMetaData element contains the metadata for a Date item. The DateDefaultOffset child element contains the default value.

Element	**DateMetaData**	
Attributes	**-**	
Parent element	**xxxMetaData**	**-**
Child elements	**DateDefaultOffset**	**Exactly one.**

DateDefaultOffset element

The DateDefaultOffset element contains the default value for a date-item. The default value is an offset value, and should be added to the current date when creating a job or order.

The element has a single relevant attribute, Null, marking that there is no value at all for this item. There is a distinction between Null and an empty (“”) or 0 value – which can still be considered values.

Element	**DateDefaultOffset**	**Element value is the default value offset (Int).**
Attributes	**Null**	**Boolean**
Parent element	**DateMetaData** **xxxDefault**	- -
Child elements	-	

ProductDefinitions element

This element stores a list of Product structures, one for each product defined in the system. There are multiple Product types, thus, multiple element types can be nested:

- Stationery
- BoundDocument
- Flyer
- BusinessCard
- NewMiscellaneous
- Legacy

At least one Legacy and NewMiscellaneous type of products must be defined.

The element doesn't have any specific attributes.

Element	**ProductDefinitions**	
Attributes	-	
Parent element	**Definition**	-
Child elements	**Stationery** **BoundDocument** **Flyer** **BusinessCard** **NewMiscellaneous** **Legacy**	**Zero or more.** **Zero or more.** **Zero or more.** **Zero or more.** **One or more.** **One or more.**

Stationery element

This element defines a product type used to order stationery items. Those are usually various print-enabled objects like mugs, USB sticks, etc.

The ItemKey attribute stores the identifier of the product. This must be a unique indentifier in the entire system of UUID type.

The IsDefault attribute is an optional Boolean flag not used for this product type so it shouldn't be set.

It has only one child element of type VisualizedProductDefinition.

Element	**Stationery**	
Attributes	**ItemKey** **IsDefault**	**UUID** **Boolean**
Parent element	**ProductDefinitions**	**-**
Child elements	**VisualizedProductDefinition**	**Exactly one.**

BoundDocument element

This element defines a product type that models a document with binding via its internal restrictions.

It has the same attributes and structure as a Stationery element.

Flyer element

Same as BoundDocument, but models a flyer-type product via its internal restrictions.

It has the same attributes and structure as a Stationery element.

BusinessCard element

This element models a special type of product used to create business cards. It is basically a VDP enabled product type and Product & order editor will present a customized interface especially for the file input area.

Still, structurally speaking, it has the same attributes and structure as a Stationery element.

NewMiscellaneous element

This type of element models a type of product also known as “Generic” as it has no internal restrictions and allows configuring everything the way user really wants even if it wouldn’t make sense.

The definition must contain at least one product of this type.

The IsDefault attribute is actually used for this type of product and it allows in Product & order editor to define a fallback product of this type that will be used in case a product is deleted from the system and there are existing Jobs based on that product. As internally each Job must be the blueprint of a product that is referenced and used for value constraints, display, etc, the default “Generic” will be the one inheriting such Jobs. Although the schema doesn’t specify this, obviously, there must be one and only one default at a given time.

The ShowRemarksIn…Intent optional flags control if remarks will be shown in the WebShop for the client user or in the Order processing workspace for the operator (PrintShop).

Element	**NewMiscellaneous**	
Attributes	**ItemKey** **IsDefault** **ShowRemarksInPrintShopIntent** **ShowRemarksInWebShopIntent**	**UUID** **Boolean** **Boolean** **Boolean**
Parent element	**ProductDefinitions**	**-**
Child elements	**ProductDefinition** **ReferencedViews**	**Exactly one.** **Exactly one.**

Legacy element

The Legacy type of product is a very special type used for backwards compatibility. It is also a generic product but without visualization support (previewer, etc), only used to provide support for legacy jobs. Exactly one of this type of product must exist in the system. From Product & order editor it cannot be removed and other instances cannot be created.

The ShowRemarksIn…Intent and IsDefault optional flags are not used for this product type so it shouldn't be set.

Element	**Legacy**	
Attributes	**ItemKey** **IsDefault** **ShowRemarksInPrintShopIntent** **ShowRemarksInWebShopIntent**	**UUID** **Boolean** **Boolean** **Boolean**
Parent element	**ProductDefinitions**	**-**
Child elements	**ProductDefinition** **ReferencedViews**	**Exactly one.** **Exactly one.**

VisualizedProductDefinition element

This element is just a wrapper element used for all product types except NewMiscellaneous and Legacy and it is used to indicate that the product is of “visualized type” meaning that it supports preview (via the Previewer component available for both the client and the operator). Although not reflected in the structure, NewMiscellaneous also is a “visualized type”, Legacy type being the only one left.

However, for compatibility reasons, the wrapper must be used as indicated.

It has no relevant attributes.

Element	**VisualizedProductDefinition**	
Attributes	**-**	
Parent element	**Stationery** **BoundDocument** **Flyer** **BusinessCard**	**-** **-** **-** **-**
Child elements	**ProductDefinition** **ReferencedViews**	**Exactly one.** **Exactly one.**

ProductDefinition element

This element defines some of the structural properties of the defined product tied to the Definition. The rest of the properties (mainly visual) are defined in another file, not a part of OpenInterface specification.

The ItemKey uniquely identifies the product type and it is redundant information also defined at product level (Stationery, Flyer, etc.).

The ProductFileInputType is a value indicating what input file configuration is available for the product and it can be one of the following very different types:

- Stationery – always set for Stationery – type products that have no such an area as it is not about documents and files
- Normal – marks regular document-oriented products
- Static – for products that have a preconfigured attached file to be ordered and that can't be changed from anywhere outside the Product & order editor (product configuration)
- VDP – for VDP-enabled products

The VDPProductType applies for VDP-enabled products and it is an optional attribute with the following possible values:

- LateBound – for manually chosen (by the user/operator) master documents
- EarlyBound – for preconfigured (Product & order editor) master documents
- None

The ItemDisplayRoot child element isn't actually used but must have the same structure as a DefinitionText element. All values can be set to 0 for Integers and "" (empty) for Strings.

The Renamings element contains caption customizations for item definitions contained by the product views.

The Languages element is redundant and for now it should be set the same everywhere it appears. See for more information.

Element	**ProductDefinition**	
Attributes	**ItemKey** **ProductFileInputType** **VDPProductType**	**UUID** **Enumeration** **Enumeration**
Parent element	**VisualizedProductDefinition** **NewMiscellaneous** **Legacy**	**-** **-** **-**
Child elements	**ItemDisplayRoot** **Renamings** **Languages**	**Exactly one.** **Exactly one.** **Exactly one.**

Renamings element

This element is only a list container that holds customizations of item definition appearance names (captions). It should be noted that currently any customization performed for an item definition present in the Product is shared for all views so currently having the same ItemDefinition named “X” in WebShop View and “Y” in PrintShop View is not supported.

Items that are present in any views of the product but are not present in this list will use the default captions defined at ItemDefinition level.

This element has no specific attributes.

Element	**Renamings**	
Attributes	-	
Parent element	ProductDefinition OrderDefinition	- -
Child elements	Renaming	Zero or more.

Renaming element

This element defines a customization of an ItemDefinition name performed in the Product & order editor. Item definitions have default captions defined for all system active languages but they can be overridden at product level by defining such an entry.

The ItemDefinition attribute links the renaming to the respective item definition and this value must be the same as the ItemKey of the referred ItemDefinition.

The element contains one or more ItemDisplayChild that define the new captions.

Element	**Renaming**	
Attributes	**ItemDefinition**	**String**
Parent element	**Renamings**	**-**
Child elements	**ItemDisplayChild**	**Exactly one.**

ItemDisplayChild element

This element contains a list of Translation elements, one for each active Language in the definition.

It has no specific attributes.

Element	**ItemDisplayChild**	
Attributes	**-**	
Parent element	**Renaming**	**-**
Child elements	**Translation**	**Zero or more.**

ReferencedViews element

This is also a list container element that allows the definition of links between the product and its corresponding View elements. It has exactly 4 elements of type ReferencedView, one for each type of view.

It has no specific attributes.

Element	**ReferencedViews**	
Attributes	-	
Parent element	**VisualizedProductDefinition** **NewMiscellaneous** **Legacy** **OrderDefinition**	- - - -
Child elements	**ReferencedView**	**Four of child element**

Referencedview element

The content of this element is a View ItemKey that links the Product/Order definition to a defined View.

It has no specific attributes.

Element	**ReferencedView**	**The value is the View ItemKey**
Attributes	-	
Parent element	**ReferencedViews**	-
Child elements	-	

OrderDefinition element

PRISMAdirect supports a single type of Order that can be globally configured in Product & order editor. This element is fixed and defines the associated properties.

The ItemKey is an UUID that can be randomly generated and must be then used in related View ItemKey creation.

The ItemDisplayRoot child element isn't actually used but must have the same structure as a DefinitionText element. All values can be set to 0 for Integers and "" (empty) for Strings.

The Renamings element contains caption customizations for order item definitions contained by the product views.

The Languages element is redundant and for now it should be set the same everywhere it appears.

The ReferencedViews, as for a ProductDefinition, make the link to order Views.

Element	**ProductDefinition**	
Attributes	**ItemKey** **ProductFileInputType** **VDPProductType**	**UUID** **Enumeration** **Enumeration**
Parent element	**VisualizedProductDefinition** **NewMiscellaneous** **Legacy**	- -. -
Child elements	**ItemDisplayRoot** **Renamings** **Languages** **ReferencedViews**	**Exactly one.** **Exactly one.** **Exactly one.** **Exactly one.**

Icons element

This element store the list icons optionally used by Enum-type ItemDefinitions. It is simply a container list of IconReference elements.

It has no relevant attributes.

Element	**Icons**	
Attributes	**-**	
Parent element	**Definition**	**-**
Child elements	**IconReference**	**Zero or more.**

IconReference element

This element is a wrapper of an Icon element that attaches the IconIndex attribute.

The IconIndex attribute is a positive Integer uniquely identifying an Icon element. This value must be used in references to this icon.

Element	**IconReference**	
Attributes	**IconIndex**	**Integer**
Parent element	**Icons**	**-**
Child elements	**Icon**	**Exactly one.**

Icon element

This element stores the actual information about the icon.

The URL and Image child elements are used to identify or store the actual image. There are two cases:

- Images embedded in the default definition:
- 1. URL is set to a value that identifies the resource and looks like: embeddedimage:// entityassembly/embeddedresources/images/sea24_cover_front.png
  2. o Image is set to Null and has no content
- - Images uploaded by the product administrator:
- 1. o URL is set to: embeddedimage://definition/
  2. o Image contains the Base64 encoding of the image itself.

Both URL and Image child elements have a Null Boolean attribute to mark if the URL or Image content is actually null. There is a distinction between Null and an empty ("") or 0 value – which can still be considered values.

Element	Icon	
Attributes	-	
Parent element	IconReference	-
Child elements	URL Icon	Exactly one. Exactly one.

LookupMetadata element

This is a wrapper element for the ODBCSettings element that holds the settings for a lookup item definition.

It has no specific attributes.

Element	**LookupMetadata**	
Attributes	**-**	
Parent element	**xxxMetaData**	**-**
Child elements	**ODBCSettings**	**Exactly one.**

ODBCSettings element

This element stores the actual retrieval settings for a lookup item definition.

The ConnectionDataSource attribute holds a string value containing the ODBC connection name.

The ConnectionUsername and ConnectionPassword store the credentials used to access the data store.

The QueryString attribute stores as a string value the query to use to retrieve the items from the data source.

Element	**ODBCSettings**	
Attributes	**ConnectionDataSource** **ConnectionUsername** **ConnectionPassword** **QueryString**	**String** **String** **String** **String**
Parent element	**LookupMetadata**	-
Child elements	-	

Rules

This paragraph contains a set of rules and concepts that apply to the XML Schema. This paragraph is referenced too from the element descriptions.

xxxKey

ItemKey, AttachmentKey, ViewKey and EnumKey attributes are used as identifiers. The value of these attributes is subjected to some rules:

- Valid characters are a-z, A-Z, 0-9 and '_'.
- The Key cannot start with '_'.

LanguageKey

A LanguageKey attribute (enumeration) specifies a language.

The possible values are:

- en-us English (US)
- en-gb English (UK)
- nl-nl Dutch
- de-de German
- fr-fr French
- da-dk Danish
- it-it Italian
- nb-no Norwegian
- pt-pt Portuguese
- es-es Spanish
- sv-se Swedish
- fi-fi Finnish
- pl-pl Polish
- ch-cz Czech
- hu-hu Hungarian
- zh-cn Simplified Chinese
- ru-ru Russian
- ja-jp Japanese

UUID & GUID

A UUID is the same as a GUID. GUID stands for Global Unique Identifier. UUID for Universal Unique Identifier.

These identifiers are based on the MAC21-address of the network card of the machine they are generated on. Additionally, a sequence number is included. This makes a UUID or GUID a garantied unique number.

UUIDs and GUIDs can be generated with GuidGen.exe. This tool comes with the MS Platform SDK and MS Visual Studio 6.

An example GUID is: “A959FAA1-CC73-11d4-9B62-0008C752E41D”. Valid characters are 0-9 and A-F. Note that the ‘-‘ characters must be placed at the positions shown in the example above.

Implementation Issues

Encoding Only the standard encodings supported by MS XML are supported. In normal operation the XML files must be UTF-16 encoded.

Adding elements The complete schema is closed. This means that no items can be present except for the ones specified in the schema.

Chapter 6
JDF mapping in PRISMAdirect

JDF / JMF interface

The JDF/JMF interface enables job submission to PRISMAdirect and provides job status feedback for JDF-enabled client applications. It does this by enabling one or more configurable JMF endpoints which listen for and react to JMF messages received over HTTP from the JDF Clients.

Job submission

The JDF Clients can submit a job to PRISMAdirect by sending a specific JMF message and specifying the JDF ticket and the Data files to be attached. The following scenarios are supported for submission of a new job:

1. **JMF message with link to JDF ticket**
   The JDF Client sends a JMF message that specifies a URL (HTTP, FILE) where the JDF ticket can be found. The JDF ticket must be available at the specified URL. The JDF ticket will contain URL(s) that refers to the Data Files that must be attached to the job. The Data Files must be available at the specified URL(s). The JMF Server must handle the download of the JDF ticket and the Data Files from the specified URL(s).
2. **MIME with link(s) to Data Files**
   The JDF Client sends a MIME multipart message that contains the JMF message and the JDF ticket. The JMF message must contain a CID URL pointing to the body part of the MIME message that contains the JDF ticket. The JDF ticket contains URL(s) referencing the Data Files. The Data Files must be available at the location specified by the URL(s). The JMF Server must handle the download of the data files from the specified URL(s).
3. **MIME that includes the Data Files**
   The JDF Client sends a MIME multipart message containing the JMF message together with the JDF ticket and the Data Files. The JMF message contains a CID URL pointing to the part of the message that specifies the JDF ticket. The JDF ticket must contain one or more CID URL(s) that point to the part(s) of the MIME message that contain the Data File(s).

For more details and examples related to the described scenarios, see **SubmitQueueEntry** (section 5.11.10) and **JDF Packaging** (section 8.3) in the JDF Specification 1.3.

The JMF Server must obtain the JDF ticket and the attached data files which will be used to create a new job in PRISMAdirect. Because internally PRISMAdirect works with a PD ticket, the JDF ticket must be “converted” and the values from it must be transferred to a PD ticket. For this the system will use a **mapping table** (JDF-to-PD) which specifies “what values to take from the JDF ticket and where to put them in the PD ticket”.

PRISMAdirect will also create an Order into the system as jobs in PRISMAdirect must belong to Orders. The ItemValues in the orderticket will also be mapped based on the same mapping table

Status feedback

When required by the JDF Client, the system will expose the status of the jobs. The values for the status are specific to PRISMAdirect. The system will offer the possibility to customize these values to Client specific job statuses. For this again a **mapping table** (PD-to-JDF) will be used. The table will map each value of the status attributes in PD to another value.

The interface can be configured from the PRISMAdirect Web Configuration workspace where the product administrator can add, remove and configure the JMF Endpoints.

Each JMF endpoint configured in the system must have one JDF-to-PD mapping table and one PD-to-JDF mapping table for job status assigned to it. The mapping tables will be XML files stored on disk. At installation, the system copies the default mapping files in a special location in the asset folder on the Central Server machine: “$(OCE_STORAGE)\Asset\JMF Endpoints”.

When the product administrator adds a new JMF endpoint from the MMC he can choose to use the default mappings or assign custom mappings for the JDF-to-PD mapping and/or for the PD-to-JDF mapping. To use custom mappings, the product administrator can select another file on disk. The selected files must have the .XML extension and it must be validated against a predefined xml schema.

At any moment in the lifecycle of a JMF endpoint the product administrator can reconfigure it and also change or assign other mapping files.

General concepts and mechanisms

In the context of the JDF/JMF interface to PRISMAdirect (PD) two types of mapping tables are used, one to map the JDF ticket to the PD ticket and one to map the PD job status to a customer specific job status. This document is meant to be a specification of the two mapping tables. It describes the syntax of the mapping files and also the customization possibilities.

The document should serve system engineers and C&SS as support for understanding the structure of the mapping tables and for adapting them to the needs of the customer if necessary.

The JDF/JMF interface enables job submission to PRISMAdirect and provides job status feedback for JDF-enabled client applications. It does this by enabling one or more configurable JMF endpoints which listen for and react to messages received over HTTP from the JDF Clients.

- **Job submission**
  The JDF Clients can submit a job to PRISMAdirect by sending a JDF ticket and the Data files attached. Because internally PRISMAdirect works with a proprietary ticket, the JDF ticket must be "converted" and the values from it must be transferred to a PD ticket. For this the system will use a **mapping table** (JDF-to-PD) which specifies "what values to take from the JDF ticket and where to put them in the PD ticket".
- **Status feedback**
  When required by the JDF Client, the system will expose the status of the jobs. The values for the status are specific to PRISMAdirect. The system will offer the possibility to customize these values to Client specific job statuses. For this again a **mapping table** (PD-to-JDF) will be used. The table will map each value of the status attributes in PD to another value.

The interface can be configured from the administration interface where the product administrator can add, remove and configure the JMF Endpoints.

Each JMF endpoint configured in the system will have one JDF-to-PD mapping table and one PD-to-JDF mapping table assigned to it. The mapping tables will be XML files stored separately for each endpoint. After installation the system provides two default mapping tables.

When adding a new JMF endpoint, the product administrator can associate the default mapping tables with it or choose other mapping tables. At any moment after this, the product administrator can change the assignment to another mapping table or he can modify the existing mapping table directly by editing the XML file.

This offers freedom to the customers to adapt the functionality of the JDF/JMF interface so that it fits to their own JDF structure and also to their own job status values.

To each JMF endpoint, the product administrator can assign a different couple of mapping tables. So, on the same PRISMAdirect installation, different JDF Client applications can interact with the system based on different mappings JDF-to-PD and PD-to-JDF.

Mapping file deployment

When the product administrator adds a new JMF endpoint from the administration interface he must assign one mapping file for the JDF-to-PD mapping and one for the PD-to-JDF mapping. The system will create a folder on disk especially for the new endpoint and will copy the mapping files selected by the product administrator to that folder. The selected files must have the .XML extension. The sub-folders for each endpoint will be created in the asset folder on the Central Server machine of the PRISMAdirect installation: “$(OCE_STORAGE)\Asset\JmfEndpoints”.

If the administrator does not select specific mapping files, the system will copy the default mapping files to the same folder.

At any moment in the lifecycle of a JMF endpoint the product administrator can reconfigure it and assign other mapping files.

JDF to PRISMAdirect: mapping for job submission

File structure

The mapping file is used for extracting the job characteristics from the JDF ticket and creating the PRISMAdirect ticket with it. The structure of the file is quite simple; it consists in a list of mapping nodes. For each ticket item in the default PRISMAdirect ticket there is a mapping node Sample from the default JDF-to-PD mapping file:

```
<? xml version = "1.0" encoding="utf-8"?>

<jc:Mappings SchemaVersion="1.0" AttachmentPath="/jdf:JDF/jdf:ResourcePool/
jdf:FileSpec/@URL" xsi:schemaLocation="oce-com-pa-jc GrayBoxMappings.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:jc="oce-com-pa-
jc">

<jc:TextMapping Prefix="" Name="FirstName" Optional="false" Separator="">

<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/…/@FirstName"/>

</jc:TextMapping>

…

<jc:NumberMapping Name="Copies" Optional="false">

<jc:JdfField XPath="/jdf:JDF/jdf:ResourceLinkPool/…/@Amount"/>

</jc:NumberMapping>

…

<jc:EnumMapping Name="DocumentMediaColor" Optional="true">

<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool…/@MediaColorName"/>

<jc:EnumValueMapping JdfValue="White" AccessEnumValue="White"/>

<jc:EnumValueMapping JdfValue="Blue" AccessEnumValue="Blue"/>

<jc:EnumValueMapping JdfValue="Red" AccessEnumValue="Red"/>

<jc:EnumValueMapping JdfValue="Green" AccessEnumValue="Green"/>

<jc:EnumValueMapping JdfValue="Yellow" AccessEnumValue="Yellow"/>

<jc:EnumValueMapping JdfValue="Gray" AccessEnumValue="Gray"/>

<jc:EnumValueMapping JdfValue="Orange" AccessEnumValue="Orange"/>

<jc:EnumValueMapping JdfValue="Pink" AccessEnumValue="Pink"/>
<jc:EnumValueMapping JdfValue="Black" AccessEnumValue="Black"/>

</jc:EnumMapping>

…

<jc:ConditionalEnumMapping Name="CoverPlace" Optional="false">

<jc:ConditionalEnumValue AccessEnumValue="None">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/…/@FrontCoatings"
ExpectedValue="None"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/…/@BackCoatings"
ExpectedValue="None"/>
```

```
</jc:ConditionalEnumV

<jc:ConditionalEnumValue AccessEnumValue="Front">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/…/@FrontCoatings"
ExpectedValue="Coated"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/…/@BackCoatings"
ExpectedValue="None"/>

</jc:ConditionalEnumValue >

<jc:ConditionalEnumValue AccessEnumValue="Back">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/…/@FrontCoatings"
ExpectedValue="None"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/…/@BackCoatings"
ExpectedValue="Coated"/>

</jc:ConditionalEnumValue>

</jc:ConditionalEnumMapping>

…

</jc:Mappings>
```

Depending on the type of the ticket item in PRISMAdirect there are several types of mapping nodes:

- **NumberMapping**
- **DateMapping**
- **TextMapping**
- **EnumMapping**
- **ConditionalEnumMappin**
- **BooleanMapping**
- **TimeSpanMapping**
- **MediaMapping (MediaEnumMapping, MediaConditionalEnumMapping)**

Common fields

There are some common fields for all the mapping nodes independent of the type. Let's take a random example of a mapping node:

```
<jc:TextMapping Prefix="" Name="FirstName" Optional="false" Separator="">

<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/…/@FirstName"/>

</jc:TextMapping>
```

`XPath="/jdf:JDF/…/@FirstName"` – This is an essential attribute in a mapping node. It specifies the location in the JDF structure where the value to be set in the PRISMAdirect ticket will be taken from. It uses an XPath expression to point to the desired location in the xml structure of the JDF.

It is not in the same place in each type of node, but every mapping node must contain at least one XPath expression.

`Name="FirstName"` – The attribute specifies the name of the ticket item in PRISMAdirect whose value will be changed with the one obtained from the JDF.

`Optional="[true/false]"` – The attribute is related to the interpretation of the mapping node. It specifies if applying the mapping is mandatory or not.

NumberMapping

```
<jc:NumberMapping Name="Copies" Optional="false">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourceLinkPool/…/@Amount"/>
</jc:NumberMapping>
```

The NumberMapping node is a simple node. It contains only the common attributes defined above.

DateMapping

```
<jc:DateMapping Name="Date" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/…/@LastEnd"/>
</jc:DateMapping>
```

The DateMapping node is similar to the number mapping. It contains only the common attributes defined above.

TextMapping

```
<jc:TextMapping Prefix="Address:" Name="Location" Optional="false"
Separator=",">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/…/jdf:Address/@Street"/>
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/…/jdf:Address/@City"/>
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/…/jdf:Address/@PostalCode"/>
</jc:TextMapping>
```

The TextMapping can contain more than one XPath expression.

`Prefix="Address:"` -
The attribute specifies a prefix to be placed in the PRISMAdirect ticket item in front of the value retrieved from JDF.

`Separator=","` – The attribute specifies a character or a string to be used as separator between multiple values taken from the JDF.

EnumMapping

```
<jc:EnumMapping Name="DocumentMediaColor" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/…/@MediaColorName"/>
<jc:EnumValueMapping JdfValue="White" AccessEnumValue="White"/>
<jc:EnumValueMapping JdfValue="Blue" AccessEnumValue="Blue"/>
<jc:EnumValueMapping JdfValue="Red" AccessEnumValue="Red"/>
<jc:EnumValueMapping JdfValue="Green" AccessEnumValue="Green"/>
<jc:EnumValueMapping JdfValue="Yellow"AccessEnumValue="Yellow"/>
<jc:EnumValueMapping JdfValue="Gray"AccessEnumValue="Gray"/>
<jc:EnumValueMapping JdfValue="Orange"AccessEnumValue="Orange"/>
<jc:EnumValueMapping JdfValue="Pink"AccessEnumValue="Pink"/>
<jc:EnumValueMapping JdfValue="Black"AccessEnumValue="Black"/>
```

```
</jc:EnumMapping>
```

This mapping type is usually used for the PRISMAdirect ticket items of type choice which have a list of options defined.

Aside from the common attributes the EnumMapping contains a list of EnumValueMapping nodes each for one of the options of the ticket item in the PRISMAdirect ticket.

`JdfValue="White"` - The attribute refers to the value found in the JDF ticket at the specified XPath.

`AccessEnumValue="White"` - The attribute represents the corresponding option in the ticket item in PRISMAdirect.

ConditionalEnumMapping

```
<jc:ConditionalEnumMapping Name="CoverPlace"Optional="false">

<jc:ConditionalEnumValue AccessEnumValue="None">

<jc:StringCondition JdfField="/jdf:JDF/…/jdf:Media/
@FrontCoatings"ExpectedValue="None"/>

<jc:StringCondition JdfField="/jdf:JDF/…/jdf:Media/
@BackCoatings"ExpectedValue="None"/>

</jc:ConditionalEnu

<jc:ConditionalEnumValue AccessEnumValue="Front">

<jc:StringCondition JdfField="/jdf:JDF/…/jdf:Media/
@FrontCoatings"ExpectedValue="Coated"/>

<jc:StringCondition JdfField="/jdf:JDF/…/jdf:Media/
@BackCoatings"ExpectedValue="None"/>

</jc:ConditionalEn

<jc:ConditionalEnumValue AccessEnumValue="Back"

<jc:StringCondition JdfField="/jdf:JDF/…/jdf:Media/
@FrontCoatings"ExpectedValue="None"/>

<jc:StringCondition JdfField="/jdf:JDF/…/jdf:Media/
@BackCoatings"ExpectedValue="Coated"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="FrontAndBack">

<jc:StringConditionJdfField="/jdf:JDF/…/jdf:Media/
@FrontCoatings"ExpectedValue="Coated"/>

<jc:StringCondition JdfField="/jdf:JDF/…/jdf:Media/
@BackCoatings"ExpectedValue="Coated"/>

</jc:ConditionalEnumValue>

</jc:ConditionalEnumMapping>
```

This mapping type is usually used for the PRISMAdirect ticket items of type choice which have a list of options defined.

The mapping node contains a list of ConditionalEnumValue nodes with an attribute:

`AccessEnumValue="None"` – The attribute refers to one of the options of the choice ticket item in the PRISMAdirect ticket.

It is recommended that the mapping contains one ConditionalEnumValue node for each option in the choice ticket item specified in the Name. But this is not mandatory.

Each ConditionalEnumValue element contains one or more conditions represented by StringCondition or NumericJdfFieldCondition or NumericComparisonCondition nodes (see Conditions below).

BooleanMapping

```
<jc:BooleanMapping Name="Collate" Optional="true" EvaluateTo="false">
<jc:StringCondition JdfField="/jdf:JDF/…/@Collate"ExpectedValue="None"/>
<jc:StringCondition JdfField="/jdf:JDF/
@Types"ContainedValue="DigitalPrinting"/>
</jc:BooleanMapping>
```

This mapping type is used for PRISMAdirect ticket items that have a true/false value. The node can contain one or more condition nodes (see Conditions below).

Additionally it contains a specific attribute:

`EvaluateTo="true/false"` – The attribute specifies the value of the ticket item as a result of the evaluation of the contained condition nodes.

TimeSpanMapping

```
<jc:TimeSpanMapping Name="FinishingTime" Optional="true">
<jc:TimeSpan Start="/jdf:JDF/jdf:AuditPool/…/@Start" End="/jdf:JDF/
jdf:AuditPool/…/@End"/>
</jc:TimeSpanMapping>
```

This type of mapping is used for ticket items that hold a numeric value representing a time span. It contains a TimeSpan node with two attributes:

`Start="/jdf:JDF/jdf:AuditPool/…/@Start"` – The attribute contains an XPath that should point to a date value in the JDF ticket. This is the start date for the time span.

`End="/jdf:JDF/jdf:AuditPool/…/@End"` - The attribute contains an XPath that should point to a date value in the JDF ticket. This is the end date for the time span.

MediaMapping

Two types of nodes are used for mapping the Media attributes. Both are specific nodes and cannot be used for ticket items other than the ones related to Media: **MediaEnumMapping** and **MediaConditionalEnumMapping**.

Examples:

```
<jc:MediaEnumMapping Name="CoverMediaColor" Type="Cover" MediaPartitioner="/
jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams" Optional="true">
<jc:JdfField XPath="/jdf:JDF/…/jdf:Media[@ID='${MediaID}']/@MediaColorName"/>
<jc:EnumValueMapping JdfValue="White" AccessEnumValue="White"/>
<jc:EnumValueMapping JdfValue="Blue" AccessEnumValue="Blue"/>
```

```
...
<jc:EnumValueMapping JdfValue="Black" AccessEnumValue="Black"/>
</jc:MediaEnumMapping>
<jc:MediaConditionalEnumMapping Name="DocumentMediaSize" Type="Content"
MediaPartitioner="/jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams"
Optional="true">
<jc:ConditionalEnumValue AccessEnumValue="A0">
<jc:NumericCondition ExpectedValue="2383.9344"JdfField="/jdf:JDF/…/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
<jc:NumericCondition ExpectedValue="3370.392"JdfField="/jdf:JDF/…/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="A0">
<jc:NumericCondition ExpectedValue="2383.9344"JdfField="/jdf:JDF/…/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
<jc:NumericCondition ExpectedValue="3370.392"JdfField="/jdf:JDF/…/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
</jc:ConditionalEnumValue>
...
</jc:MediaConditionalEnumMapping>
```

Both mapping structures build on existing mapping types with some additional attributes:

`Type="Cover/Content"` – A JDF ticket can defined multiple Media nodes, some to be used for the Content of the document to be printed, some for the Cover. The Type attribute specifies if the Media referred by the JdfField node should be the Media used for Cover or Content.

`MediaPartitioner="/jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams"` - A document can be printed on multiple types of Media. This attribute points to the location in the JDF ticket that specifies which media is used for which parts of the document.

The MediaEnumMapping builds on top of the EnumMapping type and has the same structure (see EnumMapping above).

The MediaConditionalEnumMapping builds on top of the ConditionalEnumMapping type and has the same structure (see ConditionalEnumMapping above).

Conditions

Additionally every type of mapping node can contain one or more condition nodes. The conditions can be: StringCondition, NumericJdfFieldCondition or NumericComparisonCondition.

StringCondition and NumericJdfFieldCondition both have a similar structure:

`JdfField="/jdf:JDF/…/jdf:Media/@FrontCoatings"` - The attribute has the same meaning as the one defined in the Common Fields section. It refers to a location in the JDF ticket where a value should be taken from.

`ExpectedValue="None"` - The attribute refers to the expected value to be found in the JDF at the XPath specified by the JdfField attribute.

`ContainedValue="Front"` – The attribute can be used instead of the ExpectedValue attribute. In this case it refers to the value that should be contained in the JDF at the XPath specified by the JdfField attribute.

Both StringCondition and NumericJdfFieldCondition nodes can be used without an ExpectedValue or a ContainedValue attribute.

```
<jc:NumericComparisonCondition Value_1="/jdf:JDF/…/@Dimension[0]"Value_2="/
jdf:JDF/…/@Dimension[1]" Comparison="LessThan"/>
```

The condition defines two values to be compared and the expected relation between them.

`Value_1="/jdf:JDF/…/@Dimension[0]"`

`Value_2="/jdf:JDF/…/@Dimension[1]"` – The attributes define the two values that will be compared. The XPath must point to a numeric value in the JDF ticket.

`Comparison="LessThan"` - The attribute specifies the expected relation between the two values. The possible values for this attribute are limited to: **LessThan, GreaterThan** and **Equal**.

Interpretation logic

As mentioned before, each JMF endpoint defined in the context of the JDF Server will have a JDF-to-PD mapping file attached. At startup of the JDF server this mapping file will be loaded from disk and kept in-memory as long as the JDF server is up and running.

When the JDF client submits a new JDF ticket (+ Data files) to the JDF server a specified mapper module will use the mapping table loaded at startup to transfer values from the JDF ticket into a default PD ticket. If all the mappings are applied correctly, the outcome should be a PD ticket containing all the ticket specifications, as if the customer had filled-in the PD ticket directly.

General rules for the mapping process:

- The module attempts to apply each mapping node in the table.
- If the mapping is applied successfully it moves on to the next node
- If applying the mapping fails and the mapping node is Optional="false" the mapping process stops and the entire submission attempt is considered failed.
- If applying the mapping fails and the mapping node is Optional="true" it is skipped and the process continues with the next mapping node in the list.
- If all the mappings or at least the ones that have Optional="false" are applied correctly, the process is considered successful and the outcome should be a customized PD ticket.

Rules for applying the different types of mapping nodes:

NumberMapping

```
<jc:NumberMapping Name="Copies" Optional="false">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourceLinkPool/…/@Amount"/>
</jc:NumberMapping>
```

The system retrieves the value at the specified XPath in the JDF ticket xml structure. In the PD ticket the Copies item gets the retrieved value.

Limitation: the retrieved value is applied in the PD ticket only if it is valid in the context of PD. For example the Copies ticket item being a number item can never be set with “John Doe” but only with a numeric value. Also the item can have constraints on the minimum and maximum acceptable value so again, if the value from the JDF is outside of these bounds, it cannot be set in the PD ticket.

DateMapping

```
<jc:DateMapping Name="Date" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/…/@LastEnd"/>
</jc:DateMapping>
```

The system retrieves the value specified by the XPath. The system then tries to convert the string obtained from the JDF ticket into a valid Date object. In the PD ticket the Date item will get the serialized Date object as value.

TextMapping

```
<jc:TextMapping Prefix="" Name="FirstName" Optional="false" Separator="">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/…/jdf:Person/@FirstName"/>
</jc:TextMapping>
```

The system retrieves the value at the specified XPath in the JDF ticket xml structure. In the PD ticket the FirstName item gets the retrieved value.

Limitation: the retrieved value is applied in the PD ticket only if it is valid in the context of PD.

For example the FirstName ticket item being a text item can have a limited length for the text. So if the text is longer than the limitation the value cannot be set.

EnumMapping

```
<jc:EnumMapping Name="DocumentMediaWeight" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/…/@Weight"/>
<jc:EnumValueMapping JdfValue="65"AccessEnumValue="65032gram047m2"/>
<jc:EnumValueMapping JdfValue="80"AccessEnumValue="80032gram047m2"/>
<jc:EnumValueMapping JdfValue="90"AccessEnumValue="90032gram047m2"/>
<jc:EnumValueMapping JdfValue="100"AccessEnumValue="100032gram047m2"/>
<jc:EnumValueMapping JdfValue="120"AccessEnumValue="120032gram047m2"/>
<jc:EnumValueMapping JdfValue="160"AccessEnumValue="160032gram047m2"/>
<jc:EnumValueMapping JdfValue="170"AccessEnumValue="170032gram047m2"/>
</jc:EnumMapping>
```

The system retrieves the value at the specified XPath in the JDF ticket xml structure. Next the system parses all the EnumValueMapping nodes trying to find the one with the JdfValue equal to the value taken from the JDF ticket.

If the node is found, the value of the AccessEnumValue attribute is set in the PD ticket for the ticket item specified by Name.

Example:

The value retrieved from the JDF ticket is ="100". The system will set the value of the DocumentMediaWeight ticket item to "100032gram047m2".

Limitation: the retrieved value is applied in the PD ticket only if it is valid in the context of PD. In this case the list of options defined for the choice item in PD must include the one retrieved from the JDF.

ConditionalEnumMapping

```
<jc:ConditionalEnumMapping Name="BindingMethod" Optional="true">
<jc:ConditionalEnumValue AccessEnumValue="SaddleStitch">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@StitchType"ExpectedValue="Saddle"/>
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@NumberOfStitches" ExpectedValue="2"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Staples_1">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@NumberOfStitches" ExpectedValue="1"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Staples_2">
```

```
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@NumberOfStitches" ExpectedValue="2"/>

</jc:ConditionalEnumValue>

....

</jc:ConditionalEnumMapping
```

The system parses the ConditionalEnumValue nodes. For each node it evaluates the conditions in the NumericJdfFieldCondition or StringCondition or NumericComparisonCondition sub-nodes (see Condition evaluation below). If all the conditions in one ConditionalEnumValue node are met, the parsing stops and the system attempts to set the value of the PD ticket item with the value of the corresponding AccessEnumValue attribute.

BooleanMapping

```
<jc:BooleanMapping Name="Collate" Optional="true" EvaluateTo="false">

<jc:StringCondition JdfField="/jdf:JDF/…/@Collate"ExpectedValue="None"/>

<jc:StringCondition JdfField="/jdf:JDF/
@Types"ContainedValue="DigitalPrinting"/>

</jc:BooleanMapping>
```

The system evaluates all the contained condition nodes (see Conditions evaluation). If all the conditions are met, the EvaluateTo attribute specifies what value to give to the ticket item.

In the case of the above mapping node, if both string conditions are met, the Collate ticket item will get the value “false”, otherwise the ticket item gets the value “”.

TimeSpanMapping

```
<jc:TimeSpanMapping Name="FinishingTime" Optional="true">

<jc:TimeSpan Start="/jdf:JDF/jdf:AuditPool/…/@Start"End="/jdf:JDF/
jdf:AuditPool/…/@End"/>

</jc:TimeSpanMapping>
```

The Start and the End attributes should refer to values representing a point in time – a date and a time of day. The system will load both values and it will compute the difference in time between the two. The result will be a span of time which will be set as the value of the PD ticket item (in this case FinishingTime).

MediaMapping (MediaEnumMapping, MediaConditionalEnumMapping)

The JDF ticket can contain multiple Media definitions, for all the types of Media used to print the document.

The system first tries to identify the Media used for “Cover” or “Content” as specified by the Type attribute. For this the location specified by the MediaPartitioner node is used to analyze which pages use which Media. As a basic rule, the Media used for the page number “0” or “-1” or the page with the highest number (the last page) will be identified as the Media for “Cover”.

After the Media has been identified, the ID of the Media is used to retrieve the needed attributes from the specified location in the JDF.

Example:

MediaEnumMapping is a specialized version of the EnumMapping type.

```
<jc:MediaEnumMapping Name="CoverMediaColor" Type="Cover"MediaPartitioner="/
jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams"Optional="true">

<jc:JdfField XPath="/jdf:JDF/…/jdf:Media[@ID='${MediaID}']/@MediaColorName"/>

<jc:EnumValueMapping JdfValue="White" AccessEnumValue="White"/>

<jc:EnumValueMapping JdfValue="Blue" AccessEnumValue="Blue"/>

...

<jc:EnumValueMapping JdfValue="Black" AccessEnumValue="Black"/>

</jc:MediaEnumMapping>
```

The system tries to identify the Media to be used for Cover (Type="Cover") by looking in the partitions node (MediaPartitioner="/jdf:JDF/…/jdf:DigitalPrintingParams").
Next the XPath in the JdfField node is updated with the ID of the chosen Media ([@ID='${MediaID}']).

Finally, the system proceeds to apply the mapping as a regular EnumMapping type.

Conditions evaluation

Rules for evaluating the conditions:

1. String condition
   - Evaluating the condition is done by comparing the value retrieved from the JDF ticket at th e XPath specified by JdfField to the ExpectedValue attribute.
   - If the node has a ContainedValue attribute instead, the condition is met if the value of the attribute is contained in the value retrieved from the JDF ticket by JdfField.
   - If the node does not contain an ExpectedValue or a ContainedValue attribute then the condition is met if the XPath specified by JdfField exists in the JDF ticket.
     Example:
     `<jc:StringConditionJdfField="/jdf:JDF/jdf:ResourcePool/jdf:GluingParams/@Status"ExpectedValue="Available"/>`
     For the StringCondition:
     IF The value in the JDF ticket at "`/jdf:JDF/jdf:ResourcePool/jdf:GluingParams/@Status`" is “Available”
     THEN The condition is met.
     ELSE The condition is not met.
2. Numeric condition
   `<jc:NumericCondition ExpectedValue="2383.9344"JdfField="/jdf:JDF/jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>`
   - The value retrieved from the JDF at the XPath specified by the JdfField attribute must be numeric. The numeric value is then compared to the ExpectedValue attribute. The condition is met if the values are equal with an accepted variation of 1.
3. Numeric comparison condition
   `<jc:NumericComparisonCondition Value_1="/jdf:JDF/jdf:ResourcePool/jdf:Media/@Dimension[0]" Value_2="/jdf:JDF/jdf:ResourcePool/jdf:Media/@Dimension[1]" Comparison="LessThan"/>`
   - The system retrieves the numeric value indicated by the Value_1 and Value_2 attributes.
   - The condition is met if the numeric relation between the two values corresponds to the one specified by the Comparison attribute.
     For the example above the evaluation can be read as:
     IF `Value_1 LessThan Value_2`
     THEN Condition is met
     ELSE Condition is not met.

Mapping file customization

The system offers a default mapping table that covers the relevant ticket items in the default ticket definition. The default mapping table contains one mapping node for one PD ticket item. This assures that in the PD job ticket all the relevant ticket items can be set by the submitter indirectly, via the JDF ticket.

The customer can add new ticket items to the ticket definition by using the Job Ticket Editor. By default these items will not be impacted since they are not targeted by the mapping file. There are two ways to cover these new fields.

Generic extension mapping

JDF permits adding of generic key-value pairs in a specific structure in the JDF ticket:

```
<JDF ID="NodeIDRoot" Type="ProcessGroup" Types="...
DigitalPrinting ..."xmlns="http://www.CIP4.org/JDFSchema_1_1"
Status="Waiting"xmlns:oce="http://www.oce.com/JDF_Extension/1_00" >

<oce:KeyValuePair Key="Key1">Value 1</oce:KeyValuePair>

<oce:KeyValuePair Key="Key2">Hello, World!</oce:KeyValuePair>...</JDF>
```

This is not a modification of the mapping file but a possible generic extension of the JDF ticket to support custom user defined ticket fields.

After mapping all the items defined in the mapping table, the system performs an additional step. It searches for all the oce:KeyValuePair nodes in the JDF ticket. The Key attribute specifies the name of the item in the PD ticket and the inner content of the node becomes the value for this item.

Example:

The default ticket definition contains a series of items related to user profile information (eg: FirstName, LastName, Department, Location, TelephoneNumber, EmailAddress, etc).

Suppose the PrintShop where PRISMAdirect is installed wants to stay connected to its customers via a social network (facebook, google+). The customer wants to extend the user information with the FaceBook account. For this he will add a new item in the PD ticket called FacebookAccount from JTE.

Additionally, the JDF tickets that are sent to the JDF/JMF interface of PRISMAdirect must contain a new node:

```
<oce:KeyValuePair Key="FacebookAccount">John.Doe</oce:KeyValuePair>
```

The job ticket in PRISMAdirect will have the FacebookAccount field set with the value “John.Doe”.

Add a new rule

Another way to solve the problem of user defined ticket items is by modifying the mapping table and including mapping nodes for these items also. The mapping mechanism is driven by the mapping table, so a new mapping node for a new item means that at submission the value for that item will be automatically taken from the JDF if possible.

Example:

If a customer wants to allow job submitters to specify the output type for a printed job, he can add a new ticket item from JTE called OutputType. It is created as a choice item and has the following options: Finisher, Tray, Booklet_Maker, Perfect_Binder.

Once the ticket definition is updated, the mapping file can be extended to cover also this item.

The following node can be added to the mapping xml:

```
<jc:EnumMapping Name="OutputType" Optional="true">

<jc:JdfFieldXPath="/jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams/
@OutputBin"/>

<jc:EnumValueMapping JdfValue="Finisher" AccessEnumValue="Finisher"/>

<jc:EnumValueMapping JdfValue="Tray" AccessEnumValue="Tray"/>

<jc:EnumValueMapping JdfValue="BookletMaker"
AccessEnumValue="Booklet_Maker"/>

<jc:EnumValueMapping
JdfValue="PerfectBinder"AccessEnumValue="Perfect_Binder"/>

</jc:EnumMapping>
```

Now the mapping file is updated. Let's say the next job that is submitted to PRISMAdirect contains the following fragment in the attached JDF ticket:

```
<JDF Types="... DigitalPrinting ..." ...>

<ResourcePool>

<DigitalPrintingParams ID="NodeIDDPP" Class="Parameter"Status="Available"
OutputBin="PerfectBinder" ...>

</ResourcePool> ... ourceLinkPool>

<DigitalPrintingParamsLink rRef="NodeIDDPP" Usage="Input" />

...

</ResourceLinkPool> </JDF>
```

After the mapping process is complete, in the new job created the item OutputType will have the value "Perfect_Binder".

Change an existing rule

Different JDF-enabled applications can store the information in the JDF ticket in different ways. If two or more of such applications want to submit jobs to PRISMAdirect, the system must be able to understand both JDF ticket formats.

For this it is possible to define two JMF endpoints one for each JDF Client app. Each JMF endpoint will use a separate mapping file which can be customized to support a different structure of the JDF ticket.

Example:

A customer has extended the PD ticket with a ticket field allowing the job submitters to ask for a special finishing: SquareBinding. There is an online finisher on some iPR printers supporting this, called “Saddle press”.

The Océ JDF syntax and the Kodak syntax for this finishing are different. Suppose the customer needs to support Océ and Kodak applications and he needs to map both syntaxes.

Two endpoints must be enabled for this each with a customized mapping table.

1. This is the Océ syntax:

```
<JDF ID="jdf_1" Type="Combined" Category="DigitalPrinting"
Types="LayoutPreparation Imposition Interpreting Rendering DigitalPrinting
Stitching">
<ResourcePool>
<Component ID="Product_4" Class="Quantity" Status="Unavailable"
ComponentType="PartialProduct"/>
<StitchingParams ID="ProcessID_5" Class="Parameter" Status="Available"
StitchType="Saddle" NumberOfStitches="2">
<GeneralID IDUsage="oce:SaddlePress" IDValue="On"/>
<GeneralID IDUsage="oce:SaddlePressAdjustment" IDValue="+5"/>
</StitchingParams>
<Component ID="Product_5" Class="Quantity" Status="Unavailable"
ComponentType="PartialProduct"/>
</ResourcePool>
<ResourceLinkPool>
<ComponentLink rRef="Product_4" Usage="Input" Orientation="Flip0"
CombinedProcessIndex="5"/>
<StitchingParamsLink rRef="ProcessID_5" Usage="Input"/>
<ComponentLink rRef="Product_5" Usage="Output" Orientation="Flip0"
CombinedProcessIndex="5"/>
</ResourceLinkPool> </JDF>
```

   The mapping node that will set the value for SquareBinding should look like this:

```
<jc:ConditionalEnumMapping Name="BindingMethod" Optional="true">
<jc:ConditionalEnumValue AccessEnumValue="SquareBinding">
<jc:StringConditionJdfField="/jdf:JDF/jdf:ResourcePool/
jdf:StitchingParams/jdf:GeneralID[@IDUsage='oce:SaddlePress']/@IDValue"
ExpectedValue="On"/>
</jc:ConditionalEnumValue>
...
</jc:ConditionalEnumMapping>
```

2. The Kodak syntax for the same finishing type is quite different:

```
<JDF Activation="Active" Category="DigitalPrinting"CommentURL="http://
127.0.0.1"ICSVersions="IDP_L1-1.0 Base_L3-1.0"
ID="n0803_4792_00074"JobID="001"JobPartID="080623_105104792_000175"
MaxVersion="1.3"SettingsPolicy="BestEffort"Status="Ready"
```

```
Type="Combined"Types="LayoutPreparation Imposition Interpreting
RenderingColorantControl DigitalPrinting Folding Trimming pod-
wf:SaddlePress"Version="1.3"xmlns="http://www.CIP4.org/
JDFSchema_1_1"xmlns:pod-wf="http://www.pod-wf.com"xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"xsi:type="Combined">
<Comment/>
<ResourcePool>
<pod-wf:SaddlePressParams ID="SPP001" Class="Parameter"NoOp="false" pod-
wf:PressLevel="-5" Status="Available"/>
...
</ResourcePool>
<ResourceLinkPool>
...
</ResourceLinkPool>
</JDF>
```

In the mapping table for Kodak, the mapping node must be changed:

```
<jc:ConditionalEnumMapping Name="BindingMethod" Optional="true">
<jc:ConditionalEnumValue AccessEnumValue="SquareBinding">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/pod-
wf:SaddlePressParams/@Status" ExpectedValue="Available"/>
</jc:ConditionalEnumValue>
...
</jc:ConditionalEnumMapping>
```

By enabling the two endpoints and assigning a custom mapping table to each one, the system is now configured to accept and interpret messages from the Océ and Kodak apps simultaneously.

PRISMAdirect to JDF: mapping for job status

File structure

This mapping file is used to associate PRISMAdirect specific values for job status to custom values defined by the customer. Job status in PRISMAdirect is represented by two items: **InternalJobState** and **CommunicationState**. The options of both these items will be exposed to the JDF Client.

Sample of the default PD-to-JDF mapping file:

NOTE
The default mapping table maps the values of the status items to themselves.

```
<Mappings>
<CommunicationState>
<StatusMapping PAValue="Incoming">Incoming</StatusMapping>
<StatusMapping PAValue="New">New</StatusMapping>
<StatusMapping PAValue="Accepted">Accepted</StatusMapping>
<StatusMapping PAValue="Resubmitted">Resubmitted</StatusMapping>
...
<StatusMapping PAValue="Ready">Ready</StatusMapping>
<StatusMapping PAValue="Closed">Closed</StatusMapping>
<StatusMapping PAValue="Failed">Failed</StatusMapping>
</CommunicationState>
<InternalJobState>
<StatusMapping PAValue="Uploading">Uploading</StatusMapping>
<StatusMapping PAValue="Uploaded">Uploaded</StatusMapping>
<StatusMapping PAValue="Edited">Edited</StatusMapping>
...
<StatusMapping PAValue="Exported">Exported</StatusMapping>
<StatusMapping PAValue="Closing">Closing</StatusMapping>
<StatusMapping PAValue="Closed">Closed</StatusMapping>
</InternalJobState>
</Mappings>
```

The file is made up of two sections, one for InternalJobState values and one for CommunicationState values. Each section consists in a list of StatusMapping nodes.

Each StatusMapping node associates the status value specified by the PAValue attribute to a free string customizable by the customer.

Interpretation Logic

At the start-up of the JDF/JMF Server, for all the enabled JMF endpoints the PD-to-JDF mapping table is loaded in memory from a file on disk.

When the JDF Server attempts to send the status of the job to the JDF Client, it first retrieves the values of InternalJobState and CommunicationState from the PRISMAdirect ticket of that job.

The mapper searches for the StatusMapping node whose PAValue attribute matches the status from the job. It does this both for InternalJobState and for CommunicationState in their respective sections in the XML.

When the matching StatusMapping node is found, the mapper returns the inner content of the node to the caller.

If the match is not found, the mapper returns the status value.

Mapping file customization

The customer will be able to change the values to which any job status value is mapped. In other words he can change the inner content for any and all of the StatusMapping nodes in the mapping file.

Example:

In the vocabulary of the customer a new job that arrives in the system might get the status “Unprocessed”. The analogue status in PRISMAdirect would be “New”. When the JDF Client gets the status of the job the customer might want that the job is presented as “Unprocessed” and not “New”. For this he simply has to update the inner content of the corresponding XML node to:

```
<StatusMapping PAValue="New">Unprocessed</StatusMapping>
```

Another use case would be the need to display the status of the job in another language other than English. This can be easily done for all the status values. The mapping file for a JDF Client that uses German as culture could look like this:

```
<Mappings>

<CommunicationState>

<StatusMapping PAValue="Incoming">Eingehend</StatusMapping>

<StatusMapping PAValue="New">Neu</StatusMapping>

<StatusMapping PAValue="Accepted">Akzeptiert</StatusMapping>

...

StatusMapping PAValue="Ready">Bereit</StatusMapping>

<StatusMapping PAValue="Closed">Geschlossen</StatusMapping>

<StatusMapping PAValue="Failed">Gescheitert</StatusMapping>

</CommunicationState>

...

</Mappings>
```

Of course this only applies to one JMF endpoint. A second endpoint could have the mapping file changed to return values in another language. However PRISMAdirect does not provide translations so modifying the mapping file is the responsibility of the customer.

InternalJobState and CommunicationState are internal ticket items of PRISMAdirect. So the list of options cannot be changed by the user during runtime. Therefore extending the mapping file after installation is not applicable in this case.

JMF Support

Through the JDF/JMF interface the system will expose on request the status of the submitted jobs to the JDF Clients. The JMF message that will contain the status is a response to the QueueStatus query:

```
<JMF TimeStamp="2006-09-29T13:18:01Z" SenderID="JMFSender"xmlns="http://
www.CIP4.org/JDFSchema_1_1"xmlns:xsi=http://www.w3.org/2001/XMLSchema-
instance Version="1.3">

<Response Type="QueueStatus"
ReturnCode="0"xsi:Type="ResponseQueueStatus"ID="Response-00000006"
refID="qStatus_123">

<Queue Status="Waiting" DeviceID="MainQueue" />

<QueueEntry Status="Running" QueueEntryID="0007-43a32dd5-e9e4-
43e4-9b1d-89afffdaab0" JobID="JDF_66988942-9aef-4dcc-84f7-
db35531270a4"JobPartID="PrintNode">

<JobPhase Status="InProgress" StatusDetails="New|Uploaded" />

...

</QueueEntry>

</Response>

</JMF>
```

The StatusDetails attribute will contain the values of both **CommunicationState** and **InternalJobState** concatenated in this order.

Debugging help

During both mapping processes JDF-to-PD and PD-to-JDF any mapping that cannot be applied will be logged in a file to disk. The log entry will contain the ID of the job being processed and the reason that the mapping could not be applied.

Default submission mappings

User information

Account

```
<jc:TextMapping Prefix="" Name="Account" Optional="true" Separator="">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:CustomerInfo/
@BillingCode"/>
</jc:TextMapping>
```

Company

```
<jc:TextMapping Prefix="" Name="Company" Optional="false" Separator="">
<jc:JdfFieldXPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Customer']/jdf:Company/@OrganizationName"/>
</jc:TextMapping>
```

Department

```
<jc:JdfFieldXPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Customer']/jdf:Company/jdf:OrganizationalUnit"/>
</jc:TextMapping>
```

EmailAddress

```
<jc:JdfFieldXPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Customer']/jdf:ComChannel[@ChannelType='Email']/@L
ocator"/>
</jc:TextMapping>
```

FirstName

```
<jc:JdfFieldXPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Customer']/jdf:Person/@FirstName"/>
</jc:TextMapping>
```

LastName

```
<jc:TextMapping Prefix="" Name="LastName" Optional="false" Separator="">
<jc:JdfFieldXPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Customer']/jdf:Person/@FamilyName"/>
</jc:TextMapping>
```

Location

```
<jc:TextMapping Prefix="" Name="Location" Optional="false" Separator="">
<jc:JdfFieldXPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Customer']/jdf:Address/@Street"/>
```

```
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Customer']/jdf:Address/@City"/>

<jc:JdfFieldXPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Customer']/jdf:Address/@PostalCode"/>

</jc:TextMapping>
```

TelephoneNumber

```
<jc:JdfFieldXPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Customer']/jdf:ComChannel[@ChannelType='Phone']/@L
ocator"/>

</jc:TextMapping>
```

UserID

```
<jc:TextMapping Prefix="" Name="UserId" Optional="true" Separator="_">

<jc:JdfFieldXPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Customer']/@*[name()='oce:UserID']"/>

</jc:TextMapping>
```

Finishing

BindingMethod

```
<jc:ConditionalEnumMapping Name="BindingMethod" Optional="true">
<jc:ConditionalEnumValue AccessEnumValue="SaddleStitch">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@StitchType" ExpectedValue="Saddle"/>
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@NumberOfStitches" ExpectedValue="2"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Staples_1">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@NumberOfStitches" ExpectedValue="1"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Staples_2">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@NumberOfStitches" ExpectedValue="2"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Staples_3">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@StitchType" ExpectedValue="Saddle"/>
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@NumberOfStitches" ExpectedValue="3"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Staples_4">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@StitchType" ExpectedValue="Saddle"/>
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@NumberOfStitches" ExpectedValue="4"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Staples_5">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@StitchType" ExpectedValue="Saddle"/>
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@NumberOfStitches" ExpectedValue="5"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Staples_6">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@StitchType" ExpectedValue="Saddle"/>
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@NumberOfStitches" ExpectedValue="6"/>
```

```
</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="WireO">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:WireCombBindingParams/@Status" ExpectedValue="Available"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="GlueBinding">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:GluingParams/
@Status" ExpectedValue="Available"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Booklet">

<jc:StringCondition JdfField="/jdf:JDF/@Types"
ContainedValue="LayoutPreparation"/>

<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="Imposition"/>

<jc:StringConditionJdfField="/jdf:JDF/jdf:ResourcePool/
jdf:LayoutPreparationParams/@PresentationDirection"
ExpectedValue="FoldCatalog"/>

<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="Folding"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:LayoutPreparationParams/@FoldCatalog"ExpectedValue="F4-1"/>

<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="Stitching"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:FoldingParams/
@FoldCatalog" ExpectedValue="F4-1"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@StitchType"ExpectedValue="Saddle"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="PerfectBound">

<jc:StringCondition JdfField="/jdf:JDF/@Types"
ContainedValue="CoverApplication"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:CoverApplicationParams/@Status" ExpectedValue="Available"/>

</jc:ConditionalEnumValue>

</jc:ConditionalEnumMapping>
```

Booklet

```
<jc:BooleanMapping Name="Booklet" Optional="true">

<jc:StringCondition JdfField="/jdf:JDF/@Types"
ContainedValue="LayoutPreparation"/>

<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="Imposition"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:LayoutPreparationParams/@PresentationDirection"
ExpectedValue="FoldCatalog"/>
```

```
<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="Folding"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:LayoutPreparationParams/@FoldCatalog" ExpectedValue="F4-1"/>

<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="Stitching"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:FoldingParams/
@FoldCatalog" ExpectedValue="F4-1"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:StitchingParams/
@StitchType" ExpectedValue="Saddle"/>

</jc:BooleanMapping>
```

BookletPageSize

```
<jc:EnumMapping Name="BookletPageSize" Optional="true">

<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:LayoutPreparationParams/
jdf:PageCell/jdf:FitPolicy/@SizePolicy"/>

<jc:EnumValueMapping JdfValue="ClipToMaxPage" AccessEnumValue="Keep"/>

<jc:EnumValueMapping JdfValue="ReduceToFit" AccessEnumValue="Reduce"/>

</jc:EnumMapping>
```

Collate

```
<jc:BooleanMapping Name="Collate" Optional="true" >

<jc:FalseCondition ExpectedValue="None" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:DigitalPrintingParams/@Collate"/>

<jc:StringCondition JdfField="/jdf:JDF/@Types"
ContainedValue="DigitalPrinting"/>

</jc:BooleanMapping>
```

CoverPlace

```
<jc:EnumMapping Name="CoverPlace" Optional="true">

<jc:StringCondition JdfField="/jdf:JDF/@Types"
ContainedValue="DigitalPrinting"/>

<jc:JdfField XPath="/jdf:JDF/jdf:Comment[@Name='oce:Covers']" />

<jc:EnumValueMapping JdfValue="Front" AccessEnumValue="Front"/>

<jc:EnumValueMapping JdfValue="Back" AccessEnumValue="Back"/>

<jc:EnumValueMapping JdfValue="FrontBack" AccessEnumValue="FrontAndBack"/>

</jc:EnumMapping>
```

CoverRemarks

```
<jc:MediaMapping Name="CoverRemarks" Type="Cover" Optional="true">

<jc:MediaParitioner XPath="/jdf:JDF/jdf:ResourcePool/
jdf:DigitalPrintingParams"/>

<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:Media/@Comment"/>
```

```
</jc:MediaMapping>
```

Folding

```
<jc:EnumMapping Name="Folding" Optional="true">
<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="Folding"/>
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:FoldingParams/
@FoldCatalog"/>
<jc:EnumValueMapping JdfValue="F4-1" AccessEnumValue="HalfFold"/>
<jc:EnumValueMapping JdfValue="F6-1" AccessEnumValue="TriFoldOut"/>
<jc:EnumValueMapping JdfValue="F6-4" AccessEnumValue="TriFoldIn"/>
<jc:EnumValueMapping JdfValue="F8-2" AccessEnumValue="ParallelFold"/>
<jc:EnumValueMapping JdfValue="F8-4" AccessEnumValue="GateFold"/>
<jc:EnumValueMapping JdfValue="F6-6" AccessEnumValue="ZFold"/>
<jc:EnumValueMapping JdfValue="F6-7" AccessEnumValue="ZFold"/>
<jc:EnumValueMapping JdfValue="F6-3" AccessEnumValue="SimpleGateFold"/>
</jc:EnumMapping>
```

OrientationAndBindingEdge

```
<jc:ConditionalEnumMapping Name="OrientationAndBindingEdge" Optional="true">
<jc:ConditionalEnumValue AccessEnumValue="Portrait_LeftBinding">
<jc:NumericComparisonCondition Value_1="/jdf:JDF/jdf:ResourcePool/jdf:Media/
@Dimension[0]" Value_2="/jdf:JDF/jdf:ResourcePool/jdf:Media/@Dimension[1]"
Comparison="LessThan"/>
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourceLinkPool/
jdf:ComponentLink/@Orientation" ExpectedValue="Rotate0"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Portrait_LeftBinding">
<jc:NumericComparisonCondition Value_1="/jdf:JDF/jdf:ResourcePool/jdf:Media/
@Dimension[0]" Value_2="/jdf:JDF/jdf:ResourcePool/jdf:Media/@Dimension[1]"
Comparison="LessThan"/>
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourceLinkPool/
jdf:ComponentLink/@Orientation" ExpectedValue="Flip0"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Landscape_LeftBinding">
<jc:NumericComparisonCondition Value_1="/jdf:JDF/jdf:ResourcePool/jdf:Media/
@Dimension[0]" Value_2="/jdf:JDF/jdf:ResourcePool/jdf:Media/@Dimension[1]"
Comparison="GreaterThan"/>
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourceLinkPool/
jdf:ComponentLink/@Orientation" ExpectedValue="Rotate0"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Landscape_LeftBinding">
```

```
<jc:NumericComparisonCondition Value_1="/jdf:JDF/jdf:ResourcePool/jdf:Media/
@Dimension[0]" Value_2="/jdf:JDF/jdf:ResourcePool/jdf:Media/@Dimension[1]"
Comparison="GreaterThan"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourceLinkPool/
jdf:ComponentLink/@Orientation" ExpectedValue="Flip0"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Portrait_TopBinding">

<jc:NumericComparisonCondition Value_1="/jdf:JDF/jdf:ResourcePool/jdf:Media/
@Dimension[0]" Value_2="/jdf:JDF/jdf:ResourcePool/jdf:Media/@Dimension[1]"
Comparison="LessThan"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourceLinkPool/
jdf:ComponentLink/@Orientation" ExpectedValue="Rotate90"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Portrait_TopBinding">

<jc:NumericComparisonCondition Value_1="/jdf:JDF/jdf:ResourcePool/jdf:Media/
@Dimension[0]" Value_2="/jdf:JDF/jdf:ResourcePool/jdf:Media/@Dimension[1]"
Comparison="LessThan"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourceLinkPool/
jdf:ComponentLink/@Orientation" ExpectedValue="Flip90"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Landscape_TopBinding">

<jc:NumericComparisonCondition Value_1="/jdf:JDF/jdf:ResourcePool/jdf:Media/
@Dimension[0]" Value_2="/jdf:JDF/jdf:ResourcePool/jdf:Media/@Dimension[1]"
Comparison="GreaterThan"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourceLinkPool/
jdf:ComponentLink/@Orientation" ExpectedValue="Flip90"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Landscape_TopBinding">

<jc:NumericComparisonCondition Value_1="/jdf:JDF/jdf:ResourcePool/jdf:Media/
@Dimension[0]" Value_2="/jdf:JDF/jdf:ResourcePool/jdf:Media/@Dimension[1]"
Comparison="GreaterThan"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourceLinkPool/
jdf:ComponentLink/@Orientation" ExpectedValue="Flip90"/>

</jc:ConditionalEnumValue>

</jc:ConditionalEnumMapping>
```

Output

```
<jc:ConditionalEnumMapping Name="Output" Optional="true">

<jc:ConditionalEnumValue AccessEnumValue="Stacker">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:DigitalPrintingParams/@OutputBin" ContainedValue="Stacker"/>

</jc:ConditionalEnumValue>
```

```
<jc:ConditionalEnumValue AccessEnumValue="Upperbin">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:DigitalPrintingParams/@OutputBin" ExpectedValue="Top"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="HCSExternalFinisher">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:DigitalPrintingParams/@OutputBin" ContainedValue="ExternalFinisher"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="booklet">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:DigitalPrintingParams/@OutputBin" ContainedValue="BookletMaker"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="BookletmakerExternalFinisher">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:DigitalPrintingParams/@OutputBin" ContainedValue="BookletMaker"/>
</jc:ConditionalEnumValue>
</jc:ConditionalEnumMapping>
```

Punching

```
<jc:ConditionalEnumMapping Name="Punching" Optional="true">
<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="HoleMaking"/>
<jc:ConditionalEnumValue AccessEnumValue="None">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="None"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Holes_2">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="R2-generic"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Holes_3">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="R3-generic"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Holes_4">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="R4-generic"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Holes_5">
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="R5-generic"/>
```

```
</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Holes_7">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="R7-generic"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Holes_11">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="R11m-7h4s"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Holes_23">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="W2_1i-round-0t"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Holes_12">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="S-generic"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@*[name()='oce:NumberOfHoles']" ExpectedValue="12"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Holes_19">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="S-generic"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@*[name()='oce:NumberOfHoles']" ExpectedValue="19"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Holes_20">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="S-generic"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@*[name()='oce:NumberOfHoles']" ExpectedValue="20"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Holes_21">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="S-generic"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@*[name()='oce:NumberOfHoles']" ExpectedValue="21"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Holes_32">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="S-generic"/>
```

```
<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@*[name()='oce:NumberOfHoles']" ExpectedValue="32"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Holes_34">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="S-generic"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@*[name()='oce:NumberOfHoles']" ExpectedValue="34"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Holes_44">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="S-generic"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@*[name()='oce:NumberOfHoles']" ExpectedValue="44"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Holes_47">

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@HoleType" ExpectedValue="S-generic"/>

<jc:StringCondition JdfField="/jdf:JDF/jdf:ResourcePool/jdf:HoleMakingParams/
@*[name()='oce:NumberOfHoles']" ExpectedValue="47"/>

</jc:ConditionalEnumValue>

</jc:ConditionalEnumMapping>
```

Media definition

CoverMediaColor

```
<jc:MediaEnumMapping Name="CoverMediaColor" Type="Cover" MediaPartitioner="/
jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams" Optional="true">

<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/
@MediaColorName"/>

<jc:EnumValueMapping JdfValue="White" AccessEnumValue="White"/>

<jc:EnumValueMapping JdfValue="Blue" AccessEnumValue="Blue"/>

<jc:EnumValueMapping JdfValue="Red" AccessEnumValue="Red"/>

<jc:EnumValueMapping JdfValue="Green" AccessEnumValue="Green"/>

<jc:EnumValueMapping JdfValue="Yellow" AccessEnumValue="Yellow"/>

<jc:EnumValueMapping JdfValue="Gray" AccessEnumValue="Gray"/>

<jc:EnumValueMapping JdfValue="Orange" AccessEnumValue="Orange"/>

<jc:EnumValueMapping JdfValue="Pink" AccessEnumValue="Pink"/>

<jc:EnumValueMapping JdfValue="Black" AccessEnumValue="Black"/>

</jc:MediaEnumMapping>
```

CoverMediaSize

```
<jc:MediaConditionalEnumMapping Name="CoverMediaSize" Type="Cover"
MediaPartitioner="/jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams"
Optional="true">

<jc:ConditionalEnumValue AccessEnumValue="A0">

<jc:NumericCondition ExpectedValue="2383.9344" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="3370.392" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A0">

<jc:NumericCondition ExpectedValue="2383.9344" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="3370.392" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A1">

<jc:NumericCondition ExpectedValue="1683.7776" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="2383.9344" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>
```

```
<jc:ConditionalEnumValue AccessEnumValue="A1">
<jc:NumericCondition ExpectedValue="1683.7776" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
<jc:NumericCondition ExpectedValue="2383.9344" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="A2">
<jc:NumericCondition ExpectedValue="1190.5488" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
<jc:NumericCondition ExpectedValue="1683.7776" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="A2">
<jc:NumericCondition ExpectedValue="1190.5488" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
<jc:NumericCondition ExpectedValue="1683.7776" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="A3">
<jc:NumericCondition ExpectedValue="1190.5488" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
<jc:NumericCondition ExpectedValue="841.8888" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="A3">
<jc:NumericCondition ExpectedValue="1190.5488" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
<jc:NumericCondition ExpectedValue="841.8888" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="A1">
<jc:NumericCondition ExpectedValue="1683.7776" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
<jc:NumericCondition ExpectedValue="2383.9344" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="A1">
<jc:NumericCondition ExpectedValue="1683.7776" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
<jc:NumericCondition ExpectedValue="2383.9344" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
```

```
</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A2">

<jc:NumericCondition ExpectedValue="1190.5488" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="1683.7776" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A2">

<jc:NumericCondition ExpectedValue="1190.5488" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="1683.7776" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A3">

<jc:NumericCondition ExpectedValue="1190.5488" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="841.8888" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A3">

<jc:NumericCondition ExpectedValue="1190.5488" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="841.8888" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A4">

<jc:NumericCondition ExpectedValue="595.2744" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="841.8888" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A4">

<jc:NumericCondition ExpectedValue="595.2744" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="841.8888" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A5">

<jc:NumericCondition ExpectedValue="595.2744" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
```

```
<jc:NumericCondition ExpectedValue="419.5296" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A5">

<jc:NumericCondition ExpectedValue="595.2744" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="419.5296" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Letter">

<jc:NumericCondition ExpectedValue="612" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="792" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Letter">

<jc:NumericCondition ExpectedValue="612" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="792" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Legal">

<jc:NumericCondition ExpectedValue="612" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="1008" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Legal">

<jc:NumericCondition ExpectedValue="1008" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="612" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Ledger">

<jc:NumericCondition ExpectedValue="1224" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="792" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Ledger">
```

```
<jc:NumericCondition ExpectedValue="1224" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="792" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Foolscap">

<jc:NumericCondition ExpectedValue="1224" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="972" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Foolscap">

<jc:NumericCondition ExpectedValue="1224" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="972" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Executive">

<jc:NumericCondition ExpectedValue="756" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="522" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Executive">

<jc:NumericCondition ExpectedValue="756" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="522" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Commercial">

<jc:NumericCondition ExpectedValue="595.2744" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="765" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Commercial">

<jc:NumericCondition ExpectedValue="595.2744" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="765" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>
```

```
<jc:ConditionalEnumValue AccessEnumValue="Statement">
<jc:NumericCondition ExpectedValue="612" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
<jc:NumericCondition ExpectedValue="396" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Statement">
<jc:NumericCondition ExpectedValue="612" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
<jc:NumericCondition ExpectedValue="396" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
</jc:ConditionalEnumValue>
</jc:MediaConditionalEnumMapping>
```

CoverMediaType

```
<jc:MediaEnumMapping Name="CoverMediaType" Type="Cover" MediaPartitioner="/
jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/
@MediaType"/>
<jc:EnumValueMapping JdfValue="Paper" AccessEnumValue="Paper"/>
<jc:EnumValueMapping JdfValue="Transparency" AccessEnumValue="Transparent"/>
<jc:EnumValueMapping JdfValue="LaminatingFoil" AccessEnumValue="Polyester"/>
<jc:EnumValueMapping JdfValue="Other" AccessEnumValue="Other"/>
</jc:MediaEnumMapping>
```

CoverMediaWeight

```
<jc:MediaEnumMapping Name="CoverMediaWeight" Type="Cover" MediaPartitioner="/
jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/
@Weight"/>
<jc:EnumValueMapping JdfValue="65" AccessEnumValue="65032gram047m2"/>
<jc:EnumValueMapping JdfValue="80" AccessEnumValue="80032gram047m2"/>
<jc:EnumValueMapping JdfValue="90" AccessEnumValue="90032gram047m2"/>
<jc:EnumValueMapping JdfValue="100" AccessEnumValue="100032gram047m2"/>
<jc:EnumValueMapping JdfValue="120" AccessEnumValue="120032gram047m2"/>
<jc:EnumValueMapping JdfValue="160" AccessEnumValue="160032gram047m2"/>
<jc:EnumValueMapping JdfValue="170" AccessEnumValue="170032gram047m2"/>
</jc:MediaEnumMapping>
```

DocumentMediaColor

```
<jc:MediaEnumMapping Name="DocumentMediaColor" Type="Content"
MediaPartitioner="/jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams"
Optional="true">

<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/
@MediaColorName"/>

<jc:EnumValueMapping JdfValue="White" AccessEnumValue="White"/>

<jc:EnumValueMapping JdfValue="Blue" AccessEnumValue="Blue"/>

<jc:EnumValueMapping JdfValue="Red" AccessEnumValue="Red"/>

<jc:EnumValueMapping JdfValue="Green" AccessEnumValue="Green"/>

<jc:EnumValueMapping JdfValue="Yellow" AccessEnumValue="Yellow"/>

<jc:EnumValueMapping JdfValue="Gray" AccessEnumValue="Gray"/>

<jc:EnumValueMapping JdfValue="Orange" AccessEnumValue="Orange"/>

<jc:EnumValueMapping JdfValue="Pink" AccessEnumValue="Pink"/>

<jc:EnumValueMapping JdfValue="Black" AccessEnumValue="Black"/>

</jc:MediaEnumMapping>
```

DocumentMediaSize

```
<jc:MediaEnumMapping Name="DocumentMediaColor" Type="Content"
MediaPartitioner="/jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams"
Optional="true">

<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/
@MediaColorName"/>

<jc:EnumValueMapping JdfValue="White" AccessEnumValue="White"/>

<jc:EnumValueMapping JdfValue="Blue" AccessEnumValue="Blue"/>

<jc:EnumValueMapping JdfValue="Red" AccessEnumValue="Red"/>

<jc:EnumValueMapping JdfValue="Green" AccessEnumValue="Green"/>

<jc:EnumValueMapping JdfValue="Yellow" AccessEnumValue="Yellow"/>

<jc:EnumValueMapping JdfValue="Gray" AccessEnumValue="Gray"/>

<jc:EnumValueMapping JdfValue="Orange" AccessEnumValue="Orange"/>

<jc:EnumValueMapping JdfValue="Pink" AccessEnumValue="Pink"/>

<jc:EnumValueMapping JdfValue="Black" AccessEnumValue="Black"/>

</jc:MediaEnumMapping>
```

DocumentMediaSize

```
<jc:MediaConditionalEnumMapping Name="DocumentMediaSize" Type="Content"
MediaPartitioner="/jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams"
Optional="true">

<jc:ConditionalEnumValue AccessEnumValue="A0">
```

```
<jc:NumericCondition ExpectedValue="2383.9344" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="3370.392" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A0">

<jc:NumericCondition ExpectedValue="2383.9344" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="3370.392" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A1">

<jc:NumericCondition ExpectedValue="1683.7776" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="2383.9344" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A1">

<jc:NumericCondition ExpectedValue="1683.7776" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="2383.9344" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A2">

<jc:NumericCondition ExpectedValue="1190.5488" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="1683.7776" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A2">

<jc:NumericCondition ExpectedValue="1190.5488" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="1683.7776" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="A3">

<jc:NumericCondition ExpectedValue="1190.5488" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="841.8888" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>
```

```
<jc:ConditionalEnumValue AccessEnumValue="A3">
<jc:NumericCondition ExpectedValue="1190.5488" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
<jc:NumericCondition ExpectedValue="841.8888" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="A4">
<jc:NumericCondition ExpectedValue="595.2744" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
<jc:NumericCondition ExpectedValue="841.8888" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="A4">
<jc:NumericCondition ExpectedValue="595.2744" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
<jc:NumericCondition ExpectedValue="841.8888" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="A5">
<jc:NumericCondition ExpectedValue="595.2744" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
<jc:NumericCondition ExpectedValue="419.5296" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="A5">
<jc:NumericCondition ExpectedValue="595.2744" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
<jc:NumericCondition ExpectedValue="419.5296" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Letter">
<jc:NumericCondition ExpectedValue="612" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
<jc:NumericCondition ExpectedValue="792" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Letter">
<jc:NumericCondition ExpectedValue="612" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>
<jc:NumericCondition ExpectedValue="792" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
```

```
</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Legal">

<jc:NumericCondition ExpectedValue="612" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="1008" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Legal">

<jc:NumericCondition ExpectedValue="1008" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="612" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Ledger">

<jc:NumericCondition ExpectedValue="1224" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="792" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Ledger">

<jc:NumericCondition ExpectedValue="1224" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="792" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Foolscap">

<jc:NumericCondition ExpectedValue="1224" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="972" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Foolscap">

<jc:NumericCondition ExpectedValue="1224" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="972" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Executive">

<jc:NumericCondition ExpectedValue="756" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>
```

```
<jc:NumericCondition ExpectedValue="522" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Executive">

<jc:NumericCondition ExpectedValue="756" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="522" <jc:NumericCondition
ExpectedValue="522"

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Commercial">

<jc:NumericCondition ExpectedValue="595.2744" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="765" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Commercial">

<jc:NumericCondition ExpectedValue="595.2744" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="765" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Statement">

<jc:NumericCondition ExpectedValue="612" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

<jc:NumericCondition ExpectedValue="396" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

</jc:ConditionalEnumValue>

<jc:ConditionalEnumValue AccessEnumValue="Statement">

<jc:NumericCondition ExpectedValue="612" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[1]"/>

<jc:NumericCondition ExpectedValue="396" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:Media[@ID='${MediaID}']/@Dimension[0]"/>

</jc:ConditionalEnumValue>

</jc:MediaConditionalEnumMapping>
```

DocumentMediaType

```
<jc:MediaEnumMapping Name="DocumentMediaType" Type="Content"
MediaPartitioner="/jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams"
Optional="false">

<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/
@MediaType"/>

<jc:EnumValueMapping JdfValue="Paper" AccessEnumValue="Paper"/>
```

```
<jc:EnumValueMapping JdfValue="Transparency" AccessEnumValue="Transparent"/>
<jc:EnumValueMapping JdfValue="LaminatingFoil" AccessEnumValue="Polyester"/>
<jc:EnumValueMapping JdfValue="Other" AccessEnumValue="Other"/>
</jc:MediaEnumMapping>
<jc:MediaEnumMapping Name="DocumentMediaWeight" Type="Content"
MediaPartitioner="/jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams"
Optional="true">
</jc:MediaEnumMapping>
```

DocumentMediaWeight

```
<jc:MediaEnumMapping Name="DocumentMediaWeight" Type="Content"
MediaPartitioner="/jdf:JDF/jdf:ResourcePool/jdf:DigitalPrintingParams"
Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:Media[@ID='${MediaID}']/
@Weight"/>
<jc:EnumValueMapping JdfValue="65" AccessEnumValue="65032gram047m2"/>
<jc:EnumValueMapping JdfValue="80" AccessEnumValue="80032gram047m2"/>
<jc:EnumValueMapping JdfValue="100" AccessEnumValue="100032gram047m2"/>
<jc:EnumValueMapping JdfValue="120" AccessEnumValue="120032gram047m2"/>
<jc:EnumValueMapping JdfValue="160" AccessEnumValue="160032gram047m2"/>
<jc:EnumValueMapping JdfValue="170" AccessEnumValue="170032gram047m2"/>
</jc:MediaEnumMapping>
```

Archiving

Archive

```
<jc:BooleanMapping Name="Archive" Optional="true">
<jc:TrueCondition ExpectedValue="true" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:NodeInfo/jdf:GeneralID[@IDUsage='oce:Archive']/@IDValue"/>
</jc:BooleanMapping>
```

ArchvingMode

```
<jc:EnumMapping Name="ArchivingMode" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:NodeInfo/
jdf:GeneralID[@IDUsage='oce:ArchivingMode']/@IDValue"/>
<jc:EnumValueMapping JdfValue="Job" AccessEnumValue="Job"/>
<jc:EnumValueMapping JdfValue="PDF" AccessEnumValue="PDF"/>
</jc:EnumMapping>
```

ArchivingSecurityClass

```
<jc:EnumMapping Name="ArchivingSecurityClass" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:NodeInfo/
jdf:GeneralID[@IDUsage='oce:ArchivingSecurityClass']/@IDValue"/>
<jc:EnumValueMapping JdfValue="Default" AccessEnumValue="*Default"/>
<jc:EnumValueMapping JdfValue="Personal" AccessEnumValue="*Personal"/>
<jc:EnumValueMapping JdfValue="Shared" AccessEnumValue="*Shared"/>
<jc:EnumValueMapping JdfValue="Public" AccessEnumValue="*Public"/>
</jc:EnumMapping>
```

CaseName

```
<jc:TextMapping Name="CaseName" Optional="true" >
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:NodeInfo/
jdf:GeneralID[@IDUsage='oce:CaseName']/@IDValue"/>
</jc:TextMapping>
```

CaseType

```
<jc:EnumMapping Name="CaseType" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:NodeInfo/
jdf:GeneralID[@IDUsage='oce:CaseType']/@IDValue"/>
<jc:EnumValueMapping JdfValue="Customer" AccessEnumValue="Customer"/>
<jc:EnumValueMapping JdfValue="Product" AccessEnumValue="Product"/>
<jc:EnumValueMapping JdfValue="Project" AccessEnumValue="Project"/>
</jc:EnumMapping>
```

PublishingDepartment

```
<jc:EnumMapping Name="PublishingDepartment" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:NodeInfo/
jdf:GeneralID[@IDUsage='oce:PublishingDepartment']/@IDValue"/>
<jc:EnumValueMapping JdfValue="Sales" AccessEnumValue="Sales"/>
<jc:EnumValueMapping JdfValue="Service" AccessEnumValue="Service"/>
<jc:EnumValueMapping JdfValue="Marketing" AccessEnumValue="Marketing"/>
<jc:EnumValueMapping JdfValue="Production" AccessEnumValue="Production"/>
<jc:EnumValueMapping JdfValue="Procurement" AccessEnumValue="Procurement"/>
<jc:EnumValueMapping JdfValue="Logistics" AccessEnumValue="Logistics"/>
<jc:EnumValueMapping JdfValue="Development" AccessEnumValue="Development"/>
</jc:EnumMapping>
```

DocumentType

```
<jc:EnumMapping Name="DocumentType" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:NodeInfo/
jdf:GeneralID[@IDUsage='oce:DocumentType']/@IDValue"/>
<jc:EnumValueMapping JdfValue="Collateral" AccessEnumValue="Collateral"/>
<jc:EnumValueMapping JdfValue="Documentation"
AccessEnumValue="Documentation"/>
<jc:EnumValueMapping JdfValue="Contract" AccessEnumValue="Contract"/>
</jc:EnumMapping>
```

Delivery

DeliveryDate

```
<jc:DateMapping Name="Date" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:NodeInfo/@LastEnd"/>
</jc:DateMapping>
```

DeliveryAddress

```
<jc:TextMapping Name="DeliveryAddress" Optional="false">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Delivery']/jdf:Address/@Street"/>
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Delivery']/jdf:Address/@City"/>
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Delivery']/jdf:Address/@PostalCode"/>
</jc:TextMapping>
```

DeliveryMethod

```
<jc:ConditionalEnumMapping Name="DeliveryMethod" Optional="true">
<jc:ConditionalEnumValue AccessEnumValue="Pickup">
<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="Delivery"/>
<jc:StringCondition ExpectedValue="true" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:DeliveryParams/@Transfer"/>
</jc:ConditionalEnumValue>
<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="Delivery"/>
<jc:StringCondition ExpectedValue="ExpressMail" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:DeliveryParams/@Method"/>
</jc:ConditionalEnumValue>
<jc:ConditionalEnumValue AccessEnumValue="Post">
<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="Delivery"/>
<jc:StringCondition ExpectedValue="InterofficeMail" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:DeliveryParams/@Method"/>
</jc:ConditionalEnumValue>
</jc:ConditionalEnumMapping>
```

DeliveryRemarks

```
<jc:TextMapping Optional="true" Name="DeliveryRemarks">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:DeliveryParams/
jdf:Comment"/>
</jc:TextMapping>
```

Emailing

EmailMessage

```
<jc:TextMapping Name="EmailMessage" Optional="true">

<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:NodeInfo/
jdf:GeneralID[@IDUsage='oce:EmailMessage']/@IDValue"/>

<jc:TrueCondition ExpectedValue="true" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:NodeInfo/jdf:GeneralID[@IDUsage='oce:EmailWhenJobAccepted']/@IDValue"/>

</jc:BooleanMapping>
```

EmailWhenJobAccepted

```
<jc:BooleanMapping Name="EmailWhenJobAccepted" Optional="true">

<jc:TrueCondition ExpectedValue="true" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:NodeInfo/jdf:GeneralID@IDUsage='oce:EmailWhenJobAccepted']/@IDValue"/>

</jc:BooleanMapping>
```

EmailWhenJobReady

```
<jc:BooleanMapping Name="EmailWhenJobReady" Optional="true">

<jc:TrueCondition ExpectedValue="true" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:NodeInfo/jdf:GeneralID[@IDUsage='oce:EmailWhenJobReady']/@IDValue"/>

</jc:BooleanMapping>
```

EmailWhenJobRejected

```
<jc:BooleanMapping Name="EmailWhenJobRejected" Optional="true">

<jc:TrueCondition ExpectedValue="true" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:NodeInfo/jdf:GeneralID[@IDUsage='oce:EmailWhenJobRejected']/@IDValue"/>

</jc:BooleanMapping>
```

Cost estimation and cost quotation

BudgetApproversRemarks

```
<jc:TextMapping Name="BudgetApproversRemarks" Optional="true">
<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="Approval"/>
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:ApprovalParams/
jdf:GeneralID[@IDUsage='oce:BudgetApproversRemarks']/@IDVa lue"/>
</jc:TextMapping>
```

BudgetOwner

```
<jc:TextMapping Name="BudgetOwner" Optional="true">
<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="Approval"/>
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:ApprovalParams/
jdf:ApprovalPerson/jdf:Contact/jdf:ComChannel[@ChannelType ='Email']/
@Locator"/>
</jc:TextMapping>
```

ChargeBackID1

```
<jc:TextMapping Name="ChargeBackID1" Optional="true" >
<jc:JdfField XPath="/jdf:JDF/jdf:NodeInfo/jdf:Employee/jdf:CostCenter/
@CostCenterID"/>
</jc:TextMapping>
```

EstimatedColorPages

```
<jc:NumberMapping Name="EstimatedColorPages" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:RunList/
jdf:GeneralID[@IDUsage='oce:EstimatedColorPages']/@IDValue"/>
</jc:NumberMapping>
```

EstimatedBWPages

```
<jc:NumberMapping Name="EstimatedBWPages" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:RunList/
jdf:GeneralID[@IDUsage='oce:EstimatedBWPages']/@IDValue"/>
</jc:NumberMapping>
```

EstimatedCost

```
<jc:TextMapping Optional="true" Name="EstimatedCost">
<jc:JdfField XPath="/jdf:JDF/jdf:Comment[@Name='Price']"/>
</jc:TextMapping>
```

FinalCost

```
<jc:TextMapping Name="FinalCost" Optional="true">
```

```
<jc:JdfField XPath="/jdf:JDF/jdf:Comment[@Name='oce:FinalCost']"/>
</jc:TextMapping>
```

ManualLabour

```
<jc:TextMapping Name="ManualLabor" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:Comment[@Name='oce:ManualLabour']"/>
</jc:TextMapping>
```

NrOfSourcePages

```
<jc:NumberMapping Optional="true" Name="NrOfSourcePages">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:RunList/@Npage"/>
</jc:NumberMapping>
```

RequestCostQuotation

```
<jc:BooleanMapping Name="RequestCostQuotation" Optional="true">
<jc:TrueCondition ExpectedValue="true" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:NodeInfo/jdf:GeneralID[@IDUsage='oce:RequestCostQuotation']/@IDValue"/>
</jc:BooleanMapping>
```

SubmittersDescription

```
<jc:TextMapping Name="SubmittersDescription" Optional="true">
<jc:StringCondition JdfField="/jdf:JDF/@Types" ContainedValue="Approval"/>
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:ApprovalParams/
jdf:GeneralID[@IDUsage='oce:SubmittersDescription']/@IDVal ue"/>
</jc:TextMapping>
```

Miscellaneous

Author

```
<jc:TextMapping Name="Author" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Author']/jdf:Person/@FirstName"/>
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/
jdf:Contact[@ContactTypes='Author']/jdf:Person/@FamilyName"/>
</jc:TextMapping>
```

Comments

```
<jc:TextMapping Optional="true" Name="Comments">
<jc:JdfField XPath="/jdf:JDF/jdf:Comment[@Name='OperatorInstruction']"/>
</jc:TextMapping>
```

Category

```
<jc:TextMapping Name="Category" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:NodeInfo/
jdf:GeneralID[@IDUsage='oce:Category']/@IDValue"/>
</jc:TextMapping>
```

ExceptionInserts

```
<jc:BooleanMapping Name="ExceptionInserts" Optional="true">
<jc:TrueCondition ExpectedValue="true" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:NodeInfo/jdf:GeneralID[@IDUsage='oce:ExceptionInserts']/@IDValue"/>
</jc:BooleanMapping>
```

ExceptionRemarks

```
<jc:TextMapping Name="ExceptionRemarks" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:NodeInfo/
jdf:GeneralID[@IDUsage='oce:ExceptionRemarks']/@IDValue"/>
</jc:TextMapping>
```

ExceptionTabs

```
<jc:BooleanMapping Name="ExceptionTabs" Optional="true">
<jc:TrueCondition ExpectedValue="true" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:NodeInfo/jdf:GeneralID[@IDUsage='oce:ExceptionTabs']/@IDValue"/>
</jc:BooleanMapping>
```

IntentRemarks

```
<jc:TextMapping Name="IntentRemarks" Optional="false">
<jc:JdfField XPath="/jdf:JDF/jdf:Comment[@Name='Instruction']"/>
```

```
</jc:TextMapping>
```

IntakeOperator

```
<jc:TextMapping Name="IntakeOperator" Optional="true" >
<jc:JdfField XPath="/jdf:JDF/jdf:NodeInfo/
jdf:Employee[@Roles='IntakeOperator']/jdf:Person/@FirstName"/>
<jc:JdfField XPath="/jdf:JDF/jdf:NodeInfo/
jdf:Employee[@Roles='IntakeOperator']/jdf:Person/@FamilyName"/>
</jc:TextMapping>
```

OffsetStacked

```
<jc:EnumMapping Name="OffsetStacked" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:StackingParams/
jdf:Disjointing/@OffsetDirection"/>
<jc:EnumValueMapping JdfValue="Alternate" AccessEnumValue="BySet"/>
<jc:EnumValueMapping JdfValue="None" AccessEnumValue="Off"/>
</jc:EnumMapping>
```

OperatorNotes

```
<jc:TextMapping Name="OperatorNotes" Optional="false">
<jc:JdfField XPath="/jdf:JDF/jdf:Comment[@Name='OperatorText']"/>
</jc:TextMapping>
```

ProofPrint (PrinterName)

```
<jc:TextMapping Name="ProofPrint" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:Device[@ID=/jdf:JDF/
jdf:AuditPool/jdf:ResourceAudit/jdf:DeviceLink/@rRef] /@DeviceID"/>
</jc:TextMapping>
```

PaperAvailable

```
<jc:BooleanMapping Name="PaperAvailable" Optional="true" >
<jc:TrueCondition ExpectedValue="Available" JdfField="/jdf:JDF/
jdf:ResourcePool/jdf:ExposedMedia/@Status"/>
</jc:BooleanMapping>
```

RequestSoftProof

```
<jc:BooleanMapping Name="RequestSoftProof" Optional="true">
<jc:TrueCondition ExpectedValue="true" JdfField="/jdf:JDF/jdf:ResourcePool/
jdf:NodeInfo/jdf:GeneralID[@IDUsage='oce:RequestSoftPoof']/@IDValue"/>
</jc:BooleanMapping>
```

JobName

```
<jc:TextMapping Name="JobName" Optional="false">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:CustomerInfo/
@CustomerJobName"/>
</jc:TextMapping
```

Copies

```
<jc:NumberMapping Name="Copies" Optional="false">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourceLinkPool/
jdf:ComponentLink[ @Usage='Output' and @rRef=/jdf:JDF/jdf:ResourcePool/
jdf:Component[@ComponentType='FinalProduct']/@ID ]/@Amount"/>
</jc:NumberMapping>
```

PrinterName

```
<jc:TextMapping Name="PrinterName" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:Device/@DeviceID"/>
</jc:TextMapping>
```

PrintInColor

```
<jc:EnumMapping Name="PrintInColor" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:ColorantControl/
@ProcessColorModel"/>
<jc:EnumValueMapping JdfValue="DeviceCMYK" AccessEnumValue="Color"/>
<jc:EnumValueMapping JdfValue="DeviceGray" AccessEnumValue="BlackAndWhite"/>
</jc:EnumMapping>
```

Plexity

```
<jc:EnumMapping Name="Plexity" Optional="false">
<jc:JdfField XPath="/jdf:JDF/jdf:ResourcePool/jdf:LayoutPreparationParams/
@Sides"/>
<jc:EnumValueMapping JdfValue="OneSidedFront" AccessEnumValue="Simplex"/>
<jc:EnumValueMapping JdfValue="OneSidedBackFlipX" AccessEnumValue="Simplex"/>
<jc:EnumValueMapping JdfValue="OneSidedBackFlipY" AccessEnumValue="Simplex"/>
<jc:EnumValueMapping JdfValue="TwoSidedFlipY" AccessEnumValue="Duplex"/>
<jc:EnumValueMapping JdfValue="TwoSidedFlipX" AccessEnumValue="Duplex"/>
</jc:EnumMapping>
```

Title

```
<jc:TextMapping Name="Title" Optional="true">
<jc:JdfField XPath="/jdf:JDF/@DescriptiveName"/>
</jc:TextMapping>
```

CreationDate

```
<jc:DateMapping Name="CreationDate" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:AuditPool/jdf:Created/@TimeStamp"/>
</jc:DateMapping>
```

FinishingRemarks

```
<jc:TextMapping Name="FinishingRemarks" Optional="true">
<jc:JdfField XPath="/jdf:JDF/jdf:AuditPool/jdf:PhaseTime/@Comment"/>
</jc:TextMapping>
```

FinishingTime

```
<jc:TimeSpanMapping Name="FinishingTime" Optional="true">
<jc:StringCondition JdfField="/jdf:JDF/jdf:AuditPool/
jdf:PhaseTime[@ModuleType='Finisher']/@Status" ExpectedValue="Completed"/>
<jc:TimeSpan Start="/jdf:JDF/jdf:AuditPool/
jdf:PhaseTime[@ModuleType='Finisher']/@Start" End="/jdf:JDF/jdf:AuditPool/
jdf:PhaseTime[@ModuleType='Finisher']/@End"/>
</jc:TimeSpanMapping>
```

PrepareTime

```
<jc:TimeSpanMapping Name="PrepareTime" Optional="true">
<jc:StringCondition JdfField="/jdf:JDF/jdf:AuditPool/
jdf:PhaseTime[@ModuleType='Imposer']/@Status" ExpectedValue="Completed"/>
<jc:TimeSpan Start="/jdf:JDF/jdf:AuditPool/
jdf:PhaseTime[@ModuleType='Imposer']/@Start" End="/jdf:JDF/jdf:AuditPool/
jdf:PhaseTime[@ModuleType='Imposer']/@End"/>
</jc:TimeSpanMapping>
```

DSF endpoint

DSF doesn't completely adhere to JDF standard, thus, customization had to be done.

When adding or configuring a JDF endpoint, the user can mark the endpoint as being DSF and additional configuration is needed that generates a new URL for DSF. This URL must be used for DSF case, not the main one presented by the JDF endpoint configuration.

Additionally, DSF-to-PD mapping file (available for automatic Import Service) can be used for ticket items as a better default and it is available on the PRISMAdirect installation machine: "$(OCE_STORAGE)\Asset\Import Service Mappings".

Canon

Canon Inc.
www.canon.com

Canon U.S.A., Inc.
www.usa.canon.com

Canon Canada Inc.
www.canon.ca

Canon Europe Ltd
www.canon-europe.com

Canon Latin America Inc.
www.cla.canon.com

Canon Australia PTY. Ltd
www.canon.com.au

Canon China Co., Ltd
www.canon.com.cn

Canon Singapore PTE. Ltd
www.canon.com.sg

Canon Hongkong Co., Ltd
www.canon.com.hk