



# Informatica<sup>TM</sup>

## Accelerators

Informatica MDM - Product 360

Version: 10.5 HotFix 3 SP 1

# Table of Contents

<b>1</b>	<b>GDSN Accelerator .....</b>	<b>12</b>
1.1	Providing an interface to the Global Data Synchronization Network (GDSN) for accurate product data exchange .....	12
1.2	The GDSN accelerator for Product 360 .....	13
1.2.1	Communicating with the GDSN data pools .....	14
1.2.1.1	Standard GDSN - Data Source .....	15
1.2.1.2	Standard GDSN - Data Recipient .....	16
1.2.1.3	Item Management - Data Source .....	17
1.2.1.4	Item Management - Data Recipient .....	18
1.2.2	Data model extension .....	18
1.2.2.1	Packaging hierarchies .....	18
1.2.3	UI components .....	19
1.2.3.1	Limitations of Product 360 Web .....	20
1.2.4	Media assets .....	21
1.3	GDSN Accelerator Package Content .....	21
1.3.1	Download GDSN Accelerator zip .....	21
1.3.2	Extract the GDSN Accelerator zip .....	21
1.3.3	The Resources zip .....	21
1.3.3.1	The "Common" folder .....	22
1.3.3.2	The "DataRecipient" folder .....	22
1.3.3.3	The "DataSource" folder .....	22
1.4	GDSN Accelerator Installation .....	23
1.4.1	Prerequisites .....	23
1.4.1.1	OpenAS2 environment .....	23
1.4.2	Installing the GDSN Accelerator .....	24
1.4.2.1	Product 360 Server .....	24
1.4.2.2	Product 360 Desktop Client .....	25
1.4.3	Application Modules .....	27
1.4.3.1	Application Modules Configuration for DSE .....	28
1.4.3.2	Application Modules Configuration for IM .....	28
1.4.4	Repository Configurations .....	29

1.4.4.1	Repository auto-adjustments.....	29
1.4.4.2	Manual repository adjustments .....	30
1.4.4.3	Repository Configuration for DSE .....	34
1.4.4.4	Repository Configuration for IM .....	34
1.5	GDSN Accelerator Setup with OpenAS2.....	34
1.5.1	Why use OpenAS2?.....	34
1.5.2	Official OpenAS2 documentation.....	35
1.5.3	How it works.....	35
1.5.3.1	"Item/Link/Publication send succeeded" communication .....	35
1.5.3.2	"Item/Link/Publication send failed" communication.....	36
1.5.3.3	"Import responses" communication .....	36
1.5.3.4	Import functions to create customers and suppliers.....	37
1.5.4	Installation of OpenAS2 .....	37
1.5.4.1	Upgrading OpenAS2 from 2.X or older to 3.x .....	38
1.5.5	Standard GDSN .....	38
1.5.5.1	GDSN Data Source Scenario .....	38
1.5.5.2	GDSN Data Recipient Scenario .....	48
1.5.6	Item Management.....	57
1.5.6.1	IM Data Source Scenario.....	57
1.5.6.2	IM Data Recipient Scenario.....	67
1.5.7	Optional configuration for Data Recipient Scenarios .....	76
1.5.7.1	Import item data into individual supplier catalogs.....	76
1.6	GDSN Accelerator operation .....	80
1.6.1	Maintain valid data .....	80
1.6.1.1	Item identifiers: GTINs .....	80
1.6.1.2	Maintain data .....	80
1.6.1.3	Validate data .....	82
1.6.2	Item status in relation to the GDSN data pool.....	82
1.6.2.1	Publication status .....	83
1.6.2.2	Confirmation status .....	83
1.6.3	Send data.....	84
1.6.3.1	Use queries to get an overview of your data .....	84
1.6.3.2	Build dynamic assortments from queries.....	84
1.6.3.3	Send item data .....	87
1.6.3.4	Publish data.....	87

1.6.4	Receive feedback .....	87
1.6.5	Export Templates.....	89
1.6.5.1	Export template parameters .....	89
1.6.5.2	Export data source configuration .....	89
1.6.5.3	Post processing.....	94
1.6.5.4	Export Templates for Standard GDSN DataSource .....	95
1.6.5.5	Export Templates for IM DataSource .....	96
1.6.6	Automated Jobs.....	98
1.6.6.1	Setup automated Jobs .....	98
1.6.6.2	Automated Jobs for Standard GDSN .....	99
1.6.6.3	Automated Jobs for IM.....	100
1.6.7	Publication .....	104
1.6.7.1	Mark items for publication .....	104
1.6.7.2	Publish items automatically.....	104
1.6.7.3	Publication process for Standard GDSN .....	105
1.6.7.4	Publication process for IM .....	105
1.7	GDSN Implementation Guidelines .....	106
1.7.1	Introduction .....	106
1.7.2	Documents .....	106
1.7.3	Implementation steps.....	107
1.7.4	Analyze requirements.....	107
1.7.4.1	Introduction .....	107
1.7.4.2	Questions.....	108
1.7.4.3	Resources .....	108
1.7.4.4	Data model - analyze the module .....	108
1.7.4.5	Summary .....	121
1.7.5	Data model.....	121
1.7.5.1	Introduction .....	121
1.7.5.2	Resources .....	122
1.7.5.3	Create a new entity .....	122
1.7.5.4	Create Logical keys .....	125
1.7.5.5	Create a new field.....	131
1.7.5.6	Deactivate a GDSN entity.....	135
1.7.5.7	Create or adjust a valid value list .....	135
1.7.5.8	Checklist: Test the module .....	144



1.7.6	UI adjustments .....	145
1.7.6.1	Creating additional elements in Desktop UI.....	145
1.7.6.2	Creating additional elements in Web UI .....	151
1.7.6.3	Interface visibility and display rights .....	154
1.7.6.4	Limitation .....	155
1.7.7	Data validations .....	155
1.7.7.1	Data model validation .....	155
1.7.7.2	Data quality checks by IDQ (Informatica Data Quality).....	155
1.7.7.3	Validation and formatting during import .....	156
1.7.7.4	Validation and formatting during export.....	158
1.7.8	Enhance Export Templates.....	160
1.7.8.1	Adjust the export template.....	160
1.7.9	Enhance Import Mappings.....	162
1.7.9.1	Adjust an import mapping .....	162
1.7.10	Testing .....	164
1.7.10.1	Test data .....	164
1.7.10.2	Test data quality (Data source) .....	167
1.7.10.3	Test against certified GDSN pool (Data source) .....	169
1.7.10.4	Test against certified GDSN pool (Data recipient).....	170
1.8	Migration Guides .....	170
1.8.1	GDSN Migration Guide from B2B to OpenAS2 .....	170
1.8.1.1	Overview .....	170
1.8.1.2	Export template migration .....	171
1.8.2	GDSN Migration Guide From 10.1 to 10.5 (OpenAS2) .....	176
1.8.2.1	Introduction .....	176
1.8.2.2	Changes affecting IM .....	176
1.8.2.3	Changes affecting Standard GDSN.....	178
1.8.3	GDSN Migration Guide for version 3.1.27 .....	178
1.8.3.1	Product 360 data model changes.....	178
1.8.3.2	Changes in data source export templates .....	182
1.8.3.3	Compatibility.....	187
1.9	GDSN Accelerator FAQ .....	188
1.9.1	General .....	188
1.9.1.1	Q: Can I also send items to the GDSN pool from a supplier catalog? .....	188

1.9.1.2	Q: What does a response message like "GDSN Numeric Rule ID 1281: The format of "Ingredient Sequence" must be 'dd.dd.dd...'. Where 'd' must be a digit, always ending in a 'dd' and never having a value of '00'." mean? .....	188
1.9.2	Export .....	188
1.9.2.1	Q: The export fails due to the error "... One of '{document}' is expected." .....	188
1.9.2.2	Q: The export fails due to XSD error "...The value " of attribute..." .....	188
1.9.2.3	Q: Not all of my items has been sent to the pool.....	189
1.9.3	Custom Implementations.....	189
1.9.3.1	Q: How do we add missing units of measurement to the GDSN unit system?.....	189
1.9.3.2	Q: Why does the deletion of a subentity of "Article" doesn't work via Service API when using a reserve logical key as qualification filter? .....	190
1.10	Appendix.....	190
1.10.1	Appendix A: Certificates (OpenAS2) .....	190
1.10.1.1	Overview .....	190
1.10.1.2	Create a private key / public certificate .....	191
1.10.1.3	How to import a new certificate.....	192
1.10.1.4	How to list all certificates (check) .....	192
1.10.1.5	How to delete a certificate.....	193
1.10.1.6	How to extract a certificate from a p7b file .....	193
1.10.2	Appendix B: Export templates.....	194
1.10.2.1	Load and save the export templates.....	194
1.10.2.2	Add XSD schema files.....	195
1.10.2.3	Add XSLT files .....	196
1.10.2.4	Disabled file upload .....	197
1.10.2.5	File split post export step .....	198
1.10.2.6	Customization .....	199
1.10.3	Appendix C: Packaging hierarchy data providers.....	200
1.10.3.1	Complete packaging hierarchy .....	200
1.10.4	Appendix D: Technical export information.....	201
1.10.4.1	Special export functions.....	201
1.10.4.2	General approach to output data .....	202
1.10.4.3	Extend/modify export templates .....	203
1.10.5	Appendix E: Import mappings and hotfolder configuration.....	205
1.10.5.1	Configure observed import hotfolders .....	205
1.10.5.2	Load and save the import mappings .....	205
1.10.5.3	Create and configure hotfolder .....	206

1.10.6	Appendix F: Field List .....	209
<b>2</b>	<b>Informatica BPM Accelerator.....</b>	<b>273</b>
2.1	Resources .....	274
2.2	Required Workflows.....	276
2.2.1	Configuration and Installation of BPM.....	276
2.2.2	Informatica BPM Installation.....	277
2.2.2.1	Installation of the Informatica BPM service.....	277
2.2.2.2	Webserver and Java.....	278
2.2.2.3	Integrated Security .....	279
2.2.3	Installation of the required default workflows.....	280
2.2.4	Configure Dispatch Services.....	283
2.2.5	Preparing Informatica BPM service for JMS based communication .....	286
2.2.5.1	Download additional library.....	286
2.2.5.2	Setup messaging service within Informatica BPM .....	286
2.3	Step Workflow .....	290
2.3.1	Preface.....	290
2.3.2	Table of Contents.....	291
2.3.3	Installation .....	291
2.3.4	Configuration .....	292
2.3.4.1	Prerequisites .....	292
2.3.4.2	StepWorkflow XML files .....	292
2.3.4.3	URN mappings.....	292
2.3.4.4	StepWorkflowMap.xml .....	294
2.3.4.5	StepWorkflow.xml.....	295
2.3.4.6	Comments.xml.....	298
2.3.4.7	Email .....	300
2.3.4.8	Google Vision Alproject.....	302
2.3.5	Examples .....	302
2.3.5.1	Steps to run the examples .....	302
2.3.5.2	Table of contents .....	304
2.3.5.3	Default workflows .....	305
2.3.5.4	Example 1 (1 Task) .....	312
2.3.5.5	Example 2 (2 Tasks).....	316
2.3.5.6	Example 3 (2 Tasks with DQ checks) .....	320

2.3.5.7	Example 4 (5 Tasks with DQ checks, parallel and sequential) .....	324
2.3.5.8	Example 5 (extending example 4 with an approval step) .....	332
2.3.5.9	Example 6 (2 tasks (incl. 1 supplier task) + modifying of fields) .....	341
2.3.5.10	Example 7 (2 Tasks + modifying of fields + approval step) .....	351
2.3.5.11	Example 8 (2 Tasks modifying of fields approval step merge) .....	360
2.3.5.12	Example 9 (Classic Tasks) .....	366
2.3.5.13	Example 10 (1 Task) .....	375
2.3.5.14	Example 11 (1 Task with DQ checks) .....	380
2.3.5.15	Example 12 (2 Triggers and Merge with Results) .....	387
2.3.5.16	Example 13 (Workflow starts another workflow) .....	397
2.3.5.17	Example 14 (Export items) .....	402
2.3.5.18	Example 15 (1 Task with DQ check and comments) .....	408
2.3.5.19	Example 16 (Approval task with comments) .....	413
2.3.5.20	StepWorkflow Sanity .....	421
2.3.6	Steps .....	424
2.3.6.1	ExecuteDqBatch-Process .....	424
2.3.6.2	GVisionAi-Process .....	426
2.3.6.3	GVisionSetBooleanState-Process .....	428
2.3.6.4	GVisionTransferFieldValue-DQ .....	429
2.3.6.5	ItemsInWorkflowTasks-Process .....	431
2.4	Creating MQ based workflows .....	432
2.4.1	Prerequisites .....	432
2.4.2	REST versus JMS based workflows .....	432
2.4.3	Mandatory Permission Required For Queue Based Communication .....	433
2.4.4	Types of interactions and their representation in the WSDL file .....	434
2.4.5	Router process .....	435
2.4.6	Core orchestration project .....	436
2.4.7	Correlation .....	436
2.4.8	Defining participants .....	437
2.4.8.1	Consumers .....	437
2.4.8.2	Producers .....	437
2.4.9	Workflow interaction examples .....	439
2.4.9.1	Workflow instance creation .....	440
2.4.9.2	P360 Service API oneway calls .....	440
2.4.9.3	P360 Service API calls with result handling .....	441

2.4.9.4	P360 DQ execution with result handling.....	442
2.4.9.5	P360 workflow tasks .....	443
2.5	<b>DAM Workflows .....</b>	<b>444</b>
2.5.1	DAM Configurator.....	444
2.5.1.1	Preface.....	444
2.5.1.2	Prerequisites .....	444
2.5.1.3	Installation .....	445
2.5.1.4	Configuration .....	445
2.5.1.5	Defining Main settings .....	448
2.5.1.6	Defining Property fields for digital assets.....	450
2.5.1.7	Defining Derivatives.....	454
2.5.1.8	Apply DAM configurations .....	458
2.5.2	DAM Event Listener .....	459
2.5.2.1	Preface.....	459
2.5.2.2	Prerequisites .....	459
2.5.2.3	Installation .....	460
2.5.2.4	Configuration .....	460
2.5.2.5	Defining Derivatives.....	465
2.5.2.6	Start the DAM Event Listener.....	469
2.5.2.7	Verification .....	470
2.5.3	DAM Hotfolder.....	471
2.5.3.1	Preface.....	471
2.5.3.2	Prerequisites .....	472
2.5.3.3	Installation .....	472
2.5.3.4	Configuration .....	472
2.5.3.5	Start the DAM hotfolder .....	479
2.5.3.6	Throughput .....	480
2.5.3.7	Limitations .....	480
2.5.3.8	Verification .....	481
3	<b>User Interface Templates .....</b>	<b>482</b>
3.1	Content.....	482
3.2	Import.....	482
3.3	Approval UI.....	483
3.4	Text Mastering UI .....	484

3.5	Media Assignment UI .....	485
3.6	Classification UI.....	485
3.7	Attribute Mastering UI.....	486
<b>4</b>	<b>Web Search Examples.....</b>	<b>487</b>
4.1	Items-only.ext .....	487
4.2	Products-only.ext.....	487
4.3	Variant.only.ext .....	488
4.4	All Supplier Catalogs.ext.....	488
4.5	2PPD.ext .....	488
4.6	3PPD.ext .....	488
4.7	Steps to use it: .....	488
<b>5</b>	<b>CLAIRE Accelerator.....</b>	<b>490</b>
5.1	CLAIRE Server Installation for ML Model Driven Use Cases .....	490
5.1.1	Hardware Requirements.....	490
5.1.2	Windows Installation .....	490
5.1.2.1	Folder creation .....	490
5.1.2.2	CLAIRE server start up.....	491
5.1.3	Linux Installation .....	491
5.1.3.1	Folder creation .....	491
5.1.3.2	CLAIRE server start up.....	492
5.1.4	Claire Server Health Check .....	492
5.2	CLAIRE Services Setup .....	492
5.2.1	Product360 configuration.....	494
5.3	CLAIRE Product Classification .....	495
5.3.1	Configuration and Operation .....	495
5.3.1.1	AI Training for Auto Classification .....	495
5.3.1.2	Batch Classification.....	501
5.3.1.3	Configuration of CLAIRE Flex UI .....	504
5.3.2	Best Practices and Recommendations .....	506
5.3.2.1	Folder structure for model maintenance .....	506
5.3.2.2	Recommendations for best results .....	507
5.3.2.3	Model accuracy measurement .....	508
5.4	CLAIRE Brand Extraction.....	509

5.4.1	Configuration and Operation .....	509
5.4.1.1	Brand Extraction Training .....	509
5.4.1.2	Configuration of Brand Extraction Flex UI .....	512
5.4.1.3	Batch Execution of Brand Extraction .....	514
5.4.2	Best Practices and Recommendations .....	516
5.4.2.1	Folder Structure for Model Maintenance .....	516
5.4.2.2	Recommendations for Best Results .....	517
5.5	CLAIRE Translation.....	517
5.5.1	Configuration and Operation .....	517
5.5.1.1	Installation .....	517
5.5.1.2	Configuration of Claire Translation Service.....	518
5.5.1.3	Configuration of Translation Flex UI .....	521
5.5.1.4	Batch Translation .....	523
5.5.1.5	Attribute Support for Claire Translation .....	528
5.5.1.6	Characteristic Record Support for Claire Translation .....	532
5.6	Migration Guide from 10.1 to 10.5 .....	536
5.6.1	Introduction .....	536
5.6.2	Changes Affecting Product Classification .....	536
<b>6</b>	<b>Swagger UI.....</b>	<b>537</b>
6.1	Purpose.....	537
6.2	Installation .....	537
6.2.1	Prerequisites .....	537
6.2.2	Terms .....	537
6.2.3	Download package .....	538
6.2.4	Operation as a Windows Service .....	538
6.2.5	Operation in Standalone Mode .....	539
6.3	Configuration and Operation .....	540
6.3.1	Configuration .....	540
6.3.2	Operation .....	540
6.3.2.1	Example : Fetch 'Article' details at Entity Level, by Items in a catalog.....	544
6.4	Best Practices, Recommendations and Known Issues .....	547

Accelerators are functioning implementations which can be used out of the box under specific circumstances only. Usage of these accelerators in the context of the concrete customer's use cases and requirements with the customer's data and within its specific environment might need adaptations like configuration, translation, additional scripting, or programming of add-on functionalities which have to be provided through additional professional services.

For example the eCommerce integration provide the fully functioning transfer of a set of data based on the Product 360's standard repository and according to the selected sample implementation of the WebShop. As soon the data set needs to be adapted to the concrete needs of the customer's web-shop or adapting to the customer's individual Product 360 data repository layout additional customization might be needed.

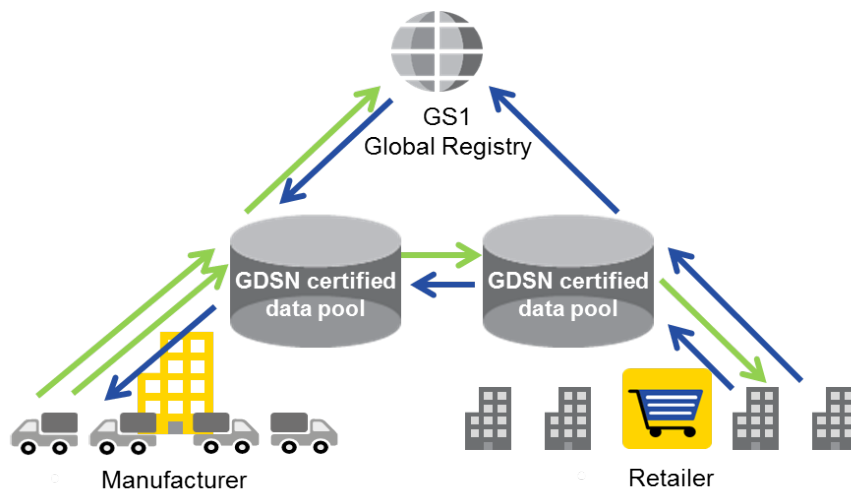
## 1 GDSN Accelerator

### 1.1 Providing an interface to the Global Data Synchronization Network (GDSN) for accurate product data exchange

This add-on to Product 360 lets you synchronize product data with GS1 GDSN standards. Throughout the lifecycle of a product, many people rely on access to information about that product and that product information must constantly be maintained and updated by manufacturers and brands. After all, reliable and even more detailed product information is required to comply with changing regulations and satisfy consumer expectations of accessing product information from any channel. Distributors, retailers, and operators are constantly challenged with insufficient and inaccurate product data across their warehouses, stores or online.

Enter the Global Data Synchronization Network (GDSN), a system of interoperable data pools and a global product classification system and registry, that enables companies to exchange standardized and synchronized supply chain data with their trading partners. Suppliers and retailers can cut down the cost of building point-to-point integrations and speed product introductions by accessing the most accurate and most current product information.





The **Informatica GDSN Accelerator** is an add-on to the Informatica Product 360 system that provides access to the 1WorldSync Item Management and Data Sync Engine - two GDSN-certified data pools. It is designed to help organizations securely and continuously exchange, update, and synchronize

product data with trading partners according to the standards defined by GS1.

After being integrated in your Product 360 system, the GDSN accelerator helps you easily transfer product data to the data pool and fully control the information shared with a specific trading partner. The solution offers great flexibility to map item attributes to the GDSN standard and can be tailored to customer specific needs like providing industry-specific or optional fields.

#### There are many benefits of using the GDSN Accelerator:

- Provides improved and consistent product data quality across the entire supply chain, increasing the accuracy of orders, reducing invoice errors, streamlining processes and slashing supply-chain costs
- Improves efficiency and collaboration with trading partners as it shares and updates product information quickly
- Streamlines and accelerates item setup processes and automates data exchange between data sources and recipients by accessing a single source of the truth. This helps speed time to market and time to shelf for new products or products with modified attributes like ingredients or package sizes.
- Can support manufacturers, suppliers and retailers to comply with regulations, and satisfies consumer and regulatory demands for more and better product information
- Informatica is a partner of 1WorldSync, atrify and GS1 Germany

#### Standard GDSN

We implemented the Accelerator against the Data Sync Engine "**DSE**" which is the data pool solution of atrify (former 1WorldSync GmbH). This pool is commonly used in Europe. In this documentation the terms "Standard GDSN" and "DSE" are used interchangeably.

#### Item Management

Item Management "**IM**" is the data pool solution of 1WorldSync inc, former 1Sync. This is mainly used in the USA.

## 1.2 The GDSN accelerator for Product 360

The GDSN accelerator consists of three main features. The first one is the data model extension. For GDSN Major Release 3, we made our data model compliant and added further parts of the GDSN Extension *Food and Beverage*. The second feature provided with the GDSN accelerator is a lot of data quality rules and rule

configurations to check if the data which will be sent to the GDSN data pool is valid. Last but not least it contains templates to export the data in the corresponding format which will be accepted by the GDSN data pool.

### 1.2.1 Communicating with the GDSN data pools

Our Accelerator supports the Data Source and Data Recipient scenarios for Standard GDSN and IM pools. Regardless of the selected solution, the communication between P360 and the pool is file based using the AS2 protocol and managed by the mediator OpenAS2.

Product 360 uses the core competence of its export to send the data to the GDSN pool. The export creates an XML file according to an XSD schema defined by the pool in use. This file will be sent to the mediator which forwards it to the connected GDSN pool.

Both, transmission confirmations and answers from the GDSN pool will be written back to Product 360, either as publication status (data source scenario) or confirmation status (data recipient scenario). Of course, this applies only for item-specific messages.

The following chapters describe the message choreographies of all supported scenarios in more detail.

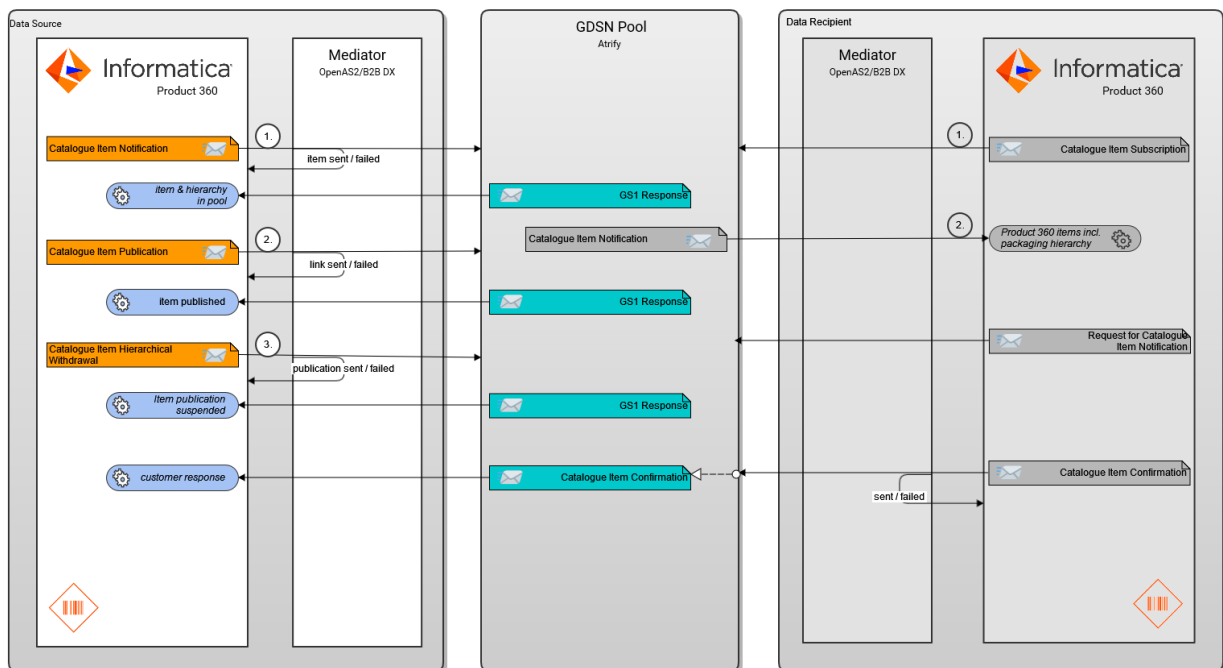
### 1.2.1.1 Standard GDSN - Data Source

The main goal of a data source scenario is to provide item data to the pool in order to make them available for retailers.

The workflow consists of two steps:

- Send item data including packaging hierarchy information by sending *Catalogue Item Notification* messages
- Publicate packaging hierarchies by sending *Catalogue Item Publication* messages

For each message sent to the pool we will receive a corresponding response message. Additionally, it is possible that we get *Catalogue Item Confirmation* messages from recipients.

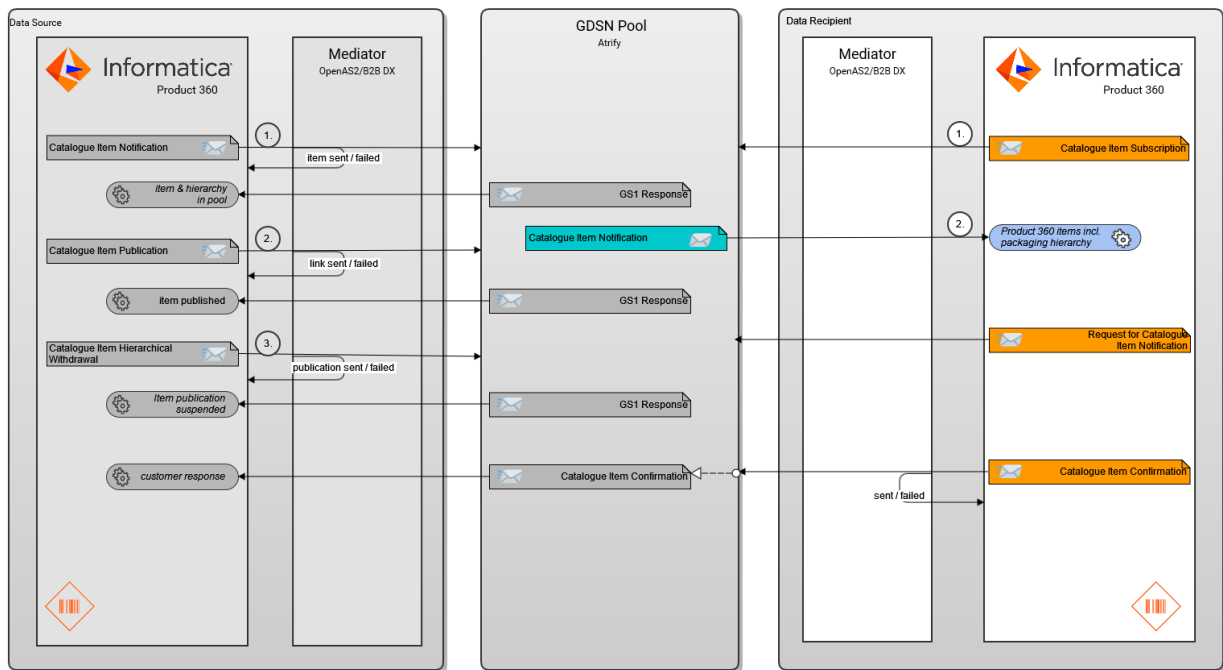


### 1.2.1.2 Standard GDSN - Data Recipient

The main goal of a data recipient scenario is to get item data from the pool.

The workflow consists of three steps:

- Subscribe to item data by various criteria like GPC categories, target markets or suppliers by sending *Catalogue Item Subscription* messages
- Receiving item data including packaging hierarchy information in *Catalogue Item Notification* messages
- Optionally, you can provide feedback about the item data to the data provider by sending *Catalogue Item Confirmation* messages



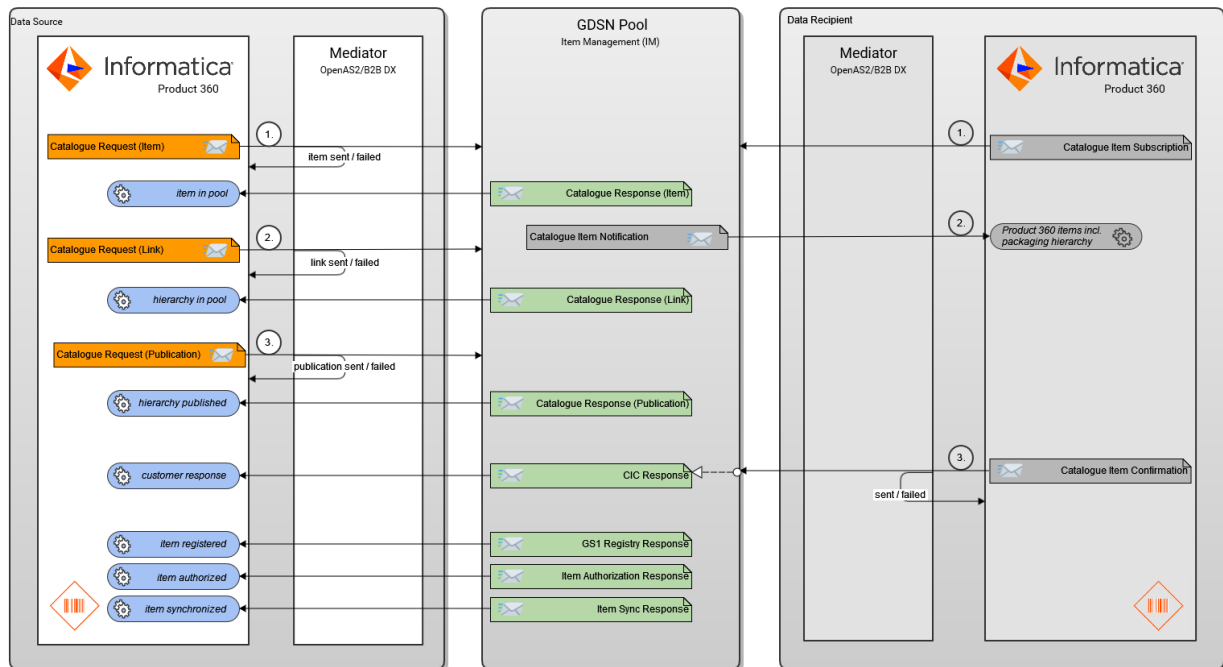
### 1.2.1.3 Item Management - Data Source

The main goal of a data source scenario is to provide item data to the pool in order to make them available for retailers.

The workflow consists of three steps:

- Send item data by sending *Catalogue Request Item* messages
- Send packaging hierarchy information by sending *Catalogue Request Link* messages
- Publicate packaging hierarchies by sending *Catalogue Request Publication* messages

For each message sent to the pool we will receive a corresponding response message. Additionally, it is possible that we get *Catalogue Item Confirmation* messages from recipients as well as some types of general administrative messages from the pool.

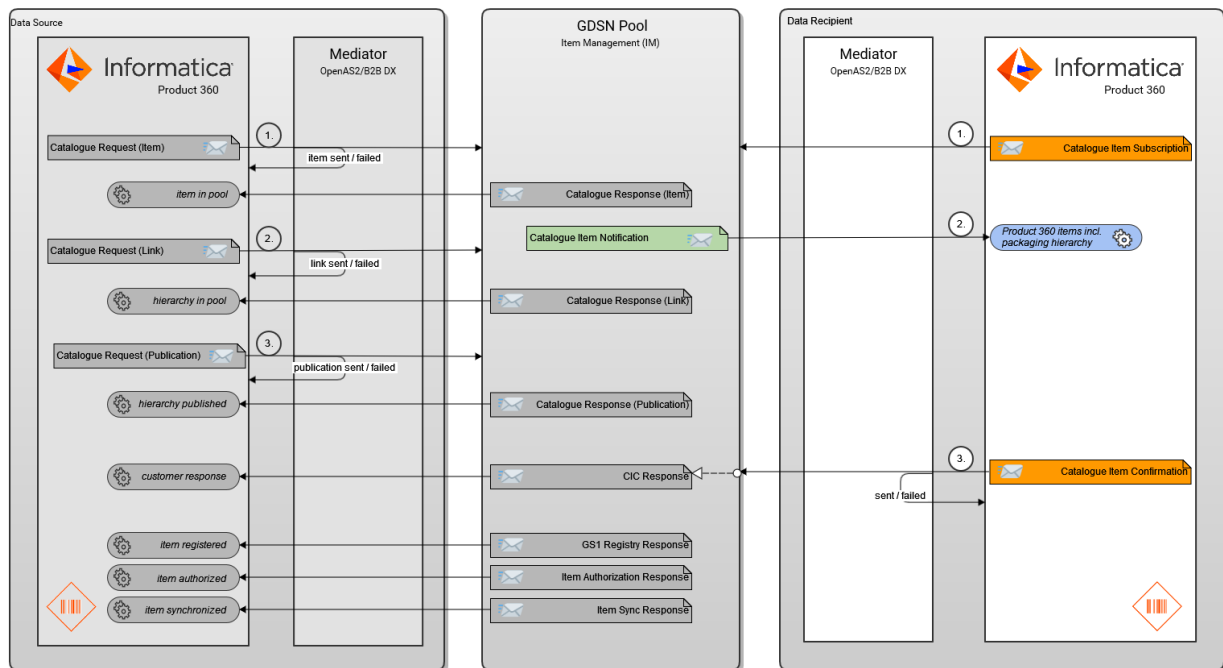


### 1.2.1.4 Item Management - Data Recipient

The main goal of a data recipient scenario is to get item data from the pool.

The workflow consists of three steps:

- Subscribe to item data by various criteria like GPC categories, target markets or suppliers by sending *Catalogue Item Subscription* messages
- Receiving item data including packaging hierarchy information in *Catalogue Item Notification* messages
- Optionally, you can provide feedback about the item data to the data provider by sending *Catalogue Item Confirmation* messages




## 1.2.2 Data model extension



The Product 360 data model supports the core GDSN data model attributes, most of the Food and Beverage extension and the Canada extension (IM only). A complete list of all provided fields can be found in the chapter [Appendix F: Field List](#) (see page 209). The included languages are English and German for field labels, value lists and descriptions, other languages need to be accessed from the local GS1 Organization, as they refer to standardized terms and cannot be translated without introducing ambiguity and confusion.

### 1.2.2.1 Packaging hierarchies

GDSN defines a special hierarchy for items, the so-called packaging hierarchy. In Product 360 this packaging hierarchy is represented by a special reference type called *Next lower level*. You can build a packaging hierarchy by creating a reference of type *Next lower level* from one item to another. The important thing here is that the parent has a reference to the child with the corresponding quantity (number). One example is

shown in the screenshot below where a case was dropped on a pallet to create a packaging hierarchy. Here the created packaging hierarchy will contain one pallet with 25 cases.

 Drag & Drop event



You have selected "Pallet" of type "Item" as the target.

☒ Create a reference between the item "Case" and the item "Pallet"

Reference type:  
Number:

Next lower level

25

☐ Create a component

The item is added to the selected kit as a static or variable component.

Component type:  
Sequence:  
Quantity:

Static

<Move to end>

1

OK

Cancel

The existing "Item references" view can be used to edit respectively delete the *Next lower level* references:

Item references (2) X

Pallet

	Reference type	Referenced object type	Referenced item	Referenced object number	Number
1	Next lower level	Item	Case A	Case A	25
2	Next lower level	Item	Case B	Case B	50

1 element selected

Reference type

### 1.2.3 UI components

With the GDSN accelerator Product 360 provides several UI perspectives like *GDSN (core)* and *Food and beverage (core)* where attributes can be displayed and maintained. Corresponding detail tabs are included in the Product 360 Web Client as well.

Due to the importance of product hierarchies in GDSN, a special view *GDSN packaging hierarchy* in both clients is part of the Accelerator.

The message history of the communication between Product 360 and the pool can be viewed in either the *Publication status* view or the *Confirmation status* view, depending on the scenario.

Please find more detailed information about available perspectives and views in chapter [GDSN Accelerator operation](#) (see page 80).

The screenshot displays the Informatica PIM 7.1 interface, which is divided into several panes. The top pane shows a list of items from the supplier catalog 'Test', with columns for Item no., GTIN, EAN UCC code (UPC/EAN Shipping Container Code, USA), and EAN UCC code (UCC-12, USA). The middle pane shows the GDSN packaging hierarchy for item 1000000000017, including a table for Target market specific GDSN data (1) with columns for Variant (English), Target market, Is active in market, Is orderable unit, Is dispatch unit, Is invoice unit, Country of origin, Primary delivery method, Total units per case, Coupon family code, GTIN variation registry, and Is private. The bottom pane shows the item details for 'Coffee Supreme X1D 1000', including a table for GDSN Target Market Data and a table for Food and beverage information.

The bottom pane shows the item details for 'Coffee Supreme X1D 1000'. The item is currently in a 'New' status. The 'GDSN Target Market Data' table shows the following data:

Header	Preview	Text	GDSN General Data	EAN UCC codes	GDSN Target Market Data	Food and beverage	Diet / Allergen information	Ingredient / Preparation information	Nutrition / Serving information	Certification information	Prices	Med
Target market			USA									
Is active in market			<input checked="" type="checkbox"/>									
Is orderable unit			<input checked="" type="checkbox"/>									
Is dispatch unit			<input checked="" type="checkbox"/>									
Is invoice unit			<input checked="" type="checkbox"/>									
Is packaging marked returnable			<input type="checkbox"/>									
Start availability date			No content									
End availability date			No content									
Discontinued date			No content									
Country of origin			No content									
Primary delivery method			No content									
Total units per case			No content									
Coupon family code			No content									
GTIN variation registry			<input checked="" type="checkbox"/>									
Is private			<input checked="" type="checkbox"/>									
Pricing on product			<input checked="" type="checkbox"/>									

The 'Food and beverage' section shows the following data:

Header	Preview	Text	GDSN General Data	EAN UCC codes	GDSN Target Market Data	Food and beverage	Diet / Allergen information	Ingredient / Preparation information	Nutrition / Serving information	Certification information	Prices	Med
Target market			USA									
Languages			English									
Functional name (English)			No content									
GDSN short description (English)			No content									
Additional description (English)			No content									
Marketing message (English)			No content									
Variant (English)			No content									
Product description (English)			No content									
Sub brand (English)			No content									
UOM types			imperial									
Height (imperial)			No content									

### 1.2.3.1 Limitations of Product 360 Web

- The is no publication status or confirmation status detail tab in Product 360 Web.
- In Product 360 Web language specific GDSN attributes can only be maintained in the key language, but it is possible to adjust the corresponding web definitions.



## 1.2.4 Media assets




The data model of media assets contains some GDSN related fields. The field *Uniform resource identifier*, for example, contains the URL to the image to add to your data in the GDSN pool.

Please note that there is currently no supported way to upload physical images to the GDSN pool.

## 1.3 GDSN Accelerator Package Content

### 1.3.1 Download GDSN Accelerator zip

The **GDSN Accelerator** is available via Informatica Shipping. It consists of the folder **PIM\_<Version>\_GDSN** in the file **PIM\_<Version>\_Accelerators.zip**.

 PIM\_10.5.0.00.00\_Rev-77519\_client\_gdsn.delta.zip  
 PIM\_10.5.0.00.00\_Rev-77519\_resources\_gdsn.delta.zip  
 PIM\_10.5.0.00.00\_Rev-77519\_server\_gdsn.delta.zip





### 1.3.2 Extract the GDSN Accelerator zip

Unpack the file **PIM\_<Version>\_Accelerators.zip** to a temporary directory on the Product 360 environment. It contains the following artefacts:

- **PIM\_<Version>\_Rev-xxxxx\_client\_gdsn.delta.zip**
  - contains PIM Desktop feature and plugins
- **PIM\_<Version>\_Rev-xxxxx\_resources\_gdsn.delta.zip**
  - contains export templates, import mappings, DQ rules and rule configurations, Product 360 Web view definitions
- **PIM\_<Version>\_Rev-xxxxx\_server\_gdsn.delta.zip**
  - contains PIM Desktop feature and plugins

### 1.3.3 The Resources zip

While the client and server zip contain the GDSN bundles which have to be simply installed via control center the "resources" package contains additional content which is shown below.

 Common  
 DataRecipient  
 DataSource  
 webdefinitions

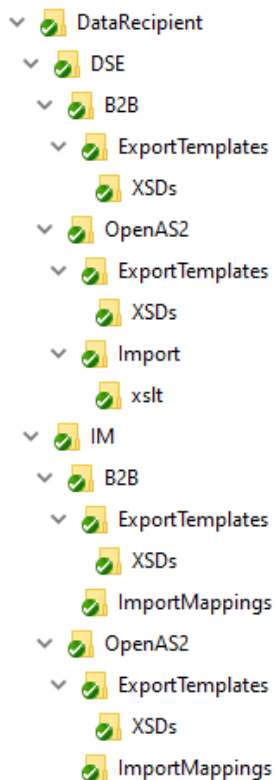
Before the GDSN accelerator is installed, it has to be clear which scenario is applicable. We support two different GDSN data pools, Item Management (IM) and Standard GDSN (e. g. DSE/atrify). For both we support the data source as well as the data recipient side which leads to four different configuration possibilities. Depending on your scenario the corresponding export templates, import mappings and data quality rules have to be selected.

#### 1.3.3.1 The "Common" folder

The "Common" folder contains things that are required for all scenarios, for example the GPC structure import which is relevant for the data recipient scenario as well as for the data source scenario and of course for both GDSN data pools.

#### 1.3.3.2 The "DataRecipient" folder

The "DataRecipient" folder contains all things necessary to install the data recipient scenario. This folder is divided again into the subfolders "DSE" and "IM" to differentiate between the GDSN data pools. For the recipient we currently only provide some export templates as well as basic import mapping to be able to import CINs into Product 360.

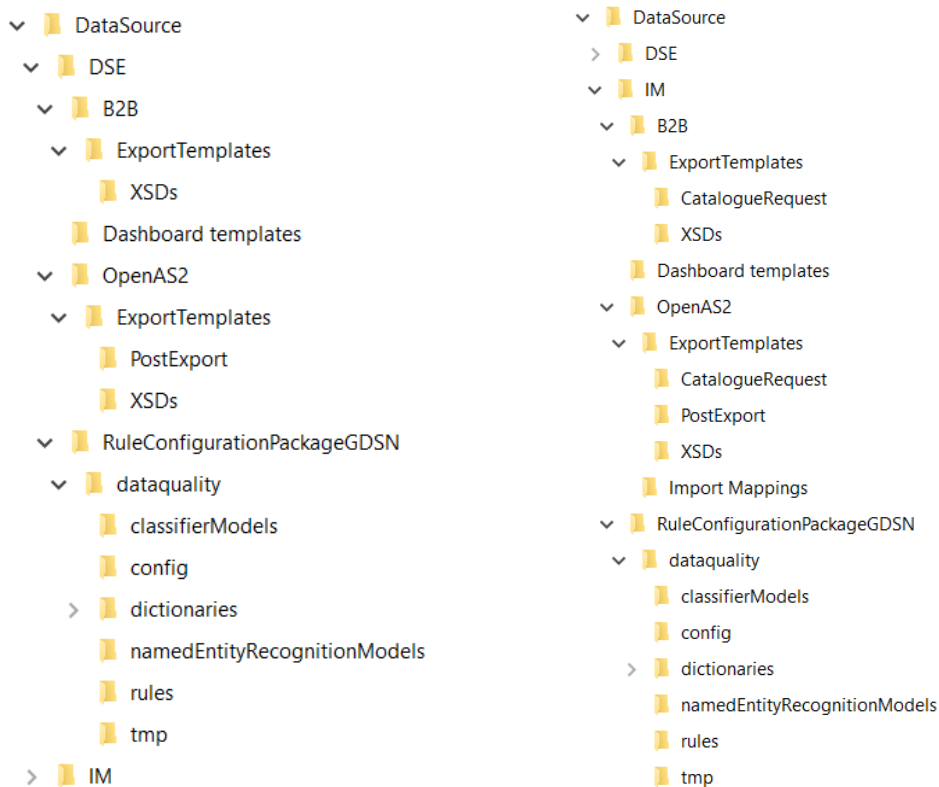


#### 1.3.3.3 The "DataSource" folder

The "DataSource" folder contains all things necessary to install the data source scenario. Like the data recipient folder it is divided into a "DSE" and "IM" subfolder which contains the corresponding Dashboard

templates, Export templates (with XSD schema files) and rule configurations. The dashboard templates are examples how a dashboard could be configured for a corresponding business user working with GDSN. Due to the fact that the choreography is different between IM and DSE the dashboards are also different. The export templates and the corresponding XSD schema files are also different depending on the GDSN data pool. This has to be considered when importing them. Last but not least there are the rule configurations files. While Product 360 provides a lot of standard GDSN rule configurations which are applicable for DSE and IM data pool, there are also a lot of additional IM specific rule configurations which are located in the IM folder.

You can use OpenAS2 for communication to the pool. Some of the mentioned things like export templates are a little bit different. That's why you find an OpenAS2 folder with suitable materials.



## 1.4 GDSN Accelerator Installation

### 1.4.1 Prerequisites

#### 1.4.1.1 OpenAS2 environment

For communication to the pool we recommend using OpenAS2. Previously Informatica B2B DX was used for communication.

OpenAS2 environment

OpenAS2 installation and configuration is described in [GDSN Accelerator Setup with OpenAS2](#) (see page 34).

## 1.4.2 Installing the GDSN Accelerator

Download and extract the GDSN Accelerator zip as described in chapter [GDSN Accelerator Package Content](#) (see page 21).

Unpack the archive **PIM\_<Version>\_Rev-xxxxx\_resources\_gdsn.delta.zip** into a temporary folder **<GDSN\_RESOURCES\_PATH>**.

### 1.4.2.1 Product 360 Server

Install Product 360 Server bundles

Unpack the archive **PIM\_<Version>\_Rev-xxxxx\_server\_gdsn.delta.zip** into the Product 360 server folder.

Install GDSN Data Quality Rule Configurations

The GDSN data quality rule configurations are used to ensure valid data according to GDSN specifications. That's why we strongly recommend it for the data source scenario.

1. Backup the Product 360 server's **dataquality** folder (**<PIM\_SERVER\_SHARED\_DIR>/dataquality**).  
Hint: **<PIM\_SERVER\_SHARED\_DIR>** is defined in server.properties under the key `filestorage.dir.shared`
2. Copy (and replace) the file **StandardDataQualityMappingProfile.xml** to the subfolder **config** of the Product 360 server's **dataquality** folder. (**<PIM\_SERVER\_SHARED\_DIR>/dataquality/config**)
  - a. For IM the StandardDataQualityMappingProfile.xml to be used can be found here:  
`\DataSource\IM\RuleConfigurationPackageGDSN\dataquality\config`
  - b. For DSE the StandardDataQualityMappingProfile.xml to be used can be found here:  
`\DataSource\DSE\RuleConfigurationPackageGDSN\dataquality\config`



If there are problems during startup due to same objects (because of same identifiers) used in both configuration sets, the server won't start. The logs give detailed information about conflicting objects.

Later after the complete GDSN accelerator installation/migration, execute DQ runs to get a status for the item for each of the new rule configurations. And the channels status is restored as well.

Adjust Repository

It is not necessary to do repository adjustments for IM. During the start of the Product 360 server the repository will be adjusted automatically according to the configured GDSN pool.

Those steps are described in the [Repository configurations](#) (see page 29) chapter as well as options for manual custom-specific repository adjustments.

If you are using the DSE pool, please have a look at [Configuration for DSE](#) (see page 34), to see the necessary adjustments.

#### Adjust application modules

In the "application\_modules.properties" file you have to activate the used GDSN modules, see chapter [Application modules](#) (see page 27) for more details.

### 1.4.2.2 Product 360 Desktop Client

#### Install Product 360 Desktop Client bundles

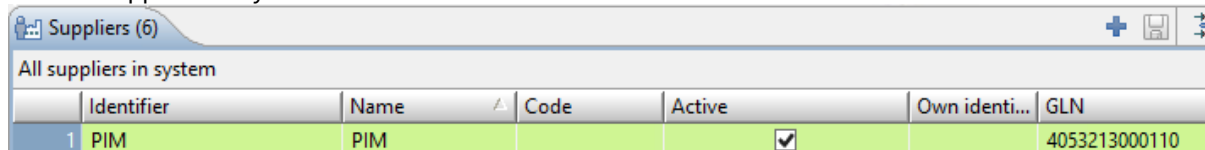
Unpack the archive **PIM\_<Version>\_Rev-xxxxx\_client\_gdsn.delta.zip** into the Product 360 client folder.

#### Create supplier and customer

In the communication with the data pool the participating parties are identified by their unique GLNs. In P360 these parties can be suppliers or customers. In order to map the responses from the pool to the original request and to process them correctly, we need the sender and receiver GLNs stored on P360 suppliers and customers.

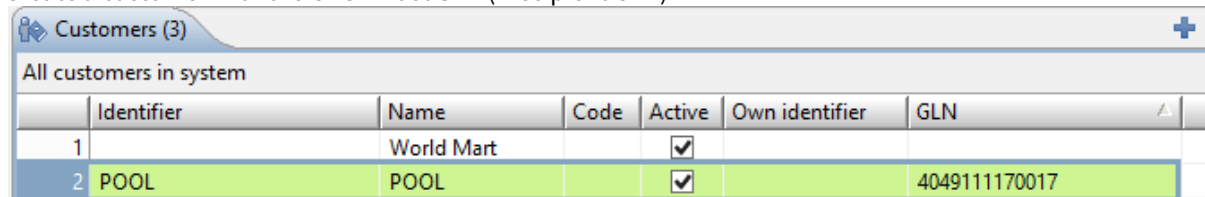
#### Data source scenario

1. Create a supplier with your **own** Information Provider GLN



Suppliers (6)						
All suppliers in system						
	Identifier	Name	Code	Active	Own identi...	GLN
1	PIM	PIM		<input checked="" type="checkbox"/>		4053213000110

2. Create a customer with your own Information Provider GLN (this is needed for publication to market groups (IM Only))
3. Create a customer with the GDSN Pool GLN (=recipient GLN)

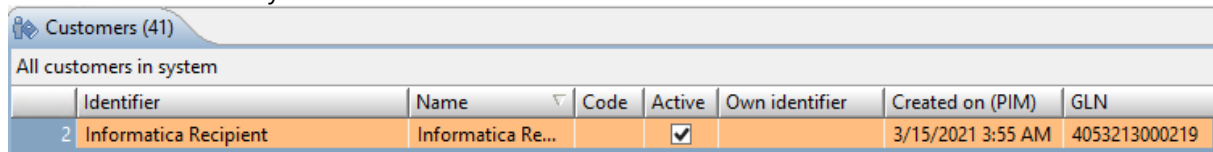


Customers (3)						
All customers in system						
	Identifier	Name	Code	Active	Own identifier	GLN
1		World Mart		<input checked="" type="checkbox"/>		
2	POOL	POOL		<input checked="" type="checkbox"/>		4049111170017

4. Maintain the GLN on all your customers

## Data recipient scenario

1. Create a customer with your **own** GLN



Customers (41)							
All customers in system							
	Identifier	Name	Code	Active	Own identifier	Created on (PIM)	GLN
2	Informatica Recipient	Informatica Re...		<input checked="" type="checkbox"/>		3/15/2021 3:55 AM	4053213000219

## Load and save export templates

The used export templates differ depending on which product is used for communicating with the pool (OpenAS2) and the exact scenario (IM/DSE, DataSource/DataRecipient). In the Accelerator Setup section you will find further information about the export templates used in your scenario and how to proceed with them.

## Install GDSN Data Quality rules and reference data

1. Add GDSN specific rules:
  - a. Open the perspective "Data quality"
  - b. Select any custom rule configuration
  - c. Open the "Select data quality rule" dialog (".." -button) (If the rule configurations are read only, create new one to be able to import rules.)
  - d. Click the button "Add rules from file"
  - e. choose the file **<GDSN\_RESOURCES\_PATH>/Common/RulePackageGDSN/Informatica\_PIM\_GDSN.xml**
  - f. Wait until the rules were added successfully
2. Add GDSN specific reference data
  - a. In the "Select data quality rule" dialog (see step 2.c. above) click the button "Add reference data from file"
  - b. Choose the file **<GDSN\_RESOURCES\_PATH>/Common/RulePackageGDSN/Informatica\_PIM\_GDSN.zip**
  - c. in the "Adding reference data" dialog leave the defaults and click "OK".
  - d. Wait, until all dictionaries are deployed, which is done automatically within a server job of type "Deployment of reference data (scheduled)".

## Import GPC Structure

1. Download the latest files from the GS1 website: <http://www.gs1.org/gpc/production>
  - a. It is recommended to download the **Combined Published Schema, or the subset of the required industry**
  - b. The excel file **GS1 Combined Published\_Schema as at 0000000.xlsx** from the zip archive is needed
2. Create a new **output** structure system with identifier "GPC". Configure that multiple assignments are not possible for this structure system.
3. Use the import mapping **<GDSN\_RESOURCES\_PATH>/Common/GPC structure import/GPC en.him** to import the downloaded excel file
4. Switch the structure system to type "classification system"

## Load Dashboard

The GDSN accelerator package contains two dashboard templates, one for IM data source scenario and one for DSE data source scenario. The templates are designed for business users who are working with GDSN

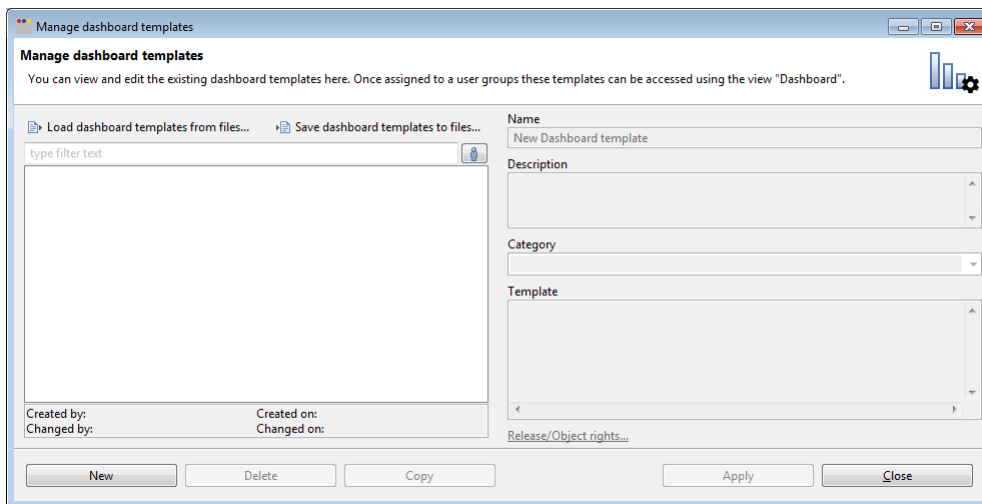
and contain typical widgets for their daily work such as a widget showing how many items are already sent to the pool or how many items are already published.

To use the dashboards please open the "Manage Dashboard templates" dialog at the menu "Management". Then click on "Load dashboard templates from file" and select your dashboard. You can assign this dashboard to your GDSN user group in the organization perspective (in the view "User groups"), so it will be opened whenever a user of this user group opens the web client.

Some parameters need to be adjusted to the company's details:

```
<parameter key="groupingField" value="Article.Status->PublicationStatusEntry.ResponseType('Super CPG','World Mart',DE,PUBLICATION_RESPONSE,$Default)"/>
```

- 'Super CPG' should be replaced with the Name of the Supplier holding the GLN of the own company
- 'World Mart' should be replaced with the Name of the default customer who should be visualized
- DE should be replaced with the required Target Market



### 1.4.3 Application Modules

With Product 360 8.0.5 the application module concept was introduced. A module can be seen as business functionality which can be enabled or disabled. Therefore a new property file called "application\_modules.properties" located in the configuration folder of the server was introduced. Currently it is possible to enable/disable different GDSN scenarios and the Food and beverage module. When activated, the corresponding perspectives, views and other things like reports and export datatypes are available for the user to import/maintain/export food and beverage or GDSN data.

**By default "GDSN" as well as "FoodAndBeverage" are set to "false" and must be activated during the accelerator installation (food and beverage only if needed).**

The Food and Beverage module can be used independently of GDSN.

An example for the application\_modules.properties file is given in the screenshot below. Please see the following chapters for a specific "DSE" or "IM" configuration.

```
# Defines if the GDSN extension is installed (true) or not (false)
GDSN = true


# Defines if the GDSN pool "IM" is used (true) or not (false) if the GDSN extension is installed.
gdsn_pool_im = false


# Defines if the GDSN pool "DSE" is used (true) or not (false) if the GDSN extension is installed.
gdsn_pool_dse = true

# Defines if the GDSN extension is used in "data source" mode (true) or not (false) if installed.
gdsn_data_source = true

# Defines if the GDSN extension is used in "data recipient" mode (true) or not (false) if installed
gdsn_data_recipient = false

# Defines if the food and beverage module is activated (true) or deactivated (false).
FoodAndBeverage = true
```

 Please note that both, running multiple pools at the same time and activating more than one role (dataSource and dataRecipient) is not supported.

 Please also note that if GDSN should be used, the corresponding GDSN plugins need to be installed in the client and server.

#### 1.4.3.1 Application Modules Configuration for DSE

The following settings are needed in the application\_modules.properties file for GDSN pool "DSE", data source scenario. If the data recipient scenario should be installed, "gdsn\_data\_source" has to be set to "false" and "gdsn\_data\_recipient" has to be set to "true". Again, having both scenarios active at the same time is not supported.

```
# Defines if the GDSN extension is installed (true) or not (false)
GDSN = true

# Defines if the GDSN pool "IM" is used (true) or not (false) if the GDSN extension is installed.
gdsn_pool_im = false

# Defines if the GDSN pool "DSE" is used (true) or not (false) if the GDSN extension is installed.
gdsn_pool_dse = true

# Defines if the GDSN extension is used in "data source" mode (true) or not (false) if installed.
gdsn_data_source = true

# Defines if the GDSN extension is used in "data recipient" mode (true) or not (false) if installed
gdsn_data_recipient = false

# Defines if the food and beverage module is activated (true) or deactivated (false).
FoodAndBeverage = true
```

#### 1.4.3.2 Application Modules Configuration for IM

The following settings are needed in the application\_modules.properties file for the GDSN pool "IM", data source scenario. If the data recipient scenario should be installed, "gdsn\_data\_source" has to be set to "false" and "gdsn\_data\_recipient" has to be set to "true". Again, having both scenarios active at the same time is not supported.



```
# Defines if the GDSN extension is installed (true) or not (false)
GDSN = true

# Defines if the GDSN pool "IM" is used (true) or not (false) if the GDSN extension is installed.
gdsn_pool_im = true

# Defines if the GDSN pool "DSE" is used (true) or not (false) if the GDSN extension is installed.
gdsn_pool_dse = false

# Defines if the GDSN extension is used in "data source" mode (true) or not (false) if installed.
gdsn_data_source = true

# Defines if the GDSN extension is used in "data recipient" mode (true) or not (false) if installed
gdsn_data_recipient = false

# Defines if the food and beverage module is activated (true) or deactivated (false).
FoodAndBeverage = true
```

## 1.4.4 Repository Configurations

### 1.4.4.1 Repository auto-adjustments

During the start of the Product 360 server the repository will be adjusted automatically according to the configured GDSN pool. The following steps will be done:

Activate GDSN repository entities

The GDSN modules are implemented as second-level entities below the "Item" root entity. In this context, the terms "GDSN modules" and "GDSN specific repository entities" are used interchangeably.

All GDSN specific repository entities are deactivated by default in the repository. If GDSN is installed and activated (see [Application modules \(see page 27\)](#)), all GDSN repository entities will be enabled during server start.

**Note:** The "Active" attribute of the repository entities is the only one that is changed here. All other attributes - like name, import purpose, ... - as well as the fields and the sub-entities can be configured in the repository as usual.

In addition to that, there are few data fields that are pool-specific. They will be disabled during the server start according to the configured GDSN pool.

**Attention:** It is strongly recommended that you don't change the "scope" settings of repository fields.

Activate enumeration entries of GDSN enumerations

The entries of GDSN enumerations defined in the "EnumFragment.repository" file will be added to the enumerations.

There're some enumeration entries that are pool-specific. That's why some enumeration entries got an additional "scope" parameter that is analysed during server startup and ensures consistent valid value lists. Not applicable enumeration entries will not be added.

**Attention:** It is strongly recommended that you don't change the "scope" settings of enumeration entries.

### EnumFragment

It is strongly recommended that the EnumFragment.repository placed in server/configuration/HPM/gdsn/ is NOT MODIFIED for any customization.

"Next lower level" item reference type

The GDSN packaging hierarchy is implemented by an additional item reference type "Next lower level". That reference type will be added to the "Enum.Article.Article.ReferenceTypes" repository enumeration, the default value is "12000". In addition to that, an enum param "GDSN.packagingHierarchy.reference" is added to that enumeration, it is needed by the business layer, e.g. to provide the hierarchy view with appropriate data.

#### 1.4.4.2 Manual repository adjustments

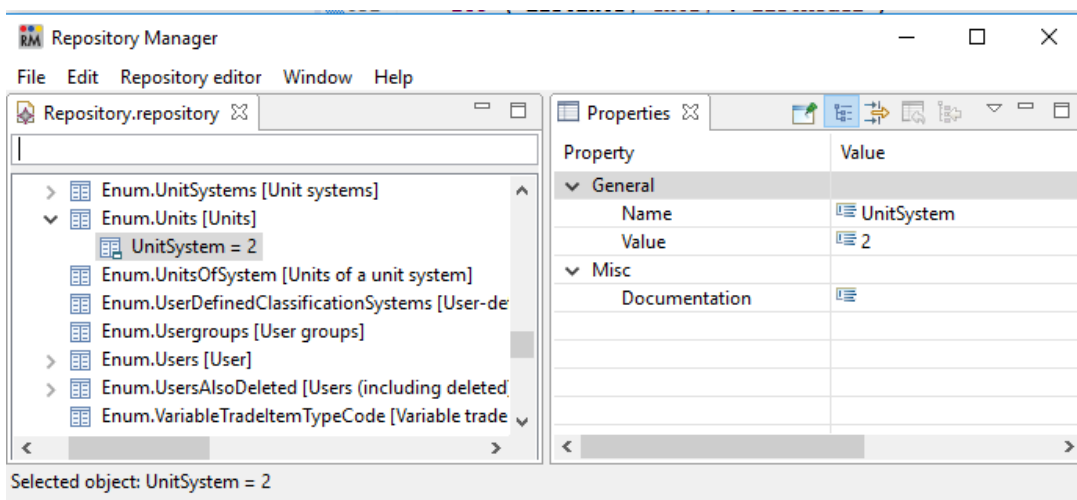
Depending on your needs, you need to make some settings manually.

**i** For all GDSN specific enumerations (e.g. Enum.PreparationType) the standard repository file (repository.repository) only contains header information. The enumeration entries itself can be found in the repository fragment file **EnumFragment.repository**. This file is located in **<PIM\_CONTROL\_CENTER\_INSTALLATION\_FOLDER>/configuration/HPM/gdsn**. It can be opened with the Repository Manager similar to the standard repository file.

Set GDSN unit system as default unit system

By default, the unit system "System units" (2) is used by the "Units" repository enumeration. All GDSN UOM fields have got own unit enumerations according to the GDSN specification.

If you want non-GDSN fields to use only GDSN units you should switch the default unit system to "70".

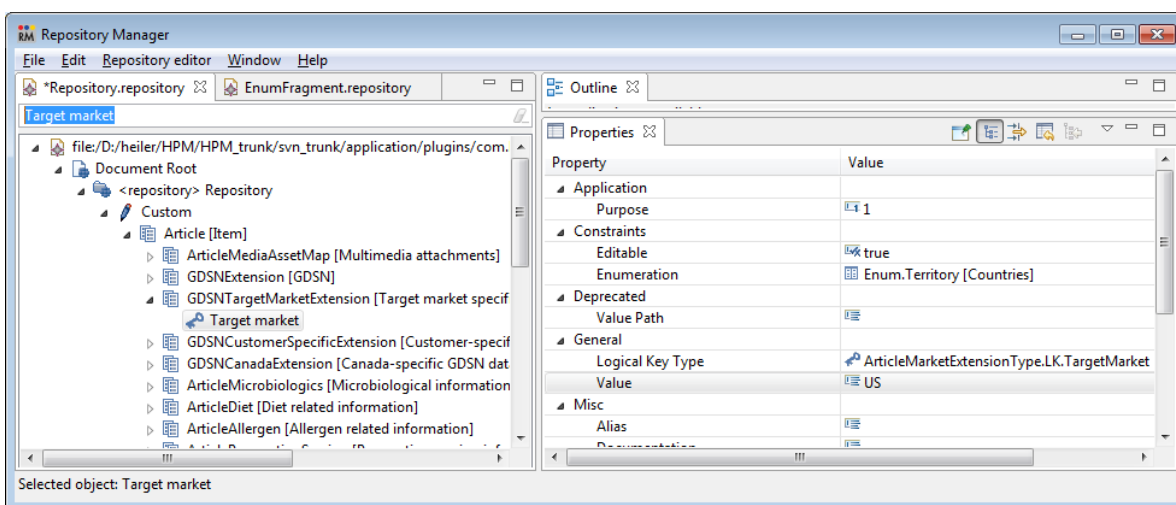


## Change default field qualification

In order to work more efficiently, you should set the default values of some logical keys to your needed value. Note that you don't have to set the values of non-editable or non-visible logical keys and fields; for editable fields it's sufficient to set the logical key value.

## Target market

There's a lot of target market specific fields. If you want to mainly use one dedicated target market you should set the default value of the corresponding logical keys and fields to that target market code.



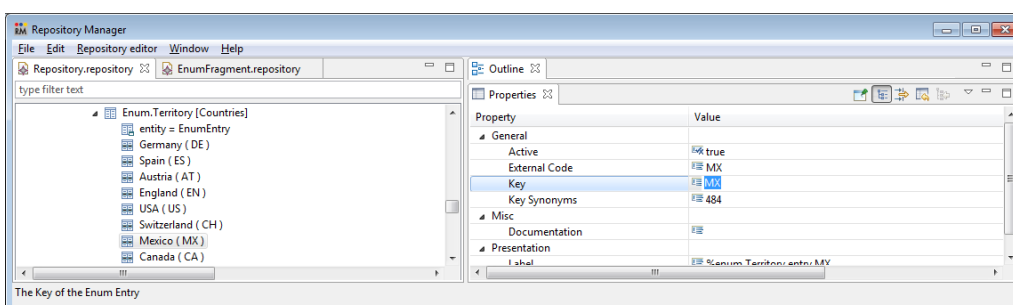
## UOM type

The default uom type value is "metric" (METRIC). If you maintain imperial unit of measures primarily you should change the default value to "IMPERIAL":

## Additional enumeration entries

## Target markets

If needed, enhance the enumeration **Countries** (Enum.Territory) with further target markets.



For each target market, also add an entry in the server's plugin\_customization.ini like the following:

**com.heiler.ppm.gdsn.core/<pool code>.Enum.Territory.<enum\_key> = <TM\_key>** (where <pool code> is 1WS (for IM) or GDSN (for DSE), <enum\_key> is the key of the enum entry, <TM\_key> is the pool specific key of the target market)

#### IM specific target market entries in plugin\_customization.ini

```
# mexico
com.heiler.ppm.gdsn.core/1WS.Enum.Territory.MX = MX
# canada
com.heiler.ppm.gdsn.core/1WS.Enum.Territory.CA = CA
```

#### DSE specific target market entries in plugin\_customization.ini

```
# mexico
com.heiler.ppm.gdsn.core/GDSN.Enum.Territory.MX = 484
# canada
com.heiler.ppm.gdsn.core/GDSN.Enum.Territory.CA = 124
```

### Customizing

#### Hide a GDSN module

When you do not want to deal with an entity which is not needed for your scenario, it is possible to deactivate the entity in the Repository Manager. Select the unwanted entity and edit the "Entity Param scope" of the entity.

The screenshot shows the Repository Manager interface. On the left, a tree view displays the hierarchy of entities. The 'ArticleDataCarrier' entity is selected, and its 'Entity Param scope' property is highlighted with a red box. On the right, a table lists the properties and their values. The 'Entity Param scope' property is listed with the value 'GDSN', which is also highlighted with a red box.

property	value
Documentation	Supported scope: GDSN
Name	scope
Value	GDSN

When deleting the value GDSN those entities won't be activated during server start anymore. Furthermore, the corresponding views are not available in the UI.

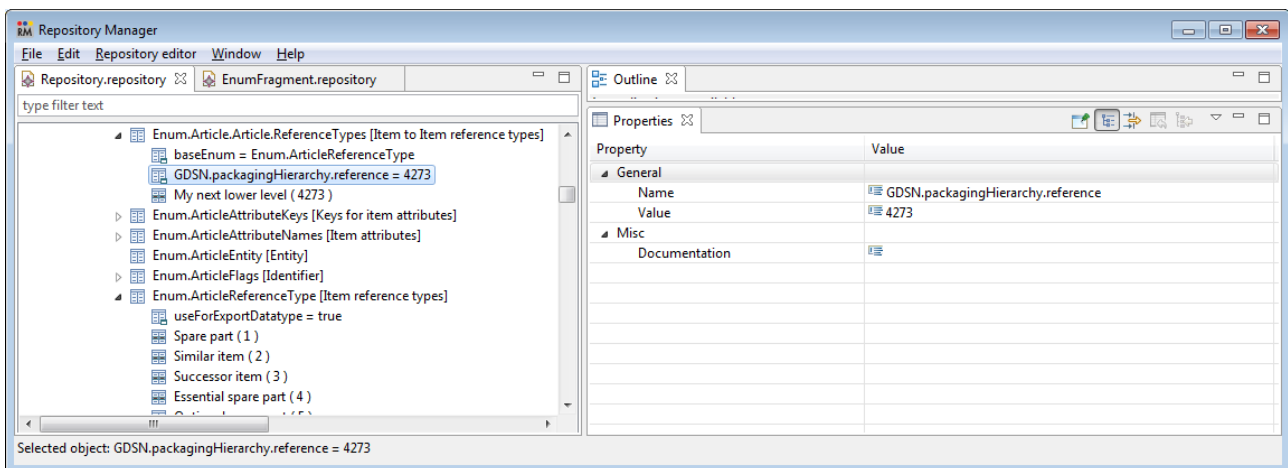
Take care when deactivating an entity because all **references** to the entity in other Product 360 components have to be deleted manually.

Example for these components are:

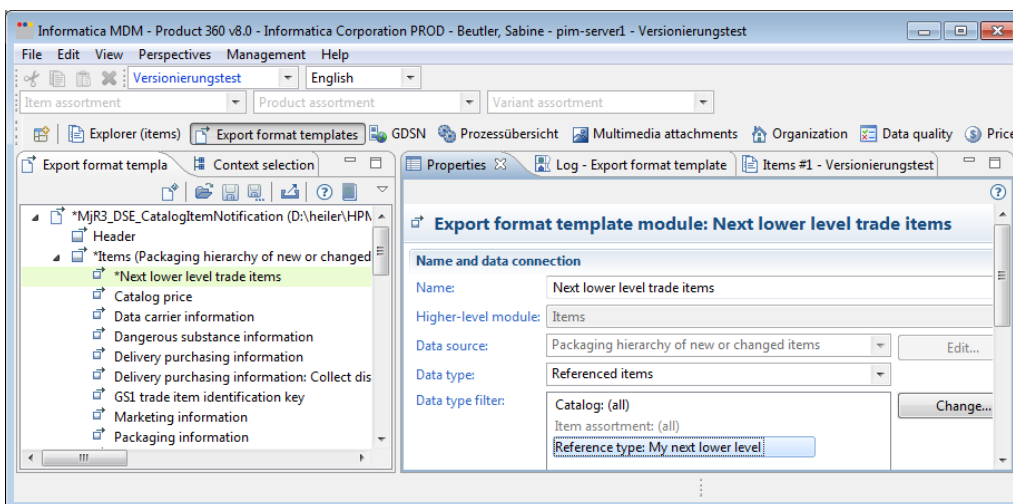
- Export templates
- Import mapping
- Custom article detail XML files
- DQ rule configurations
- Product 360 customizations

Change the "Next lower level" reference type

If you want to use another reference type for the packaging hierarchy of items, you have to ensure this reference type is part of the "Enum.Article.Article.ReferenceTypes" repository enumeration. Furthermore, you have to add an enumeration parameter "GDSN.packagingHierarchy.reference" with your reference type as value to that enumeration.



In case you're using standard GDSN (e. g. atrify) you have to adjust the CatalogItemNotification export template accordingly:



#### 1.4.4.3 Repository Configuration for DSE

There're some additional repository changes needed if you want to use a standard GDSN pool (for example atrify).

Adjust length of repository fields

Field identifier	Field length (repository)	Field length (DSE)
GDSNTargetMarketExtensionLang.FunctionalName	70	35
GDSNTargetMarketExtensionLang.Variant	70	35
Certificate.Value	200	120

**Please note: Although this could be also done automatically during startup we decided to leave it as a manual step. This has mainly transparency reasons but also has the advantage that field length can still be changed by Professional Services, e.g. when the data model is used in another context than GDSN.**

#### 1.4.4.4 Repository Configuration for IM

The out of the box repository is suitable for usage with the Item Management pool. No further manual adjustments are needed.

### 1.5 GDSN Accelerator Setup with OpenAS2

#### 1.5.1 Why use OpenAS2?

For companies interested in sharing information with the GDSN only, and not leveraging the extensive trade partner management capabilities of Informatica B2B, using the OpenAS2 software is a much more lightweight solution that especially in cost-sensitive cloud scenarios can provide a big advantage. As OpenAS2 is a very small application focused on the single use case to send/receive files using the AS2 protocol:

- no database server is required
- less memory is required (~2 GB for files up to 300 MB)
- less CPU power is required

Using fewer resources minimizes the costs of the corresponding infrastructure where the application runs. Due to focusing on this specific use case the installation and configuration efforts are lower, and startup times are quicker, which in return minimizes downtimes and the provisioning of the system.

### **Having everything in one place**

To support the GDSN choreography using OpenAS2, many functionalities have been introduced as standard Product 360 capabilities, leveraging for example the hot folder or the import. The only part which is left to OpenAS2 is encrypting / decryption the messages and the AS2 connection to the GDSN data pool. Having all the functionality in one place is an enormous advantage for any GDSN update which is made on a regular basis. You only have to update one application which lowers the efforts and costs, as well as minimizes the risk any update might contain.

Also when it comes up to customization and adjustment of standard functionality, it is easier for our customers and partners, as they are usually already familiar with the processes, capabilities and the technology of Product 360.

Finally, it is easier for any user as most of the steps including errors are now shown in Product 360 directly and it's not required anymore to check multiple applications to trace the complete process.

## **1.5.2 Official OpenAS2 documentation**

In the installation package of OpenAS2 you can find the OpenAS2HowTo.pdf file which contains a lot of useful information about OpenAS2 in general but also about installation and configuration possibilities. Although most of the things are described in the chapters below. It can help to find solutions in case of any error or if more details are necessary to any configuration.

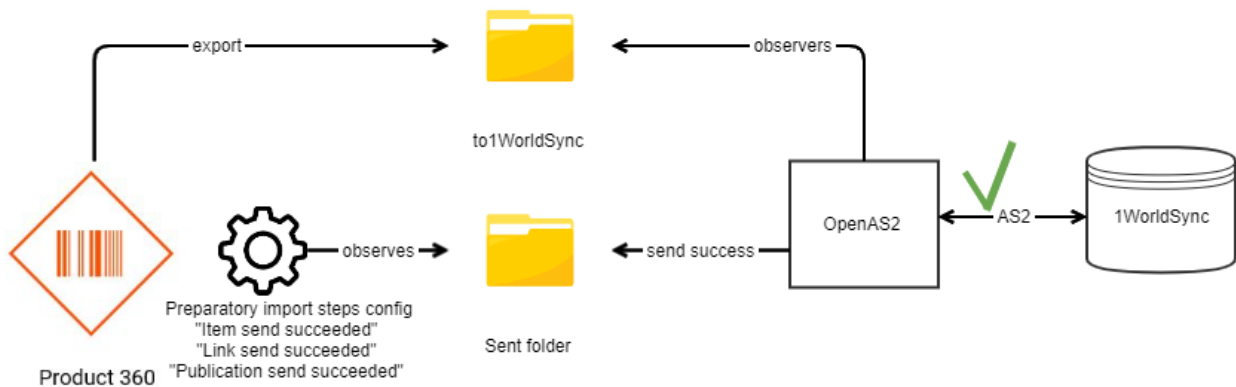
## **1.5.3 How it works**

The communication between Product360 and OpenAS2 is completely file-based. Whenever OpenAS2 receives a message file (e.g. a Publication response) from the pool, the file will be decrypted and then put to the configured folder. This folder has to be observed by Product360 using a hotfolder configuration to pick up the responses and import the corresponding publication status. How to configure OpenAS2 and the hotfolder in order to make the choreography work is described in the following scenario-specific chapters.

Next we describe with an example taken from the IM scenario what happens when you send a message and receive an answer from the pool.

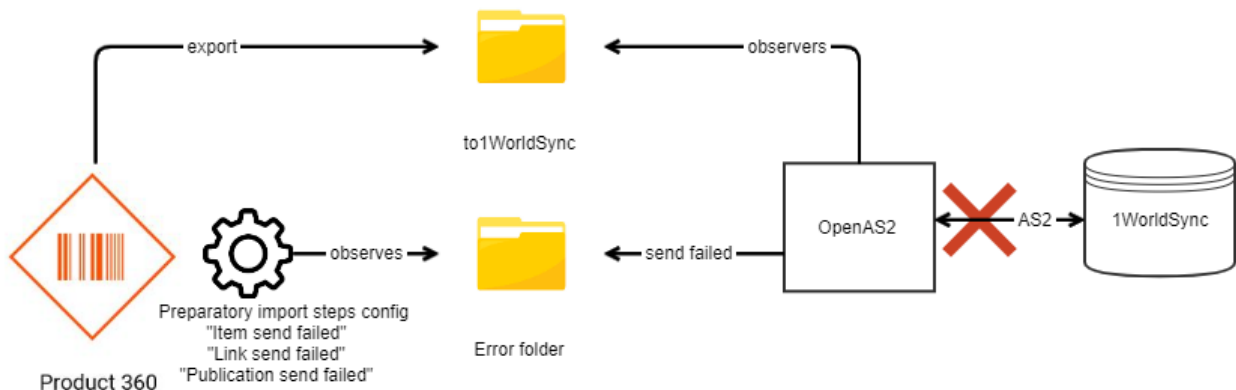
### **1.5.3.1 "Item/Link/Publication send succeeded" communication**

Whenever OpenAS2 succeeds to send a file, the file will be archived and put to the configured sent folder (see the attribute *sentdir* in the corresponding module of your config.xml). We have to configure the pre-import step of the hotfolder configuration to import this file and mark all corresponding items/links/publications as sent successfully by setting the publication status to "Item/Link/Publication transferred -> Acknowledgement".



### 1.5.3.2 "Item/Link/Publication send failed" communication

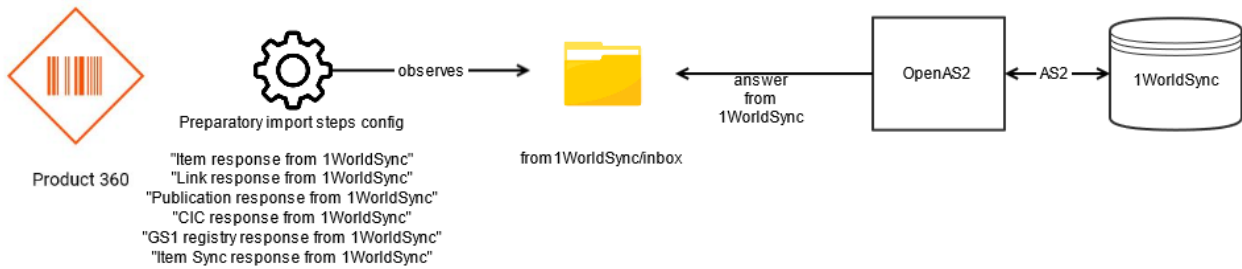
Whenever OpenAS2 is not able to send a file due to any configuration or connection issue, the file will be put to the configured error folder (see the attribute *error\_dir* in the corresponding module of your config.xml). We have to configure the pre-import step to import this file and mark all corresponding items/links/publications as not sent by setting the publication status to "Item/Link/Publication transferred -> Exception".



### 1.5.3.3 "Import responses" communication

We provide the same pre-import step which analyses the message files from the GDSN pool and uses the right import mapping and import the corresponding publication status for an item, link, publication, CIC, GS1 registry or item sync response.



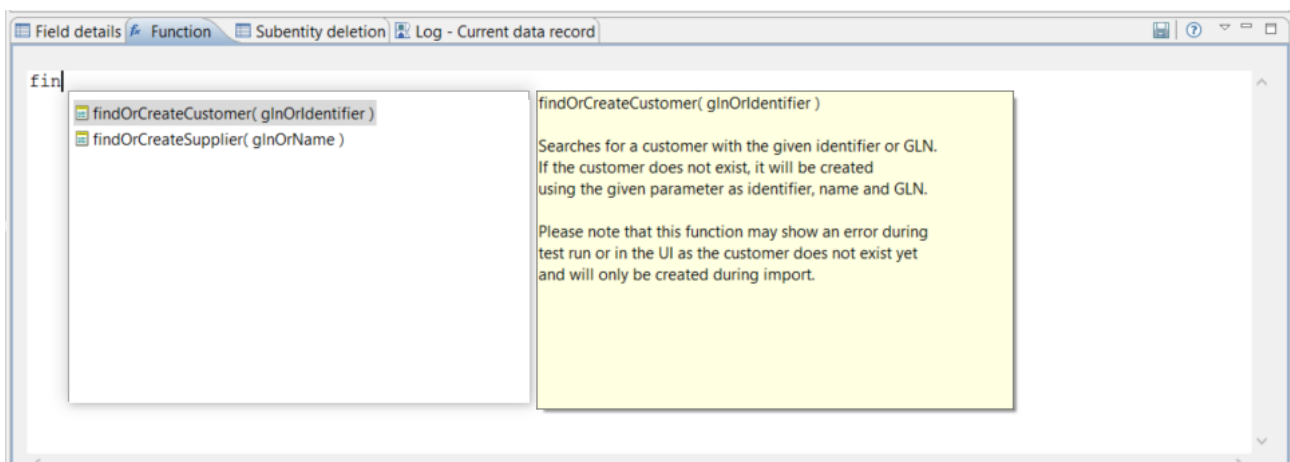


#### 1.5.3.4 Import functions to create customers and suppliers

There may be responses containing unknown customers and suppliers. This is especially true for the CatalogueItemConfirmation, which contains feedback to your items from unknown parties using the Recipient Scenario.

In order to create them in Product360, there are two new import functions "findOrCreateCustomer" and "findOrCreateSupplier". These are used by our out-of-the-box import mappings provided for the hotfolder configuration.

First they look if a customer/supplier already exists in the system. If multiple results are found, the first one is taken. If no result is found, the customer/supplier is created in the system with the given GLN as identifier, name and GLN.



#### 1.5.4 Installation of OpenAS2

1. Go to <https://sourceforge.net/projects/openas2/files> and download the OpenAS2Server zip file, in the version which is mentioned in the Product Availability Matrix.
2. Unzip the file to a directory of your choice, e.g. "C:\OpenAS2". This directory will be referenced in this documentation as <OpenAS2>



The path to your OpenAS2 directory should not contain any whitespaces. This could lead to errors when starting OpenAS2.

#### 1.5.4.1 Upgrading OpenAS2 from 2.X or older to 3.x



When updating an existing OpenAS2 installation please refer to the upgrade documentation provided by OpenAS2 inside the Release Notes file.

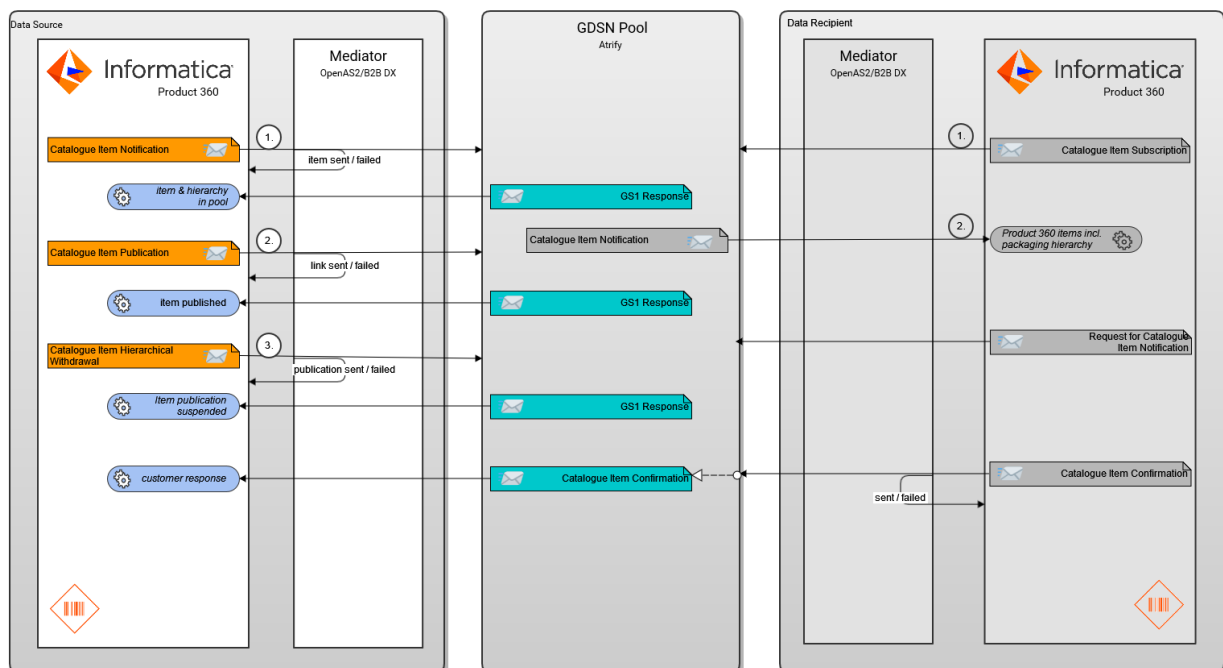
RELEASE-NOTES.md

### 1.5.5 Standard GDSN

#### 1.5.5.1 GDSN Data Source Scenario

##### Message Choreography

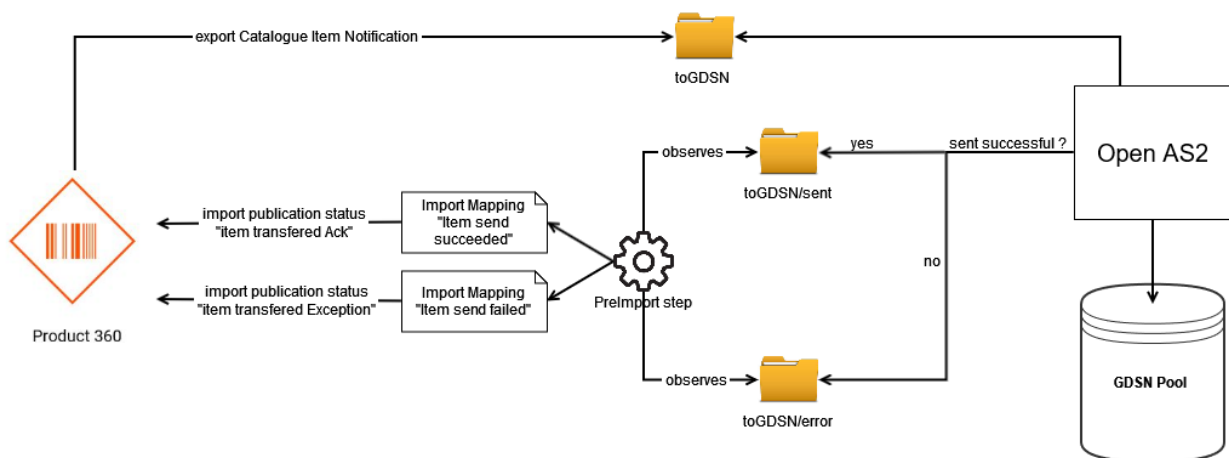
By sending a "Catalogue Item Notification" new items are added or existing ones are modified in the GDSN pool. In contrast to IM where items are linked by a separate message later, Standard GDSN messages contain one or multiple complete item hierarchies. When all needed item data has been added to the GDSN pool and the hierarchies were created, it is possible to publish it for a specific customer (data recipient) or market groups by sending a "Catalogue Item Publication" message. By doing this the data recipient has the ability to view and synchronize the published item including all the child items below that item. An existing publication for a specific customer or market group can be deleted.



The choreography "send Catalogue Item Notification" in more detail

With OpenAS2 not much things changed. Product 360 still leverages the export and put the GDSN message file to a folder. Instead of Informatica B2B, OpenAS2 will pick it up, encrypt it and send it to 1WorldSync. After receiving the confirmation, that the message could be delivered\* successfully, the message will be put to sent folder. In case something goes wrong during the encryption or sending process, OpenAS2 will try to resend it several times and if it was still not successful the message will be put to the error folder.

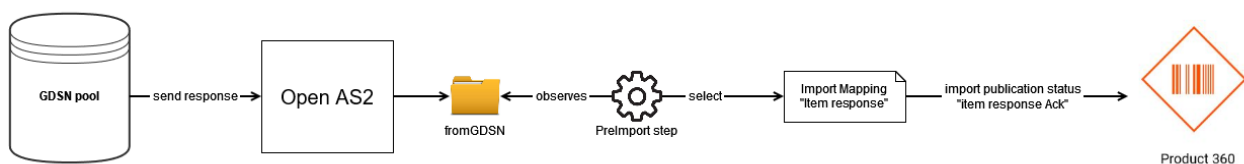
The sent as well as the error folder are configured as hot folders in Product 360 and so files will be picked up and processed by a preparatory import step. This preparatory import step will analyze the name and/or the content of the file and select the corresponding mapping to import a Publication status for each item in the file. Creating this Publication status was done before by Informatica B2B via the Service API of Product 360.



**i** \*Please note that "delivered" does only mean that the GDSN pool received the message, but neither opened nor validated it. The real GS1 Response will be sent asynchronously and is shown in the next chapter.

The choreography "receive GS1 Response" in more detail

After sending out a Catalogue Item Notification, the GDSN pool will send a GS1 Response asynchronously. OpenAS2 will receive the response, decrypt it and put it to the folder "fromGDSN" which is also observed by Product 360. Also here the configured preparatory import step will analyze the name and content of the file and selects the corresponding mapping to import a publication status for each item in the file.



## Other messages

All other messages as the "Catalogue Item Publication" etc... will be processed the same way as the "Catalogue Item Notification" which was visualized above.

## OpenAS2 Configuration

There are two main configuration files which are relevant to each installation and have to be configured, config.xml and partnership.xml. Templates for both files can be found in the *config* directory of the OpenAS2 installation package.

### config.xml

This file contains common settings, it configures the modules that will be activated by OpenAS2 server when it starts up. The following attributes are mandatory to be adjusted according to your pool registration information.

### Properties

These are the global properties used for communication between sender and receiver in your scenario.

Attribute	Example value	Description
storageBaseDir	%home%/../data	The Base storage directory, used to put and read messages to and from the receiver
as2_async_mdn_url	http://194.165.163.148:5080/as2	The public IP Address and its receiving folder used to receive messages from your communication partner
customer_name	Informatica	The communication partner name, used as the sender in this scenario

### Certificates

These are the certificate attributes used for message encryption.

Attribute	Example value	Description
filename	%home%/infa_gdsn_test.p12	The path to your p12 keystore file
password	testas2	The password with which your keystore has been encrypted with

### Partnerships

Here you can configure the partnership between your end and your communication partners.

Attribute	Example value	Description
filename	%home%/partnerships.xml	The path to your partnership.xml file

### Processor

Finally, you can configure your processor modules in order to define how the messages are to be processed. You need to add 4 more modules in order to configure the message processing correctly.



Directories you define here are also used by inbox hotfolder and export. Incoming message are usually automatically picked up by the observation mechanism of the inbox hotfolder, therefore the directories defined here and defined in the inbox hotfolder configuration have to be the same. (See [Appendix: Import mappings and hotfolder configuration \(see page 205\)](#))

In order to send exported files automatically the target folder configured in the "Copy file(s)" post export step of every GDSN export template has also to be configured here as outbox directory. (See [GDSN Accelerator operation/Export Templates \(see page 89\)](#))

### Send Message processing

Here you can define, how messages that you send to your communication partner are being processed.

Attribute	Example value	Description
classname	org.openas2.processor.receiver.AS2DirectoryPollingModule	The polling module to take in the file to be processed

Attribute	Example value	Description
outboxdir	\$properties.storageBaseDir\$/toAtrify	The path to the directory from which to send messages to your communication partner
errordir	\$properties.storageBaseDir\$/toAtrify/ error	The path to the error directory from which error message are to be received at in case of an error
stored_error_filename	OPENAS2- \$rand.UUID\$@\$msg.attributes.filename\$ _failed_toAtrify	The filename pattern of the error messages sent to your communication partner
sentdir	\$properties.storageBaseDir\$/toAtrify/ sent	The directory of the message files to be sent to you communication partner
stored_sent_filename	OPENAS2- \$rand.UUID\$@\$msg.attributes.filename\$ _succeeded_toAtrify	The filename pattern of the successfully sent files to your communication partner
defaults	sender.as2_id=INFA_DEMO05_DS_EU, receiver.as2_id=4049111170017	Your own AS2ID as the sender as well as the AS2ID of your communication partner as the receiver

**Received Message processing**

Here you can define, how messages that your communication partner sends and which you are receiving, are being processed.

Attribute	Example value	Description
classname	org.openas2.processor.receiver.AS2D irectoryPollingModule	The polling module to take in the file to be processed
outboxdir	\$properties.storageBaseDir\$/ fromAtrify	The path to the directory from which to receive messages of your communication partner

Attribute	Example value	Description
errordir	\$properties.storageBaseDir\$/fromAtrify/error	The path to the error directory from which to receive error messages from your communication partner
defaults	sender.as2_id=4049111170017, receiver.as2_id=INFA_DEMO05_DS_EU	The AS2ID of your communication partner as the sender as well as the AS2ID of yourself as the receiver

#### Error Message processing

Here you can define how errors that occurred when your communication partner sent a message are supposed to be processed.

Attribute	Example value	Description
classname	org.openas2.processor.receiver.AS2ReceiverModule	The receiver module to process received messages with
port	5080	The port for which to listen for error messages
errordir	\$properties.storageBaseDir\$/fromAtrify/inbox/error	The path to the directory to receive incoming errors from

#### Message resend processing

Finally you can also define how messages that should be resent, e.g. due to faulty communication, should be resent.

Attribute	Example value	Description
classname	org.openas2.processor.resender.DirectoryResenderModule	The resender module which resends failed messages in case of a network error

Attribute	Example value	Description
resenddir	\$properties.storageBaseDir\$/resend	The path to the directory where resent messages should be sent from
errordir	\$properties.storageBaseDir\$/toAtrify/error	The path to the error directory from which error messages are to be received at in case of an error

partnerships.xml

The partnerships.xml contains information about the connection details between the trade partners such as AS2 ids, certificates and URLs. Usually, there are two partners, you and the data pool.

In this file one partner configuration is defined for each partner as well as one partnership configuration per communication channel (you → data pool, data pool → you).

#### Source Partner configuration

Attribute	Example value	Description
name	Informatica	The name of this partner. In this case your company name
as2_id	INFA_DEMO05_DS_EU	The AS2ID of this partner. in this case your company AS2ID
x509_alias	infa_dev_to_atrify	The certificate name in your keystore file
email	someName@informatica.com	The email address to which notifications can be send to

#### Receiver Partner Configuration



Attribute	Example value	Description
name	Atrify	The name of this partner. In this case your communication partner name
as2_id	4049111170017	The AS2ID of this partner. in this case your communication partner's AS2ID
x509_alias	atrify	The certificate name in your keystore file
email	Info@Atrify.com	The email address to which notifications can be sent to. In this case the email address of your communication partner

#### Source Partnership configuration

Attribute	Example value	Description
partnership name	Informatica-to-Atrify	The name of this partnership configuration. It makes sense to name it in favor of the actual communication direction
sender	Informatica	The partner name which sends messages to the receiver
receiver	Atrify	The partner name which receives messages from the sender
attribute - subject	Sent from \$sender.name\$ to \$receiver.name\$	The subject of each message handled in this partnership. Usually used for message processing automation
attribute - as2_url	http://as2-datasync-test.atrify.com:80/as2	The public address to which messages are to be sent to

Attribute	Example value	Description
attribute - as2_mdn_to	http://as2-datasync-test.atrify.com:80/as2	The public MDN address to which messages are to be sent to

#### Receiver Partnership Configuration

Attribute	Example value	Description
partnership name	Atrify-to-Informatica	The name of this partnership configuration. It makes sense to name it in favor of the actual communication direction
sender	Atrify	The partner name which sends messages to the receiver
receiver	Informatica	The partner name which receives messages from the sender. In this case, you yourself
attribute - subject	File sent from Atrify to us	The subject of each message handled in this partnership. Usually used for message processing automation
attribute - as2_url	http://194.165.163.148:5080/as2	The public address to which messages are to be sent to. In this case, your own public IP Address where you expect answers to be received from
attribute - as2_mdn_to	http://194.165.163.148:5080/as2	The public MDN address to which messages are to be sent to. In this case, your own public IP Address where you expect answers to be received from

## Product 360 Configuration

### Import Hotfolder configuration

Whenever OpenAS2 receives a message file, the file will be put to the configured folder. This folder has to be observed by Product360 to pick up the responses and import the corresponding publication status.

Please see [Appendix: Import mappings and hotfolder configuration \(see page 205\)](#) on how to setup the hotfolder configuration. The import mappings are located in the GDSN accelerator resources package under \DataSource\DSE\OpenAS2\ImportMappings.

### Export configuration



You can find a general description how to load and save export templates and xsd files in [Appendix: Export templates \(see page 194\)](#).

Note: If you want to export to the GDSN Standard pool adjust the repository like described in [Configuration for DSE \(see page 29\)](#).

### Export templates

The export templates can be found in DataSource/DSE/OpenAS2/ExportTemplates:

- CIN\_CatalogItemNotification.ext
- CIP\_CatalogItemPublication.ext
- CIPHW\_CatalogItemPublicationHierarchyWithdrawal.ext

Load and save them in Product360.

### XSD schema files

The XSD schema files are located in DataSource/DSE/OpenAS2/ExportTemplates/XSDs

Upload the xsd files to the system.

### XSLT files

In DataSource/DSE/OpenAS2/ExportTemplates/PostExport there are four XSLT files needed to transform the export files into a file set ready to be submitted to the GDSN pool. Based on the filenames, you can see to which export template and post step they should be assigned.

Note: for "CIN\_CatalogItemNotification" you have to configure two "XSL Transformation" export steps.

<b>XSLT file</b>	<b>Export template</b>	<b>Post step</b>
CIN_CatalogueItemNotification_FileSplit.xslt	CIN_CatalogueItemNotification	XSL transformation(file split)
CIN_CatalogueItemNotification_Hierarchy.xslt	CIN_CatalogueItemNotification	XSL transformation(hierarchy)
CIP_CatalogueItemPublication_FileSplit .xslt	CIP_CatalogueItemPublication	XSL transformation(file split)
CIPHW_CatalogItemPublicationHierarchyWithdrawal_FileSplit.xslt	CIPHW_CatalogItemPublicationHierarchyWithdrawal	XSL transformation(file split)

#### Background information

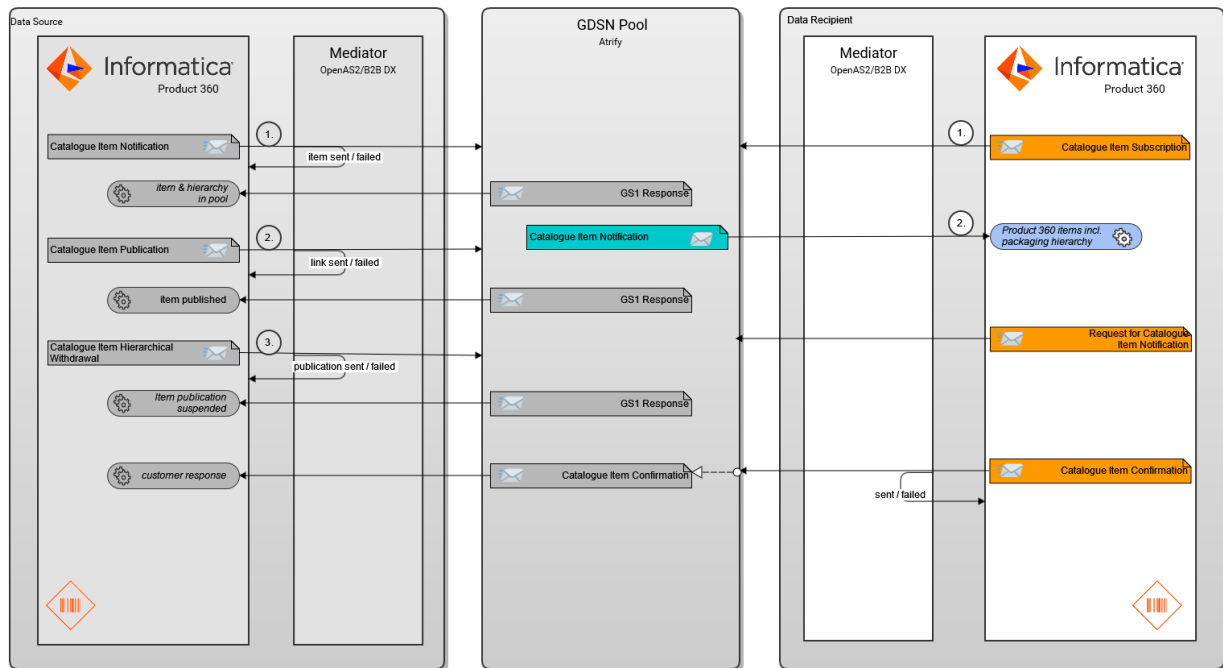
IM, like DSE, allows a file only to contain a certain number of documents (items respectively). That's why every export template uses a post export step to split the export file every 100 documents and will attach header and footer in order to create a valid file. The files have the same name as configured in the export template but a consecutive number is added at the end.

Additionally, Standard GDSN requires item data to be sent in hierarchies. For this purpose, there is a second post export step that takes the export file containing the items in a flat arrangement and reorganizes them into hierarchies based on their linked GTINs. This happens before the file split .

#### 1.5.5.2 GDSN Data Recipient Scenario

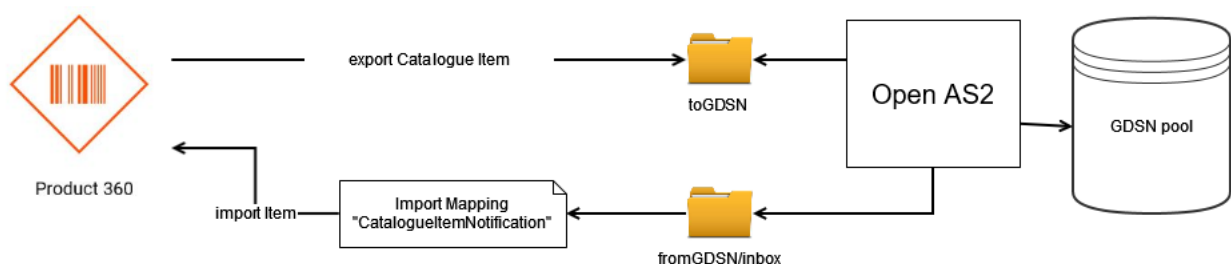
##### Message Choreography

In the Data Recipient Scenario the standard choreography requires the user to first send a Catalogue Item Subscription Message of the required GLN or GTIN. If a GTIN is requested, only the item belonging to that GTIN will be subscribed to. If a whole GLN is subscribed to, all items belonging to that GLN will be requested. Next, on PIM side, the user can create a Confirmation Status. This status provides information about the actual state of the subscribed item. This state can now be sent to the pool by sending a Catalogue Item Confirmation which contains the created status.



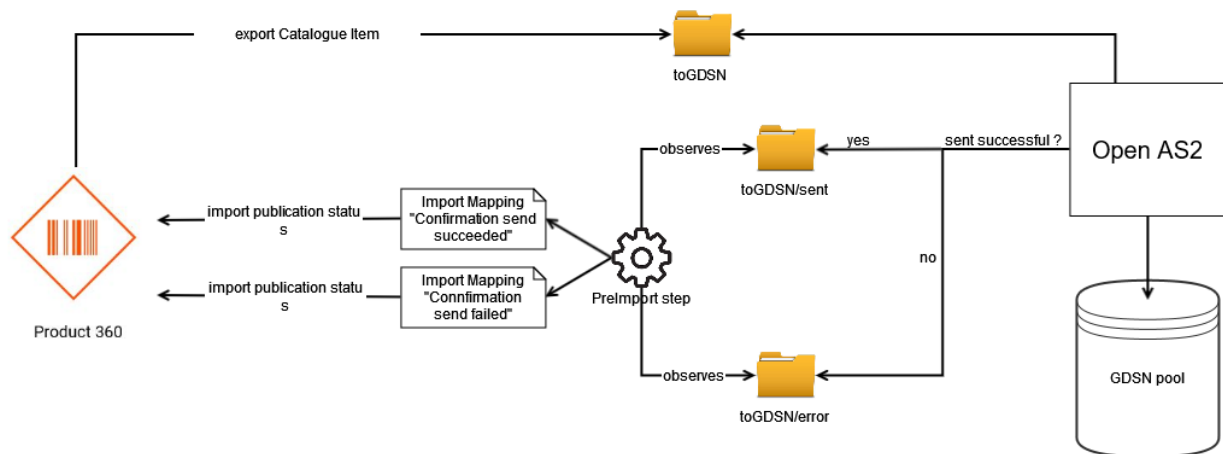
## Catalogue Item Subscription and Catalogue Item Notification

Sending a Catalog Item Subscription, will first export the required data of the selection into the toGDSN folder. The response will not be imported since there is not yet an item this information belongs to. After some time, if the item was successfully subscribed to, the requested data (Catalogue Item Notification message) will be transferred into the fromGDSN/inbox folder. Here the data will be imported by Product 360 and will finally appear in your client.



## Catalogue Item Confirmation

After creating a Confirmation status, the user is now able to send that status to the Pool via the Catalogue Item Confirmation Message. Such a message is first exported to the to1WorldSync folder and processed by OpenAS2. Next, depending on its success, the resulting status will now be written by OpenAS2 and imported by Product360.



### OpenAS2 Configuration

There are two main configuration files which are relevant to each installation and have to be configured, `config.xml` and `partnership.xml`. Templates for both files can be found in the `config` directory of the OpenAS2 installation package.

#### `config.xml`

This file contains common settings, it configures the modules that will be activated by OpenAS2 server when it starts up. The following attributes are mandatory to be adjusted according to your pool registration information.

#### Properties

These are the global properties used for communication between sender and receiver in your scenario.

Attribute	Example value	Description
storageBaseDir	%home%/../data	The Base storage directory, used to put and read messages to and from the receiver
as2_async_mdn_url	http://194.165.163.148:5080/as2	The public IP Address and its receiving folder used to receive messages from your communication partner
customer_name	Informatica	The communication partner name, used as the sender in this scenario

## Certificates

These are the certificate attributes used for message encryption.

Attribute	Example value	Description
filename	%home%/infa_gdsn_test.p12	The path to your p12 keystore file
password	testas2	The password with which your keystore has been encrypted with

## Partnerships

Here you can configure the partnership between your end and your communication partners.

Attribute	Example value	Description
filename	%home%/partnerships.xml	The path to your partnership.xml file

## Processor

Finally, you can configure your processor modules in order to define how the messages are to be processed. You need to add 4 more modules in order to configure the message processing correctly.



Directories you define here are also used by inbox hotfolder and export. Incoming message are usually automatically picked up by the observation mechanism of the inbox hotfolder, therefore the directories defined here and defined in the inbox hotfolder configuration have to be the same. (See [Appendix: Import mappings and hotfolder configuration \(see page 205\)](#))

In order to send exported files automatically the target folder configured in the "Copy file(s)" post export step of every GDSN export template has also to be configured here as outbox directory. (See [GDSN Accelerator operation/Export Templates \(see page 89\)](#))

## Send Message processing

Here you can define, how messages that you send to your communication partner are being processed.

Attribute	Example value	Description
classname	org.openas2.processor.receiver.AS2DirectoryPollingModule	The polling module to take in the file to be processed
outboxdir	\$properties.storageBaseDir\$/toGDSN	The path to the directory from which to send messages to your communication partner
errordir	\$properties.storageBaseDir\$/toGDSN/error"	The path to the error directory from which error message are to be received at in case of an error
stored_error_filename	OPENAS2-\$rand.UUID\$@\$msg.attributes.filename\$_failed_toGDSN	The filename pattern of the error messages sent to your communication partner
senddir	\$properties.storageBaseDir\$/toGDSN/sent	The directory of the message files to be sent to you communication partner
stored_sent_filename	OPENAS2-\$rand.UUID\$@\$msg.attributes.filename\$_succeeded_toGDSN	The filename pattern of the successfully sent files to your communication partner
defaults	sender.as2_id=INFA_DEMO05_DR, receiver.as2_id=4049111170017	Your own AS2ID as the sender as well as the AS2ID of your communication partner as the receiver

#### Received Message processing

Here you can define, how messages that your communication partner sends and which you are receiving, are being processed.

Attribute	Example value	Description
classname	org.openas2.processor.receiver.AS2DirectoryPollingModule	The polling module to take in the file to be processed



Attribute	Example value	Description
outboxdir	\$properties.storageBaseDir\$/fromGDSN	The path to the directory from which to receive messages of your communication partner
errordir	\$properties.storageBaseDir\$/fromGDSN/error	The path to the error directory from which to receive error messages from your communication partner
defaults	sender.as2_id=4049111170017, receiver.as2_id=INFA_DEMO05_DR	The AS2ID of your communication partner as the sender as well as the AS2ID of yourself as the receiver

#### Error Message processing

Here you can define how errors that occurred when your communication partner sent a message are supposed to be processed.

Attribute	Example value	Description
classname	org.openas2.processor.receiver.AS2ReceiverModule	The receiver module to process received messages with
port	5080	The port for which to listen for error messages
errordir	\$properties.storageBaseDir\$/fromGDSN/inbox/error	The path to the directory to receive incoming errors from

#### Message resend processing

Finally you can also define how messages that should be resent, e.g. due to faulty communication, should be resent.

Attribute	Example value	Description
classname	org.openas2.processor.resender.DirectoryResenderModule	The resender module which resends failed messages in case of a network error
resenddir	\$properties.storageBaseDir\$/resend	The path to the directory where resent messages should be sent from
errordir	\$properties.storageBaseDir\$/toGDSN/error	The path to the error directory from which error messages are to be received at in case of an error

partnerships.xml

The partnerships.xml contains information about the connection details between the trade partners such as AS2 ids, certificates and URLs. Usually, there are two partners, you and the data pool.

In this file one partner configuration is defined for each partner as well as one partnership configuration per communication channel (you → data pool, data pool → you).

#### Source Partner configuration

Attribute	Example value	Description
name	Informatica	The name of this partner. In this case your company name
as2_id	INFA_DEMO05_DR	The AS2ID of this partner. in this case your company AS2ID
x509_alias	infa_dev_from_atrify	The certificate name in your keystore file
email	someName@informatica.com	The email address to which notifications can be send to

#### Receiver Partner Configuration

Attribute	Example value	Description
name	Atrify	The name of this partner. In this case your communication partner name
as2_id	4049111170017	The AS2ID of this partner. in this case your communication partner's AS2ID
x509_alias	atrify	The certificate name in your keystore file
email	Info@Atrify.com	The email address to which notifications can be sent to. In this case the email address of your communication partner

#### Source Partnership configuration

Attribute	Example value	Description
partnership name	Informatica-to-Atrify	The name of this partnership configuration. It makes sense to name it in favor of the actual communication direction
sender	Informatica	The partner name which sends messages to the receiver
receiver	Atrify	The partner name which receives messages from the sender
attribute - subject	Sent from \$sender.name\$ to \$receiver.name\$	The subject of each message handled in this partnership. Usually used for message processing automation
attribute - as2_url	http://as2-datasync-test.atrify.com:80/as2	The public address to which messages are to be sent to

Attribute	Example value	Description
attribute - as2_mdn_to	http://as2-datasync-test.atrify.com:80/as2	The public MDN address to which messages are to be sent to

**Receiver Partnership Configuration**

Attribute	Example value	Description
partnership name	Atrify-to-Informatica	The name of this partnership configuration. It makes sense to name it in favor of the actual communication direction
sender	Atrify	The partner name which sends messages to the receiver
receiver	Informatica	The partner name which receives messages from the sender. In this case, you yourself
attribute - subject	File sent from Atrify to us	The subject of each message handled in this partnership. Usually used for message processing automation
attribute - as2_url	http://194.165.163.148:5080/as2	The public address to which messages are to be sent to. In this case, your own public IP Address where you expect answers to be received from
attribute - as2_mdn_to	http://194.165.163.148:5080/as2	The public MDN address to which messages are to be sent to. In this case, your own public IP Address where you expect answers to be received from

## Product 360 Configuration

### Import Hotfolder configuration

Whenever OpenAS2 receives a message file, the file will be put to the configured folder. This folder has to be observed by Product360 to pick up the responses and import the corresponding publication status.

Please see [Appendix: Import mappings and hotfolder configuration \(see page 205\)](#) on how to setup the hotfolder configuration. The import mappings are located in the GDSN accelerator resources package under \DataRecipient\DSE\OpenAS2\Import.

### Export configuration



You can find a general description how to load and save export templates and xsd files in [Appendix: Export templates \(see page 194\)](#).

### Export templates

The export templates can be found in DataRecipient/DSE/OpenAS2/ExportTemplates:

- CIC\_CatalogItemConfirmation.ext
- CIS\_CatalogItemSubscription.ext
- RFCIN\_RequestForCatalogItem.ext

Load and save them in Product360.

### XSD schema files

The XSD schema files are located in DataRecipient/DSE/OpenAS2/ExportTemplates/XSDs

Upload the xsd files to the system.

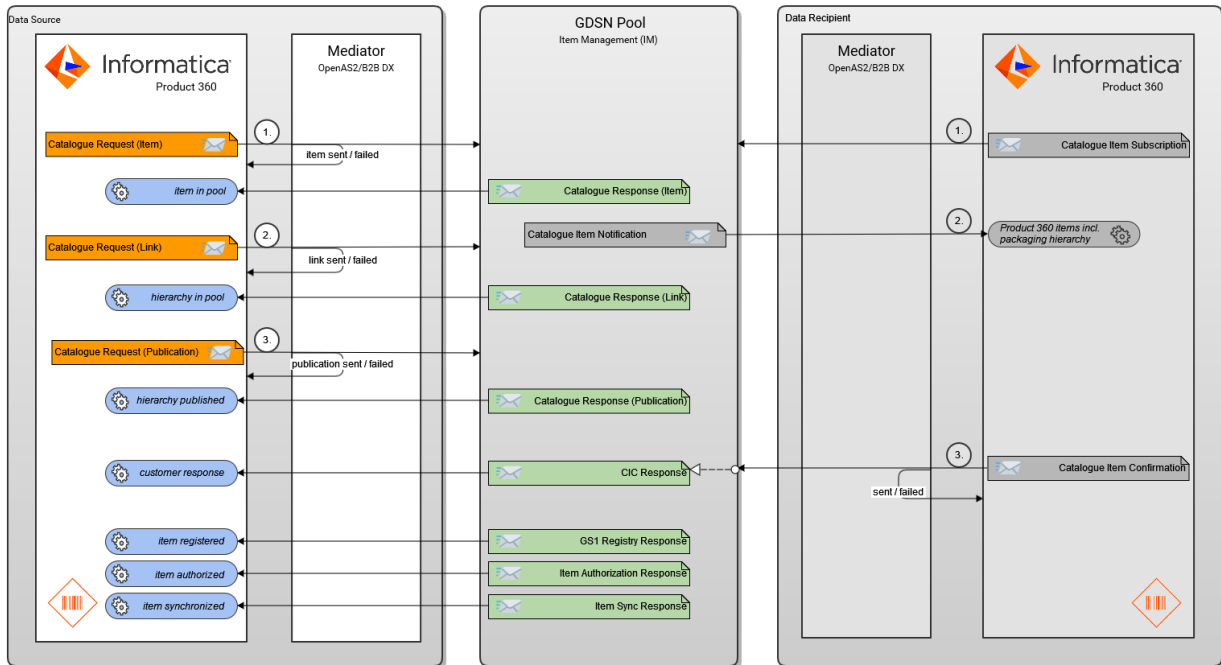
## 1.5.6 Item Management

### 1.5.6.1 IM Data Source Scenario

#### Message Choreography

By sending a "Catalogue Request" message of type "Item" new items are added or existing ones are modified in the GDSN pool. After all necessary items have been successfully added to the GDSN pool, hierarchies can be created by linking these items together. Items can also be removed from hierarchies by unlinking them again. This is done by sending a "Catalogue Request" message of type "Link". When all needed item data has been added to the GDSN pool and the hierarchies were created, it is possible to publish it for a specific

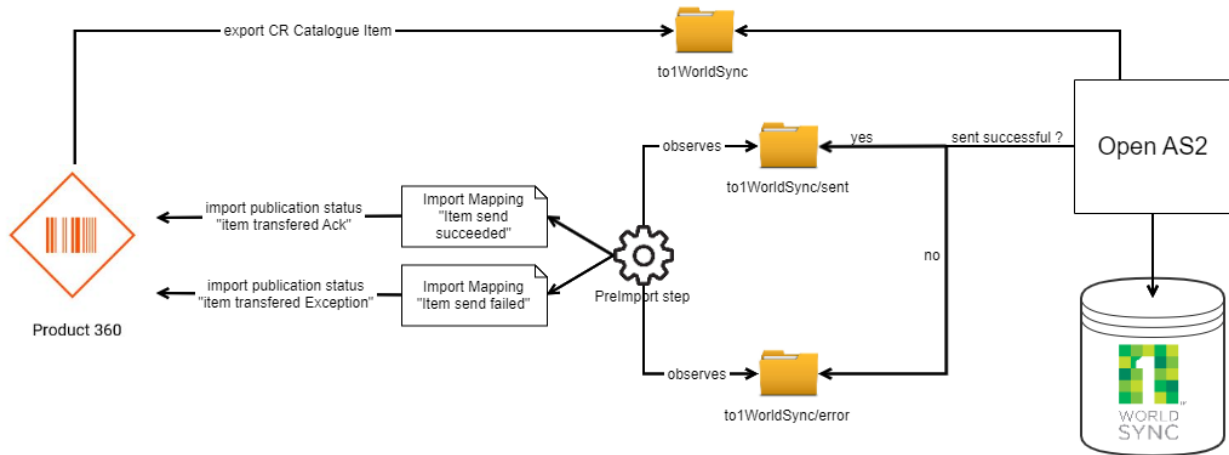
customer (data recipient) or market groups by sending a "Catalogue Request" of type "Publication" message. By doing this the data recipient has the ability to view and synchronize the published item including all the child items below that item. An existing publication for a specific customer or market group can be deleted.



The choreography "send Catalogue Request item" in more detail

With OpenAS2 not much things changed. Product 360 still leverages the export and put the GDSN message file to a folder. Instead of Informatica B2B, OpenAS2 will pick it up, encrypt it and send it to 1WorldSync. After receiving the confirmation, that the message could be successful delivered\*, the message will be put to sent folder. In case something goes wrong during the encryption or sending process, OpenAS2 will try to resend it several times and if it was still not successful the message will be put to the error folder.

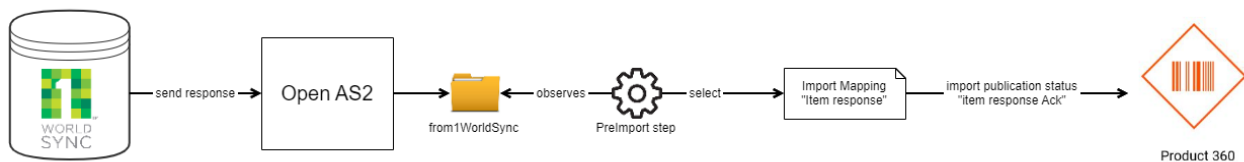
The sent as well as the error folder are configured as hot folders in Product 360 and so files will be picked up and processed by a preparatory import step. This preparatory import step will analyze the name and/or the content of the file and select the corresponding mapping to import a Publication status for each item in the file. Creating this Publication status was done before by Informatica B2B via the Service API of Product 360.



**i** \*Please note that "delivered" does only mean that 1WorldSync received the message, but neither opened nor validated it. The real Catalogue Response will be sent asynchronously and is shown in the next chapter.

The choreography "receive Catalogue Response item" in more detail

After sending out a Catalogue Request item, 1WorldSync will send a Catalogue Response item asynchronously. OpenAS2 will receive the response, decrypt it and put it to the folder "from1WorldSync" which is also observed by Product 360. Also here the configured preparatory import step will analyze the name and content of the file and selects the corresponding mapping to import a publication status for each item in the file.



## Other messages

All other messages as the "Catalogue Request Link", "Catalogue Request publication" etc... will be processed the same way as the "Catalogue Request Item" which was visualized above.

## OpenAS2 Configuration

There are two main configuration files which are relevant to each installation and have to be configured, config.xml and partnership.xml. Templates for both files can be found in the *config* directory of the OpenAS2 installation package.

config.xml

This file contains common settings, it configures the modules that will be activated by OpenAS2 server when it starts up. The following attributes are mandatory to be adjusted according to your pool registration information.

#### Properties

These are the global properties used for communication between sender and receiver in your scenario.

Attribute	Example value	Description
storageBaseDir	%home%/../data	The Base storage directory, used to put and read messages to and from the receiver
as2_async_mdn_url	http://194.165.163.149:5080	The public IP Address and its receiving folder used to receive messages from your communication partner
customer_name	Informatica	The communication partner name, used as the sender in this scenario

#### Certificates

These are the certificate attributes used for message encryption.

Attribute	Example value	Description
filename	%home%/as2_certs.p12	The path to your p12 keystore file
password	testas2	The password with which your keystore has been encrypted with

#### Partnerships

Here you can configure the partnership between your end and your communication partners.



Attribute	Example value	Description
filename	%home%/partnerships.xml	The path to your partnership.xml file

#### Processor

Finally, you can configure your processor modules in order to define how the messages are to be processed. You need to add 4 more modules in order to configure the message processing correctly.



Directories you define here are also used by inbox hotfolder and export. Incoming message are usually automatically picked up by the observation mechanism of the inbox hotfolder, therefore the directories defined here and defined in the inbox hotfolder configuration have to be the same. (See [Appendix: Import mappings and hotfolder configuration \(see page 205\)](#))

In order to send exported files automatically the target folder configured in the "Copy file(s)" post export step of every GDSN export template has also to be configured here as outbox directory. (See [GDSN Accelerator operation/Export Templates \(see page 89\)](#))

#### Send Message processing

Here you can define, how messages that you send to your communication partner are being processed.

Attribute	Example value	Description
classname	org.openas2.processor.receiver.AS2DirectoryPollingModule	The polling module to take in the file to be processed
outboxdir	\$properties.storageBaseDir\$/to1WorldSync	The path to the directory from which to send messages to your communication partner
errordir	\$properties.storageBaseDir\$/to1WorldSync/error	The path to the error directory from which error message are to be received at in case of an error
stored_error_filename	OPENAS2-\$rand.UUID\$@\$msg.attributes.filename\$_failed_to1WorldSync	The filename pattern of the error messages sent to your communication partner

Attribute	Example value	Description
senddir	\$properties.storageBaseDir\$/to1WorldSync/sent	The directory of the message files to be sent to you communication partner
stored_sent_filename	OPENAS2-\$rand.UUID\$@\$msg.attributes.filename\$_succeeded_to1WorldSync	The filename pattern of the successfully sent files to your communication partner
defaults	sender.as2_id=INFA_DEV_8_DS, receiver.as2_id=8380160030003	Your own AS2ID as the sender as well as the AS2ID of your communication partner as the receiver

#### Received Message processing

Here you can define, how messages that your communication partner sends and which you are receiving, are being processed.

Attribute	Example value	Description
classname	org.openas2.processor.receiver.AS2DirectoryPollingModule	The polling module to take in the file to be processed
outboxdir	\$properties.storageBaseDir\$/from1WorldSync	The path to the directory from which to receive messages of your communication partner
errordir	\$properties.storageBaseDir\$/from1WorldSync/error	The path to the error directory from which to receive error messages from your communication partner
defaults	sender.as2_id=8380160030003, receiver.as2_id=INFA_DEV_8_DS	The AS2ID of your communication partner as the sender as well as the AS2ID of yourself as the receiver

#### Error Message processing

Here you can define how errors that occurred when your communication partner sent a message are supposed to be processed.

Attribute	Example value	Description
classname	org.openas2.processor.receiver.AS2ReceiverModule	The receiver module to process received messages with
port	5080	The port for which to listen for error messages
errordir	\$properties.storageBaseDir\$/from1WorldSync/inbox/error	The path to the directory to receive incoming errors from

#### Message resend processing

Finally you can also define how messages that should be resent, e.g. due to faulty communication, should be resent.

Attribute	Example value	Description
classname	org.openas2.processor.resender.DirectoryResenderModule	The resender module which resends failed messages in case of a network error
resenddir	\$properties.storageBaseDir\$/resend	The path to the directory where resent messages should be sent from
errordir	\$properties.storageBaseDir\$/to1WorldSync/error	The path to the error directory from which error messages are to be received at in case of an error

#### partnerships.xml

The partnerships.xml contains information about the connection details between the trade partners such as AS2 ids, certificates and URLs. Usually, there are two partners, you and the data pool.

In this file one partner configuration is defined for each partner as well as one partnership configuration per communication channel (you → data pool, data pool → you).

**Source Partner configuration**

Attribute	Example value	Description
name	Informatica	The name of this partner. In this case your company name
as2_id	INFA_DEV_8_DS	The AS2ID of this partner. in this case your company AS2ID
x509_alias	infa_to_1ws	The certificate name in your keystore file
email	someName@informatica.com	The email address to which notifications can be send to

**Receiver Partner Configuration**

Attribute	Example value	Description
name	1WorldSync	The name of this partner. In this case your communication partner name
as2_id	8380160030003	The AS2ID of this partner. in this case your communication partner's AS2ID
x509_alias	1ws	The certificate name in your keystore file
email	Info@1WorldSync.com	The email address to which notifications can be sent to. In this case the email address of your communication partner

**Source Partnership configuration**

Attribute	Example value	Description
partnership name	Informatica-to-1WorldSync	The name of this partnership configuration. It makes sense to name it in favor of the actual communication direction
sender	Informatica	The partner name which sends messages to the receiver
receiver	1WorldSync	The partner name which receives messages from the sender
attribute - subject	Sent from \$sender.name\$ to \$receiver.name\$	The subject of each message handled in this partnership. Usually used for message processing automation
attribute - as2_url	http://as2.preprod.1worldsync.com:4080/exchange/8380160030003	The public address to which messages are to be sent to
attribute - as2_mdn_to	http://as2.preprod.1worldsync.com:4080/exchange/8380160030003	The public MDN address to which messages are to be sent to

**Receiver Partnership Configuration**

Attribute	Example value	Description
partnership name	1WorldySync-to-Informatica	The name of this partnership configuration. It makes sense to name it in favor of the actual communication direction
sender	1WorldSync	The partner name which sends messages to the receiver

Attribute	Example value	Description
receiver	Informatica	The partner name which receives messages from the sender. In this case, you yourself
attribute - subject	File sent from 1WorldSync to us	The subject of each message handled in this partnership. Usually used for message processing automation
attribute - as2_url	http://194.165.163.149:5080/as2	The public address to which messages are to be sent to. In this case, your own public IP Address where you expect answers to be received from
attribute - as2_mdn_to	http://194.165.163.149:5080/as2	The public MDN address to which messages are to be sent to. In this case, your own public IP Address where you expect answers to be received from

## Product 360 Configuration

### Import Hotfolder configuration

Whenever OpenAS2 receives a message file, the file will be put to the configured folder. This folder has to be observed by Product360 to pick up the responses and import the corresponding publication status.

Please see [Appendix E: Import mappings and hotfolder configuration \(see page 205\)](#) on how to setup the hotfolder configuration. The import mappings are located in the GDSN accelerator resources package under \DataSource\IM\OpenAS2\ImportMappings.

### Export configuration



You can find a general description how to load and save export templates and xsd files in [Appendix: Export templates \(see page 194\)](#).

### Export templates

The export templates can be found in DataSource/IM/OpenAS2/ExportTemplates/CatalogueRequest:

- CR\_CatalogueRequest Item.ext
- CR\_CatalogueRequest Link ADD.ext

- CR\_CatalogueRequest Link DELETE.ext
- CR\_CatalogueRequest Publication HW.ext
- CR\_CatalogueRequest Publication.ext
- CR\_CatalogueRequest Publication\_ByVariables.ext

Load and save them in Product360.

#### **XSD schema files**

The XSD schema files are located in DataSource/IM/OpenAS2/ExportTemplates/XSDs

Upload the xsd files to the system.

#### **XSLT files**

In DataSource/IM/OpenAS2/ExportTemplates/PostExport there is a XSLT file CR\_CatalogueRequest\_FileSplit.xslt needed to transform the export files into a file set ready to be submitted to the 1WS data pool.

#### **Background information**

IM, like DSE, allows a file only to contain a certain number of documents (items respectively). That's why every export template uses a post export step to split the export file every 100 documents and will attach header and footer in order to create a valid file. The files have the same name as configured in the export template but a consecutive number is added at the end.

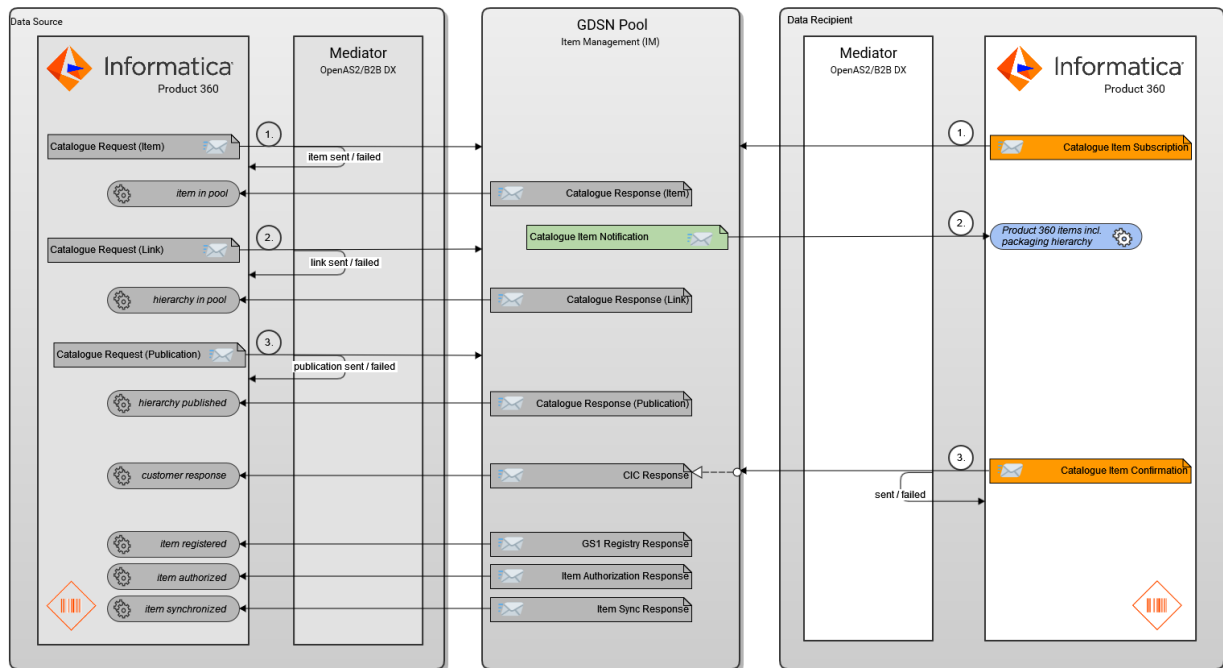
Data Quality rule configurations

To ensure valid data we provide data quality rule configurations. Please see [GDSN Accelerator Installation \(see page 23\)](#) on how to install them.

### **1.5.6.2 IM Data Recipient Scenario**

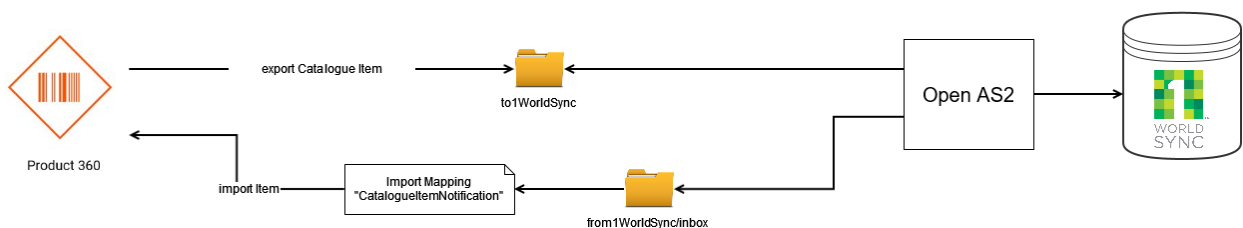
Message Choreography

In the DataRecipient Scenario the standard choreography requires the user to first send a Catalogue Item Subscription Message of the required GLN or GTIN. If a GTIN is requested, only the item belonging to that GTIN will be subscribed to. If a whole GLN is subscribed to, all items belonging to that GLN will be requested. Next, on PIM side, the user can create a Confirmation Status. This status provides information about the actual state of the subscribed item. This state can now be send to the Pool by sending a Catalogue Item Confirmation which contains the created status.



## Catalogue Item Subscription

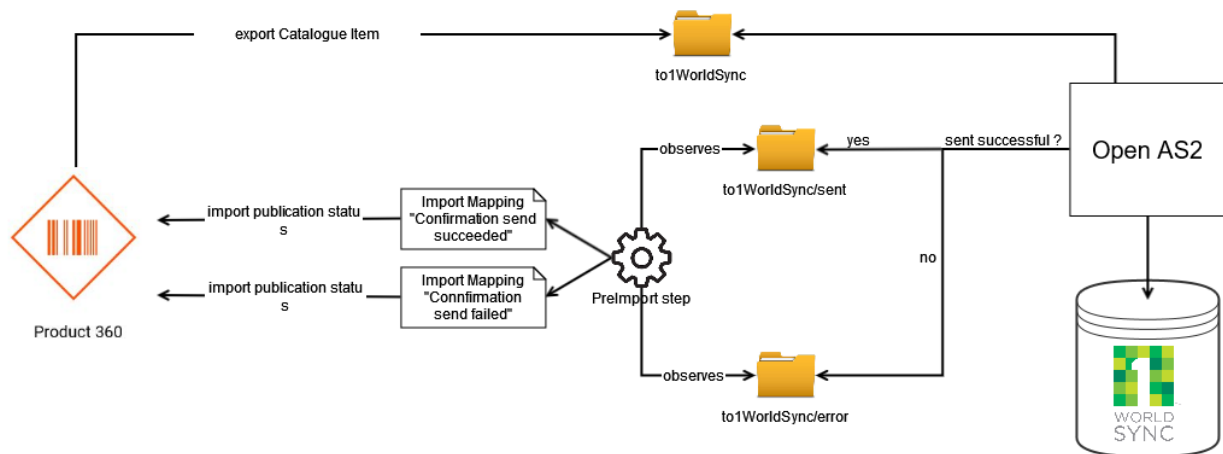
Sending a Catalog Item Subscription, will first export the required data of the selection into the to1WorldSync folder. The response will not be imported since there is not yet an item this information belongs to. After some time, if the item was successfully subscribed to, the requested data (Catalogue Item Notification message) will be transferred into the from1WorldSync/inbox folder. Here the data will be imported by Product 360 and will finally appear in your client.



## Catalogue Item Confirmation

After creating a Confirmation status, the user is now able to send that status to the Pool via the Catalogue Item Confirmation Message. Such a message is first exported to the to1WorldSync folder and processed by OpenAS2. Next, depending on its success, the resulting status will now be written by OpenAS2 and imported by Product360.





### OpenAS2 Configuration

There are two main configuration files which are relevant to each installation and have to be configured, `config.xml` and `partnership.xml`. Templates for both files can be found in the `config` directory of the OpenAS2 installation package.

#### `config.xml`

This file contains common settings, it configures the modules that will be activated by OpenAS2 server when it starts up. The following attributes are mandatory to be adjusted according to your pool registration information.

#### Properties

These are the global properties used for communication between sender and receiver in your scenario.

Attribute	Example value	Description
storageBaseDir	%home%/../data	The Base storage directory, used to put and read messages to and from the receiver
as2_async_mdn_url	http://194.165.163.149:5080	The public IP Address and its receiving folder used to receive messages from your communication partner
customer_name	Informatica	The communication partner name, used as the sender in this scenario

## Certificates

These are the certificate attributes used for message encryption.

Attribute	Example value	Description
filename	%home%/as2_certs.p12	The path to your p12 keystore file
password	testas2	The password with which your keystore has been encrypted with

## Partnerships

Here you can configure the partnership between your end and your communication partners.

Attribute	Example value	Description
filename	%home%/partnerships.xml	The path to your partnership.xml file

## Processor

Finally, you can configure your processor modules in order to define how the messages are to be processed. You need to add 4 more modules in order to configure the message processing correctly.



Directories you define here are also used by inbox hotfolder and export. Incoming message are usually automatically picked up by the observation mechanism of the inbox hotfolder, therefore the directories defined here and defined in the inbox hotfolder configuration have to be the same. (See [Appendix: Import mappings and hotfolder configuration \(see page 205\)](#))

In order to send exported files automatically the target folder configured in the "Copy file(s)" post export step of every GDSN export template has also to be configured here as outbox directory. (See [GDSN Accelerator operation/Export Templates \(see page 89\)](#))

## Send Message processing

Here you can define, how messages that you send to your communication partner are being processed.

Attribute	Example value	Description
classname	org.openas2.processor.receiver.AS2DirectoryPollingModule	The polling module to take in the file to be processed
outboxdir	\$properties.storageBaseDir\$/to1WorldSync	The path to the directory from which to send messages to your communication partner
errordir	\$properties.storageBaseDir\$/to1WorldSync/error	The path to the error directory from which error message are to be received at in case of an error
stored_error_filename	OPENAS2-\$rand.UUID\$@\$msg.attributes.filename\$_failed_to1WorldSync	The filename pattern of the error messages sent to your communication partner
senddir	\$properties.storageBaseDir\$/to1WorldSync/sent	The directory of the message files to be sent to you communication partner
stored_sent_filename	OPENAS2-\$rand.UUID\$@\$msg.attributes.filename\$_succeeded_to1WorldSync	The filename pattern of the successfully sent files to your communication partner
defaults	sender.as2_id=Informatica_recipient_as2_id, receiver.as2_id=8380160030003	Your own AS2ID as the sender as well as the AS2ID of your communication partner as the receiver

#### Received Message processing

Here you can define, how messages that your communication partner sends and which you are receiving, are being processed.

Attribute	Example value	Description
classname	org.openas2.processor.receiver.AS2DirectoryPollingModule	The polling module to take in the file to be processed

Attribute	Example value	Description
outboxdir	\$properties.storageBaseDir\$/from1WorldSync	The path to the directory from which to receive messages of your communication partner
errordir	\$properties.storageBaseDir\$/from1WorldSync/error	The path to the error directory from which to receive error messages from your communication partner
defaults	sender.as2_id=8380160030003, receiver.as2_id=Informatica_recipient_ as2_id	The AS2ID of your communication partner as the sender as well as the AS2ID of yourself as the receiver

#### Error Message processing

Here you can define how errors that occurred when your communication partner sent a message are supposed to be processed.

Attribute	Example value	Description
classname	org.openas2.processor.receiver.AS2ReceiverModule	The receiver module to process received messages with
port	5080	The port for which to listen for error messages
errordir	\$properties.storageBaseDir\$/from1WorldSync/inbox/error	The path to the directory to receive incoming errors from

#### Message resend processing

Finally you can also define how messages that should be resent, e.g. due to faulty communication, should be resent.

Attribute	Example value	Description
classname	org.openas2.processor.resender.DirectoryResenderModule	The resender module which resends failed messages in case of a network error
resenddir	\$properties.storageBaseDir\$/resend	The path to the directory where resent messages should be sent from
errordir	\$properties.storageBaseDir\$/to1WorldSync/error	The path to the error directory from which error messages are to be received at in case of an error

partnerships.xml

The partnerships.xml contains information about the connection details between the trade partners such as AS2 ids, certificates and URLs. Usually, there are two partners, you and the data pool.

In this file one partner configuration is defined for each partner as well as one partnership configuration per communication channel (you → data pool, data pool → you).

#### Source Partner configuration

Attribute	Example value	Description
name	Informatica	The name of this partner. In this case your company name
as2_id	Informatica_recipient_as2_id	The AS2ID of this partner. in this case your company AS2ID
x509_alias	informatica_dev_recipient_to_1ws	The certificate name in your keystore file
email	someName@informatica.com	The email address to which notifications can be send to

#### Receiver Partner Configuration

Attribute	Example value	Description
name	1WorldSync	The name of this partner. In this case your communication partner name
as2_id	8380160030003	The AS2ID of this partner. in this case your communication partner's AS2ID
x509_alias	1ws	The certificate name in your keystore file
email	Info@1WorldSync.com	The email address to which notifications can be sent to. In this case the email address of your communication partner

**Source Partnership configuration**

Attribute	Example value	Description
partnership name	Informatica-to-1WorldSync	The name of this partnership configuration. It makes sense to name it in favor of the actual communication direction
sender	Informatica	The partner name which sends messages to the receiver
receiver	1WorldSync	The partner name which receives messages from the sender
attribute - subject	Sent from \$sender.name\$ to \$receiver.name\$	The subject of each message handled in this partnership. Usually used for message processing automation
attribute - as2_url	http://as2.preprod.1worldsync.com:4080/exchange/8380160030003	The public address to which messages are to be sent to

Attribute	Example value	Description
attribute - as2_mdn_to	http://as2.preprod.1worldsync.com:4080/exchange/8380160030003	The public MDN address to which messages are to be sent to

**Receiver Partnership Configuration**

Attribute	Example value	Description
partnership name	1WorldySync-to-Informatica	The name of this partnership configuration. It makes sense to name it in favor of the actual communication direction
sender	1WorldSync	The partner name which sends messages to the receiver
receiver	Informatica	The partner name which receives messages from the sender. In this case, you yourself
attribute - subject	File sent from 1WorldSync to us	The subject of each message handled in this partnership. Usually used for message processing automation
attribute - as2_url	http://194.165.163.149:5080/as2	The public address to which messages are to be sent to. In this case, your own public IP Address where you expect answers to be received from
attribute - as2_mdn_to	http://194.165.163.149:5080/as2	The public MDN address to which messages are to be sent to. In this case, your own public IP Address where you expect answers to be received from

## Product 360 Configuration

### Import Hotfolder configuration

Whenever OpenAS2 receives a message file, the file will be put to the configured folder. This folder has to be observed by Product360 to pick up the responses and import the corresponding publication status.

Please see [Appendix: Import mappings and hotfolder configuration \(see page 205\)](#) on how to setup the hotfolder configuration. The import mappings are located in the GDSN accelerator resources package under \DataRecipient\IM\OpenAS2\ImportMappings.

### Export configuration



You can find a general description how to load and save export templates and xsd files in [Appendix: Export templates \(see page 194\)](#).

### Export templates

The export templates can be found in DataRecipient/IM/OpenAS2/ExportTemplates.

- CIC\_CatalogItemConfirmation.ext
- CIS\_CatalogItemSubscription.ext

Load and save them in Product360.

### XSD schema files

The XSD schema files are located in DataRecipient/IM/OpenAS2/ExportTemplates/XSDs

Upload the xsd files to the system.

## 1.5.7 Optional configuration for Data Recipient Scenarios

### 1.5.7.1 Import item data into individual supplier catalogs

When Product 360 is used with a GDSN data pool connection in data recipient mode, one of the first steps during configuration is to decide which catalog(s) to use for importing the item data. By default, the target catalog is determined based on the information provider of the respective message. However, it is also possible to configure different behavior.



## Configuration settings

There are two settings you can use to configure the identification of the target catalog dependend on a message that is to be processed by the hotfolder. You can find them in `plugin_customization.ini` file in the section "GDSN preferences".



If you want to use the configuration options described below, please make sure not to use special characters. If it should be necessary nevertheless, then these special characters must be escaped, as it is made also in properties files, which are used for translated texts.

Example: "Prices (€)" has to be escaped as "Prices (\u20AC)"

## Define a static target catalog

If you have already used a GDSN connection with Product 360 in the past and don't want to change the behavior regarding the target catalog, you can specify the identifier of the target catalog here. By default, no value is configured.

### plugin\_customization.ini - targetCatalog

```
# If the default behavior is not desired but all item data should be imported into
# one catalog, the identifier of that
# catalog should be configured here. It could also be the identifier of the master
# catalog
com.heiler.ppm.gdsn.core/targetCatalog =
```

## Define a pattern for the target supplier catalog

The second option is to define a pattern for the target supplier catalogs. This is useful in case there could be more than one catalog per supplier, or if you just want to specially mark the supplier catalogs that contain data from the GDSN data pool. By default, no value is configured.

### plugin\_customization.ini - targetCatalogIdentifierPattern

```
# In case multiple supplier catalogs are mapped to a single supplier, a pattern can
# be specified to identify the catalog
# to be used unambiguously. This pattern should contain exactly one placeholder ('*')
#
# It will also be used to create missing supplier catalogs automatically. In this
# case, the placeholder will be replaced
# by the GLN of the supplier to build the identifier of the new catalog.
# Example:
```

```
# For a supplier '00000000000007' the pattern 'GDSN_*' would result in a catalog with
the identifier 'GDSN_00000000000007'
com.heiler.ppm.gdsn.core/targetCatalogIdentifierPattern =
```

What cannot be configured?

When a message is received from the data pool, it is processed by the hotfolder. After the message has been analysed by the corresponding preparatory import step, the corresponding supplier is searched. If it does not yet exist, it is created automatically. Next, the target catalog is created if it is not yet there. This behavior cannot be changed.

Default behavior

If the default values are used and no target catalog could be found for the information provider of a message (which is a supplier in Product 360), a new supplier catalog will be created. The identifier and the name of that new supplier catalog will be the GLN of the supplier.

If there are multiple catalogs for the supplier, an exception will be thrown. No messages can be processed for the information provider until the data has been cleaned and a target catalog can be found unambiguously.

Detailed procedure

The following chapter explains where these settings come into play.

In data recipient scenarios, we have two important message types, *catalogue item notification* and *catalogue item confirmation*. While we receive *catalogue item notification* messages from the data pool, we send *catalogue item confirmation* message to the data pool and re-import its data to have a complete history of all data exchanges with the data pool.

Catalogue item notification message

#### **Step 1: Read and analyse the message file by the preparatory import step**

The information provider is read from the message, usually there is a GLN. A corresponding supplier has to be created if it cannot be found.

The target catalog is determined based on the configuration, it will be created if it cannot be found.

#### **Step 2: Delete existing confirmation status entries by the preparatory import step**

A simple import would update existing confirmation status entries. That's why all affected confirmation status entries will be deleted first. This makes it possible to have the complete history of all communications with the data pool.

The affected items will be searched in the target catalog.

### **Step 3: Choose import mapping and set target catalog by the preparatory import step**

According to the message, the import mappings are chosen. In addition to that, the target catalog will be set to item related import mappings.

### **Step 4: Import item data**

Item data is imported into the target catalog which was chosen by the preparatory import step.

### **Step 5: Import item meta data**

As the very last step, confirmation status entries will be imported.

The status objects will be selected according to the affected items. Several import functions are used to find items by their GTINs, to find status objects by items, and to find the catalog by the supplier.

Note that the import function to find a catalog by a supplier doesn't create a new catalog. At this point in time the target catalog was already created by the preparatory import step. Nevertheless, the import function uses the same algorithm to find the catalog as the preparatory import step, thus also the same configuration settings.

Catalogue item confirmation message

A *catalogue item confirmation* message is written by export in Product 360. As every message we create by an export and transfer to the data pool via the OpenAS2 connection, also *catalogue item notification* messages will be re-imported using the hotfolder.

### **Step 1: Read and analyse the message file by the preparatory import step**

This step is similar to the first step of processing catalog item notification messages. The target catalog is determined based on the configuration, it should not be necessary to create a new catalog.

### **Step 2: Delete existing confirmation status entries by the preparatory import step**

Not only confirmation status entries of message type "CIC transferred" will be deleted but also corresponding entries of type "CIC" and "CIC transfer failed". This ensures a clean status of the affected items and prevents multiple sent catalogue item confirmation messages.

### **Step 3: Import item meta data**

The last step is the import of new confirmation status entries of type "CIC transferred".

The corresponding import mapping is also using import functions to find the target catalog by the information provider and to find status objects by items.

## 1.6 GDSN Accelerator operation

### 1.6.1 Maintain valid data

#### 1.6.1.1 Item identifiers: GTINs

##### **GTINs for GDSN and food and beverage data**

All items which should be sent to the pool **must** have 14 digits GTINs. Otherwise there will be no answers from the pool written back to Product 360.

When the GTIN of an item which should be sent does not contain 14 digits you can add leading zeros until it matches the 14 digits criteria.

#### 1.6.1.2 Maintain data

Product 360 contains several perspectives to maintain *GDSN* and *Food and beverage* data. These are described below.

##### **GDSN (core)**

In the *GDSN (core)* perspective you find the views to maintain the most important aspects of GDSN. These include the target market specific data, the item references to build the packaging hierarchy and the *Packaging hierarchy* view to see a visualization of the packaging hierarchy, *Trade item lifespan information* and the *Publication status* view to see in which state your item is regarding the GDSN message chain. *Sustainability information* view is only available for the DSE pool.

##### **GDSN (supply chain data)**

This perspective enables you to maintain all data that is relevant in the supply chain. Views to maintain data of the following modules are included: Packaging-specific data, Trade item handling, Data carrier information, Delivery purchasing information

##### **GDSN (Canada-specific data)**

This perspective is only available for the IM pool. It contains the same views as the *GDSN (core)* perspective plus two additional views to maintain the data of the Canada extension.

##### **Food and beverage (core)**

In this perspective you can maintain the most important data of your food or beverage. These include diet related information, allergen related information, nutrient information and ingredients.

##### **Food and beverage (misc)**

This perspective contains views to maintain data about special aspects of the product. For example preparation and serving related information but also physiochemical and microbiological information. Additionally there are views to maintain data about certifications you might have.

##### **Ingredients**

You can maintain the ingredients of your product in the *Food and beverage (core)* perspective, however, first you have to make them available for the whole system in this perspective. You can think of this perspective as the pool of ingredients you use and the data of your product only references ingredients in this pool.

### **Certifications**

This is the pool of certifications you can use in the system. You can add references to these certifications in two places. First, you can add a certification to your product in the *Item certification* view and second you can add a certification for a specific diet type in the *Diet related information* view.

### **Third parties**

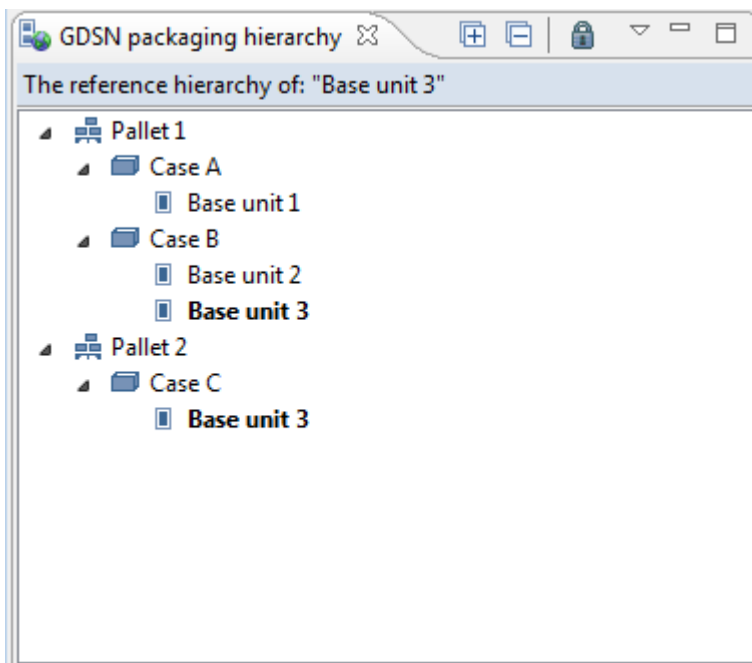
Third parties are all entities you interact with which are neither customers nor suppliers. There are three categories of third parties in the standard, Brand owner, Certification organisation and Manufacturer of goods. For use with certifications, it is necessary to add the category Certification organisation, otherwise they won't be shown when you create a certification.

### **Valid values**

If there are valid value lists given by GDSN and/or the respective pool, only those values will be shown in the drop down lists of the field. In case Product 360 requires a value for the specific field but GDSN does not, there is an additional value <No code> contained in the proposal list.

The packaging hierarchy view

The "GDSN packaging hierarchy" view can be used to visualize the packaging hierarchies of items:



The view shows the complete packaging hierarchy outgoing from one or more items. The hierarchy includes all higher level and all lower level items of each outgoing item. The outgoing items are highlighted in the view. The content of the view depends on the selection. This means that each time one or more items are selected in any other view the packaging hierarchy view loads its content based on the selected items. The view can be locked to retain the current content. The functionality "Obtain content exclusively from" is

available in the view to link it with a specific view. The labels of the tree nodes which represent the items of the hierarchy can be configured by using the view menu entry "Configure node labels...". Up to five fields can be defined to be used as part of the node labels. By a double-click on an item in the view the current outgoing item of the displayed hierarchy can be changed. In this way the user can "navigate" through the complete packaging hierarchy and see possible other hierarchies for items which are contained in more than one packaging hierarchy. The packaging hierarchy view allows multi-selection which can be used e.g. to show all items of the current hierarchy in a separate item view (by using the context menu entry "Show selected items"). It can be used e.g. for maintenance of all items of a packaging hierarchy at once, since the packaging hierarchy view itself is read-only.

### 1.6.1.3 Validate data

There are three levels of data validations for *GDSN* and *Food and beverage* data in Product 360:

#### Data model validation

The Product 360 data model and repository configuration ensures for example the correct field length and valid value lists.

#### Data quality checks

There is a whole package checking rules given by GDSN or IM. Most of them check dependencies between two or more field values.

Examples:

- Check 'Canceled date' is empty if 'Discontinued date' is populated
- Check 'Is dispatch unit' items have a 'Gross weight (imperial)' populated
- Check 'Number of items per layer' is greater than zero if 'Number of layers per pallet' is greater than zero

#### XSD validation

The XSD validation is executed during the export of the data. It is a final check, that the output has the correct format.

Example: Check if mandatory fields have values.

## 1.6.2 Item status in relation to the GDSN data pool

To manage the current status of items in the synchronization process with the pool, there are publication or confirmation status entries. Whenever a message is sent to or received from the data pool, corresponding information is stored to the affected items as new status entry. This allows the user to see the current item status, the history of communication with the data pool, and to filter items suitable for sending certain messages to the data pool.

The publication status is used in data source scenarios in contrast to the confirmation status which is used in data recipient scenarios.

### 1.6.2.1 Publication status

With the publication status you are able to indicate if the item is in the GDSN pool, to which customer the item is published or if the item has already been transferred to the GDSN data pool but no response has arrived yet. Furthermore it is easily possible to find out problems for a specific item or regarding the hierarchy. Therefore there are different "Message types" which indicate the current state of the item.

#### Publication status message types

- Item transferred
  - Item was delivered to Informatica B2B DX and sent to the GDSN pool after
- Item response
  - Item was received by the GDSN pool which validated the data and answered with this status information
- Registry response
  - After an item has been added or an existing item was modified (when GPC code, cancel date or discontinued date is updated) a message was sent to the Global Registry. On successful receipt of a response from the Global Registry the item information is updated and this message will be sent to data source.
- Link transferred
  - The according link (reference) was delivered to Informatica B2B DX and sent to the GDSN pool afterwards
- Link response
  - The according link (reference) was delivered to the GDSN pool which answered with this status information
- Publication transferred
  - The publication for this item was delivered to Informatica B2B DX and will be sent to the GDSN pool afterwards
- Publication response
  - The publication for this item was delivered to the GDSN pool which answered with this status information
- Sync response
  - After an item has been published to a data recipient who is also subscribing it and the item data or hierarchy fails at 1SYNC due to incorrect or insufficient data or fails the recipient data pool CIN validations, the GDSN pool will answer with this status information
- Authorization response
  - Data recipient provides a confirmation state of the item hierarchy (that has been synchronized to them) and sends it to the GDSN pool which will inform the data source with this message

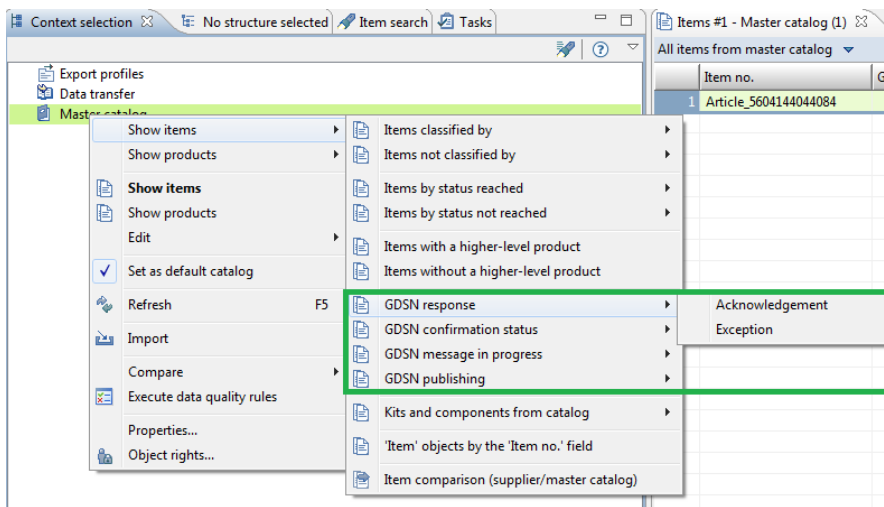
### 1.6.2.2 Confirmation status

The confirmation status contains the information about the item data provider and can also be used as basis to send feedback to the data provider.

## 1.6.3 Send data

### 1.6.3.1 Use queries to get an overview of your data

Queries are provided to find items based on their status related to the GDSN pool (using the publication status). That means the user has the possibility to easily find all items which have already been sent to the pool successfully or those where the GDSN pool has sent an exception for. To show the different kinds of possible queries just right click on a catalog and select "Show items" and see all submenu entries starting with "GDSN".



The queries are separated into four categories:

- **GDSN response:** Find all items which got a specified response from the GDSN pool, e.g. all items which received an "Acknowledgement" response for an "Item add" message or all items which received an "Exception" response for a "Link delete" message.
- **GDSN confirmation status:** These queries find all items with a specific severity of item authorization response messages. It is used as a helper to be able to keep an overview over specific responses of the data recipients.
- **GDSN message in progress:** Find all items which are currently processed, that means all items which have been sent to the pool with any message, e.g. "Link add" but not received a response yet. Due to the fact that it can take a while until a response from the GDSN pool is received these queries are useful to determine if the item has already been sent. This query category is only available for the IM pool.
- **GDSN publishing:** Find all items which should be published, republished or unpublished.

### 1.6.3.2 Build dynamic assortments from queries

All items which have been added to the pool

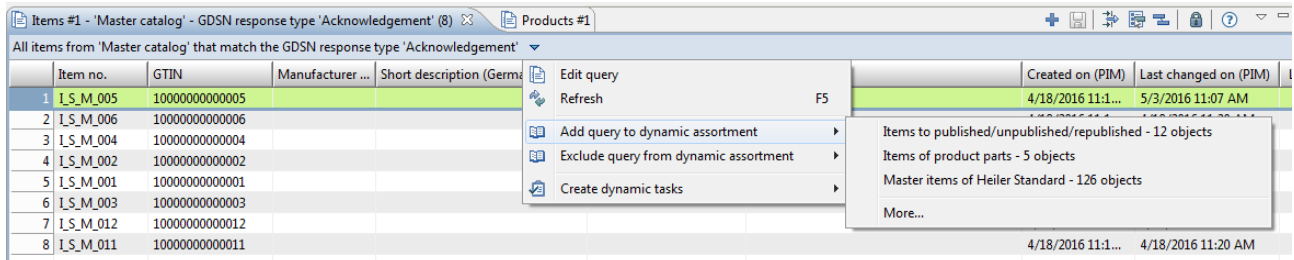
Step 1: Find all items that have been added to the pool

Choose "GDSN response - Acknowledgement" query from the context menu. In the query configuration dialog set "Message type" to "Item response" and then execute the query.



Step 2: Create an assortment from the result set

Select "Add query to dynamic assortment" from table header context bar and create new assortment using "More..."



Step 3: Find all items which have been added to the pool and got a modify exception

Choose "GDSN response - Exception" query from the context menu. In the query configuration dialog set "Message type" to "Item response", "Executed operation" to "Modify" and then execute the query.

Step 4: Add items to the assortment

Select "Add query to dynamic assortment" from table header context bar and choose your new assortment.

If everything was configured correctly you can verify the assortment by clicking on "View->Item assortments". Your assortment should look similar to the picture below:

Items which are in GDSN pool	82 items
Items from 'Master catalog' with GDSN response type 'Acknowledgement', 'Item response'	78 items
Items from 'Master catalog' with GDSN response type 'Exception', 'Item response', 'Modify'	4 items

All items which have not been added to the pool

Step 1: Create an item assortment containing all items that are to be transferred to the GDSN pool

This could be all items that are mapped to the GPC structure system

Step 2: Find all items that have been added to the pool

See above

Step 3: Exclude all items that have been added to the pool from the assortment

Select "Exclude query from dynamic assortment" from table header context bar and choose your new assortment.

The screenshot shows a table titled 'All items from 'Master catalog' that match the GDSN response type 'Acknowledgement'. The table has columns: Item no., GTIN, Manufacturer..., Short description (German), Created on (PIM), and Last changed on (PIM). A context menu is open over the table, showing options: Edit query, Refresh (F5), Add query to dynamic assortment, Exclude query from dynamic assortment, and Create dynamic tasks. A sub-menu is also visible, listing: Items to published/unpublished/republished - 12 objects, Items of product parts - 5 objects, Master items of Heiler Standard - 126 objects, and More...

Item no.	GTIN	Manufacturer...	Short description (German)	Created on (PIM)	Last changed on (PIM)
1	I_S_M_005	100000000000005		4/18/2016 11:1...	5/3/2016 11:07 AM
2	I_S_M_006	100000000000006		4/18/2016 11:1...	4/18/2016 11:20 AM
3	I_S_M_004	100000000000004			
4	I_S_M_002	100000000000002			
5	I_S_M_001	100000000000001			
6	I_S_M_003	100000000000003			
7	I_S_M_012	100000000000012			
8	I_S_M_011	100000000000011			

Step 4: Find all items which have been added to the pool and got a modify exception

See above

Step 5: Exclude items from the assortment

See above

All items which are marked for publication

To build an assortment with all items which are marked for publication you have to use the "GDSN publishing" query three times and add each result to an assortment

Step 1: Find all items that are marked to be published

Choose "GDSN publishing - To be published" query from the context menu and execute the query.

Add the result to a new assortment using "Add query to dynamic assortment" from table header context bar.

Step 2: Find all items that are marked to be unpublished

Choose "GDSN publishing - To be unpublished" query from the context menu and execute the query.

Add the result to your new created assortment using "Add query to dynamic assortment" from table header context bar.

Step 3: Find all items that are marked to be republished

Execute step 2 with the "GDSN publishing - To be republished" query.

If everything was configured correctly you can verify the assortment by clicking on "View->Item assortments". Your assortment should look similar to the picture below:

Items to published/unpublished/republished	12 items
<input checked="" type="checkbox"/> Items from 'Master catalog' with GDSN message type 'To be published'	12 items
<input checked="" type="checkbox"/> Items from 'Master catalog' with GDSN message type 'To be republished'	0 items
<input checked="" type="checkbox"/> Items from 'Master catalog' with GDSN message type 'To be unpublished'	0 items

### 1.6.3.3 Send item data

In GDSN you first ADD, LINK and MODIFY your data in the pool before you PUBLISH it to your customers. For all these steps you create a message to the pool using the Product 360 export with the corresponding export templates described in [Export templates \(see page 89\)](#). The export works on an assortment of items. To build an assortment you can, for example, use the search. Further information on assortment creation can be found in the Product 360 user manual.

You can either execute each of the export templates manually or automated by using a scheduled job. How to do that is described in [Setup automated jobs \(see page 98\)](#).

### 1.6.3.4 Publish data

You most likely want to publish your data in an automated way using a job. To make that work, you have to mark the items which have to be published, so that the job can identify them. The items can be marked by importing a specific publication status as described in [Publication \(see page 104\)](#).

## 1.6.4 Receive feedback

With the publication status you are able to see the current status in the synchronization process of each item. This means you are able to indicate if the item is in the GDSN pool, to which customer the item is published or if the item has already been transferred to the GDSN pool but no response has arrived yet. Furthermore it is easily possible to find out problems for a specific item or regarding the hierarchy. Therefore there are different "Message types" which indicate the current state of the item, for example

- Item response
  - Item was received by the GDSN pool which validated the data and answered with this status information
- Publication response
  - The publication for this item was delivered to the GDSN pool which answered with this status information
- CIC response
  - Feedback message from the target market or customer for an item which you published. The CIC responses are different for DSE and IM.

- DSE

For this message there will be two publication status entries in Product 360:

1. A status containing a context: This publication status includes publication messages with errors which are item specific. The context indicates the item with the highest hierarchy level.
2. A status without context: This publication status informs the user about the general status of the CIC and is only present in the item with the highest hierarchy level. No publication status messages are shown.

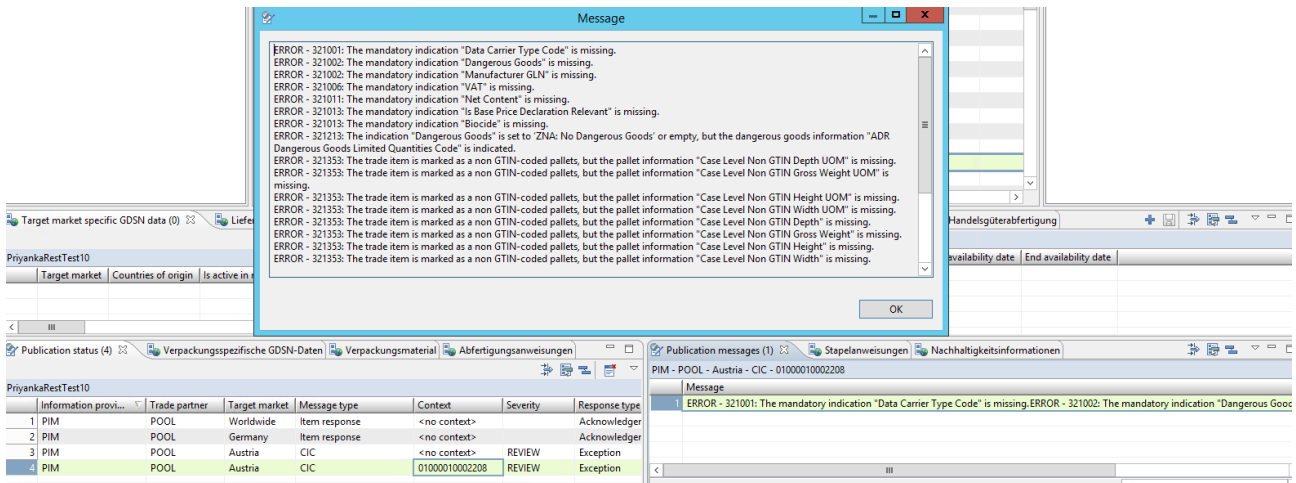


With B2B GDSN Accelerator 10.1.0.1 the mapping for the "Trade partner" got changed. From now on the recipient of the data will be mapped instead of the customer with the pool GLN. This change is leading to clear information about the current state of the hierarchy for each customer.

- **IM**

The CIC publication status is only written in the item with the highest hierarchy level and contains all publication messages for the items in his hierarchy.

When the pool responds with an exception there could be different reason for that. To see all relevant exceptions for an item you can double click on the message in the publication message view to see all messages which were sent from the pool.



### Publication status entries

Only the newest entries are shown in the publication status view for an item. If you want to see all messages including the ones which were overwritten you can click on the button marked in **red** and see the old entries written in *italic*.

The screenshot shows the 'Publication status' table in the Informatica MDM interface. The table has columns for Information provider, Trade partner, Target market, Message type, Context, Severity, and Response type. The table lists several entries, with the last one (row 43) highlighted in green, indicating a 'REVIEW' status with an 'Exception' response type. A red circle highlights a button in the top right corner of the table, which is used to view all messages including the ones which were overwritten.

	Information provi...	Trade partner	Target market	Message type	Context	Severity	Response t
36	PIM	POOL	Germany	Item response	<no context>		Acknowledged
37	PIM	POOL	Germany	Item response	<no context>		Acknowledged
38	PIM	POOL	Germany	Item response	<no context>		Acknowledged
39	PIM	POOL	Germany	Item response	<no context>		Acknowledged
40	PIM	POOL	Austria	CIC	<no context>	REVIEW	Exception
41	PIM	POOL	Austria	CIC	01000010002208	REVIEW	Exception
42	PIM	POOL	Austria	CIC	<no context>	REVIEW	Exception
43	PIM	POOL	Austria	CIC	01000010002208	REVIEW	Exception

## 1.6.5 Export Templates

### 1.6.5.1 Export template parameters

Export template parameters can be parameters of data sources, parameters of export post steps or variables. Furthermore variables can in turn be used as parameters of data sources or export post steps.

There are a lot of parameters you can configure in the export templates. All of them can be assigned to one of the following two categories:

#### Identify

Each of the export templates has two (GDSN) or three (IM) mandatory parameters needed for the communication with the GDSN pool. It is recommended to set the correct values and make those parameters invisible.

- "Information provider" and "Information provider GLN" respectively  
This is the supplier representing your own GLN
- "Recipient" and "Recipient GLN" respectively  
This is the customer representing the 1WorldSync pool GLN
- "User Id"  
**IM only:** This is the user id to authenticate yourself against the IM pool

The "Information provider" and the "Recipient" have to exist in Product 360, otherwise the answers from the pool will not be written back to Product 360. Make sure you created the values like described in the section about the creation of supplier and customer.

#### Select data

Most parameters are used to qualify which data should be exported.

First of all, there is a catalog and an item assortment to determine which items are to be exported.

In addition to that, the target market, a language, a price type, etc. specify which details of items are to be exported.

### 1.6.5.2 Export data source configuration

#### IM:

All IM export templates use the "Item list" as data source.

#### "Changed and new items"

If you want to schedule repeated export jobs, you usually want to send only new and changed items to the pool. That is why you have to change the data source of the corresponding export template.

To change the data source to "Changed and new items" you need to do following steps:

1. Open the properties dialog of your export format template and go to the "Data sources" tab

2. Press the "New..." button and choose the "Changed and new items" data source. Then close the property dialog by pressing "OK"
3. Open the "Items" module of your export format template by double click
4. Change the data source from "Item list" to "Changed and new items" in the property view of the module
5. Go back to properties dialog and open "Data sources" tab
6. Select data source "Item list" and press "Delete" button

### Standard GDSN:

Most Standard GDSN export templates use a form of the hierarchy of new and changed items data provider as data source. If you want to schedule repeated export jobs, you usually want to send only new and changed items to the pool.

### Both:

The following settings are recommended for the data source:

Setting	Value	Editable
Catalog	Master catalog	true
Assortment	<empty>	true
Reference date	<i>Variable "ReferenceDate", see next chapter</i>	true
Type of change	New and changed items	false
Update assortment	<empty>	true
Version	Working version	false

You can specify appropriate values for the visible parameters in the export profiles you create from the export templates.

It is recommended to set the "Update assortment" value to "true" for scheduled export jobs, but if you want to start a single export it may not be necessary to update the assortment.

### "Items by new or changed publication status" (IM)

The export template for creating publication messages also uses the "Item list" data source by default. If you want to use the recommended publication process (see [Publication \(see page 104\)](#)), you should use another item data source for the corresponding publication export template (see [Export templates for IM \(see page 96\)](#)), the "Items by new or changed publication status" data source. The reason is that items are not changed

when they are marked for publication and therefore are not picked up by the "Changed and new items" data source. The "Items by new or changed publication status" data source checks the publication status entries of items and processes all items that have new publication markings.

To change the data source to "Items by new or changed publication status" you need to do following steps:

1. Open the properties dialog of your export format template and go to the "Data sources" tab
2. Press the "New..." button and choose the "Items by new or changed publication status" data source. Then close the property dialog by pressing "OK"
3. Open the "Documents" module of your export format template by double click
4. Change the data source from "Item list" to "Items by new or changed publication status" in the property view of the module
5. Go back to properties dialog and open "Data sources" tab
6. Select data source "Item list" and press "Delete" button

The following settings are recommended for the "Items by new or changed publication status" data source:

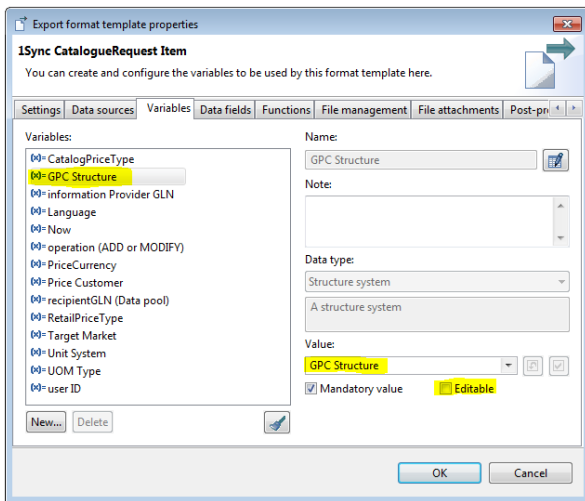
Setting	Value	Editable
Catalog	Master catalog	true
Assortment	<empty>	true
Reference date	Variable "ReferenceDate", see next chapter	true
Update assortment	empty	true
Version	Working version	false

You can specify appropriate values for the visible parameters in the export profiles you create from the export templates.

It is recommended to set the "Update assortment" value to "true" for scheduled export jobs, but if you want to start a single export it may not be necessary to update the assortment.

#### Parameters

Most parameters of the export template have to have fixed values. You should set those values to the export templates and switch the corresponding parameters to "Not editable" to hide them for further use.



### 1 Hide a mandatory parameter: set a value, switch to not editable

If you then use an export template to create an export profile, you will see only parameters that are really needed to be configured, for example the operation "ADD" or "MODIFY" in the export profile.

#### Reference date

If you want a [scheduled export job](#) (see [page 98](#)) to export all items that have been created or updated after the last run, you can use a variable that gets updated with the corresponding date and time for every export job run.

First, you create a variable of type "Date and time". Then you assign this variable as value for the "Reference date" parameter of the "Changed and new items" data source. The last step is to configure the variable in an export profile: select "Update" and "after exporting" for the variable.



**Export format template properties**  
**1Sync CatalogueRequest Item**  
 You can create and configure the variables to be used by this format template here.

Settings | Data sources | **Variables** | Data fields | Functions | File management | File attachments | Post-processing

Variables:

- CatalogPriceType
- GPC Structure
- information Provider GLN
- Language
- Now
- operation (ADD or MODIFY)
- PriceCurrency
- Price Customer
- recipientGLN (Data pool)
- ReferenceDate**
- RetailPriceType
- Target Market
- Unit System

Name: ReferenceDate

Note:

Data type: Date and time  
 A date and the time. The default value is the current date. The input format is 8/18/2015 11:24 AM or 2015-08-18

Value:

☒ Mandatory value ☒ Editable

OK Cancel

**Export format template properties**  
**1Sync CatalogueRequest Item**  
 You can select and configure the data sources to be used by this format template here.

Settings | **Data sources** | Variables | Data fields | Functions | File management | File attachments | Post-processing

Data sources:

- Changed and new items
- Item list

Name: Changed and new items

Parameter:

Catalog: Master catalog

☒ Mandatory value ☒ Editable

Assortment:

☐ Mandatory value ☐ Editable

Reference date:

ReferenceDate

New... Delete

**New export profile - 1Sync CatalogueRequest Item 2**  
**Export parameter configuration**  
 Set the export parameters for executing this export.

Data source: "Changed and new items"

Catalog: Master catalog

Assortment:

Update assortment: Yes

Variables

Now: Default value (8/18/2015)

**ReferenceDate:** 8/18/2015 4:22 PM ☒ Update: after exporting

operation (ADD or MODIFY): ADD

< Back Next > Finish Cancel

XSD schema files (Standard GDSN)

For DSE all available XSD schema files for all GDSN modules are delivered although only some of them are supported out of the box and so are really needed.

They are provided to make it easier for IPS or partners to extend the datamodel with new modules and have a consistent set of XSD files. This is also important because B2B has the same set of XSD schema files and they have to match to guarantee a working process.

Please note that B2B also has the full XSD schema file set which means there is nothing to do on B2B side when adding new modules or attributes.

Informatica B2B DX receive endpoint directory

The path to the B2B DX receive endpoint directory is set by the value for "Target directory" of the "Copy export file" export post step. As this is a fixed value you should set this parameter to not editable.

Use export templates

After you have configured the export templates you can use them to create export profiles for manual export or for scheduled export jobs.

The manual export can also be triggered by an immediate export. Therefore your template must match the needs of the immediate export:

- The purpose "Available for immediate export" must be set in the template
- The export format template for the immediate export must contain the value "Editable" in the data sources for the "Catalog" and "Assortment" parameters.

### 1.6.5.3 Post processing

Validate

All created XML files by the export will be validated against the corresponding XSD file. The "Validate XML file(s)" export post step will cancel the export if the validation fails.

*Export post step name:* Validate XML file(s)

*Parameter name:* Cancel export in case of error

Transfer

After an export has been finished successfully, the created XML file will be transferred to B2B DX or OpenAS2. The path to that folder is a parameter of the "Copy export file" export post step.

As this is a fixed value you should set this parameter to not editable.

*Export post step name:* Copy export file

*Parameter name:* Target directory

For usage with B2B set the parameter to the B2B DX receive endpoint directory.

For usage with OpenAS2, have a look in the OpenAS2 config.xml and search for the attribute 'outboxdir' in the tag processor/module

*Example:*

```

<openas2>
...
  <processor ...>
    <module classname="org.openas2.processor.receiver.AS2DirectoryPollingModule"
      outboxdir="$properties.storageBaseDir$/toAtrify"
      errordir="$properties.storageBaseDir$/toAtrify/error"
    ... />
    ...
  </processor>
  ...
</openas2>

```

resolving to something like this: C:/Informatica/OpenAS2/data/toAtrify

#### 1.6.5.4 Export Templates for Standard GDSN DataSource

##### Catalog Item Notification

The *Catalog Item Notification* export template outputs a "Catalog Item Notification" GDSN message containing items. Depending on the configuration, you can transfer new and/or modified trade item information with all detailed data maintained.

More information about provided data fields can be found in chapter [GDSN Accelerator field list](#) (see page 209).

##### Available Operations:

**ADD:** Adds the item to the pool (can also be used to update the current item). This is the operation that should be used for the automated *Catalog Item Notification* export job.

**CORRECT:** Is used to update the values for the current item in the pool. It can be used for manual triggered *Catalog Item Notification* exports.

**CHANGE\_BY\_REFRESH:** Updates the current hierarchy for the item in the pool if there is no active publication. This operation should only be used for dedicated item hierarchies that are to be transferred to the GDSN pool by manual started *Catalog Item Notification* exports.

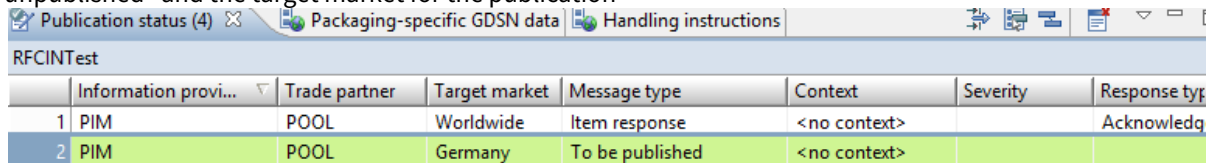
##### Catalog Item Publication

The *Catalog Item Publication* export template creates a GDSN message containing publications for all given items. Depending on the publication status of the items, you can add or delete publications. Furthermore, the message can be sent to a target market or for all GLNs an item is assigned to. You can find more information about the publication in the corresponding [Publication](#) (see page 104) chapter.

A publication will be sent if the items to publish follow these conditions:

- The items have been sent to the pool already, there's an Item response publication status for each item

- All items which should be published have to have a publication status entry with "To be published" or "To be unpublished" and the target market for the publication



	Information provi...	Trade partner	Target market	Message type	Context	Severity	Response typ
1	PIM	POOL	Worldwide	Item response	<no context>		Acknowledg
2	PIM	POOL	Germany	To be published	<no context>		

- The configured target market in the export format template corresponds to the target market of the item in "Target market specific GDSN data"
- If you want to publish to specific customers, you have to add a valid trade partner with a valid GLN.

**Note:** If there is the "<Public>" customer specified instead of a valid trade partner for the publication of the item it will be published to the whole target market taken from the parameter "Target market".

### Catalog Item Publication Withdrawal

The *Catalog Item Publication Withdrawal* export template is used to manually modify a publication of single items for a given GLN. Depending on the configuration, you can delete the active GLN publication on an item or change the hierarchy within the item.

A publication withdrawal will be sent if the item follows these conditions:

- The configured target market in the export format template corresponds to the target market of the item in "Target market specific GDSN data".

### Available Operations:

**PUBLICATION WITHDRAWAL:** Used when deleting an active publication.

**HIERARCHY\_LINK\_CORRECTION:** Used if you want to change the hierarchy of the specified item. After sending this operation, you can send your item with the changed hierarchy information with a Catalog Item Notification. The publication will be automatically resumed when the new Catalog Item Notification was sent.

## 1.6.5.5 Export Templates for IM DataSource

### Catalog Request Item

The *CR\_CatalogueRequest Item* export template outputs a "Catalog request" GDSN message containing items. Depending on the configuration, you can transfer new or modified trade item information with all detailed data maintained.

More information about provided data fields can be found in chapter "GDSN field list".

**Allowed values:** ADD, MODIFY

**Note:** The operation (ADD or MODIFY) specified in the GDSN message is a parameter of the export template. You have to ensure the correct value for that parameter, otherwise the pool will send an error response.

## Catalog Request Link Add

The *1CR\_CatalogueRequest Link ADD* export template outputs all packaging hierarchy information that has not been sent to the GDSN pool yet.

Technically, new item references of the "Next lower level" type are exported. The following rules apply:

- The referencing item (the parent item) and the referenced item (the child item) have been successfully sent at least once to the pool
- The link between the two items does not exist in the pool

**Note:** The export of item reference information has to analyse existing publication status data of according items. The publication status of items will be updated as a result of the response messages sent from the GDSN pool. That is why it may take a while until a new reference for a new item will be picked up by a "Link ADD" export.

## Catalog Request Link Delete

The *CR\_CatalogueRequest Link DELETE* export template outputs all deleted packaging hierarchy information that has been sent to the GDSN pool before.

Technically, deleted item references of the "Next lower level" type are exported. The following rules apply:

- The referencing item (the parent item) and the referenced item (the child item) have been successfully sent at least once to the pool
- The link between the two items exists in the pool
- The link between the two items has been deleted

## Catalog Request Publication

The *1Sync CatalogueRequest Publication CR\_CatalogueRequest Publication* export template creates a "Catalog request" GDSN message containing publications for all given items. Depending on the publication status, you can add or delete publications. Furthermore, the message can be sent for a market group or for all GLNs an item is assigned to. You can find more information in the chapter [Publication](#) (see page 104).

A publication will be sent if the items to publish follow these conditions:

- The configured target market in the export format template corresponds to the target market of the item in "Target market specific GDSN data"
- If you publish to a GLN, then the field "Trade partner" must be filled with the according customer
- If you want to publish to a market group you have to add the market group to the field "Context" and leave the field "Trade partner" blank

**Note:** Either the field context has to be filled with the market group or at least one "Trade partner" entry has to exist for each exported item. Otherwise an invalid XML file will be created and the export will be canceled.

As alternative you can use the *CR\_CatalogueRequest Publication\_ByVariables* export template export template in order to configure all the parameters manually.

## Catalog Request Publication Withdrawal

The *CR\_CatalogueRequest Publication HW* export template is used to withdraw a published hierarchy from the pool. It can be specified to unpublish the article from the market group or the GLN.

Afterwards the hierarchy can be changed and republished again. This has to be done with the Catalog Request Publication template.

## 1.6.6 Automated Jobs

### 1.6.6.1 Setup automated Jobs

Usually business users don't want to care about sending their new created or modified items to the GDSN pool. Due to the fact that the Product 360 export is used to send data to the pool, you can easily define repeating export jobs which send all matching items to the pool every hour for example. The setup described below has to be done once by a skilled consultant or administrator and not touched by any business user.

Please make sure that all templates are configured already as described in the chapter [Export templates \(see page 89\)](#).

Which automated jobs are needed

You can create automated jobs to send all GDSN messages in a scheduled way.

Please note, that you could cause exceptions if you send data twice to the GDSN pool by manual and automated exports at the same time.

How do automated export jobs work

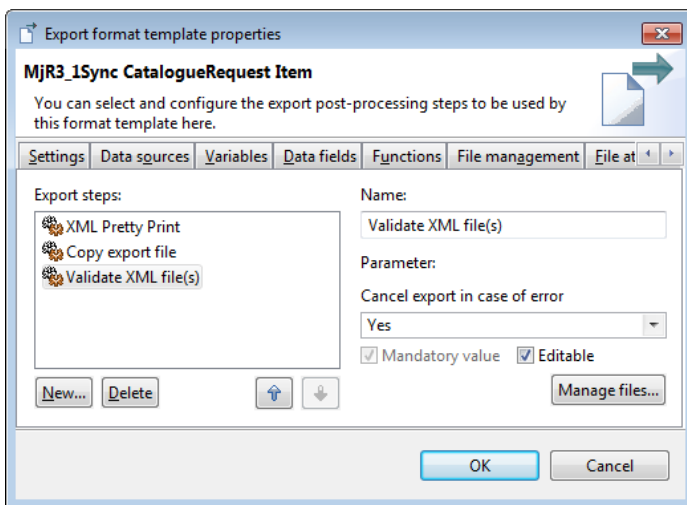
Automated jobs start exports of an export profile at specified times or in recurring intervals. The items to be sent are retrieved by the evaluation of the defined export parameters, for example assortment rules or reference date.

How to find the repeat interval for a job

There are no recommendations concerning the repeat interval of these exports. But please always have in mind that the answer from the GDSN pool could take a while and if you define the interval too short, it could be possible that a message is sent twice which will maybe result in an error.

What happens if there's no data to be exported

If an automated export job runs but there're no items to be exported, an invalid xml file will be created. To prevent such an invalid file to be transferred to the GDSN pool, the export has to be cancelled by the failing "Validate XML file(s)" export post step.



## 2 Cause export job to fail in case of invalid XML output file

16	Mandato...	Export post-processi...	catalogueRequest	Datei "1Sync CatalogueRequest Item.xml": Fehler in Zeile:
17	Summary	Post-export step	Validate XML file(s)	Export canceled due to XML validation error(s)

## 3 Export cancelled

The next run of the job will be started as configured and pick up all items defined by the assigned assortment rules.

### 1.6.6.2 Automated Jobs for Standard GDSN

Setting up a repeated job to automatically send all new and changed items to the GDSN pool

In the Standard GDSN scenario the item data together with their hierarchy data are sent in a single message called *CatalogItemNotification* to the GDSN pool. The corresponding export template to be used is *CIN\_CatalogItemNotification*.

There are two parameters you have to specify in order to find the items to be sent. The first is the catalog and item assortment, the second is the reference date used to find all items that have been created or modified since the last execution of the job.

Note: For the initial setup you should use a date in the distant past to ensure that all item are considered. For further information see [Export templates for Standard GDSN](#) (see page 95).

Setting up a repeated job to automatically send item publications to the GDSN pool

In order to provide the item data to the customers you need to send publication messages to the GDSN pool. The first step is to mark your items for publication using the publication status as described in [Publication](#) (see page 104).

Next you have to build a dynamic assortment containing all items marked for publication, see [GDSN Accelerator operation](#) (see page 80), section "Create an assortment containing all items which are marked for publication".

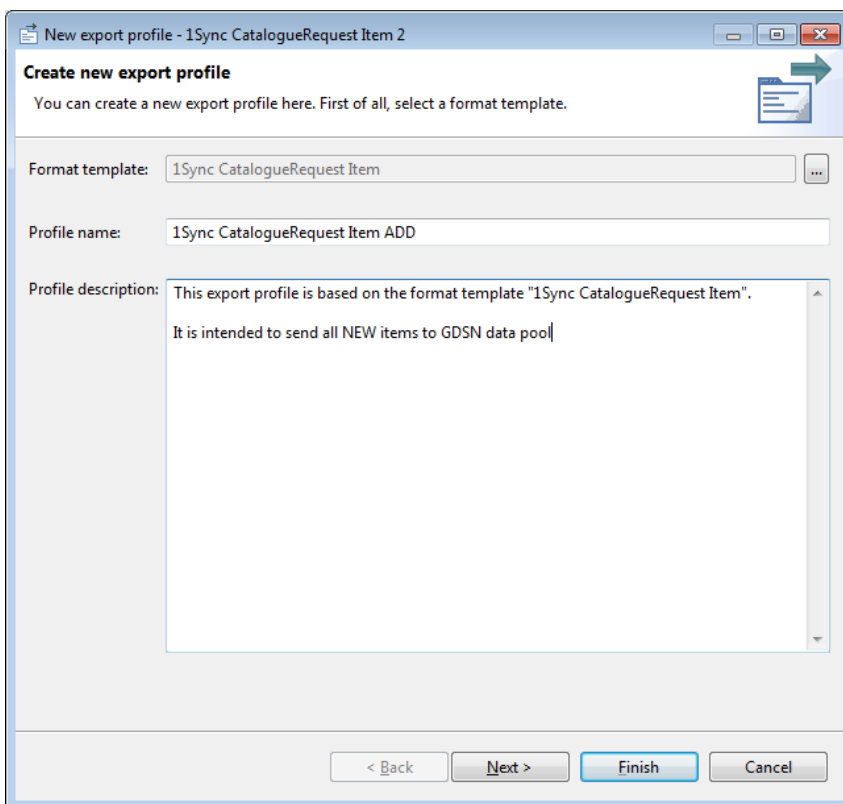
Finally you have to create the export job using the export template *CIP\_CatalogItemPublication* which should be already configured as described in the chapter [Export templates](#) (see page 99). Before scheduling the job set the assortment mentioned above to the "Assortment" parameter.

### 1.6.6.3 Automated Jobs for IM

Setup repeating export job to automatically send all new or changed items to the GDSN pool

Send all new items

For every repeating export job we define an export profile. The base for this export profile is an assortment containing all items which are not sent to the GDSN pool yet. The chapter [GDSN Accelerator operation \(see page 80\)](#) describes how to create this kind of assortment. After you have created the assortment open the "Explorer (Items)" perspective, go the "Context selection" tab, right-click on "Export profiles" and select "New export profile...". The dialog in the screenshot below pops up where you are able to define a name and select a format template.



The chosen export format template should be already configured as described in the chapter [Export templates \(see page 96\)](#) and so there should be only a few configurations left. You can choose the according catalog but it is highly recommended that you only send items from your master catalog. Additionally you can choose an assortment, which should be of course the one you just created in the step before. Last but not least the operation has to be defined, which will be set to ADD in this case because we want to "add" new items to the GDSN pool.



**Edit export profile - 1Sync CatalogueRequest Item**

**Export parameter configuration**  
Set the export parameters for executing this export.

Data source "Changed and new items"

Catalog:\* Master catalog

Assortment: GDSN - Items which are NOT in the GDSN p

Variables

operation (ADD or MODIFY):\* ADD

< Back Next > Finish Cancel

After you have successfully created an export profile you can schedule a repeating export job. To do that, right-click on the created export profile and select "Export". The parameters should already be well defined so you only have to configure the interval options on the last wizard page. An example is shown below:

**Export export profile - 1Sync CatalogueRequest Item**

**Select output destination**  
Plan the execution time of the export here and specify additional settings based on this.

Execution: Schedule export (repeating) From 8/18/2015 at 11:08 AM every 1 Hours.

Start date: 8/18/2015

Start time: 11:08 AM

Comment:

☐ Save export file(s) locally

Which files do you want to download?

☐ Data file(s) (1 files)

☐ Multimedia archive

Do you want the data files to be packed in a ZIP archive?

☐ Pack file(s)

Which local directory do you want to download the files to?

Directory

< Back Next > Finish Cancel

Send all changed items

This scenario is nearly the same as before with the small difference that the items are edited instead of created. One main difference is the assortment which should contain of course all items which are already in the GDSN pool. Please see chapter [GDSN Accelerator operation \(see page 80\)](#) and create this assortment. After this, create a new export profile but compared to the "new items" scenario described above use your new assortment and set the operation to MODIFY. It is also recommended to name the export profile clearly (e.g. "CR\_CatalogueRequest Item MODIFY").

**Export export profile - 1Sync CatalogueRequest Item MODIFY**

**Export parameter configuration**  
Set the export parameters for executing this export.

Data source "Changed and new items"

Catalog:\* Master catalog

Assortment: GDSN - Items which are already in the GDSN

Variables

operation (ADD or MODIFY):\* MODIFY

< Back Next > Finish Cancel

Last but not least schedule a repeating export job based on the created export profile and define a corresponding interval.

Please note: When you add or delete a reference and have a job which sends updates automatically, a "Link add" or "Link delete" will be sent. Due to the fact that the item has changed, also a "Item modify" message will be sent to the GDSN pool. This is unnecessary and a technical limitation but does not affect your data in the system or in the GDSN pool.

Setup repeating export job to automatically send packaging hierarchies to the GDSN pool

In the IM scenario it is required to send "Link" messages to create or delete packaging hierarchies. As pre requirement all items which should be linked as a hierarchy have to be added to the GDSN pool first. To ensure this, it is recommended that you use an assortment which contains only items which are already in the GDSN pool (see chapter ["GDSN Accelerator operation \(see page 80\)"](#), if you already created this assortment you can also reuse it).

Send all created packaging hierarchies

In a next step you have to create an export profile as in all other scenarios. This time you need a different export template named *CR\_CatalogueRequest Link ADD* which should be already configured as described in the chapter [Export templates \(see page 96\)](#). Furthermore select the catalog and the corresponding assortment and create afterwards a repeating export job as described above.

**Please note:**

- There is no "Link modify" scenario what means that if you want to change any information related to a reference you first have to send a "Link delete" message followed by a new "Link add" message with the new values.
- The "Link" messages are one of the most important differences between the IM and the Standard GDSN scenario. There are no link messages in the Standard GDSN scenario because the hierarchies are already defined in the CIN messages.

**Finding the repeat interval for sending all created packaging hierarchies**

Due to the fact that the "Link" messages are only sent for items which are already in the pool, it makes sense to schedule the jobs with a time lag to the job which sends new created items to the GDSN pool. As it takes the longest time to add newly created items to the GDSN pool it is recommended that you schedule the "Link add" job after the "Item add" job with an interval according to your data volume. That means if you are sending thousands of new items to the GDSN pool you probably have to define a longer interval than if you only send ten items.

Send all deleted packaging hierarchies

If you remove an item from the packaging hierarchy a "Link delete" message should be sent to update the changes in the GDSN pool. The scenario is basically the same as the "Link add" message but we have a different export format template. Create a new export profile using the export template *CR\_CatalogueRequest Link DELETE*, select the catalog and the corresponding assortment. Finally, as in all other scenarios create a repeating export job based on this profile and define a corresponding interval.

**Note:** It probably makes sense not to schedule the job the same time you send all created packaging hierarchies to not get confused but there is no real need to do that.

Setup repeating export job to automatically send item publications to the GDSN pool

In order to provide the item data to the customers you need to send publication messages to the GDSN pool. The first step is to mark your items for publication using the publication status as described in [Publication \(see page 104\)](#).

Next you have to build a dynamic assortment containing all items marked for publication, see [GDSN Accelerator operation \(see page 80\)](#).

Finally you have to create an export profile as in all other scenarios. This time you need a different export template named *CR\_CatalogueRequest Publication* which should be already configured as described in the chapter [Export templates \(see page 96\)](#). Furthermore select the catalog and the corresponding assortment and create afterwards a repeating export job.

## 1.6.7 Publication

After item data has been added to the GDSN pool and the hierarchies were created, it is possible to publish it for a specific customer (data recipient), target markets (Standard GDSN only) or market groups (IM only) by sending a GDSN publication message. By doing this the data recipient has the ability to view and synchronize the published item including all the child items below that item. An existing publication for a specific customer, market group or target market can be deleted.

*Note: Only the root item of a hierarchy has to be published actively. The rest of the hierarchy will be published with it automatically. The root item of a hierarchy may be target market specific.*

After an item has been published and subscribed to by a data recipient, some additional validations are triggered for the published GTINs and their hierarchies. If an item or a hierarchy fails due to incorrect or insufficient data, a corresponding response GDSN message will be generated by the GDSN pool. Using the information from the received message, a new publication status entry for the affected item is created.

In order to make the publication process more traceable, you shouldn't publish individual items manually but use the approach described below.

### 1.6.7.1 Mark items for publication

After the item data and item hierarchies are maintained completely and have been sent to the GDSN pool they are ready for publication. Some items may have to be unpublished since they are deprecated, some may have to be republished to a certain data recipient.

You can add all that information to the affected items using an import, it is not possible at the moment to add those information in the Product 360 UI directly. The detailed description of the data to be imported is described in the pool specific chapters below.

### 1.6.7.2 Publish items automatically

Once you've added the publication status information to the items, the next run of the scheduled publication export job picks up all that data and creates corresponding GDSN messages. Of course, it's also possible to start a publication export manually.

*Note: GDSN publication messages are only created for items that had been sent to the GDSN pool before. Even if an item is marked for publication, it won't be published if it hasn't been sent to the GDSN pool successfully.*

There's a special algorithm which analyses the publication status entries created by the import of publication markings and by responses sent by the GDSN pool. The following rules apply:

- In general, a publication message will only be created if the item has successfully been sent to the GDSN pool before.
- If there's a publication status of message type "Item response" or "Publication response" created after the publication marking, no publication message will be created.
- Publication messages of type "republish" or "unpublish" will only be created if the item has been published successfully before.

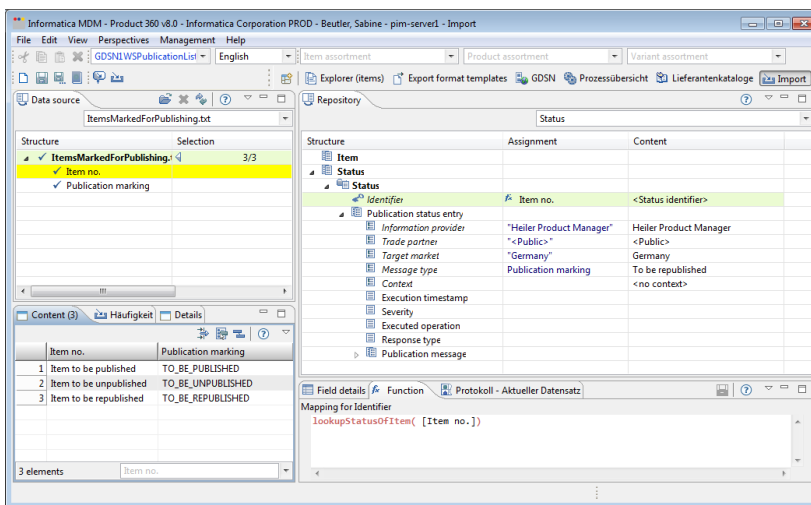
### 1.6.7.3 Publication process for Standard GDSN

#### Mark items for publication

You can publish items for a specific customer (data recipient) or for target markets. If you want to publish items for a target market you have to choose the "<Public>" customer as trade partner, otherwise select the desired customer.

**Note:** All customers maintained in your Product 360 should have a valid GLN, it's needed to identify them as trade partner against the GDSN pool.

All you need for the import of publication markings for items is a file containing the item numbers or GTINs and the type of publication operation you want to create (publish, unpublish, republish).



**Note:** Publication status information objects are independent objects and mapped to items. That's why you need a special import function to find the corresponding status entries for the items.

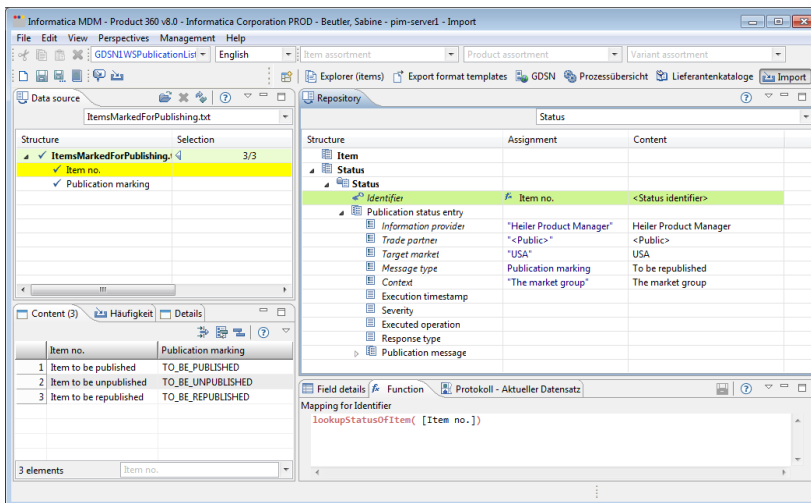
### 1.6.7.4 Publication process for IM

#### Mark items for publication (IM)

You can publish items for a specific customer (data recipient) or for a market group. If you want to publish items for a market group you have to map the name of the market group to the *Context* field, otherwise select the desired customer.

**Note:** All customers maintained in your Product 360 should have a valid GLN, it's needed to identify them as trade partner against the GDSN pool.

All you need for the import of publication markings for items is a file containing the item numbers or GTINs and the type of publication operation you want to create (publish, unpublish, republish).



Note: Publication status information objects are independent objects and mapped to items. That's why you need a special import function to find the corresponding status entries for the items.

## 1.7 GDSN Implementation Guidelines

### 1.7.1 Introduction

The scope of the GDSN implementation guidelines is to support adjustments of the provided GDSN Accelerator. This includes detailed technical instructions as well as things to consider. Furthermore it contains a list of things you need to clarify as a prerequisite, information about best practices when testing and information what to consider and think about if a different data pool should be supported.

To be able to use the GDSN implementation guidelines you need to have the following technical skills:

- You have a general understanding of GDSN.
- You are familiar with the Product 360 repository and you know about the available properties of the objects.
- You are able to create Product 360 import mappings and know how to use different import functionalities.
- You are able to export Product 360 data and know how to format the data correctly by using export functions.
- You are able to write Data Quality rules. (Optional, only needed if no standard DQ rule fits your needs.)
- You are able to configure Data Quality checks. (Optional)

If you use B2B solution

- You have basic knowledge of B2B Data Exchange
- You have basic knowledge of Informatica PowerCenter

In addition the requirements for the needed GDSN attributes of your customer should be defined.

### 1.7.2 Documents

This documentation refers to already existing Product 360 documentation. Therefore you should have following documentation nearby:

- Previous chapters of GDSN Accelerator documentation
- Product 360 documents, which can be found at the Informatica partner webspace, especially the "Informatica MDM - Product 360 - <Version> - Knowledgebase, Installation and Customization.zip"
- Product 360 User Manual, which is part of the Product 360 delivery
- Data Quality documents, which are part of the Data Quality delivery

Furthermore this documentation refers to external documentation provided by your GDSN certified data pool. This includes documentation about

- the GDSN attributes (name, data type, description, valid value lists,...)
- valid value lists and their entries
- validations
- structure of the files used in the communication (if applicable)

The used documentations in this guideline are mentioned at the beginning of each chapter.

### 1.7.3 Implementation steps

Depending on the scenario you are using, the following chapters are important for you

Data source:

- Analyze requirements
- Data model
- UI adjustments
- Data validations
- Export templates
- Testing

Data recipient:

- Analyze requirements
- Data model
- UI adjustments
- Import mappings
- Testing

If you want to implement a connection to a pool that we don't support out of the box, have a look at chapter GDSN Implementation Guidelines/Communication.

### 1.7.4 Analyze requirements

#### 1.7.4.1 Introduction

This chapter describes how to analyze your GDSN requirements. First, you need to compare the needed GDSN attributes with the "[Appendix F: Field List \(see page 209\)](#)" to get a rough overview of how many new attributes need to be added. Please take care that you might need to adjust existing fields as well. This applies to new GDSN versions as well as implementing a new module and moving existing parts or fields respectively in the new module. As result of the analysis you should be able to call out which changes are needed and in case of data model changes, how the according entity and fields must be designed. Based on

this analysis, the chapter "[Data model \(see page 121\)](#)" describes how to add new GDSN modules and GDSN attributes as well as creating or adjusting GDSN valid value lists.

In order to be able to analyze your customer's GDSN requirements, you need to have a good understanding of the Product 360 repository and the architecture it is based on.

#### 1.7.4.2 Questions

The first step is to ask the right questions:

- Which scenario is used?
  - Data source or data recipient?
  - IM or DSE?
- Which modules are already available?
- What kind of change has to be implemented?
  - Is it a missing data field of an existing module?
  - Is it a missing module?
  - Is it a missing validation?
  - Is it a missing value of a valid value list?
  - ...



You can find all supported GDSN attributes by the GDSN Accelerator in the chapter "[GDSN Accelerator field list \(see page 209\)](#)".

#### 1.7.4.3 Resources

It is essential to have all necessary documents available. These are the GDSN specification documents you can get from the corresponding GS1 home page.

There are several PDF and Excel files describing fields, data types, validations. And there are XSD files you should use to get detailed information about the structure of the GDSN message files, especially of the files containing item data.

In the following examples we use these files:

- XML Schemas
- IM\_Participant\_Dictionary\_R7.1.0\_v1 (especially tabs IM Participant dictionary and IM Valid Values)
- GDSN module PDF files
- Data\_Source\_1WS\_XML\_Guide\_IM7.0v6.pdf called "Data Source XML Guide" in the rest of the document
- IM\_Validations\_Document\_R7.1.0\_v1.xlsx

#### 1.7.4.4 Data model - analyze the module

So let's work together on a first "module"... We've been told by the customer, that he needs to store microbiological information and wants to make it available using *GDSN IM*. Where do we start?





- AGM - attribute group many

For further information on flex attributes and structure types see explanations and examples in "1WorldSync Item Management - Data Source 1WS XML Guide".

**1WS Item Structure Type:** The "O" tells us, that the attribute group is optional.

**Module:** In the "Module" column it says "FoodAndBeveragePropertiesInformation", so there is no separate module for microbiological information. We should have a look at what else is contained in the "FoodAndBeveragePropertiesInformation" module in order to decide how to design our entities. We note that down and go on for now.

**Next we find four lines that seem to be fields**

1WS Catalogue Request Attribute	1WS XML Structure Type	1WS Item Structure Type	1WS Item Data Type	1WS Item Data Length (Min)	1WS Item Data Length (Max)	Qualifier	Module	FLX
foodAndBevMicrobiological/ organismCode	A	O	string	1	80		FoodAndBeveragePropertiesInformation	Y
foodAndBevMicrobiological/ organismMaximumValue	AQ	O	ufloat	15	15	uom	FoodAndBeveragePropertiesInformation	Y
foodAndBevMicrobiological/ organismReferenceValue	AQ	O	ufloat	33	2	uom	FoodAndBeveragePropertiesInformation	Y
foodAndBevMicrobiological/ organismWarningValue	AQ	O	ufloat	33	2	uom	FoodAndBeveragePropertiesInformation	Y

*What information do we get here?*

In the group "foodAndBevMicrobiological" there are four fields

1. The attribute "organismCode" which is a simple string attribute with a max. length of 80 that is optional.
2. The qualified attribute "organismMaximumValue" which is a decimal value with a min. length of 15 and a max. length of 15. The Qualifier "uom" is an indicator that we need another field to store the value for the qualifier. In some cases, like for example the language, the qualifier can be a logical key.
3. and 4. are the qualified fields "organismReferenceValue" and "organismWarningValue", both qualified with a unit, both decimal, both optional, both with a min. length of 33 and a max. length of 2. Well, the length can't be correct. We should check that later in the Participant dictionary. Write that down and go on.



"Uom" stands for "unit of measure". We use the terms "uom" and "unit" interchangeably.

Measurement values always consist of a pair of value and uom.

#### If we keep searching we find these lines

1WS Catalogue Request Attribute	1WS XML Structure Type	1WS Item Structure Type	1WS Item Data Type	1WS Item Data Length (Min)	1WS Item Data Length (Max)	Qualifier	Module	FLX
componentInformation/ foodAndBeveragePropertiesInformation/ foodAndBevMicrobiological/ organismCode	A	O	string	1	80		ComponentInformation	Y
componentInformation/ foodAndBeveragePropertiesInformation/ foodAndBevMicrobiological/ organismMaximumValue	AQ	O	ufloat	15	15	uom	ComponentInformation	Y

1WS Catalogue Request Attribute	1WS XML Structure Type	1WS Item Structure Type	1WS Item Data Type	1WS Item Data Length (Min)	1WS Item Data Length (Max)	Qualifier	Module	FLX
componentInformation/foodAndBeveragePropertiesInformation/foodAndBevMicrobiological/organismReferenceValue	AQ	O	ufloat	15	15	uom	ComponentInformation	Y
componentInformation/foodAndBeveragePropertiesInformation/foodAndBevMicrobiological/organismWarningValue	AQ	O	ufloat	15	15	uom	ComponentInformation	Y

If you compare this set of fields with the ones we found before you will notice that the attribute names are the same but the paths are different. The module is different, too. It's "ComponentInformation".

Components are a MjR3 feature that we don't support at the moment, so ignore these fields. If you want to know more about components have a look at the GDSN homepage (<http://www.gs1.org/gdsn>).

Get a better idea of the structure

Since the fields we found are flex attributes, we won't see much of them in the IM XSDs. Sometimes it's hard to imagine the complete structure of a module based only on the textual information given in the table. But we can get a little help from the GDSN XSDs to get a better idea how the structure might look like.



If we look at the "FoodAndBeverageInformationType", marked in the green we see the occurrences are "0..\*", so we can have multiple entries for "FoodAndBeverageMicrobiologicalInformation". This means we have to find a logical key for our data model. We have the organismCode. Since the rest are measurement values, it seems to make sense to have one set of values for each organism. So this is a good candidate for a logical key.

In GDSN there is only the "organismMaximumValue". We also found that field in the "1WorldSync Item Management - Data Source 1WS XML Guide" but additionally there were "organismReferenceValue" and "organismWarningValue". In the XSD we can see that there is an additional "unitOfMeasure" belonging to the "organismMaximumValue". This is no surprise, we already saw that the fields are qualified with a unit and knew that we needed to store this information somewhere.

### So let's recap:

We have the "foodAndBeverageMicrobiological" module, which is an attribute group many. This sounds like an entity, doesn't it?

We have identified the fields

- organismCode
- organismMaximumValue

- organismMaximumValueUOM
- organismReferenceValue
- organismReferenceValueUOM
- organismWarningValue
- organismWarningValueUOM

We also know that "organismCode" is a candidate for a logical key.

Design the entity

You don't know yet, but it will be described in the section "[Data model \(see page 121\)](#)" that there is the entity type `ArticleDomainType` that is suitable for implementing new modules and that has a sub entity for measurement values. There is an additional key `UOMType`. Possible values are: `METRIC` and `IMPERIAL`. We need to be able to store multiple values, but since units are convertible we don't need to store each and every value we might be using in an output. That's the reason we don't use the unit as the logical key.

So what we have to do is to create an entity like this:

- Entity: ArticleMicrobiological (based on ArticleDomainType)
  - Logical key: organismCode
  - Field: organismCode
- Entity: ArticleMicrobiologicalUOM (based on ArticleDomainUOMType)
  - Logical key: UOMType
  - Field: UOMType
  - Field: organismMaximumValue
  - Field: organismMaximumValueUOM
  - Field: organismReferenceValue
  - Field: organismReferenceValueUOM
  - Field: organismWarningValue
  - Field: organismWarningValueUOM

Now, we have the basic structure of our sub entity.

Check the details

Now, we have to check for the details. The details can be found in the Participant dictionary.

	C	D	E	F	G	H	I	J	K	L	M	N	O
	GUI Location	GUI Name	IM XML Name	GSN XML Name	Mandatory/Optional	Definition	Data Type	Length (All Rights), Min Length (All non-Restr. Data Types)	Precision (All Rights), Max Length (All non-Restr. Data Types)	Global/Target Market Specific	Occurrence	Qualifier Type	Qual Value List
1	Components Tab	Organism Code	componentInformation/foodAndBeveragePropertiesInformation/foodAndBevMicrobiological/organismCode	CatalogItemNotification/CatalogItem/variantItem/variantItemComponents/foodAndBeveragePropertiesInformationModule/microbiologicalInformation/microbiologicalOrganismCode	0	Code indicating the type of microbiological organism.	V/P/NB/organismCode	1	80	Target Market	0..1		
547	Components Tab	Organism Maximum Value	componentInformation/foodAndBeveragePropertiesInformation/foodAndBevMicrobiological/organismMaximumValue	CatalogItemNotification/CatalogItem/variantItem/variantItemComponents/foodAndBeveragePropertiesInformationModule/microbiologicalInformation/microbiologicalOrganismMaximumValue	0	Maximum allowable value of the microbiological organism.	ufloat	15	15	Target Market	0..1	uom	uom
548													

Search for the attributes we found earlier:

### foodAndBevMicrobiological/organismCode

GUI Name: Organism Code

This is your English display label. By convention in Product 360, the first letter of the first word is upper case, all following words start with a lower case character except it is a name or another word which is correctly spelled with an upper case character in English.

*IM XML Name:* foodAndBevMicrobiological/organismCode

This is where you find the path given in the Data Source XML Guide

*GDSN XML Name:*

Where you find it in the XSDs of the GDSN XML structure.

*Mandatory/Optional:*

Most of the fields are optional, some of the fields are "M within O group" which means they are mandatory if you form the group. These fields can be set to mandatory in the repository (upper and lower bound = 1). Check if these fields are suitable as logical keys.

*Definition:*

This is your description in English. Read it, correct it, if it is a whole sentence add a '.' at the end and if there is no valuable descriptive content in there, don't use it.

*Datatype:* VV/FNBOrganismCode

"VV" means there is a valid value list. "FNBOrganismCode" is the name of the list. The values will be found on the tab "IM Valid Values". Valid value lists most likely contain strings.

Other common data types apart from valid value lists are dateTime, uinteger and ufloat.

→ We need to add an enumeration to the repository as well.

*Min. length and max. length:*

The field can have values of strings with a length up to 80. Even they say the min. length is 1, since the value is optional I would use a lower bound of 0.

*Global/Target market specific:* Target Market

Defines if it is possible to maintain different values for different target markets.

→ We need an additional target market key.

*Occurrence:* 0..1

This can be a problem if we want to use organism code as a logical key. Logical keys are mandatory.

**foodAndBevMicrobiological/organismMaximumValue**

Most of the information is similar to the above.

The data type is ufloat and the columns of the min./max. length are called "**Length (All floats)**, Min. Length (All non-float Data Types)" and "**Precision (All floats)**, Max. Length (All non-float)". This explains the values 33/2 which made no sense earlier.

*Example:* Imagine an attribute defined with 15/15. What this means is that 1234567890,12345 is valid and 12345,1234567890 is valid but 12345678,12345678 is not valid because the complete length is greater than 15.

Product 360 can't persist such huge numbers. A BigDecimal16/6 is always used which means the complete number is at most 16 places long - 10 places before the decimal separator and 6 decimal places. However, if the definition is for example 5/2 the max. range should be set to 99999,99 with scale 2. Be aware that this means you can store 88888,888888 in the database anyway because the scale is just a matter of formatting.

**foodAndBevMicrobiological/organismReferenceValue** and **foodAndBevMicrobiological/organismWarningValue**

Most of the information is similar to the above.

Have a look at the length and precision. Here in the Participant dictionary it says 15/15 not 33/2 as it did in the Data Source XML Guide. This is an example of conflicting documentation. Note this on your test list and send dummy data to the data pool later. Determine what is correct on the error messages you get back from the data pool.

## UOM

We know we need the unit fields as well. We won't find them as separate lines in the participant dictionary, only as qualifier in the line of the attribute they belong to. At a first glance you might wonder how you will be able to create the field with so little information. However since units will always be stored as

`UnitProxies` and corresponding field types are already provided in the `ArticleDomainUOMType`, there is not much left to be configured.

The only question we have to answer is, which units should be available to the user or in other words which enumeration do we need to add to the unit field. If we go back to the Participant dictionary there is no VV entry in the datatype column, which makes sense because this line is about the measurement value that is a numeric value.

GUI Name	IM XML Name	DataType	Length (All floats), Min Length (All non-float Data Types)	Precision (All floats), Max Length (All non-float)	Qualifier Type	Qual Valid Value List
Organism Warning Value	componentInformation/foodAndBeveragePropertiesInformation/foodAndBevMicrobiological/organismWarningValue	ufloat	15	15	uom	uom

But there are two other columns which will give us the answer we need. There is the qualifier type "uom" and "Qual Valid Value List" "uom". If you go to the tab "IM Valid Values" you will find a list with that name. In the "Attribute Name" column of that tab you will also find the "organismWarningValue" attribute and the other value fields. Check the existing enumerations if there is already one with the matching values or create your own. See section ["Data model \(see page 121\)"](#) for information how to do that.



It has proven to be useful to create an excel sheet with all the information relevant to you. This may include:

- GDSN attribute name
- PIM display label
- Field identifier
- Data type in GDSN
- Data type in PIM
- Valid values
- Is field mandatory?
- ...

## Logical keys

Let's come back to the hardest decision. What do we use as logical key(s)?

- *Do we need a logical key?*

Yes, foodAndBevMicrobiological is AGM, so we need to be able to store more than one set of values per target market.



- *Why should we use organismCode as logical key?*  
It makes sense to have one set of measurement values per organism. It doesn't make much sense to have multiple warning values for the same organism from a business point of view.
- *Why shouldn't we use organismCode as logical key?*  
Because organismCode is optional. This means GDSN allows to have measurement values not belonging to one of the organisms in the valid values list.
- *Is there an alternative?*  
Can't think of one.

What is the solution then?

The solution is to use "organismCode" as logical key but tweak the valid values a little. Make an enumeration "with optional code". This enumeration has one or more additional values. Most standard enumerations with optional code have one additional value, for example "NONE". This allows the user to store additional value sets, it works with all the generic mechanisms in Product 360 and in the export there is a mechanism which will ensure that this value is not sent to the GDSN data Pool. How many additional entries (if at all) you need depends on the requirements of the customer.

See how to create an enumeration with optional code in the chapter ["Data model \(see page 121\)"](#).

See how to handle enumerations with optional codes in the export in section ["Technical details \(see page 201\)"](#).

Compacting the structure

At the beginning we saw that the attributes related to microbiological information belong to the module "FoodAndBeveragePropertiesInformation". We ignored that up to now. But you might ask yourself if you have to fit a complete module into one Article sub entity.

The answer is definitely 'no'.

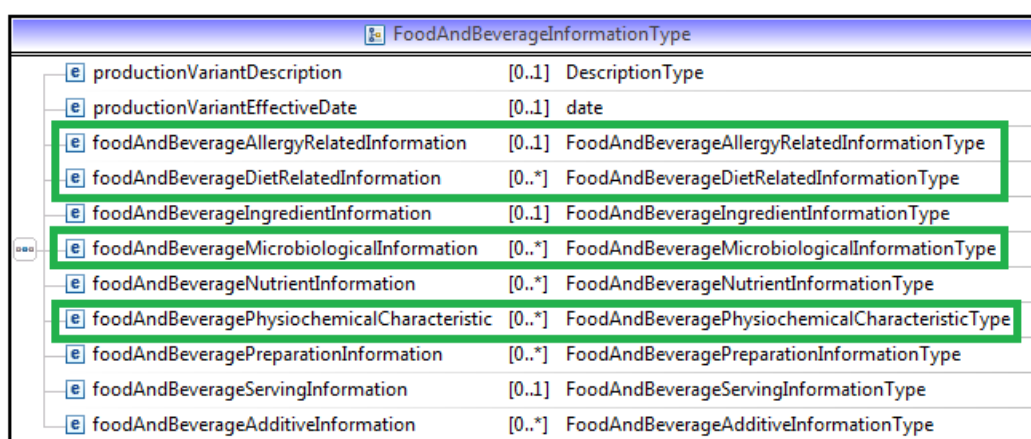
Let's see what else is contained in the module "FoodAndBeveragePropertiesInformation": What we find are physiochemical properties.

1WS Catalogue Request Attribute	1WS XML Structure Type	1WS Item Structure Type	Module	FLX
physioChemicalProperties	AGM	O	FoodAndBeveragePropertiesInformation	Y

1WS Catalogue Request Attribute	1WS XML Structure Type	1WS Item Structure Type	1WS Item Data Type	1WS Item Data Length (Min)	1WS Item Data Length (Max)	Qualifier	Module	FLX
physioChemicalProperties/ physioChemicalCharacteristic Code	A	O	string	1	80		FoodAndBeveragePropertiesInformation	Y
physioChemicalProperties/ physioChemicalCharacteristic Value	AQM	O	ufloat	15	15	uom	FoodAndBeveragePropertiesInformation	Y

Does this information have a relation to microbiological information?

No. So we probably can implement the physiochemical properties in its own sub entity. When we look at the GDSN XSDs, we get a confirmation of our assumption. **FoodAndBeverageMicrobiologicalInformationType** and **FoodAndBeveragePhysioChemicalCharacteristicType** are two separate types on the same level as **FoodAndBeverageAllergyRelatedInformation** and **FoodAndBeverageDietRelatedInformation**.



## Deep structures

Some modules, for example the ingredient information, have a pretty deep structure with up to ~ 10 nested levels of XML tags. The `ArticleDomainType` has a depth of 3 (+1 for the item itself).

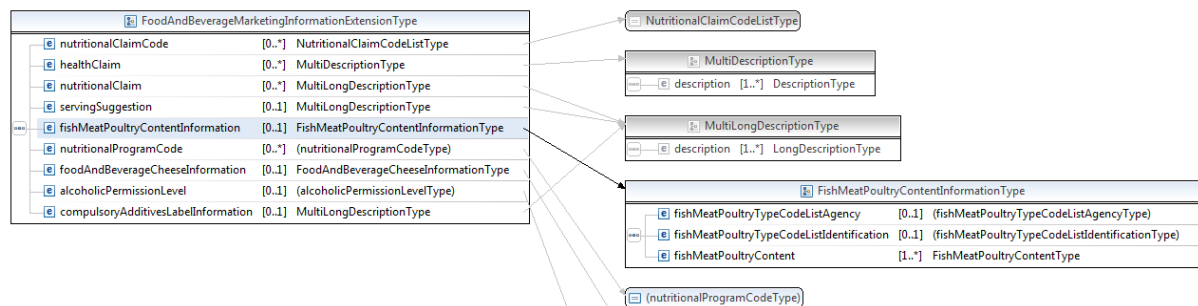
If you encounter a module with a deep structure you have to get creative and shrink it down to 3 levels.

To get you started here are two possible ways to do that:

### 1. Extraction

Certifications are an example of Extraction. You can add a certification for the item itself and you can add a certification for a specific diet. The XML sub structure that stores the certification information is the same in both cases. Furthermore if you think object oriented, a certification is a self-contained complete object. So in Product 360 that part was extracted into its own root entity and only the entity proxy is stored in the corresponding sub entities of the item.

### 2. Compacting



As you can see "fishMeatPoultryContentInformation" is contained in "FoodAndBeverageMarketingInformationExtension". But it has no relation to the rest of the information in the marketing information. This is a case where the container is not really needed for the logical structure of the information and we can skip this layer. The occurrence of "0..1" confirms that no sub entity layer with logical keys is needed at this point.

## Ways to ensure data consistency

In the course of creating the data model, you should start thinking about data consistency or in other words validations.

There are two sources for information about validations. The Participant dictionary contains a lot of basic validations like the max. field length. Complex validations, you have to take into account for the design of your entity, are listed in the "IM Validations Document".

## Data model

There are lower level validations only affecting a single field that can be configured in the repository.

### Valid value lists

*Description:* Some fields only allow a certain set of values.

*Where to find:* Information can be found in the Participant dictionary in column "DataType", for qualifiers in column "Qual Valid Value List". List entries are found in tab "IM Valid Values".

*How to implement:* In Product 360 this kind of validation is ensured by the enumeration you add to a field.

*Example:* "organismCode" has valid value list "FNBOrganismCode" and is implemented as

`Enum.OrganismCode.WithOptionalCode` at logical key

`ArticleMicrobiologics.LK.OrganismCode` and field

`ArticleMicrobiologics.OrganismCode`

## Min. and max. values

*Description:* Numeric values, especially measurement values can be restricted to a certain range.

*Where to find:* Information can be found either in the Participant dictionary, in the columns "Length (All floats), Min. Length (All non-float Data Types)" and "Precision (All floats), Max. Length (All non-float)" or in the Validations document.

*How to implement:* In Product 360 this kind of validation is ensured by the entries in the field properties "Min. Range" and "Max. Range"

*Example:* "OrganismMaximumValue" has a max. length of 15 places

Other examples from the validations document:

- The value in - Qty of Next Level Item(s) (formerly Pack) is greater than 1 and less than 999999.
- If fatPercentageInDryMatter is not empty then value must be greater than or equal to 0 and less than or equal to 100.00.

Closely related to min. and max. values is the max. length of string values.

*Example:* GTIN Name: Value must be between 1 and 40 characters.

In this case use properties min. and max. length.

## Mandatory fields

*Description:* Some fields are mandatory globally or mandatory in an optional or mandatory group.

*Where to find:* Information can either be found in the Participant dictionary, in column "Mandatory/Optional", or in the validations document.

*How to implement:* Set the lower bound to 1 in order to make a field mandatory within an entry of a sub entity.

*Hint:* Logical keys are always mandatory in Product 360.

*Examples:* GTINName is a required field

More information on configuration of the repository can be found in the section "[Data model \(see page 121\)](#)"

You should think about these kind of validations now!

## Data Quality

Then there are more complex validations that are affecting multiple fields at once, depend on a specific value or target market. Most likely they will be implemented using DQ rule configurations.

Examples:

- If promotionalTypeCode is populated, then isConsumerUnit must be true.
- For each occurrence of the Loopgroup “promotional”, attributes freeQtyOfNextLowerLevel and freeQtyOfProduct cannot both be populated.
- If targetMarketCountryCode is equal to '752' then packagingMaterialTypeCode and packagingMaterialCompositionQuantity are used in pairs. I.e. if one is populated the other one must be populated, too.
- If grossWeight and netWeight are provided on the same record, grossWeight must be greater than or equal to netWeight
- There must be at most one iteration of minimumFishMeatPoultryContent per Unit Of Measure

For further information see chapter "[Data validations \(see page 155\)](#)"

## Export

When you have to output data into export files you should ensure to create well-formatted values, details can be found in the chapter "[Data validations \(see page 155\)](#)".

### 1.7.4.5 Summary

#### Analyze Module Summary

1. Get your documentation documents
2. Collect the fields of your module
3. Get an idea of the structure intended by GDSN
4. Try to fit the structure in an existing entity type
5. Find your logical keys
6. Implement the entity with the information from the section "Data Model"
7. Think about validations
8. In the process note all assumptions, discrepancies and open questions for later testing.

## 1.7.5 Data model

### 1.7.5.1 Introduction

This chapter describes how to implement a module as an entity. However, references to the additional documentation about the repository are made and can be found in the chapter Domain Model (Repository) of the Product 360 Documentation.

After you analyzed the new module as described in the chapter [Analyze requirements \(see page 107\)](#) of this documentation you have an idea what the entity will look like. This site describes in more detail the technical implementation and configuration possibilities. There are the following sections

- Create entity (Types area, custom area and sub entities)
- Create logical keys
- Create fields
- Create enumerations (valid value lists)
- Test checklist

If you only need to add a specific field, you can skip the first two sections.

### 1.7.5.2 Resources

In the following chapters we use these files:

- Profile Overview: FMCG\_DIY\_ARGO\_ProfileOverview\_Codelists\_<Version>.xlsx
- Participant Dictionary: IM\_Participant\_Dictionary\_R<Version>.xlsx
- Product 360 Documentation: Informatica MDM - Product 360 - v<Version> - Knowledgebase, Installation and Customization.zip

### 1.7.5.3 Create a new entity

Create a new entity - Types area

In the types area, there are three GDSN related sub entity types of `ArticleType`

- `ArticleDomainType`
- `ArticleMarketExtensionType`
- `ArticleDomainExtensionType`

`ArticleDomainType`

If you create a new module, this entity type should be your first choice when searching for an entity type to build your entity on.

It contains lots of field types of various data types. There are two main types:

- `ArticleDomainType<Subentity>.Std_<datatype>_<fieldNumber>`
- `ArticleDomainType<Subentity>.Res_<datatype>_<fieldNumber>`

The existing fields for GDSN modules in the standard are using the

`ArticleDomainType<Subentity>.Std_<datatype>_<fieldNumber>` fields for GDSN attributes.

Whereas the `ArticleDomainType<Subentity>.Res_<datatype>_<fieldNumber>` can be used for enriching those GDSN modules by customizings.

If you create an entirely new module, you are allowed to use

`ArticleDomainType<Subentity>.Std_<datatype>_<fieldNumber>` . If you enhance an existing module, please only use the `ArticleDomainType<Subentity>.Res_<datatype>_<fieldNumber>` fields.

There is a sub entity called `ArticleIngredientLangType` . Don't use it. It is a special purpose sub entity only used by the standard.

If your running out of fields, you either create another entity or file a request via Product360 Support for the necessary changes.

## ArticleMarketExtensionType

In contrast to the `ArticleDomainType` the `ArticleMarketExtensionType` has a different set of logical keys. It has a `PartyProxy` logical key, but misses the two `<entity>.LK.Std_LK_<datatype>_<keyNumber>` logical keys. It depends on your module which entity type to choose.

## ArticleDomainExtensionType

Don't use this entity type. It is deprecated.



Use `ArticleDomainType` for new Modules.

Use `ArticleDomainType<Subentity>.Res_<datatype>_<fieldNumber>` to enhance existing modules.

Use `ArticleDomainType<Subentity>.Std_<datatype>_<fieldNumber>` fields in new modules, preferably.

## Available ArticleDomainType sub entity types

`ArticleDomainType` has several sub entity types commonly used in GDSN for different purposes

- **ArticleDomainLangType**

Entities based on this entity type contain language specific data. It is similar to `ArticleLang` and has logical keys `EntityId` and `Language`.

- **ArticleDomainUOMType**

Entities based on this entity type usually contain measurement values - a numeric value combined with a unit. The key `UOMType` can have the one of the values "metric" and "imperial". It is based on the assumption that different metric units like gram and kilogram can be converted in each other and therefore it would be an error source to store values for both units. But it may not be possible to automatically convert from metric units into imperial units.

*Example:* `ArticlePackagingUOM.PackagingWeight` and

`ArticlePackagingUOM.PackagingWeightUOM`

- **ArticleDomainPartyType**

Entities based on this entity type usually contain data related to a party like customer specific data.

- **ArticleSubDomainType**

Entities based on this entity type usually contains data that together forms a list.

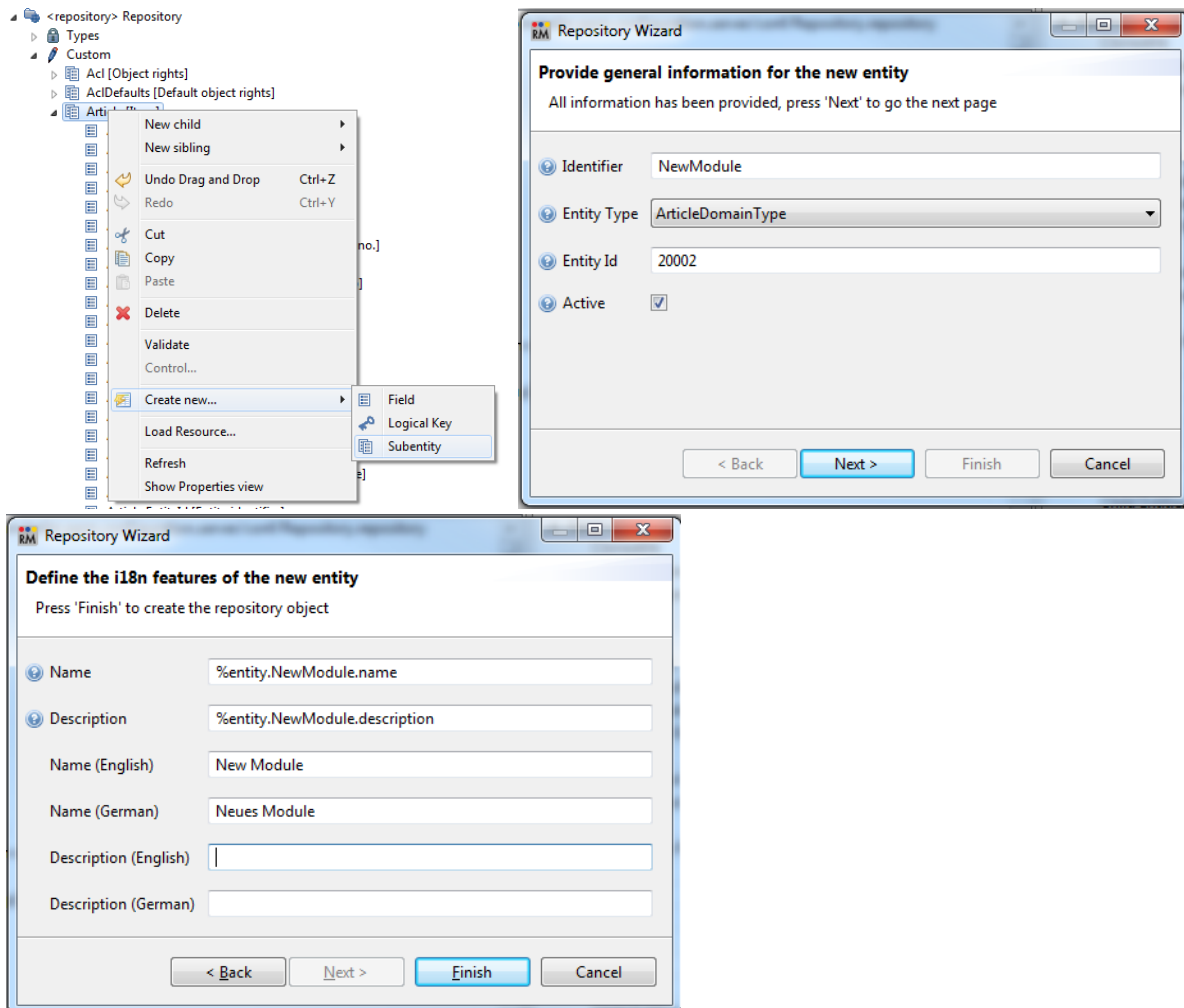
There are subentities supporting language specific data and measurement values for each list entry.

*Example:* `ArticlePackagingMaterial` contains a list of different packaging materials. A bottle of wine may be packaged in a hardwood box, padded with hemp fibers and surrounded by a cardboard box.

Create a new entity - Custom area

Wizard

Included in the Repository Manager are wizards helping you to create sub entities, fields, logical keys etc.



Create the entity of your module

There is nothing special to a GDSN module entity. It should support standard functionality like:

- Import
- Export
- Merge
- Clone
- Search
- Data Quality
- Service API



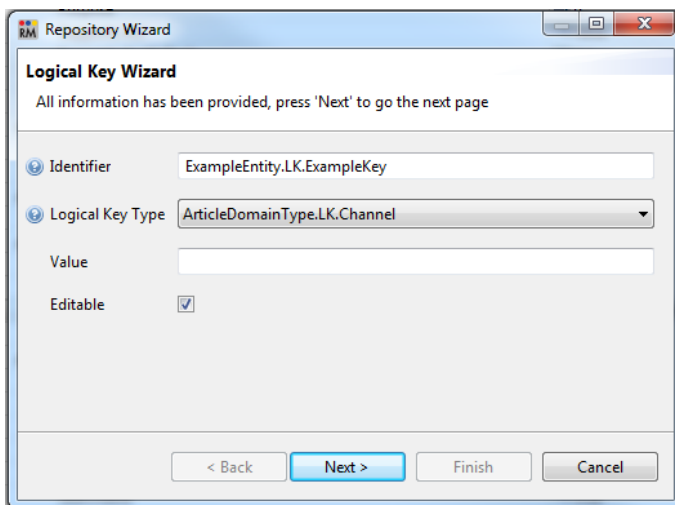


Entity IDs must be 20002 or higher.

Name and Description have to be externalized and be available in your needed client languages.  
Please note that the default language is English.

#### 1.7.5.4 Create Logical keys

This is the wizard to create new logical keys:



**Repository Wizard**

**Logical Key Wizard**

All information has been provided, press 'Next' to go the next page

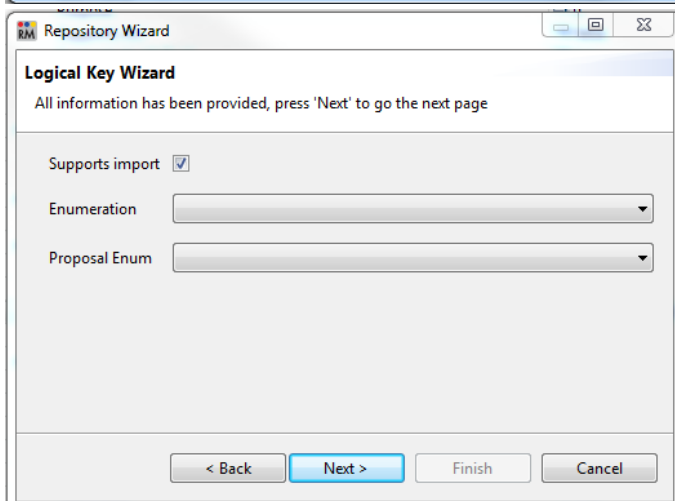
Identifier: ExampleEntity.LK.ExampleKey

Logical Key Type: ArticleDomainType.LK.Channel

Value:

Editable: ☒

< Back Next > Finish Cancel



**Repository Wizard**

**Logical Key Wizard**

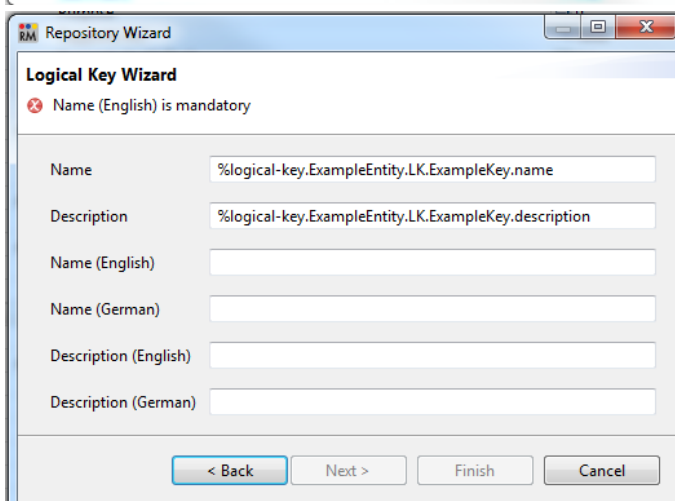
All information has been provided, press 'Next' to go the next page

Supports import: ☒

Enumeration:

Proposal Enum:

< Back Next > Finish Cancel



**Repository Wizard**

**Logical Key Wizard**

✖ Name (English) is mandatory

Name: %logical-key.ExampleEntity.LK.ExampleKey.name

Description: %logical-key.ExampleEntity.LK.ExampleKey.description

Name (English):

Name (German):

Description (English):

Description (German):

< Back Next > Finish Cancel

In the new entity, there need to be a logical key for each logical key type of the underlying entity type. If you don't want to use all of them all you can deactivate these logical keys. Don't forget to add a default value in this case.

Each logical key needs a corresponding field and if you want to restrict the values for the logical key to a defined set using an enumeration, don't forget to add the enumeration to the logical key as well as to the field.



- Create a logical key for each logical key type in the underlying entity type.
- Each logical key needs a corresponding field.

If you want a logical key to be *active*, check this:

- Purpose is set to 1
- Editable is set to true
- It has an identifier
- Supports import is set to true

If you want a logical key to be *inactive*, check this:

- Purpose is set to 0
- Editable is set to false
- It has a value (default value)
- supports import is set to false

### Logical keys - Service API

In the Service API, if you want to read or write a certain value, you need to specify the qualification. The qualification consists of values for all the logical keys on the path to the target field. To identify a logical key, the alias is used. You can define the alias in the selected line in the screenshot of the Repository Manager below. The alias gives you the opportunity to make the Service API requests more easily readable.

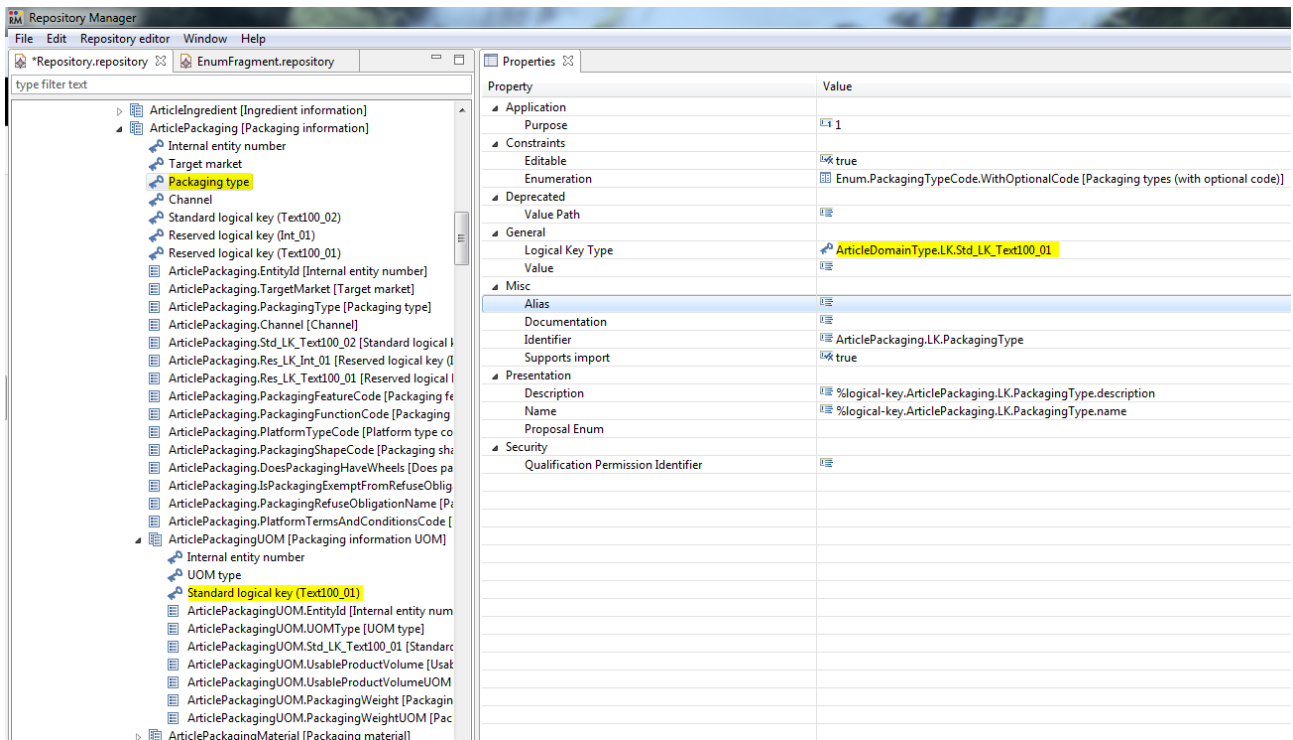
If there is no alias, the object name of the mapped field of the logical key is used. In the

`ArticleDomainType`, the object names are generic because this entity type is the basis for a couple of different entities. Therefore it is recommended to use the alias.



The alias (or object name respectively) needs to be unique in the path to the target field.

*Example:* Let's have a look at the field `ArticlePackagingUOM.PackagingWeight`. Let's further assume a customer is using the Standard logical key (`Text100_01`) ( set editable = true) in the entity `ArticlePackagingUOM`. Note that there is a logical key `Packaging type` that is based on the Standard logical key (`Text100_01`) in the entity `ArticlePackaging`.



The corresponding fields of both logical keys have the object name " std\_LK\_Text100\_01 ".

You will notice that you get the values via Service API but not the complete qualification. In the code block below the packaging type is missing.

**GET <http://localhost:1512/rest/V1.0/list/Article/ArticlePackaging/byCatalog?fields=ArticlePackagingUOM.PackagingWeight&catalog=MASTER>**

```
{
  "cacheId": "20170329_111826_0",
  "entityIdentifier": "ArticlePackaging",
  "totalSize": 66,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 2,
  "columnCount": 0,
  "columns": [],
  "rows": [
    {
      "object": {
        "id": "224435@1",
        "label": "Item1",
        "entityId": 1000
      },
      "qualification": {
        "targetMarket": "Barbados",
        "uomType": "metric",
```

```

        "std_LK_Text100_01": "DEFAULT"
    },
    "values": [
        "9.9"
    ]
},
{
    "object": {
        "id": "224435@1",
        "label": "Item1",
        "entityId": 1000
    },
    "qualification": {
        "targetMarket": "Barbados",
        "uomType": "metric",
        "std_LK_Text100_01": "DEFAULT"
    },
    "values": [
        "11.11"
    ]
}
]
}

```

If you try to write values and you specify values for all logical keys as in the example below ....

#### POST <http://localhost:1512/rest/V1.0/list/Article/ArticlePackaging>

```

{
    "columns": [
        {
            "identifier": "ArticlePackagingUOM.PackagingWeight"
        }
    ],
    "rows": [
        {
            "object": {
                "id": "'Item1'@'MASTER'"
            },
            "qualification": {
                "targetMarket": "Barbados",
                "std_LK_Text100_01": "Ampoule",
                "uomType": "METRIC",
                "std_LK_Text100_01": "DEFAULT"
            },
            "values": [
                "8.8"
            ]
        }
    ]
}

```

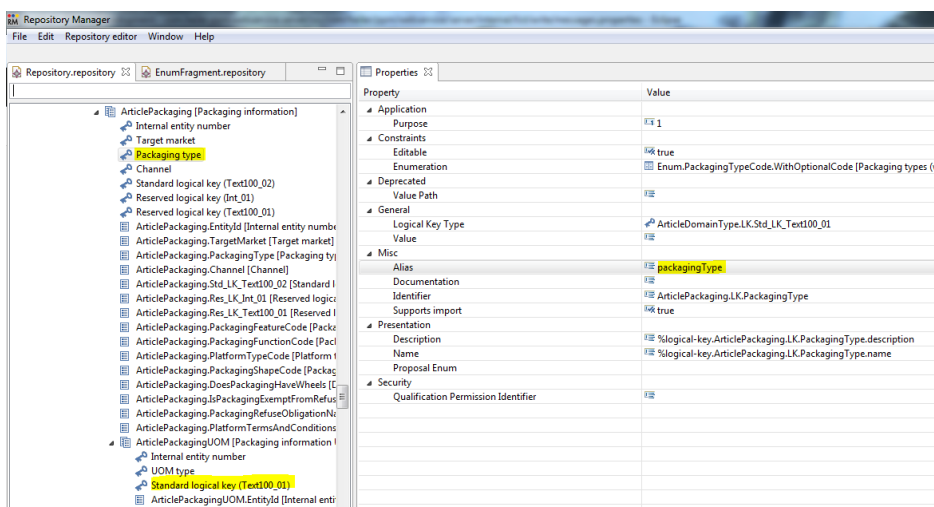
```
    ]
}
```

You will get this error message:

## Result

Found two identical logical key types std\_LK\_Text100\_01 for the same entity hierarchy. Please use the property 'alias' on the logical key in the repository to specify which logical key you want to use.

To fix this issue use a unique alias for all logical keys as recommended.



Don't forget to use the alias in your requests.

## POST [http://localhost:1512/rest/V1.0/list/Article/ArticlePackaging \(with alias\)](http://localhost:1512/rest/V1.0/list/Article/ArticlePackaging (with alias))

```
{
  "columns": [
    {
      "identifier": "ArticlePackagingUOM.PackagingWeight"
    }
  ],
  "rows": [
    {
      "object": {
        "id": "'Item1'@'MASTER'"
      },
      "qualification": {
        "targetMarket": "Barbados",
        "packagingType": "Ampoule",

```

```

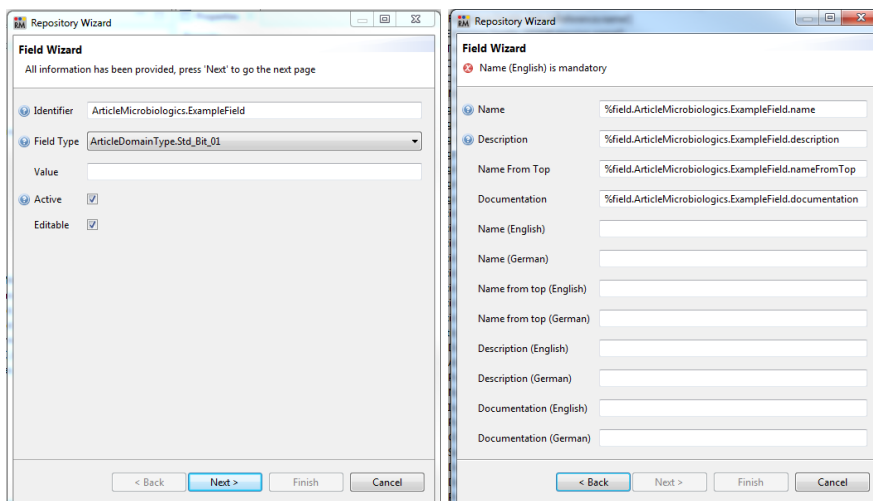
        "uomType": "METRIC",
        "std_LK_Text100_01": "DEFAULT"
    },
    "values": [
        "8.8"
    ]
}
]
}

```

### 1.7.5.5 Create a new field

#### Wizard

The Repository Manager also contains a wizard to create fields. A field has a lot of properties to configure. We won't show you all seven pages of the wizard at this point. Most important properties will be described below. More details can be found in the repository documentation mentioned above.



In the second screenshot you see the language specific properties of a field. The first four fields contain the property identifier for the externalization of the text. From the fifth field on you input the real labels you want to see in the UI.

#### Field Properties

##### Mandatory properties

First of all you need an **identifier** for your field. If there is no reason not to comply to the rule, this will be the same name as in the GDSN documentation. A Product 360 convention is that all fields of type Boolean start with the prefix "Is", e. g. "IsBaseUnit", "IsService", "IsConsumerUnit". As you can see in the examples, another convention is to use camel case for field identifier consisting of multiple words.

Next you have to choose a **field type**. Make sure the field type is not in use already. If you're using `ArticleDomainType`, you have to be careful with text field types.

`ArticleDomainType.Std_Text100_01` to `ArticleDomainType.Std_Text100_10` have upper bound -1 meaning these are list types. Please always try to use a type field which is already configured as list/single value according to your needs.

If you have a list of values, choose a *string field type* so that the whole list can be put in the single database field. To choose the correct length for a field, multiply the length of a single value including a separator between the values with the number of expected values.

*Example:* If you have a field with an enumeration with four entries and the key of each enum entry is a single character, then it is safe to use a `Std_Text10_<number>` field.

Make sure the chosen field type is activated in the types area (`Inactive = false`).

If this field does not correspond to a non editable logical key, you should check that `Editable = true` is set.

#### Language specific properties

In order to see the field in the UI, it needs at least a **name**. You can define a **name from top** which is for example shown in the main table and normally includes the logical key values. For more information see section "Referencing on logical keys in field names" of the above mentioned documentation.

The **description** will be displayed as a tool tip in the field selection dialog and in the import. In general there is an English description in the GDSN documentation which should be used.

- Make sure the description is really a description of the data that should go in this field. Read the description before you add it to Product 360. Delete meaningless information like "0 to 80 character text field" or "Choose value from the drop down". This information is stored elsewhere in the repository.
- Use a meaningful description or leave it empty.
- If the description is a complete sentence, end it with a '.', if not don't use a '.'.

All labels have to be externalized in your client languages. For more information see section "Multi-language support of the repository" in the above mentioned documentation.

There is also a field **documentation** in the repository. The value of this field does not appear in the UI and does not need to be internationalized. It can be used for any documentation purpose and will be visible in the Repository Manager.

#### Support different functionalities

There are some properties defining if a certain field supports a specific functionality which are self explaining like "Cloneable", "Mergeable", "Searchable" and "Supports Data Quality".

#### Export

If the Export purpose is set to '0' the field cannot be exported. The export supports different purposes as described in the section "Export Purpose" of the above mentioned documentation. If you are not sure, start with "Export Purpose" = 1.



**Import**

If the Import purpose is set to '0' the field cannot be imported. If the field should be importable, set the "Import Purpose" = 1. For historic reasons make sure that the deprecated property "Purpose" always has the same value as the Import Purpose and that supports import is set accordingly.  
To determine where this field should be displayed in the "Repository" tree view of the import perspective use the property "Category". That's why it's recommended to keep all fields of one GDSN module in the same category. However, if you have reasons to do it differently, it is possible. The category itself also needs to be created in the repository.

**Service API**

If the field should be available in the Service API, you need to set "Supports service API" = true. For technical reasons the field has to support the service API if you want to use Data Quality (Supports Data Quality = true).

**Constraints**

**Occurrences:** The properties "Upper Bound" and "Lower Bound" define how many values can be stored here.

- For a mandatory value set both properties to 1.
- For a non-mandatory field with one value set lower bound = 0 and upper bound = 1.
- For a list of values set lower bound = 0 and upper bound = -1.

You can define if it should be an INFO, a WARNING or an ERROR in case the upper or lower bound is not met. Normally this should be set to ERROR, otherwise the system will store these values regardless of the field configuration.

**Valid value lists:** There are two places where you can add an enumeration to the field.

- Enumeration - The value of this field must be one of this enumeration. Otherwise the user will get an error and the value won't be stored. If the database already contains values not contained in the enumeration these will no longer be displayed in the UI.
- Proposal Enum - Use the proposal list if the user is allowed to store values which are not contained in the list.

Since the enumeration keys are stored as the values for this field, the chosen field type has to have a matching data type with the key class of the enumeration.

**Field length:** GDSN often defines the max. field length.

- For string values use the properties "Min Length" and "Max Length". Make sure the values you choose here are not lower resp. higher than the min. and max. length in the types area for the corresponding field type. If the field contains a list of values (upper bound = -1), the min. and max. length refer to a single entry, not the entire list.
- For numeric values the "Min Length" and "Max Length" are not used. If you want to restrict a numeric value you have to use "Range Min" and "Range Max". Make sure the values are not lower resp. higher than the min. and max. range defined for the corresponding field type. The decimal separator used here is '.'.  
You can define how many decimal places should be displayed in the UI by using the property "Scale". The displayed value will be rounded. Note that this is only a matter of presentation. In the database all decimal places given in the UI are stored.

To ensure that only values in the defined range will be stored, set "Validation Severity Range" to ERROR. The default value is WARNING.

- For dates also use "Range Min" and "Range Max". This is an example of the pattern you have to use: "yyyy-MM-dd HH:mm:ss". For example: 9999-12-31 23:59:59

If you don't restrict the values in the custom area the default values from the corresponding field type will be used.

**Transitions:** If you store a proxy in the field and you want to make fields of the proxy available (for example in the field selection dialog) you can set the matching root entity to the property "Proxy Transition Entity". More information about transition fields can be found in the knowledge base article at chapter Transition fields - Group name of the Product 360 Documentation.

## Presentation

### "Visible" and "Visible From Top":

- Your field belongs to a deactivated logical key: Set "Visible" and "Visible From Top" to false. The field should not appear in the UI.
- Your field belongs to an activated logical key: Set "Visible" to true, so it will be visible in the sub entity views. Set "Visible From Top" to false since we don't want fields which have to be qualified with the value they contain in the main table.
- Your field doesn't belong to a logical key: By default set "Visible" and "Visible From Top" to true. The field should be available in the main table as well as in the sub entity views.

In order to have a display name in the UI, don't forget to fill in the property "Name" if "Visible" is set to true and property "Name From Top" if "Visible From Top" is true.

### View configuration:

You can configure that a field is present in the default configuration of the corresponding sub entity view. Set "Display By Default" = true. This configuration will be overwritten by the layout stored in the client's workspace. You can also configure in which order the fields should appear in the table using "Default Column Order". The index is 0 based.

You can do the same for the main table using "Display By Default From Top" and "Display Column Order From Top". However this is not recommended for GDSN modules.

## Miscellaneous

**Default value:** In case you create a field belonging to a logical key that is not editable, not only the logical key needs a default value but the field needs the same default value. The default value may also make sense for other fields, especially for mandatory fields.

## Limitations

- In some cases the Item Management data model allows values from 0 - 9,999,999,999 for attributes with datatype "Integer". Due to technical restrictions Product 360 allows only values from 0 to 2,147,483,647 for attributes with datatype "Integer". In real life this should not cause any issues because it's unlikely to have such a large value for any attribute.
- IM defines decimal values in a flexible way by defining the complete length and the max. number of decimal places.

Example: complete length 15 - max. decimal places 15

- 1234567890,12345 is valid
- 12345,1234567890 is valid
- 12345678,12345678 is not valid because the complete length is greater than 15

Product 360 does not support this flexible definition. There is a technical limitation to 10 places before the decimal separator and 6 decimal places. Higher numbers are not supported.

- There is no way to limit the decimal places that are stored in the database. The "Scale" value is only used to format a value in the UI.

#### 1.7.5.6 Deactivate a GDSN entity

If you want to deactivate a GDSN entity read the chapter "Hide a GDSN module" in the [Repository configurations](#) (see page 29) section of the GDSN Accelerator documentation.

#### 1.7.5.7 Create or adjust a valid value list

Many fields need a valid value list. In most cases this is a simple list of some kind of codes mapped to language-dependent labels. Sometimes you need a list of units of measure. First of all, you should check if the needed valid value list is already available.

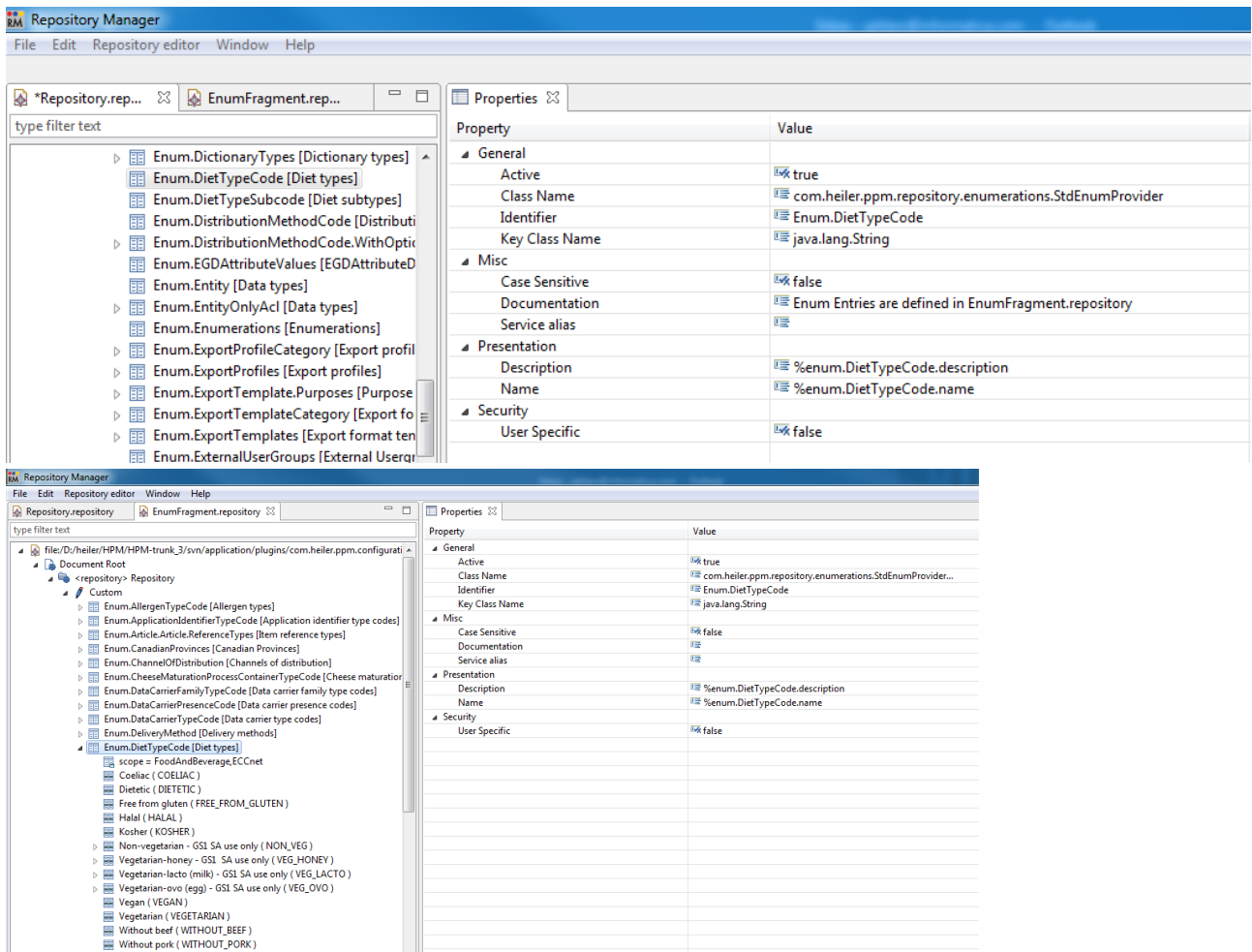


Synonyms for "valid value list" are "look up values", "preset values" and in technical terms "enumeration".

Generate or adjust a repository enumeration

Repository.repository vs. EnumFragment.repository

The "Repository.repository" file contains all enumerations used by repository fields. For GDSN or food and beverage enumerations the repository only contains the definition of the enumeration. The actual enumeration entries are contained in a second file called "EnumFragment.repository". This will keep the repository more organized because there are some GDSN valid value lists with high numbers of entries.



## Create enumeration

Use the Repository Wizard of the Repository Manager to create a new enumeration.

**Repository Wizard**  
Enumeration Wizard  
Configure the general options of an enumeration

Identifier: Enum.

Class Name: com.heiler.ppm.repository.enumerations.StdEnumProvider

Key Class Name: java.lang.String

Active: ☒

< Back Next > Finish Cancel

**Repository Wizard**  
Enumeration Wizard  
All information has been provided, press 'Next' to go the next page

Case Sensitive: ☐

User Specific: ☐

< Back Next > Finish Cancel

**Repository Wizard**  
Enumeration Wizard  
Name (English) is mandatory

Name: %enum.ExampleEnum.name

Description: %enum.ExampleEnum.description

Name (English):

Name (German):

Description (English):

Description (German):

< Back Next > Finish Cancel

Use an identifier that makes it easy to find the corresponding field.

Usually you need to use the key class "string" when creating an enumeration for GDSN.

Enter the properties identifier in the fields "Name" and "Description", shown in the third screen. They should have the pattern `%enum.<identifier without substring 'Enum.'>.<name|description>` as can be seen in the screenshot above. There is no "Name from top", but name and description should be provided at least in English. In case you are using more client languages, please maintain those names in the according "repository.properties" file for your language.

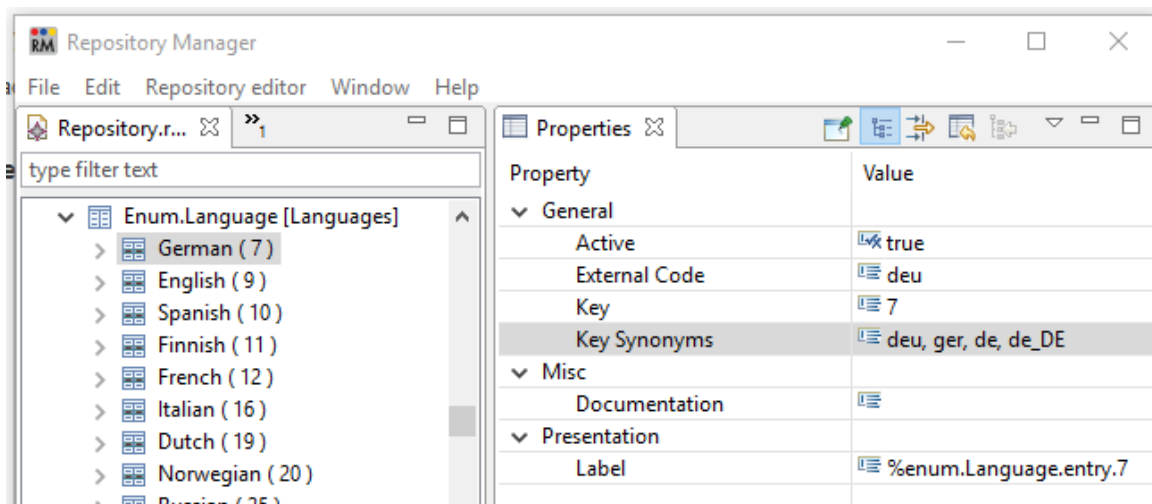
Copy the enumeration into "EnumFragment.repository" in order to add entries in the next step.

For a good trace-ability of your enumeration definition, it is recommended to add a documentation to your enumeration in the "Repository.repository" file: "Enumeration entries are defined in EnumFragment.repository".

#### Add enumeration entries

Key synonyms come in handy if there are multiple different values in the data source which should result in the same value in Product 360.

*Example:* Enum.Language contains enum entries with synonyms. For example you can import "deu", "ger", "de" or "de\_DE" and either value will result in the language key 7 in the system.



Make sure the label is externalized and available in English and your required client languages.

#### Delete enumeration entries

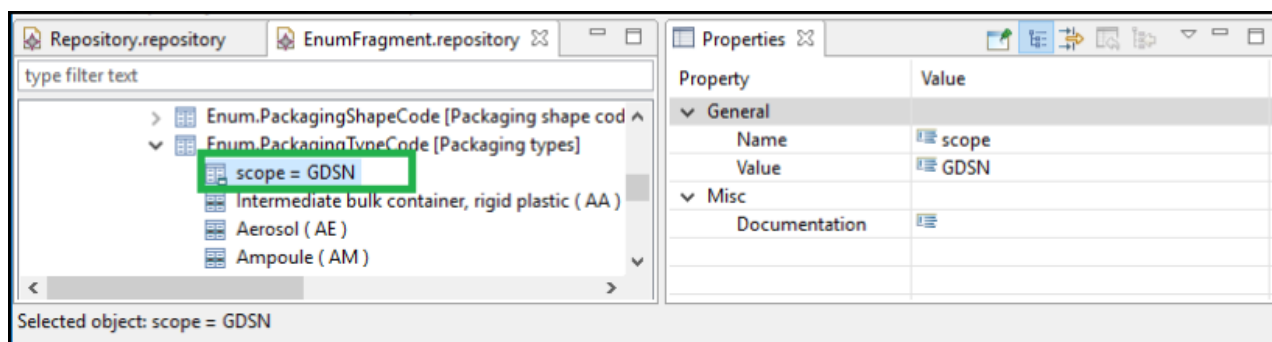
In a customizing, it is allowed to delete enumeration entries.

*Example:* If your customer is a manufacturer of vegan food, you may want to delete the enumeration entry "MEAT" in Enum.DietTypeSubcode in order to reduce false data entered by accident.

Be aware that if the keys of the entries you delete are already in the database or could be imported from another system, these values won't be displayed anymore.

#### Scopes on enumerations and enumeration entries

You may have seen parameters at enumerations or enumeration entries named "scope".

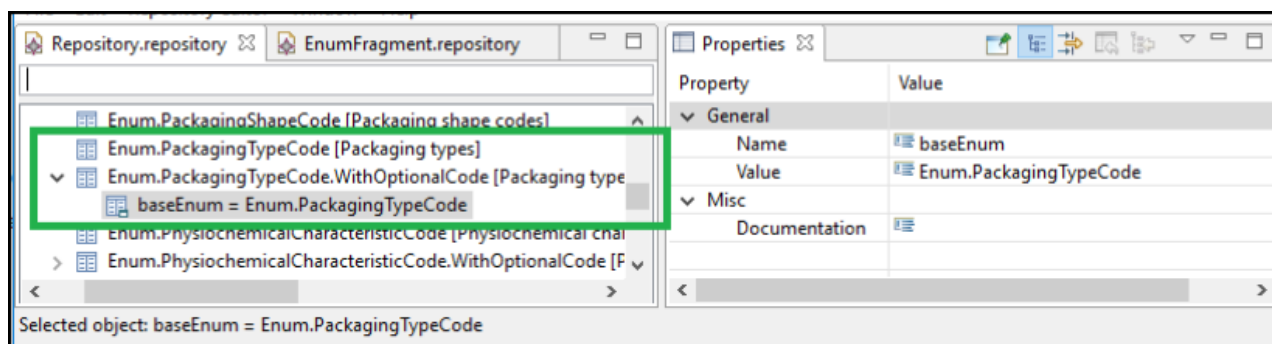


You should not change it or create such parameters at your enumerations or enumeration entries. Those parameters are needed for internal purposes only; they are used to make automatic repository adjustments during startup according to the configuration defined in "application\_modules.properties" file.

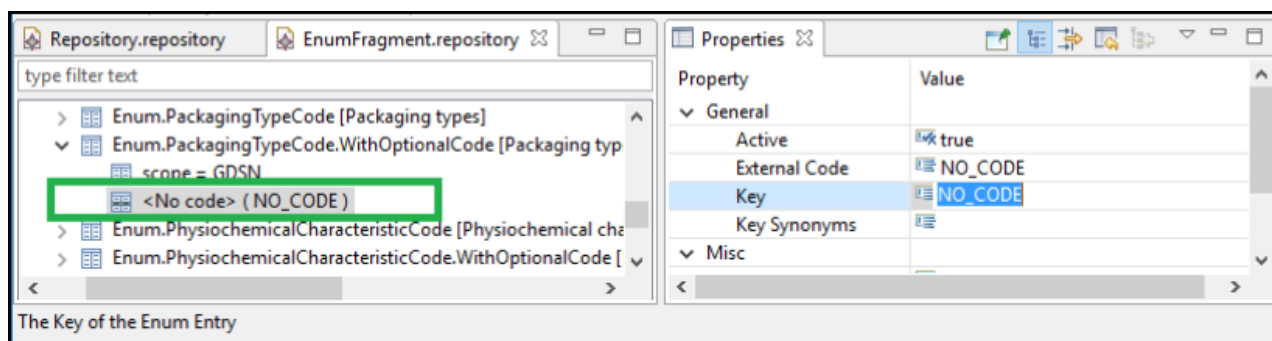
#### Create enumerations with optional code

If you have to enable a mandatory field with a valid value list to get "no value" for an entry (such fields are usually logical key fields), you have to use an enumeration with optional code(s). Such enumerations provide all values of the corresponding valid value list and one or more values that will not be transferred to the GDSN pool.

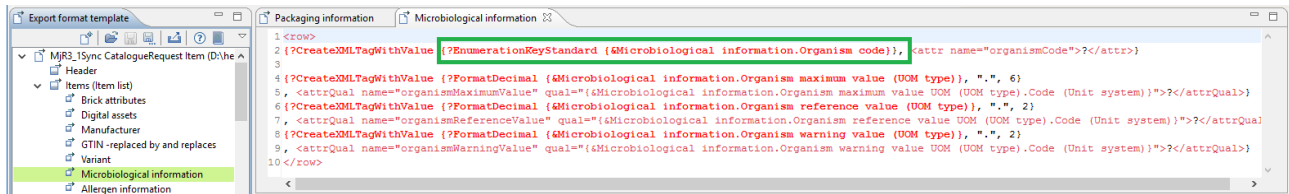
First you create an enumeration as described above, the empty enumeration definition in "Repository.repository" and the entries in "EnumFragment.repository". Then you create a second enumeration with an identifier like "<Identifier of first enumeration>.WithOptionalCode". That second enumeration gets an entry standing for "no code". All other enumeration entries will be used from the "parent" enumeration.



#### 4 Define the enumeration using a base enumeration



If you have to adjust the corresponding export template to transfer item data to the GDSN pool you should use the `EnumerationKeyStandard` export function. Details on that function can be found in the "Technical details (see page 201)" chapter of this GDSN documentation.



## 5 Use export function to output only valid GDSN values

Create or adjust a unit of measure list

The GDSN unit system contains many units of measure. Usually, a valid value list for a field only uses some of those units.

Therefore units are separated into unit categories but unfortunately the categories are different depending on the GDSN pool which is used.

Product 360 contains predefined categories for the IM GDSN system units. For DSE the same categories can be used or new categories can be created as described in the chapters below.

To see a list of valid units for a field open the appropriate GDSN document. Use the Participant Dictionary when using the IM GDSN pool or the Profile Overview when using the DSE GDSN pool.

### GDSN unit categories

The GDSN units are separated into categories. One unit can be in different categories and one field can have valid values of multiple categories.

A list of units and their category can be seen in the *Units* perspective of Product 360. Select the GDSN unit system and add the field "Category" with the field selection dialog. Make sure to qualify the "Category" field with the GDSN parameter.

All units in the unit system "Own GDSN"			
	Code (Own GDSN)	Name (Own GDSN, English)	Category (Own GDSN)
1	BFT	board foot	Volume units
2	EA	each	Count units
3	BP	hundred board feet	Volume units
4	HC	hundred count	Count units
5	CWA	hundred pounds (cwt)/hun...	Mass units
6	MIU	million international unit (...)	Count units
7	LTN	ton (UK) or long ton (US)	Mass units
8	STN	ton (US) or short ton (UK/US)	Mass units

### Use an existing unit enumeration

In order to use a category at a specific field it is necessary to use an unit enumeration. Product 360 has predefined enumerations named after the categories for the IM data pool. In order to map a unit enumeration to a specific field it is best practice to use the Repository Manager.



Property	Value
▼ Application	
Cloneable	true
Export Purpose	1
Import Purpose	1
Mergeable	true
Purpose	1
Searchable	true
▼ Constraints	
Editable	true
Enumeration	Enum.GDSNAreaUnits [GDSN Area units] ▼
Lower Bound	Enum.GDSNAreaUnits [GDSN Area units]
Max Length	Enum.GDSNConfirmationMessageCode [GDSN confirmation message code]
Min Length	Enum.GDSNCountUnits [GDSN Count units]
Picture Clause	Enum.GDSNDataRecipientMessageType [GDSN Data Recipient message type]
Proxy Transition Entity	Enum.GDSNDataSourceMessageType [GDSN Data Source message type]
Range Max	Enum.GDSNDataSourcePublicationMode [GDSN Publication mode]
Range Min	Enum.GDSNDataSourcePublicationRecipient [GDSN Publishate to]
Upper Bound	Enum.GDSNDataSourcePublishMessageType [GDSN Data Source publishing message type]
Validation Severity Enum	Enum.GDSNDensityUnits [GDSN Density units]
Validation Severity Lower Bound	Enum.GDSNDimensionUnits [GDSN Dimensions]
Validation Severity Range	Enum.GDSNEnergyUnits [GDSN Energy units]
Validation Severity Upper Bound	Enum.GDSNExecutedOperation [GDSN executed operation]
▼ General	
Active	true
Field Type	Enum.GDSNMessageInProgress [GDSN Message In Progress]
Identifier	Enum.GDSNMiscUnits [GDSN Miscellaneous units]
Value	Enum.GDSNNutritionQuantityUnits [GDSN Nutrition Quantity units]
▼ Misc	
Documentation	Enum.GDSNOrderUnits [GDSN Order units]
Supports Data Quality	Enum.GDSNPackagingUnits [GDSN Packaging units]
	Enum.GDSNPowerUnits [GDSN Power units]
	Enum.GDSNPressureUnits [GDSN Pressure units]
	Enum.GDSNProductYieldUnits [GDSN Product yield units]

### Creating new unit enumerations

When the predefined unit enumerations do not meet the requirements of the field you want to add, it is possible to create a new enumeration in the repository by using the Repository Manager and include all categories which are needed separated by a ",".

It is important that the `unitSystem` property matches the unit system which should be used for the enumeration. The standard GDSN unit system is **70**.

- > Enum.GDSNProductYieldUnits [GDSN Product yield units]
- > Enum.GDSNProportionUnits [GDSN Proportion units]
- > Enum.GDSNPublicationMessageCode [GDSN publication message code]
- ▼ Enum.GDSNQuantityUnits [GDSN Quantity units]
 

UnitSystem = 70  
 UnitCategory = Area,Count,Dimensions,InfoStorage,Mass,Volume
- Enum.GDSNResponseType [GDSN response type]

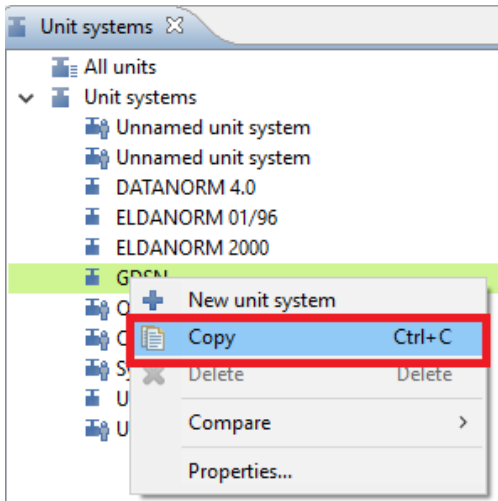
After creating the unit enumeration it can be referenced as enumeration at a field as shown above.

### Creating a custom unit system



Beware when using a custom unit system. All changes made in the standard GDSN unit system are not automatically added to any custom unit system and have to be maintained manually!

When the predefined unit categories are insufficient or not usable for your project, it is possible to create a custom unit system based on the GDSN unit system. To do this simply copy the GDSN unit system in the *Units* perspective.



After that it is possible to add or modify units and their categories for your needs in the *Unit maintenance* view.

To use the unit system created this way some additional steps have to be done in order to use it in the whole application:

#### **Get the custom unit system ID**

As stated above GDSN has a default unit system and in order to change that the ID of the new unit system is needed.

To get the unit system ID the Service API can be used as shown in the picture below.

GET
http://localhost:1512/rest/V1.0/list/UnitSystem/UnitSystemLang/bySearch?query=UnitSystemLang.Name = "Own GDSN"

Body
Cookies
Headers (3)
Tests

Pretty
Raw
Preview
JSON

```

1 {
2   "cacheId": "20170407_123539_0",
3   "entityIdentifier": "UnitSystemLang",
4   "totalSize": 1,
5   "startIndex": 0,
6   "pageSize": 100,
7   "rowCount": 2,
8   "columnCount": 0,
9   "columns": [],
10  "rows": [
11    {
12      "object": {
13        "id": "10003",
14        "label": "Own GDSN",
15        "entityId": 3200
16      },
17      "qualification": {},
18      "values": []
19    }
20  ]
21 }

```

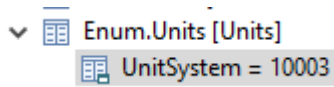


It is important to add a blank space before and after the "=" sign when qualifying UnitSystemLang.Name with a value.

The ID marked in red is the unit system ID of the custom unit system. To have Product 360 use the custom unit system instead of the GDSN unit system open the Repository Manager.

#### (Optional) Use the new unit system in all unit fields

To use the new unit system in all UOM fields, including the fields which are not GDSN related, navigate to the enumeration `Enum.Units` and change the parameter `UnitSystem` to the ID retrieved from the Service API.



When restarting Product 360 the custom unit system will be used instead of the standard unit system.

#### Create category enumerations

As described in the section "Creating new unit enumerations", it is needed to add unit enumerations that use the custom unit system as enumeration parameter.

#### Change export UOM fields to use custom unit system

Make sure to use the new unit system in the export for all UOM fields.

The screenshot shows the 'Variables' tab in the Informatica MDM interface. On the left, a list of variables is shown, with 'Unit system' selected. On the right, the details for the 'Unit system' variable are displayed. The 'Name' field contains 'Unit system'. The 'Data type' is set to 'Unit system'. The 'Value' field contains 'Own GDSN', which is highlighted with a red box. Below the 'Value' field, the 'Mandatory value' and 'Editable' checkboxes are both checked.

Create a new unit

If a unit is missing in the GDSN unit system and is available in the official GDSN documents please contact the support as they will provide a database script to add the unit.

### 1.7.5.8 Checklist: Test the module

#### Check now

- ☐ Start validation in repository editor (right-click on a node you want to be validated, choose "Validate")
  - ☐ Validate the new (sub) entity
  - ☐ Validate whole custom node
- ☐ Check data type, field length and valid values
 

This can be checked at different places. Try to write values of the correct and incorrect data type and field length or write correct and incorrect valid values using the import, the Service API or the UI. The positive check is not enough. If the import does not report an error when writing a valid value, it can either mean you added the valid value list correctly or it can mean you forgot to attach the enumeration to your field. Always try to get the errors you expect.
- ☐ Check import
  - ☐ Check category
  - ☐ Check name (It's the name of the logical key, not the corresponding field, displayed in the import UI)
  - ☐ Check in the view "Field details" if the properties are as expected
- ☐ Check export
  - ☐ Only needed fields can be exported
  - ☐ No deactivated logical keys appear in the list of fields
  - ☐ Only needed entities are available as export sub-data types
  - ☐ The field names should contain all visible qualifications (logical keys)
- ☐ Check Service API
  - ☐ Look at the meta API
    - ☐ Invisible fields are not available
    - ☐ Names and descriptions are available
    - ☐ The valid value lists are correct
  - ☐ Read fields
  - ☐ Write fields

- ☐ Check that each visible logical key has a unique alias within its path to the root entity

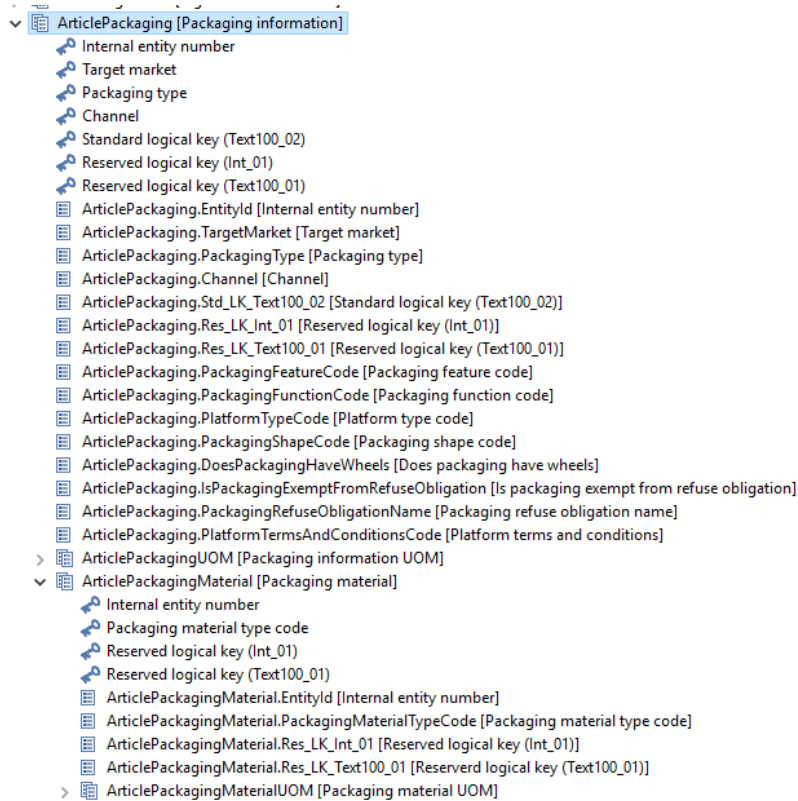
### Check after you built the UI

- ☐ Are table or detail views available in the desktop client? Is detail tab in web client available?
- ☐ Check that all fields are visible
  - ☐ Check in main table (qualified access)
  - ☐ Check in detail table and detail tab (unqualified access)
- ☐ Check if data can be created, edited and be deleted in desktop client and web client
- ☐ Check datatype, field length and valid values (if not already done outside the UI)
- ☐ Check correct display names and descriptions
  - ☐ Are labels available in English and needed client languages?
  - ☐ Do names from top contain all necessary logical keys? Are the logical keys in the expected order?  
You can check this either in the column headers or in the field selection dialog of the main table. It is recommended that the order of the logical keys is determined by the containing entity. Go from the root entity to the display fields and collect the visible logical keys on this path.
- ☐ Check if the item is still clone-able and merge-able after you maintained data
- ☐ Check if the item search works correctly
- ☐ Make sure the desktop view(s) contain(s) a suitable set of default columns and the layout of the web detail tab is suitable

## 1.7.6 UI adjustments

### 1.7.6.1 Creating additional elements in Desktop UI

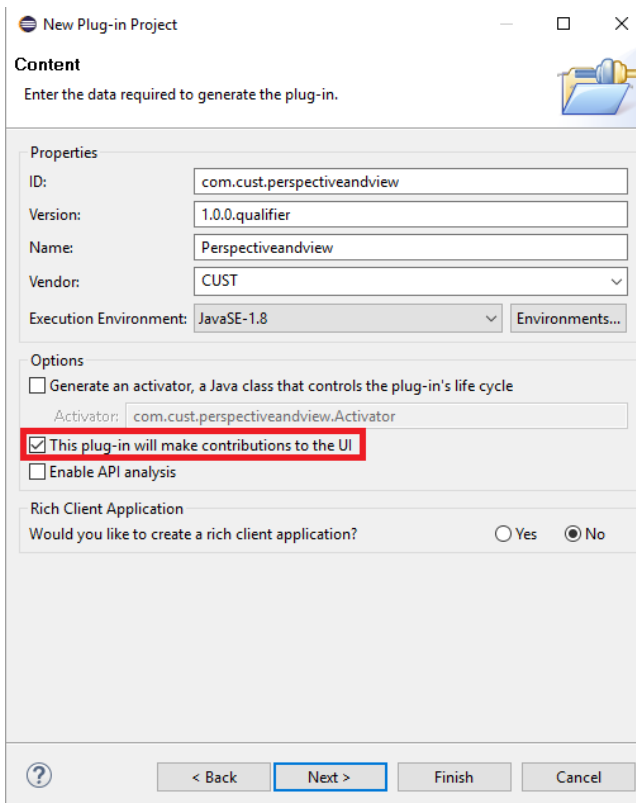
After we created our new entity `ArticlePackaging` we want to edit data using the UI of Product 360.



We will need a view for the sub entity `ArticlePackaging` and one for the sub entity `ArticlePackagingMaterial`. The customer decided that no additional views for the UOM or the Lang sub entities are needed as they can be seen in their appropriate higher level sub entity.

#### General information

To have all UI elements concentrated in a specific place it is best practice to create a separate plugin for the UI elements ending with `.ui`, for example `com.informatica.customizing.additionalviews.ui`. When creating the plugin please make sure that the checkbox "This pug-in will make contributions to the UI" is checked.



**New Plug-in Project**

Content  
Enter the data required to generate the plug-in.

**Properties**

ID:

Version:

Name:

Vendor:

Execution Environment:  [Environments...](#)

**Options**

☐ Generate an activator, a Java class that controls the plug-in's life cycle

Activator:

☒ This plug-in will make contributions to the UI

☐ Enable API analysis

**Rich Client Application**

Would you like to create a rich client application? ☐ Yes ☒ No

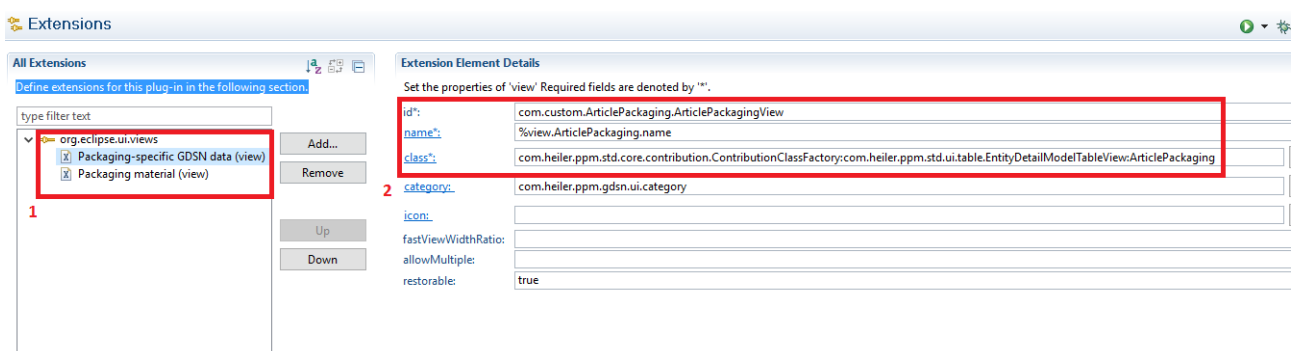
[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

When the box is checked a `plugin.xml` file will be created where we can add contributions to specific extension points.

After each headline the result can be seen by adding the new UI plugin to the plugin directory of Product 360 client.

## Adding views

A view is used to show every field from the given entity. Every sub entity, besides UOM and Lang entities, needs a separate view.



**Extensions**

All Extensions

Define extensions for this plug-in in the following section

type filter text

1 **org.eclipse.ui.views**

- ☒ Packaging-specific GDSN data (view)
- ☒ Packaging material (view)

[Add...](#) [Remove](#)

[Up](#) [Down](#)

**Extension Element Details**

Set the properties of 'view' Required fields are denoted by '\*'.

2

id:	com.custom.ArticlePackaging.ArticlePackagingView
name:	%view.ArticlePackaging.name
class:	com.heiler.ppm.std.core.contribution.ContributionClassFactory:com.heiler.ppm.std.ui.table.EntityDetailModelTableView:ArticlePackaging
category:	com.heiler.ppm.gdsn.ui.category
icon:	
fastViewWidthRatio:	
allowMultiple:	
restorable:	true

1. Contribute the views to the extension point `org.eclipse.ui.views`

2. Complete the mandatory properties marked with a \*. Details about the properties can be found in the Creating custom perspectives section (chapter "Customizing") in the document "Informatica MDM - Product 360 - <VERSION>- Knowledgebase, Installation and Customization".

Because the `ArticlePackaging` entity is a direct sub entity of the root entity `Article` we can use the `com.std.ui.table.EntityDetailModelTableView` which is used by direct sub entities.

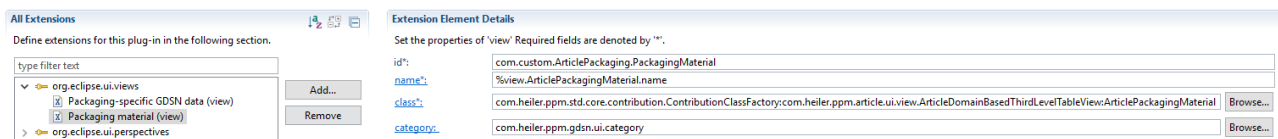
Sub entities which are not direct sub entities from a root entity are called third level hierarchy entities. They must use a different `TableView`.

### Third level hierarchy

In our example `ArticlePackagingMaterial` is a third level hierarchy entity (`Article` → `ArticlePackaging` → `ArticlePackagingMaterial`). When having a third level hierarchy in the repository it is necessary to use another value at the class text field because the content which is shown in the `ArticlePackagingMaterial` view should depend on the selection made in the `ArticlePackaging` view.

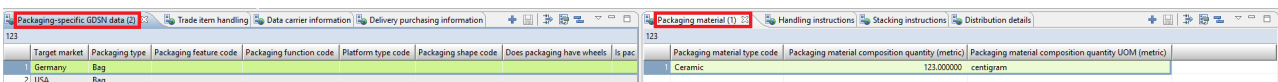
The `TableView` which we have to use is: **`com.heiler.ppm.article.ui.view.ArticleDomainBasedThirdLevelTableView`**

So the configuration for the third level hierarchy view looks like this:



### **Take care when placing the ArticlePackagingMaterial view**

It is necessary that both views can be visible at the same time and therefore we recommend to place the `ArticlePackaging` view on the opposite side. Otherwise the selection paradigm cannot work. This has to be done for every third level hierarchy view and their counterparts.



### Adding perspectives

To add a perspective we first need to add a new java class to the UI plugin containing the new perspective with a perspective identifier.



**GDSNCorePerspective**

```

1  public class PackagingPerspective implements IPerspectiveFactory
2  {
3      public static String PERSPECTIVE_ID =
4          "com.custom.perspectiveandview.ArticlePackagingPerspective"; //$NON-NLS-1$
5
6      @Override
7      public void createInitialLayout( IPageLayout layout )
8      {
9          layout.setEditorAreaVisible( false );
10     }

```

Now we have to add a contribution to the `org.eclipse.ui.perspectives` extension point.

1. Add perspective to the extension point.
2. Fill the mandatory properties marked with a \*. More information about perspective properties can be found in the Creating custom perspectives section (chapter "Customizing") in the document "Informatica MDM - Product 360 - <VERSION>- Knowledgebase, Installation and Customization".

In our case the value for the `class` attribute is the class we just created. The name has to be fully qualified with the package where the class is located.

Here: `com.custom.perspectiveandview.PackagingPerspective`.

The perspective can now be seen in the client but the perspective is empty. In the next chapter we will add views to the perspective.

### Adding views to a perspective

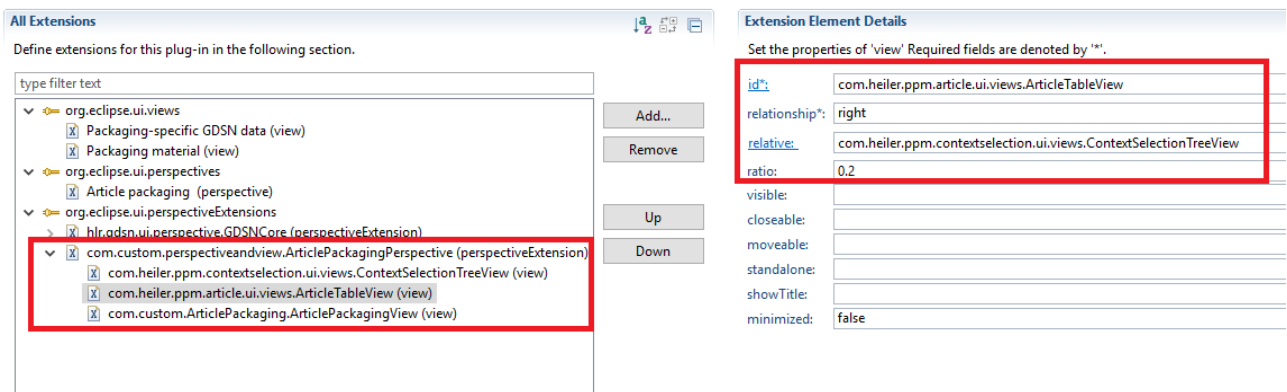
To add a view to an existing perspective you have to add a contribution to the `perspectiveExtensions` extension point .

1. The `target id` of the contribution `perspectiveExtension` has to be the id of the perspective where you want to make the modifications. In our example we want to add a view to our "ArticlePackagingPerspective" we created in the section above. So we use the id `com.custom.perspectiveandview.ArticlePackagingPerspective` . We can then right click the perspective extension contribution and add a view.
2. In the "Extension Element Details" (2) we can then fill the values for the view. More information about the creation of perspective extensions can be found in the Creating custom perspectives section (chapter "Customizing") in the document "Informatica MDM - Product 360 - <VERSION>- Knowledgebase, Installation and Customization".

### Adding standard views to a custom perspective

When creating a new perspective most of the time it is necessary to add standard views like the *Article table (Items #1)* or the *Context selection* view to it. To do this, the same procedure for adding views in the `perspectiveExtension` has to be done.

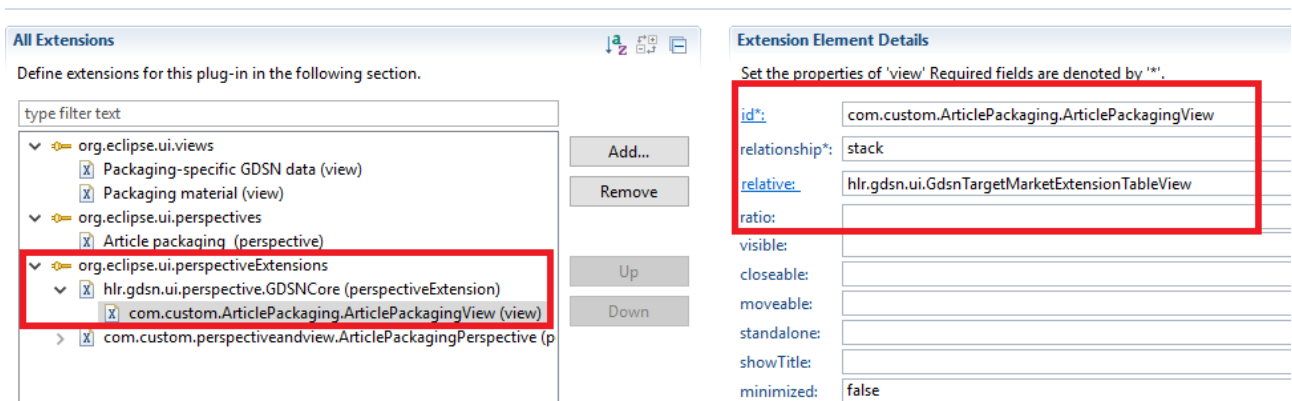
The view ids can be found in the **plugin.xml** of standard `com.heiler.*.ui` plugins.



### Extend a standard GDSN perspective with custom views

It is possible to add a view to a standard GDSN perspective. To do this the same procedure like adding views to perspectives is used. That means only a contribution to the `perspectiveExtensions` extension point is needed. To add for example a view to the GDSN core perspective, the id "hlr.gdsn.ui.perspective.GDSNCore" has to be declared in the relative value in the "Extension Element Details".

All perspective and view ids from GDSN can be found in **plugin.xml** of the `com.heiler.ppm.gdsn.ui` plugin.



### 1.7.6.2 Creating additional elements in Web UI

#### General information

For editing detail tabs in the web, a detailed documentation can be found in the chapter Detail Tab Definition Examples in the document "Informatica MDM - Product 360 - <VERSION>- Knowledgebase, Installation and Customization".

To add new tabs when working with items in the web UI we have to copy the generated `article.detailtab.xml` from the `server/configuration/HPM/webdefinitions/default` directory to the `server/configuration/HPM/webdefinitions` directory. That way it will not get overwritten when restarting the Product 360 Server.

We can now edit the detail tab for our needs.

**i** When referencing Product 360 repository fields, make sure to specify the field identifier from the **custom area** but not the field type identifier from the types area.

#### Adding a new field

When only adding a new field to an existing module just add the tag `field` with the identifier of the new field to the already existing field group or table group.

```
[...]
<fieldGroup displaySectionWidget="true" subEntityId="ArticlePackaging">
[...]
  <field identifier="ArticlePackaging.NewFieldCreatedInTheRepository"/>
[...]
```

Adding a new module

Depending on the module, you can choose between a field group and a table group.

A field form group can be seen as a single field list. We will use the `fieldGroup` for the `ArticlePackaging` and `ArticlePackagingUOM` entity.

Make sure to use the field identifier from the custom area of the repository.

#### Article packaging definition

```
<definition debugId="article_packaging_tab" i18NKey="%web.article.detail.tab.packaging" permissionId="web.article.detail.tab.packaging" position="509" rootEntity="Article">
  <column/>
  <column>
    <enumGrouping caption="%field.targetMarket.name" enumIdentifier="Enum.Territory" selectable="true" value="US"/>
    <enumGrouping caption="%field.packagingType.name" enumIdentifier="Enum.PackagingTypeCode.WithOptionalCode" selectable="true" value="NO_CODE"/>
    <fieldGroup displaySectionWidget="true" subEntityId="ArticlePackaging">
      <field identifier="ArticlePackaging.PackagingFeatureCode"/>
      <field identifier="ArticlePackaging.PackagingFunctionCode"/>
      <field identifier="ArticlePackaging.PlatformTypeCode"/>
      <field identifier="ArticlePackaging.PackagingShapeCode"/>
      <field identifier="ArticlePackaging.DoesPackagingHaveWheels"/>
      <field identifier="ArticlePackaging.IsPackagingExemptFromRefuseObligation"/>
      <field identifier="ArticlePackaging.PackagingRefuseObligationName"/>
      <field identifier="ArticlePackaging.PlatformTermsAndConditionsCode"/>
      <logicalKey identifier="ArticleDomainType.LK.Std_LK_Text100_01" selectable="true" value="&lt;No code&gt;"/>
      <logicalKey identifier="ArticleDomainType.LK.TargetMarket" selectable="true" value="USA"/>
    </fieldGroup>
    <fieldGroup displaySectionWidget="true" subEntityId="ArticlePackagingUOM">
      <field identifier="ArticlePackagingUOM.UsableProductVolume"/>
      <field identifier="ArticlePackagingUOM.UsableProductVolumeUOM"/>
      <field identifier="ArticlePackagingUOM.PackagingWeight"/>
      <field identifier="ArticlePackagingUOM.PackagingWeightUOM"/>
      <logicalKey identifier="ArticleDomainType.LK.Std_LK_Text100_01" selectable="true" value="&lt;No code&gt;"/>
      <logicalKey identifier="ArticleDomainType.LK.TargetMarket" selectable="true" value="USA"/>
      <logicalKey identifier="ArticleDomainUOMType.LK.UOMType" selectable="true" value="metric"/>
    </fieldGroup>
  </column>
</definition>
```

</definition>

The result will look like this:

Trade item lifespan	Data carrier information	Item certifications	Sustainability information	Delivery purchasing information	Packaging-specific GDSN data
Target market	USA				
Packaging type	<No code>				
Packaging feature code:	No content				
Packaging function code:	No content				
Platform type code:	No content				
Packaging shape code:	No content				
Does packaging have wheels:	No content				
Is packaging exempt from refuse obligation:	No content				
Packaging refuse obligation name:	No content				
Platform terms and conditions:	No content				
UOM types	metric				
Usable product volume (metric):	No content				
Usable product volume UOM (metric):	No content				
Packaging weight (metric):	No content				
Packaging weight UOM (metric):	No content				

For the sub entity `ArticlePackagingMaterial` we will use the `tableGroup` to show the fields in a table.

#### Article packaging material definition

```
<definition debugId="article_packagingMaterial_tab" i18NKey="%web.article.detail.tab.packagingMaterial" permissionId="web.article.detail.tab.packagingMaterial" position="510" rootEntity="Article">
  <column/>
  <column>
    <tableGroup>
      <actionPanel>
        <actionButton action="create" i18NKey="%web.article.detail.tab.create.packagingMaterial"/>
        <actionButton action="edit" i18NKey="%web.article.detail.tab.edit.packagingMaterial"/>
        <actionButton action="delete" i18NKey="%web.article.detail.tab.delete.packagingMaterial"/>
      </actionPanel>
    </tableGroup>
  </column>
</definition>
```

```

<fieldFormGroup displaySectionWidget="true" subEntityId="ArticleP
ackagingMaterialUOM">
  <field identifier="ArticlePackaging.TargetMarket"/>
  <field identifier="ArticlePackaging.PackagingType"/>
  <field identifier="ArticlePackagingMaterial.PackagingMaterial
TypeCode"/>
  <field identifier="ArticlePackagingMaterialUOM.UOMType"/>
  <field caption="%field.packagingMaterialCompositionQuantity.n
ame" identifier="ArticlePackagingMaterialUOM.PackagingMaterialCompositionQuantity"/>
  <field caption="%field.packagingMaterialCompositionQuantityUO
M.name" identifier="ArticlePackagingMaterialUOM.PackagingMaterialCompositionQuantityU
OM"/>
</fieldFormGroup>
</actionPanel>
<tableDefinition i18NKey="" identifier="detail_packagingMaterial"
rootEntity="ArticlePackagingMaterialUOM">
  <field identifier="ArticlePackaging.TargetMarket" sortable="true"
/>
  <field identifier="ArticlePackaging.PackagingType" sortable="true"
"/>
  <field identifier="ArticlePackagingMaterial.PackagingMaterialType
Code" sortable="true"/>
  <field caption="%field.packagingMaterialCompositionQuantity.name"
identifier="ArticlePackagingMaterialUOM.PackagingMaterialCompositionQuantity"
sortable="true"/>
  <field caption="%field.packagingMaterialCompositionQuantityUOM.na
me" identifier="ArticlePackagingMaterialUOM.PackagingMaterialCompositionQuantityUOM"
sortable="true"/>
</tableDefinition>
</tableGroup>
</column>
</definition>

```

The result will look like this:

Trade item lifespan	Data carrier information	Item certifications	Sustainability information	Delivery purchasing information	Packaging-specific GDSN data	Packaging material	Trade item h
<div><div><div><div>+</div><div></div><div></div></div></div></div>							
Target market ▲	Packaging type	Packaging material type code	Packaging material composition quantity (metric)	Packaging material composition quantity UOM (me			
Germany	Bag	Ceramic	123.000000	centigram			
USA	Bag	Ceramic	125.000000	centigram			

Note that the action panel, marked in red, has to be explicitly declared in the XML as `actionButton` to be shown.

### 1.7.6.3 Interface visibility and display rights

When creating a new view and contributing it to the extension point the display right is automatically generated and can be seen in the "Interface visibility" view in the desktop client.

For more information about the visibility of the new views and perspectives please read the "Frontend visibility" chapter in the "Informatica MDM - Product 360 - <VERSION>- Knowledgebase, Installation and Customization" document. There is a description how to manage your views and perspectives with action and visibility rights.

#### 1.7.6.4 Limitation

It is not possible to display sub entities which exceed the 4th level repository hierarchy. When having such a case have a look at the "[Deep structures](#)" (see page 107) chapter in the GDSN Implementation Guidelines.

### 1.7.7 Data validations

Product 360 provides several ways to validate and format your product information.

- [Data model validation](#) (see page 155)
- [Data quality checks by IDQ \(Informatica Data Quality\)](#) (see page 155)
- [Validation and formatting during import](#) (see page 156)
- [Validation and formatting during export](#) (see page 158)

For GDSN those are needed to get the data correctly into the data model of Product 360, maintain the data correctly in Product 360 and export the data in the expected format of 1WS.

This chapter is referring to the "Informatica MDM - Product 360 - Desktop\_ <Version>\_UserManual\_en.pdf" which will be called "Product 360 User Manual" in the following sections.

#### 1.7.7.1 Data model validation

The data model validation is based on the repository data model. During importing and maintaining the data, Product 360 validates the length of character strings, the ranges of numeric values and whether the entered value is compatible to the defined enumeration if existent. In addition it checks in most cases whether it is a mandatory value or not.

See chapter "[Data model](#) (see page 121)" for more information.

#### 1.7.7.2 Data quality checks by IDQ (Informatica Data Quality)

It is possible to check your product information automatically and manual by using IDQ. Product 360 is delivered with a set of standard rules which can be adapted to your requirements by configuration. It can be that you cannot find a standard quality rule which fulfills your needs. In this case you need to create your own rule. Based on rules you can create your specific data quality configuration which is described in the "Product 360 User Manual" in chapter "Data quality checks". For GDSN the data quality checks are mainly used for data source scenario because in data recipient scenario you get only valid data from 1WS.

##### Creating a data quality rule

In the case you cannot define your required check by using a standard data quality rule you need to write a custom data quality rule. Please take a look at the documentation for Data Quality to be able to do so.

## Creating a data quality configuration

You can define a specific data quality check by using a data quality configuration which is based on a data quality rule. The necessary steps to define such data quality configuration are described in the "Product 360 User Manual" in chapter "Creating quality rules".

In GDSN you often need a rule which is checking if field A is not empty, and if so field B must not be empty as well. A typical example for this is a UOM (unit of measure) value and the according UOM field like "Usable product volume" and "Usable product volume UOM". The following picture shows how the configuration is looking for this:

The screenshot shows the 'Data quality configuration' window. The left pane shows a tree view with the following structure:

- GDSN3
  - GDSN\_Customizing
    - MyGroup (Item)
      - Check items with 'Usable product volume UOM' populated**
    - LMIV

The main pane displays the configuration for the selected rule:

**Name:** Check items with 'Usable product volume' have a 'Usable product volume UOM' populated ☐ Hidden

**Description:** Check items with 'Usable product volume' have a 'Usable product volume UOM' populated

**Rule name:** Check\_IfNotEmptyConditionNotEmpty

**Data type:** Packaging information

Input port	Data type	Field
inObjectID	string(512)	Object code number
inConditionField	string(512)	Usable product volume
inCheckField	string(512)	Usable product volume UOM
inStatusMessage	string(50)	

**Data type:** Item

Output port	Data type	Field
outObjectID	string(512)	Object code number
outStatusMessage	string(4096)	Message
outStatusCode	string(10)	Status

**Data preprocessor:** [Dropdown menu]

## Automatic data quality check execution configuration

Product 360 provides some triggers like "Export started", on which a data quality check will be automatically executed. The "Product 360 User Manual" describes in chapter "Planning quality checks" how to use the triggers.

### 1.7.7.3 Validation and formatting during import

In the import you can specify functions for the information to be imported. The "Product 360 User Manual" contains the list of many available import functions (import function reference) and also all functionality in the import. Furthermore you can create your own customized import function or validate your file on your own via your customized pre-import step (see documentation for Customizing Import and Hot folder).



Some examples

- The value "1001.101" contained in the import file will be formatted with 2 decimal places and will show the thousand separator. The imported value would be "1,001.1".

The screenshot displays the Informatica MDM interface. On the left, the 'Data source' pane shows a tree structure for 'PackagingInformationTestData\_10\_Items.xlsx (Sheet1)'. The 'Structure' pane on the right shows a tree for 'Packaging information'. The 'Content' pane at the bottom shows a table with columns 'Usable product volume' and 'Content'. The 'Field details' pane on the right shows the mapping for 'Usable product volume' with the formula: `format([Usable product volume],2,true)`.

Structure	Selection	Content
✓ PackagingInformationTestData_1		
✓ Item no.		
✓ Target market		
✓ Packaging type		
✓ Packaging feature code		
✓ Packaging function code		
✓ Platform type code		
✓ Packaging shape code		
✓ Packaging refuse obligation name		
✓ Platform terms and conditions		
✓ is packaging exempt from refuse		
✓ Does packaging have wheels		
✓ UOM type (usable product volume)		
✓ Usable product volume		

Structure	Assignment	Content
✓ Packaging information		
✓ Packaging information		
Target market	Target market	USA
Packaging type	Packaging type	Intermediate bulk container, rigid ...
Packaging feature code	Packaging feature code	Lid
Packaging function code	Packaging function code	Dispenser
Platform type code	Platform type code	ISO 0 pallet
Packaging shape code	Packaging shape code	Bar
Does packaging have wheels	Does packaging have wheels	Not applicable
Is packaging exempt from refuse obligation	is packaging exempt from refuse o...	Unspecified
Packaging refuse obligation name	Packaging refuse obligation name	obligation name 1
Platform terms and conditions	Platform terms and conditions	No exchange / no return
✓ Packaging information UOM		
UOM type	UOM type (usable product volume)	imperial
Usable product volume	Usable product volume	1,001.100000

Content	Frequency	Details
Usable product v...		
1	1001.101	

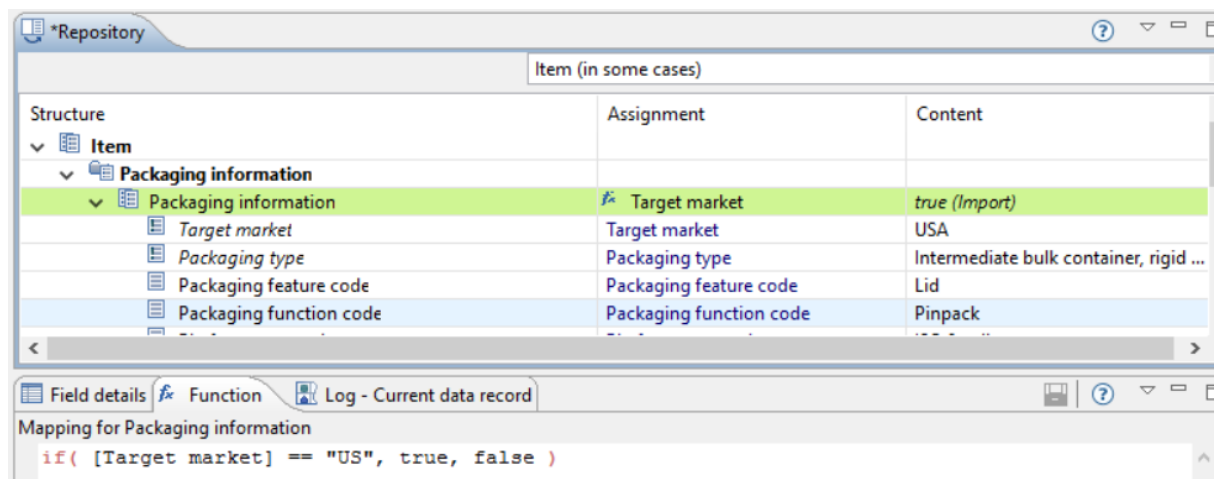
- Define the used pattern for decimal values and date format pattern for the whole import file or for each field of your data source via context menu.

The screenshot shows the 'Data source' pane with a tree structure for 'PackagingInformationTestData\_10\_Items.xlsx (Sheet1)'. A context menu is open over the 'Usable product volume' node, showing options: 'Rename data source', 'Collapse from here', 'Collapse all nodes', 'Expand from here', 'Expand all nodes', and 'Format settings' (highlighted).

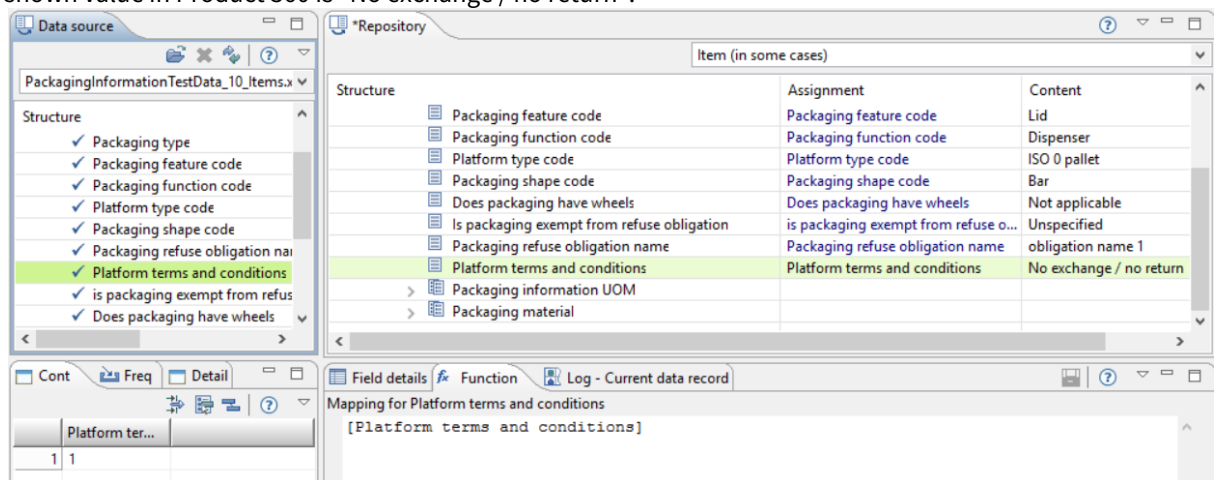
Structure	Selection	Content
✓ UOM type (usable product volume)		IMPERIAL
✓ Usable product volume		1122.313
✓ Usable product volume		4G
✓ Packaging type		FOAM
✓ UOM type		METRIC
✓ Packaging type		10.0001

Content (1)	Usable product volume
1	1122.313

- Define if an entity should be imported by using functions. Here the packaging information will be only imported if the "Target market" is "US".



- It is recommended for fields with enumeration to import their values by using the defined key in the repository. In the example the "Platform terms and conditions" value in the Data Source is "1" but the shown value in Product 360 is "No exchange / no return".



#### 1.7.7.4 Validation and formatting during export

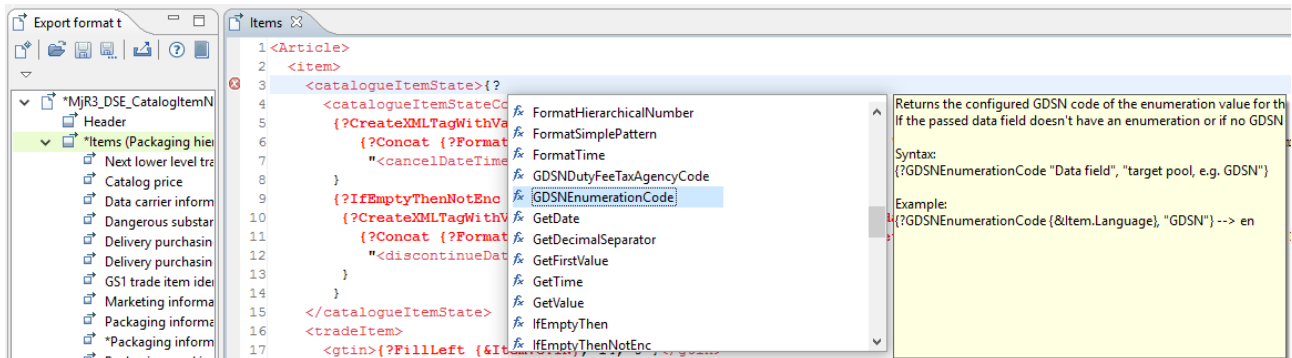
The export can check the data for specific criteria and also format the data in the expected format. Those configurations have to be done within the export format template.

##### Validation rules for data fields

You can define validation rules for each data field. Those have the focus on the according field value and cannot take references to other field values. In the chapter "Validation rules for data fields" of "Product 360 User Manual" it is described how to use this kind of validation.

## Validate and format by export functions

The export comes with many functions where the Product 360 User Manual describes many of them in chapter "Export function reference". The "Product 360 User Manual" describes many of them in chapter "Export function reference". The currently available list of functions, including the GDSN specific functions, can be displayed in Export template perspective of Product 360 directly. You can access those via typing "{?" and pressing "Ctrl+Space".



By using the correct function you can ensure that the information is exported in the expected format. Please format all your numbers and enumeration values as shown below. In addition always format your date and boolean values as well. You can use the delivered export templates as a reference in case you are unsure how to export it correctly.

```
<packagingMaterial>
  {?CreateXMLTagWithValue {?EnumerationKey {&Packaging material.Packaging material type code}}}
  , "packagingMaterialTypeCode"?</packagingMaterialTypeCode>"}
  {?CreateXMLTagWithValue {?FormatDecimal {&Packaging material.Packaging material composition quantity (UOM type)},".",6},
  ?<packagingMaterialCompositionQuantity measurementUnitCode="{&Packaging material.Packaging material composition quantity UOM (UOM type).Code (Unit system)}"}>
  ?
</packagingMaterialCompositionQuantity>"}
</packagingMaterial>
```

You can find more details in chapter [Technical details](#) (see page 201) of GDSN Accelerator documentation.

## Validate the exported file

The GDSN communication is based on XML files. Therefore each message has defined XSD schemata which the export can use to validate the exported file. This validation will fail in case of exporting data in a not expected way. If an XSD validation error occurs, the export will be canceled and you can check if you might missed to format one of your values. Please ensure that you are not exporting an XML tag without value because this is leading to an XSD failure as well.

### Limitations:

- If the export fails due to an XSD validation error, the file will not be saved. This could be a problem when the XSD validation error message is not clear. This is especially the case for flex attributes which have generic XML tags and so the error messages are not unambiguous. A possible workaround is to setup a post export step (at the very beginning) which copies the file to a local directory. Then you can check the line number from the error message and see what's really wrong.
- If no items are exported because the assortment is empty or all of the items are "sorted out" due to a failing DQ rule per item, the XML file will not contain any item. This leads to an XSD validation error and the file will not be sent to the GDSN pool. Although the behavior is correct but it is not obvious to the user what happened.

- If the export fails due to an XSD validation error the process overview shows "Cancelled by user". This is currently a technical restriction which has no bad influence on any data but can be confusing.

You can find more details in chapter [Technical details](#) (see page 201) of GDSN Accelerator documentation.

## 1.7.8 Enhance Export Templates

All messages we send to one of the supported GDSN pools are created by the P360 export. If you enhance the data model, especially those messages containing item data have to be adjusted. This applies mainly to data source scenarios.

### 1.7.8.1 Adjust the export template

The first step is to find out where to place the additional data in the XML structure of the message. This is defined in the corresponding XSD files. If you have an example file by hand, it can be easier to have a look in it.

In this chapter we will explain how we add a new GDSN module using the IM data source scenario. We will show how we extend the existing export template *CR\_CatalogueRequest Item* so the data of the new module will be part of the message *Catalogue Request Item* that is sent to the pool.

You can find general information about how to work with the Product 360 Export in the "Product 360 User Manual" chapter "Export". Furthermore also consider the according chapters regarding export in "[Data validation](#) (see page 155)" and "[Testing](#) (see page 164)" of this documentation.

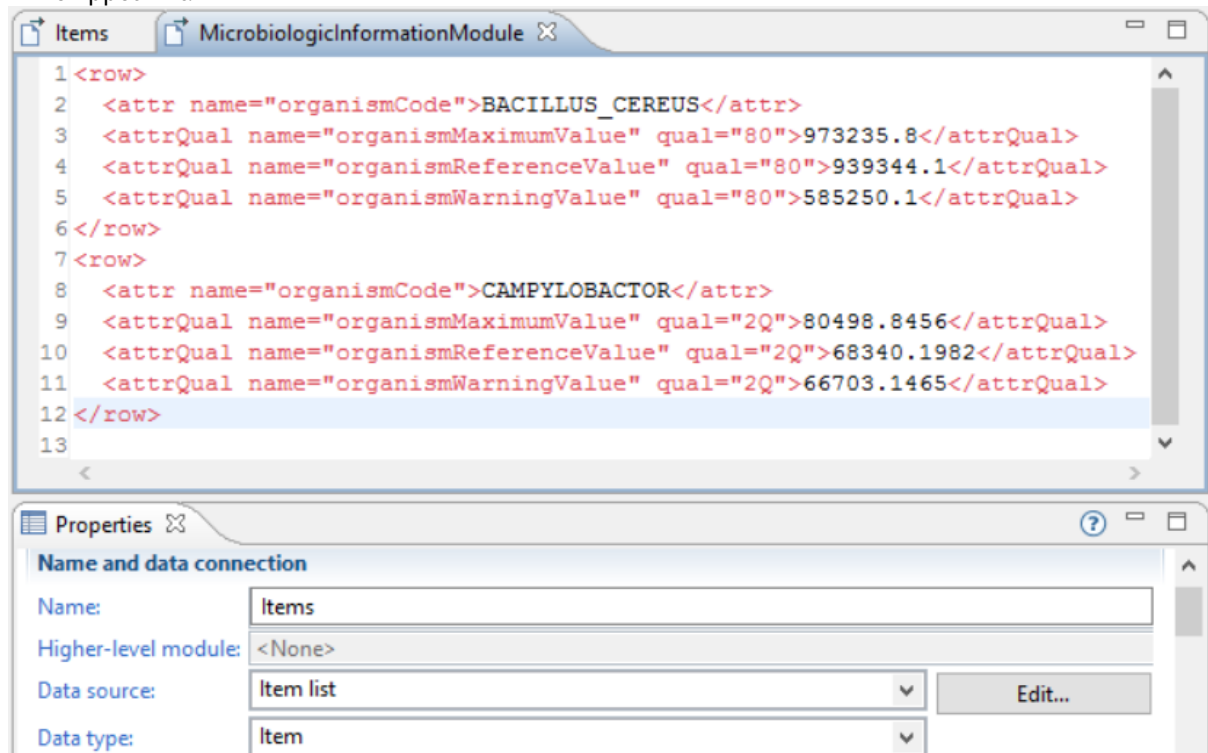
Example

We assume that we've added the GDSN module *MicrobiologicInformationModule* to the Product 360 data model and want to send those product information to the pool by using the IM connection.

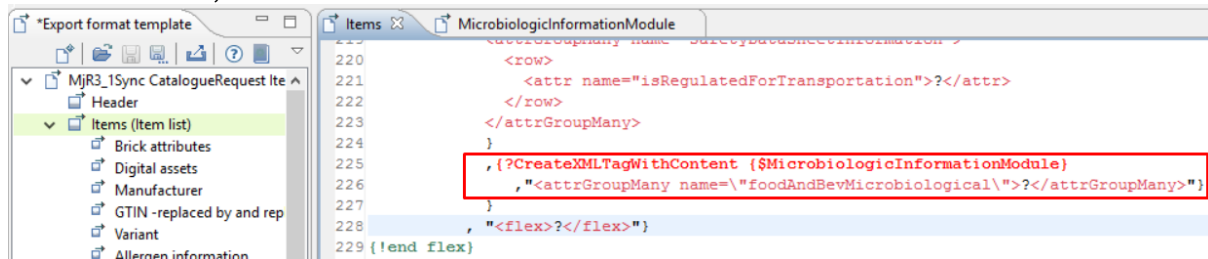
1. Get a XML snippet of the *MicrobiologicInformationModule* in the *Catalogue Request Item* message and ensure that it contains valid data. Usually you find GDSN modules mapped to "flex" attributes when using IM.

```
<attrGroupMany name="psychotropicSubstance">
<attrGroupMany name="FoodAndBevMicrobiological">
  <row>
    <attr name="organismCode">BACILLUS_CEREUS</attr>
    <attrQual name="organismMaximumValue" qual="80">973235.8</attrQual>
    <attrQual name="organismReferenceValue" qual="80">939344.1</attrQual>
    <attrQual name="organismWarningValue" qual="80">585250.1</attrQual>
  </row>
  <row>
    <attr name="organismCode">CAMPYLOBACTOR</attr>
    <attrQual name="organismMaximumValue" qual="2Q">80498.8456</attrQual>
    <attrQual name="organismReferenceValue" qual="2Q">68340.1982</attrQual>
    <attrQual name="organismWarningValue" qual="2Q">66703.1465</attrQual>
  </row>
</attrGroupMany>
<attrGroupMany name="physioChemicalProperties">
```

2. Create a sub-module for the *MicrobiologicInformationModule* in the export format template and copy the XML snippet in it.



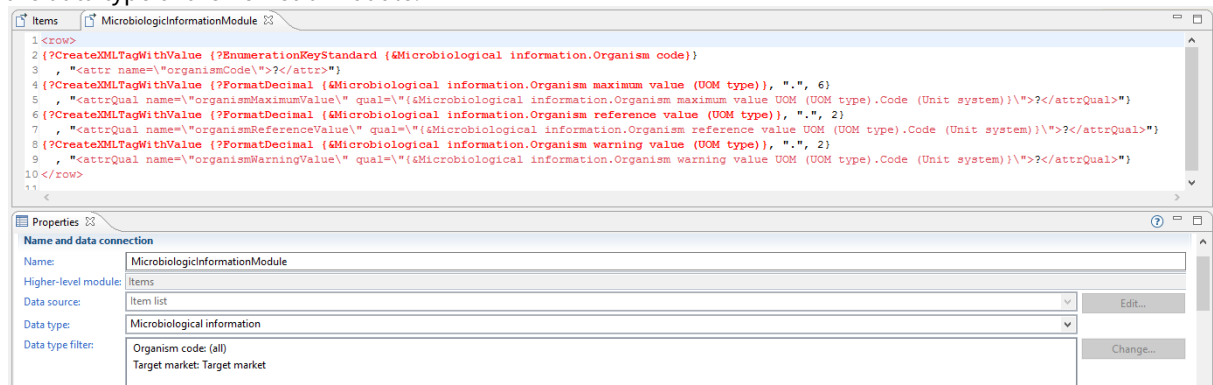
3. Call the sub-module in the *Items* module of the export format template. Please note that there is no ordering of "flex"-Attributes, so we can add it at the end.



4. Execute an export against the data pool with a single item and check if the product information for the GDSN module really was in the generated file and if the response from the data pool was without errors.

Publication status (3)							
DS_666540418579335							
	Informa...	Trade partner	Target...	Message type	Context	Response type	Executed operation
1	Inform...	MjR3 Pool	USA	Item response	<no context>	Acknowledgement	Modify
2	Inform...	MjR3 Pool	USA	Item transferred	<no context>		Modify
3	Inform...	MjR3 Pool	USA	Item registry response	<no context>	Acknowledgement	Add

5. Exchange the static values with the field values and test with a single item against the data pool. Now adjust the data type of the new sub-module.



6. Test the template against the data pool with several items as explained in chapter "[Testing \(see page 164\)](#)".

## 1.7.9 Enhance Import Mappings

All item related messages we receive from one of the supported GDSN pools are processed by the P360 import. If you enhance the data model the corresponding import mappings have to be adjusted. This applies mainly to data recipient scenarios.

### 1.7.9.1 Adjust an import mapping

We will show how to map the data fields of a new GDSN module using an example from the Item Management data recipient scenario.

You can find general information about how to work with the Product 360 Import in the "Product 360 User Manual" chapter "Data import". In addition to that, please consider the according chapters "[Data validation \(see page 155\)](#)" and "[Testing \(see page 164\)](#)" of this documentation.

#### Example

We will assume that we will have to add the mapping of the new GDSN module *MicrobiologicInformationModule* of the IM data pool.

1. Pick up a CIN file in the hotfolder of Product 360 which contains product information for the *MicrobiologicInformationModule*.

The product information could look like this:

```
<attrGroupMany name="psychotropicSubstance">
<attrGroupMany name="foodAndBevMicrobiological">
  <row>
    <attr name="organismCode">BACILLUS_CEREUS</attr>
    <attrQual name="organismMaximumValue" qual="80">973235.8</attrQual>
    <attrQual name="organismReferenceValue" qual="80">939344.1</attrQual>
    <attrQual name="organismWarningValue" qual="80">585250.1</attrQual>
  </row>
  <row>
    <attr name="organismCode">CAMPYLOBACTOR</attr>
    <attrQual name="organismMaximumValue" qual="2Q">80498.8456</attrQual>
    <attrQual name="organismReferenceValue" qual="2Q">68340.1982</attrQual>
    <attrQual name="organismWarningValue" qual="2Q">66703.1465</attrQual>
  </row>
</attrGroupMany>
<attrGroupMany name="physioChemicalProperties">
```

2. Create a new import project with the XML file of step 1 in the *Import* perspective of Product 360. Select the corresponding stored import mapping to complete the import project.
3. In the *Data Source* view navigate to the "foodAndBevMicrobiological" flex attributes. The XML path has been looked up in the chapter "Analyze requirements (see page 107)".

Structure	Selection	Content
flex		
attr (1)	1/193	true
attrGroup (1)	1/2	
attrGroupMany (2)	33/85	
name (9)		foodAndBevMicrobiological
row (2)	2/2	
attr (4)		CAMPYLOBACTOR
name (10)		organismCode
attrGroupMany (3)		
attrMany (6)		
attrQual (6)	1/3	80498.8456
name (42)		organismMaximumValue
qual (12)		2Q
attrQualMany (7)		
attrGroup (5)		
attrQual (10)	1/24	7189795167338534963346.1
attrQualMany (13)	1/45	
attrMany (13)	1/47	
cancelDate		

4. Map the fields accordingly to the Product 360 data model.

Structure	Assignment	Content
flex		
attr (1)		true (Import)
name (2)		USA
attrGroup (1)		
attrGroupMany (2)		
name (9)		foodAndBevMicrobiological
row (2)		
attr (4)		BACILLUS_CEREUS
name (10)		organismCode
attrGroupMany (3)		
attrMany (6)		
attrQual (6)	3/3	585250.1
name (42)		organismWarningValue
qual (12)		80
attrQualMany (7)		
attrGroup (5)		
attrQual (10)	1/24	7189795167338534963346.1
attrQualMany (13)	1/45	
attrMany (13)	1/47	

5. Execute the import and check if the product information is available in Product 360 as expected.



6. Execute the complete workflow of the GDSN Accelerator for receiving a CIN.

## 1.7.10 Testing

There are several ways to ensure your changes are correct. In this chapter we will describe how your test data has to look like to test the correct behavior in Product 360. Those can be used to test different modules of Product 360, like import, maintain functionality, data quality checks and exporting the data.

### 1.7.10.1 Test data

The test data should cover various data constellations. Therefore you always have to create the data with following specifications:

- Always test with several datasets of each entity of your created/adjusted data model
- Always test limits of your created/adjusted data model
- Always test with data which are valid and with data which are invalid (also known as good and bad test data). Check if the invalid test data lead to the expected severity (error/warning/info).
- Always create some "complete" items, that means items containing all data you want to send to the GDSN pool.

You usually can reuse the test data for testing all modules of Product 360 except invalid test data.



If you enhance your test data with a remarks field you can note your expected result in it.

#### Test data model

In the chapter [Data model \(see page 121\)](#) we introduced how to add or adjust GDSN modules and their fields in Product 360 which includes a [Data model check list \(see page 121\)](#). This check list ensures a correct defined data model. It's also checking if the changes in the Product 360 data model are visible by the generic functionality of Product 360. For the mentioned check list you should create test data for exhaustive testing.

For example you can create an Excel file as import file, like shown below for the fields `Packaging type`, `Packaging feature code`, `Usable product volume` and `Usable product volume UOM` of the sub entity `ArticlePackaging`, which is representing the GDSN Module

`PackagingInformationModule`:



Item no.	Target market (LK) (Enumeration)	Packaging type (LK) (Enumeration)	Packaging refuse obligation name (String)	UOM type (LK) (Enumeration)	Usable product volume (Decimal value)	Usable product volume UOM (Enumeration)	Remarks
TestPackInfoMod_1	US	AA	obligation name	IMPERIAL	1001.101	4G	Should pass
TestPackInfoMod_1	US	AA	obligation name	IMPERIAL	123456.12345		Should pass, but data quality check will fail because of missing UOM
TestPackInfoMod_1	US	CNG	obligation name	IMPERIAL	9999999999.999999	LTR	Should pass
TestPackInfoMod_1	US	CNG		METRIC	1253.6264	LTR	Should pass
TestPackInfoMod_1	ES	PUG	<String of 200 characters>				Should pass
TestPackInfoMod_2	US	AA		IMPERIAL	0		Should pass, but data quality check will fail because of missing UOM
TestPackInfoMod_3	BB	ABC	obligation name				Should fail due to not existent Packaging type

Item no.	Target market (LK) (Enumeration)	Packaging type (LK) (Enumeration)	Packaging refuse obligation name (String)	UOM type (LK) (Enumeration)	Usable product volume (Decimal value)	Usable product volume UOM (Enumeration)	Remarks
TestPackInfoMod_3	BB	STR	obligation name	METRIC	1000000000.000000	LTR	Should fail due to a range failure at Usable product volume
TestPackInfoMod_3	BB	STR	<String of 201 characters>				Should fail due to a range failure at Packaging obligation name
TestPackInfoMod_3	BB	STR	obligation name	TEST	234.3	4G	Should fail due to not existent UOM type

(Note: this table does not contain all scenarios to test)



If you write the max value "9,999,999,999.999999" for decimal values, Product 360 is actually transforming it to "9,999,999,999.999998".

#### Test import

Create an import mapping for your excel file and execute the import. Ensure that all expected errors are printed in the import log. Please check the created data after for correctness.

#### Test export

Create an export to check if you can export the data as expected. This also ensures that all fields are visible with the needed qualification.



This test is usually not made with the GDSN XML schemata.

## Test Service API

If you are planning to use the Service API for your fields, ensure that you have tested it.

### 1.7.10.2 Test data quality (Data source)

This chapter is only needed in case you created data quality rules or data quality configurations.

#### Testing created data quality rules

There are many ways to test your created data quality rules. For example you can test the created data quality rules within the Informatica Developer. The necessary steps are explained by using the "IfNotEmptyConditionNotEmpty" rule:

1. Create an Excel file which contains all input ports and an expected behavior as description.

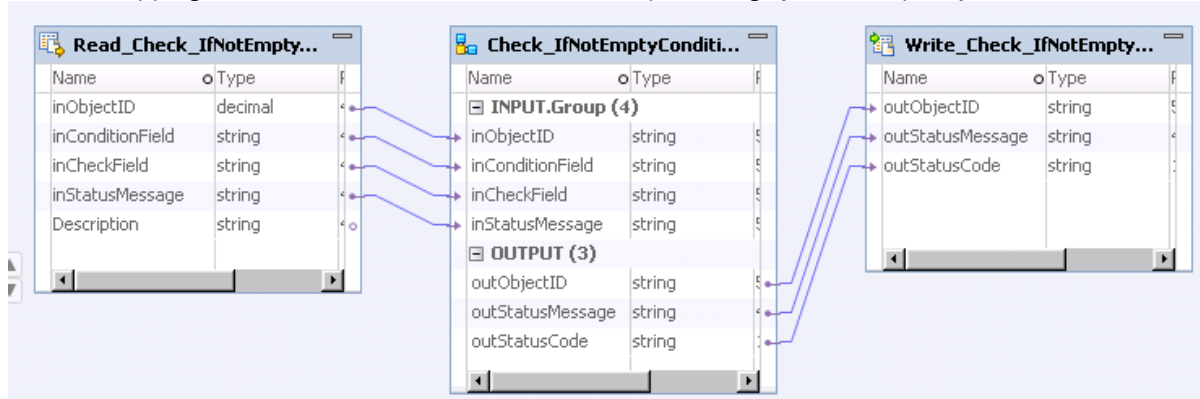
inObjectID	inConditionField	inCheckField	inStatusMessage	Description
10001			GDSN1	both inputs empty, ok
10002		somevalue	GDSN1	first input empty, ok
10003	somevalue		GDSN1	first input not empty, but second one, error
10004	somevalue	another	GDSN1	both inputs not empty, ok
10005	somevalue			first input not empty, but second one, with default error message
10101	somevalue		GDSN1	aggregated, not ok
10101	somevalue	another	GDSN1	

(Note: not all possible test scenarios are covered here)

2. Create an Excel file which contains the expected output after the data quality rule was running.

outObjectID	outStatusMessage	outStatusCode
10001	No Error	1
10002	No Error	1
10003	No Error	1
10004	GTIN is required but not provided	0
10005	inCheckField must be empty because inConditionField is not empty	0
10101	GTIN is required but not provided	0

3. Create a mapping to send the Excel file data created at step 1 through your data quality rule.



4. Compare the expected result from step 2 with the actual result of step 3.

#### Testing created data quality configuration

Depending on your data quality check, you can reuse the data you already created for the data model test. But you might need to extend those with further data constellations.

A recommended way to test if your data quality configuration is working as expected is:

1. Think about all possible data constellations and create your data accordingly.
2. Execute the data quality check for your configuration on the data of step 1.
3. Evaluate if your expectation and the actual result is equal.

In case the data quality configuration result is differently as your assumption, please check following:

1. Are the correct fields mapped to your data quality configuration?
2. If existent, is the correct qualification of the fields used?
3. If existent, is the port "InObjectID" mapped?
4. Did you use the correct data type?
5. In case you are using a sub entity of item as data type, then there must be one data entry for this sub entity to execute the rule.  
If you want to ensure that a value is in a specific field of a sub entity use the item as data type instead of the sub entity and qualify the field accordingly. You could also implement a data quality rule which is checking the field via service API.
6. Is the used data quality rule the correct rule for your validation? Please check the description of the rule again.

### 1.7.10.3 Test against certified GDSN pool (Data source)

If you have done all the testing and you ensured that all data is correct in Product 360 you need to enhance your export format template for GDSN accordingly. In chapter [Data validation](#) (see page 155) is already described how to manage the correct formatting of each field in the export.

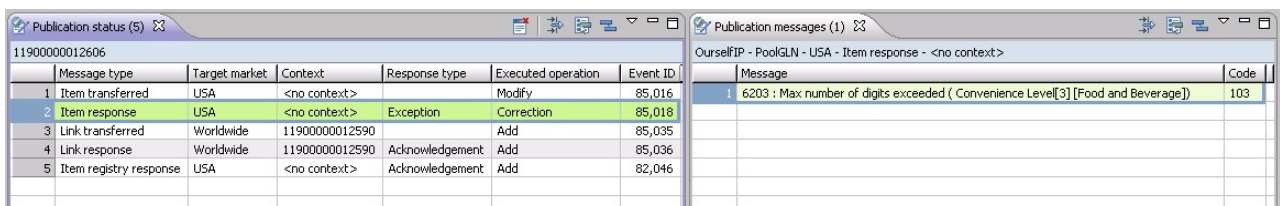
You can reuse your test data created for the data model testing but in addition, you have to ensure that all mandatory fields of GDSN are maintained. Otherwise the XSD validation will fail during export.

When sending data to the certified pool, ensure that you have the data to export validated. The exported data should get accepted by your data pool. In case of using 1WS data pool, you can verify the correctness of the data in the XML file by opening the "Publication status" view in Product 360. You will see an entry with "Item response" as message type and the according response type. If the response type is "Acknowledgement", then your changes seem to work. Please check the exported file to ensure that data for changes are exported.



	Message type	Information provider	Created on	Trade partner	Target market	Context	Severity	Response type	Executed operation	Event ID
1	Item registry response	OurselFIP	8/21/2015 4:54 AM	PoolGLN	USA	<no context>		Acknowledgement	Add	81,072
2	Link response	OurselFIP	8/11/2015 12:50...	PoolGLN	Worldwide	1190000012514		Acknowledgement	Add	82,006
3	Link transferred	OurselFIP	8/11/2015 12:49...	PoolGLN	Worldwide	1190000012514			Add	82,005
4	Item response	OurselFIP	8/10/2015 6:36 AM	PoolGLN	USA	<no context>		Acknowledgement	Modify	81,075
5	Item transferred	OurselFIP	8/10/2015 6:35 AM	PoolGLN	USA	<no context>			Modify	81,074

In case of an exception as response type something is wrong at your data. Please evaluate what the exact error message is saying. Sometimes it is already helpful to just investigate the exported XML file for "strange" things.



	Message type	Target market	Context	Response type	Executed operation	Event ID
1	Item transferred	USA	<no context>		Modify	85,016
2	Item response	USA	<no context>	Exception	Correction	85,018
3	Link transferred	Worldwide	1190000012590		Add	85,035
4	Link response	Worldwide	1190000012590	Acknowledgement	Add	85,036
5	Item registry response	USA	<no context>	Acknowledgement	Add	82,046

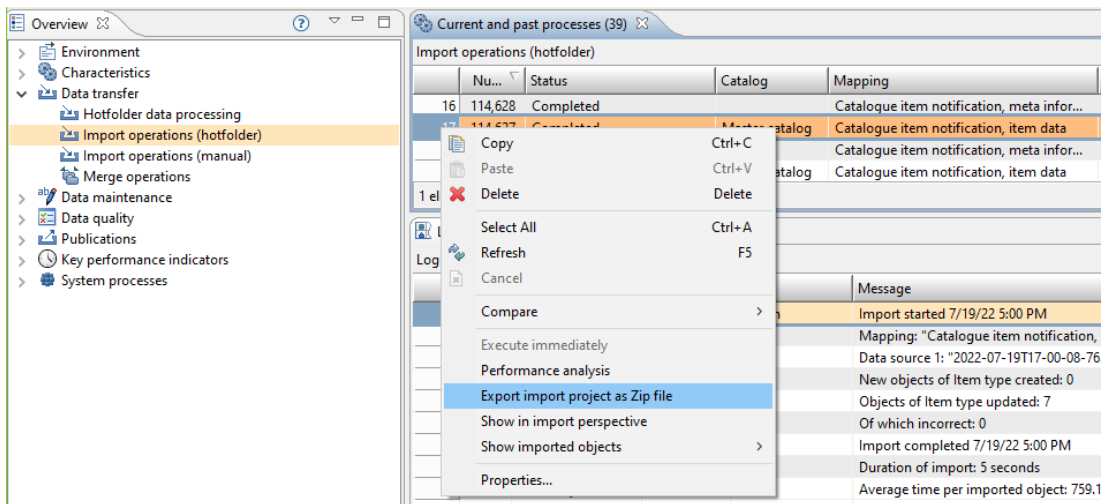
	Message	Code
1	6203 : Max number of digits exceeded ( Convenience Level[3] [Food and Beverage])	103

Please always publish your data to the 1WS data pool. This is needed since some validations are only running during the publish process.

#### 1.7.10.4 Test against certified GDSN pool (Data recipient)

In order to test the adjusted import mappings containing additional fields, we recommend to reimport a Catalogue Item Notification (CIN) message.

In the process overview in node "Import operations (hotfolder)", find an entry using a mapping called "Catalogue item notification, item data". Download the data source by using context menu entry "Export import project as Zip file".



Open the downloaded data file and check that it contains values for the fields you added to your data model.

Create a new import project using the new enhanced mapping and the downloaded data file and start the import.

Check if import has been successful in process overview and in the item UI check for the imported values.

## 1.8 Migration Guides

### 1.8.1 GDSN Migration Guide from B2B to OpenAS2

This page describes the migration of the Standard GDSN Data Source scenario.

#### 1.8.1.1 Overview

In contrast to B2B, with OpenAS2 all the incoming communication is handled internal to Product 360 using the import. Therefore a hotfolder configuration needs to be setup. Instructions can be found in "Product 360 Configuration" subchapter according to your scenario in [GDSN Accelerator Setup with OpenAS2](#) (see page 34).

The outgoing communication is still done via the export, some adjustments are necessary which are described below.

### 1.8.1.2 Export template migration

#### All export templates

##### File split

In older versions, data transformations in B2B Data Exchange split the files in packages of a certain number of documents the pool can handle. Using OpenAS2 this is now done by a post export step.

Please see [Appendix: Export templates \(see page 194\)](#) on how to customize it.

##### Copy to hotfolder

The exported files will now be sent to the pool by OpenAS2. This means the export template has to be adjusted to copy the exported files to the OpenAS2 hotfolder instead of the B2B DX ones.

Please see [GDSN Accelerator operation/Export Templates \(see page 89\)](#) on how to change the used hotfolder.

#### CIN\_CatalogItemNotification

##### Hierarchies for DSE

In older versions, B2B Data Exchange was used to create hierarchies of items. Using OpenAS2 this is now done by a post export step.

Please see [Product 360 Configuration \(Data Source, Standard GDSN\) \(see page 47\)](#) for more information.

##### Changed content

Changes in CIN\_CatalogItemNotification were needed to comply with the standard XSD files.

##### Module: Header

The complete content of the Header module of the provided export template should be used.

##### Module: Items

Lines 1 to 2 were replaced by new lines 1 to 27:

Changes in Items module (1)	
1	<transaction>
2	<transactionIdentification>

```

3      <entityIdentification>tran_{?ValueGet "globalIdentifier"}_{?
DatasetCounter}</entityIdentification>
4      <contentOwner>
5          <gln>{%Information provider GLN}</gln>
6      </contentOwner>
7  </transactionIdentification>
8  <documentCommand>
9      <documentCommandHeader type="{%Operation (ADD, CHANGE_BY_REFRESH or
CORRECT)}">
10         <documentCommandIdentification>
11             <entityIdentification>cmd_{?ValueGet "globalIdentifier"}_{?
DatasetCounter}</entityIdentification>
12             <contentOwner>
13                 <gln>{%Information provider GLN}</gln>
14             </contentOwner>
15         </documentCommandIdentification>
16     </documentCommandHeader>
17     <catalogue_item_notification:catalogueItemNotification>
18         <creationDateTime>{?FormatDate {?GetDate}, "yyyy-MM-dd"}T{?
FormatTime {?GetTime}, "HH:mm:ss"}</creationDateTime>
19         <documentStatusCode>ORIGINAL</documentStatusCode>
20         <catalogueItemNotificationIdentification>
21             <entityIdentification>cin_{?ValueGet "globalIdentifier"}_{?
DatasetCounter}</entityIdentification>
22             <contentOwner>
23                 <gln>{%Information provider GLN}</gln>
24             </contentOwner>
25         </catalogueItemNotificationIdentification>
26         <isReload>true</isReload>
27     <catalogueItem>
28         <catalogueItemState>
29             <catalogueItemStateCode>REGISTERED</catalogueItemStateCode>

```

Lines 254 to 259 (last 6 lines) were replaced by new lines 279 to 282:

#### Changes in Items (2)

```

1          </tradeItemSynchronisationDates>
2      </tradeItem>
3  </catalogueItem>
4  </catalogue_item_notification:catalogueItemNotification>
5  </documentCommand>
6  </transaction>

```

Module: Footer



The complete content of the Footer module of the provided export template should be used.

#### CIP\_CatalogItemPublication

In addition to the changes for the new export post step to split the exported file which is described above there are changes to create unique identification values.

Module: Header

Line 3 was added, line 15 was changed

It is recommended to use the complete content of this module

#### Module: Header

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <catalogue_item_publication:catalogueItemPublicationMessage
   xmlns:catalogue_item_publication="urn:gs1:gdsn:catalogue_item_publication:
   xsd:3" xmlns:sh="http://www.unece.org/cefact/namespaces/
   StandardBusinessDocumentHeader" xmlns:xsi="http://www.w3.org/2001/
   XMLSchema-instance" xsi:schemaLocation="urn:gs1:gdsn:catalogue_item_public
   ation:xsd:3 http://www.gdsregistry.org/3.1/schemas/gs1/gdsn/
   CatalogueItemPublication.xsd">
3  {?ValueSet "globalIdentifier", {?FormatDate {?GetDate}, "yyyy-MM-dd"}_{?
   FormatTime {?GetTime}, "HH:mm:ss.SSS"}}
4      <sh:StandardBusinessDocumentHeader>
5          <sh:HeaderVersion>1.0</sh:HeaderVersion>
6          <sh:Sender>
7              <sh:Identifier Authority="GS1">{%Information provider GLN}</sh:Ident
   ifier>
8          </sh:Sender>
9          <sh:Receiver>
10             <sh:Identifier Authority="GS1">{%Recipient GLN (data pool)}</sh:Iden
   tifier>
11         </sh:Receiver>
12         <sh:DocumentIdentification>
13             <sh:Standard>GS1</sh:Standard>
14             <sh:TypeVersion>3.1</sh:TypeVersion>
15             <sh:InstanceIdentifier>CIP_{?ValueGet "globalIdentifier"}</sh:Instan
   ceIdentifier>
16             <sh:Type>catalogueItemPublication</sh:Type>
17             <sh:MultipleType>false</sh:MultipleType>
18             <sh:CreationDateAndTime>{?FormatDate {?GetDate}, "yyyy-MM-dd"}T{?
   FormatTime {?GetTime}, "HH:mm:ss"}</sh:CreationDateAndTime>
19         </sh:DocumentIdentification>
20     </sh:StandardBusinessDocumentHeader>

```

Module: Publication status DSE

Line 1 was added, lines 4, 12 and 23 were changed

Module: Publication status DSE	
1	{?NumberIncrement "counter"}
2	<transaction>
3	<transactionIdentification>
4	<entityIdentification>cip_tran_{?ValueGet "globalIdentifier"}_{?
	NumberGet "counter"></entityIdentification>
5	<contentOwner>
6	<gln>{&GDSN 1WS DSE Publication.Information provider.GLN}</gln>
7	</contentOwner>
8	</transactionIdentification>
9	<documentCommand>
10	<documentCommandHeader type="{?EnumerationKey {&GDSN 1WS DSE
	Publication.Publication operation}}">
11	<documentCommandIdentification>
12	<entityIdentification>cip_cmd_{?ValueGet "globalIdentifier"}_{?
	NumberGet "counter"></entityIdentification>
13	<contentOwner>
14	<gln>{&GDSN 1WS DSE Publication.Information provider.GLN}</gln>
15	</contentOwner>
16	</documentCommandIdentification>
17	</documentCommandHeader>
18	<catalogue_item_publication:catalogueItemPublication>
19	<creationDateTime>{?FormatDate {?GetDate}, "yyyy-MM-dd"}T{?
	FormatTime {?GetTime}, "HH:mm:ss"></creationDateTime>
20	<documentStatusCode>ORIGINAL</documentStatusCode>
21	<documentStructureVersion>3.1</documentStructureVersion>
22	<catalogueItemPublicationIdentification>
23	<entityIdentification>cip_{?ValueGet "globalIdentifier"}_{?
	NumberGet "counter"></entityIdentification>
24	<contentOwner>
25	<gln>{&GDSN 1WS DSE Publication.Information provider.GLN}</gln>

CIPHW\_CatalogItemPublicationHierarchyWithdrawal

In addition to the changes for the new export post step to split the exported file which is described above there are changes to create unique identification values.

Module: Header

Line 3 was added, line 15 was changed

It is recommended to use the complete content of this module

**Module: Header**

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <catalogue_item_hierarchical_withdrawal:catalogueItemHierarchicalWithdrawalMessage xmlns:catalogue_item_hierarchical_withdrawal="urn:gs1:gdsn:catalogue_item_hierarchical_withdrawal:xsd:3" xmlns:sh="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:gs1:gdsn:catalogue_item_hierarchical_withdrawal:xsd:3 http://www.gdsregistry.org/3.1/schemas/gs1/gdsn/CatalogueItemHierarchicalWithdrawal.xsd">
3  {?ValueSet "globalIdentifier", {?FormatDate {?GetDate}, "yyyy-MM-dd"}_{?FormatTime {?GetTime}, "HH:mm:ss.SSS"}}
4      <sh:StandardBusinessDocumentHeader>
5          <sh:HeaderVersion>1.0</sh:HeaderVersion>
6          <sh:Sender>
7              <sh:Identifier Authority="GS1">{%Information provider GLN}</sh:Identifier>
8          </sh:Sender>
9          <sh:Receiver>
10             <sh:Identifier Authority="GS1">{%Recipient GLN (data pool)}</sh:Identifier>
11         </sh:Receiver>
12         <sh:DocumentIdentification>
13             <sh:Standard>GS1</sh:Standard>
14             <sh:TypeVersion>3.1</sh:TypeVersion>
15             <sh:InstanceIdentifier>CIPHW_{?ValueGet "globalIdentifier"}</sh:InstanceIdentifier>
16             <sh:Type>catalogueItemHierarchicalWithdrawal</sh:Type>
17             <sh:MultipleType>>false</sh:MultipleType>
18             <sh:CreationDateAndTime>{?FormatDate {?GetDate}, "yyyy-MM-dd"}T{?FormatTime {?GetTime}, "HH:mm:ss"}</sh:CreationDateAndTime>
19         </sh:DocumentIdentification>
20     </sh:StandardBusinessDocumentHeader>

```

Module: Publication withdrawal

Line 1 was added, lines 4, 12 and 23 were changed

**Module: Publication withdrawal**

```

1  {?NumberIncrement "counter"}
2  <transaction>
3      <transactionIdentification>
4          <entityIdentification>ciphw_tran_{?ValueGet "globalIdentifier"}_{?NumberGet "counter"}</entityIdentification>
5          <contentOwner>

```

```

6      <gln>{%Information provider GLN}</gln>
7      </contentOwner>
8    </transactionIdentification>
9    <documentCommand>
10      <documentCommandHeader type="DELETE">
11        <documentCommandIdentification>
12          <entityIdentification>ciphw_cmd_{?ValueGet "globalIdentifier"}
13_?{?NumberGet "counter"}</entityIdentification>
14          <contentOwner>
15            <gln>{%Information provider GLN}</gln>
16          </contentOwner>
17        </documentCommandIdentification>
18      </documentCommandHeader>
19      <catalogue_item_hierarchical_withdrawal:catalogueItemHierarchicalWith
20drawal>
21        <creationDateTime>{?FormatDate {?GetDate}, "yyyy-MM-dd"}T{?
22FormatTime {?GetTime}, "HH:mm:ss"}</creationDateTime>
23        <documentStatusCode>ORIGINAL</documentStatusCode>
24        <documentStructureVersion>3.1</documentStructureVersion>
25        <catalogueItemHierarchicalWithdrawalIdentification>
26          <entityIdentification>ciphw_{?ValueGet "globalIdentifier"}_{?
27NumberGet "counter"}</entityIdentification>
28        <contentOwner>

```

## 1.8.2 GDSN Migration Guide From 10.1 to 10.5 (OpenAS2)

### 1.8.2.1 Introduction

This section covers all the changes that we have made from Product 360 versions **10.1 and prior** to version **10.5**.

For OpenAS2 scenarios, we have changed several internal processes to be able to keep histories of some data, however this also caused some changes to export templates and import mappings.

Additionally we added support to be able to use OpenAS2 with Standard GDSN.

### 1.8.2.2 Changes affecting IM

#### IM export template changes

All IM export templates have been slightly changed. If no individual adjustments have been made to them, they can simply be loaded again as described [here \(see page 194\)](#). If individual adjustments have been made to these export templates, the following changes should be done manually.

The content of the tag <documentId> has been changed for all export templates:

File Name	Module	Line	Previously	Now
CR_CatalogueRequestItem.ext	Items	2	<documentId>{?GDSNDocumentId}</documentId>	<documentId>{?GDSNDocumentIdentifier}</documentId>
CR_CatalogueRequestLink ADD.ext	references	2	<documentId>{?GDSNParentDocumentId}_{?FillLeft {&GDSN 1WS IM Link ADD.Referenced item.GTIN},14,0}</documentId>	<documentId>{?GDSNParentDocumentIdentifier}_{?FillLeft {&GDSN 1WS IM Link ADD.Referenced item.GTIN},14,0}</documentId>
CR_CatalogueRequestLink DELETE.ext	references	2	<documentId>{?GDSNParentDocumentId}_{?FillLeft {&GDSN 1WS IM Link DELETE.Referenced item.GTIN},14,0}</documentId>	<documentId>{?GDSNParentDocumentIdentifier}_{?FillLeft {&GDSN 1WS IM Link DELETE.Referenced item.GTIN},14,0}</documentId>
CR_CatalogueRequestPublication HW.ext	Documents	2	<documentId>{?GDSNDocumentId}</documentId>	<documentId>{?GDSNDocumentIdentifier}</documentId>
CR_CatalogueRequestPublication.ext	GDSN 1WS IM Publication	16	<documentId>{&GDSN 1WS IM Publication.Publication document id}</documentId>	<documentId>{&GDSN 1WS IM Publication.Publication document identifier}</documentId>
CR_CatalogueRequestPublication_ByVariables.ext	Documents	2	<documentId>{?GDSNDocumentId}</documentId>	<documentId>{?GDSNDocumentIdentifier}</documentId>

IM import mapping changes

All IM import mappings have been adjusted, so they need to be imported again as described [here \(see page 194\)](#).

### 1.8.2.3 Changes affecting Standard GDSN

Directory structure changes

With 10.5 we are adding OpenAS2 GDSN support for Standard GDSN. As such we added some files that are used for its Post-Export Step.

These files can be found in a new directory: `\DataSource\DSE\OpenAS2\ExportTemplates\PostExport`

Export template changes

All default export templates for Standard GDSN have been adjusted to use a new Post-Export Step for file splitting. The corresponding file can be found in the directory mentioned above.

In addition, the export template *CIN\_CatalogItemNotification* has been adjusted to use a new Post-Export Step for creating a hierarchy. The corresponding file can also be found in the directory mentioned above.

## 1.8.3 GDSN Migration Guide for version 3.1.27

### 1.8.3.1 Product 360 data model changes

Deleted data fields

NA

Moved data fields

The following table lists fields that have been moved to another entity

GDSN attribute name	GDSN module	GD SN/F&B	P360 field name	P360 entity	Remarks
CannabisCBDTypeCode	Health related information	GD SN	From: ArticleHealthCare.CannabisCBDTypeCode  To: ArticleCannabisInformation.CannabisCBDTypeCode	From: Health related information  To: Cannabis information	

New modules/ data fields

The following table lists all new fields we have added.

GDSN module/attribute name	GDSN module	GD SN/F&B	P360 module/field identifier	P360 entity
Cannabis Information(Module)	Product information → Cannabis Information	GD SN	ArticleCannabisInformation	Cannabis information
CannabinoidTypeCode(Attribute)	Product information → Cannabis Information	GD SN	ArticleCannabisInformation.CannabinoidTypeCode	Cannabinoid type code


GDSN module/attribute name	GDSN module	GD SN/F&B	P360 module/field identifier	P360 entity
MicrobiologicalOrganismStrainCode (Attribute)	Microbiological information	GD SN	ArticleMicrobiologics.MicrobiologicalOrganismStrainCode	Microbiological Organism Strain Code

Checks before migration/upgrading



Please complete the checks listed below **before** updating.



Field(Entity)	Enumeration	Comment
Microbiological organism strain code	MicrobiologicalOrganisms trainCodeType	<p>ArticleMicrobiologics.LK.Res_LK_Text100_02 has been used for this new mandatory field.</p> <p>In DB the column ArticleDomain.Std_LK_Text100_02 will be updated with value 'NO_CODE', the default value for MicrobiologicalOrganismStrainCode.</p> <div style="border: 1px solid red; padding: 10px; margin: 10px 0;"> <p> Note: If there are already entries in ArticleDomain WHERE Std_LK_Text100_02 &lt;&gt;'DEFAULT' AND EntityID = 10010 the migration script will fail. This is intended as you need to adjust your data before migration. Update ArticleDomain entries to have Std_LK_Text100_02 = 'DEFAULT', where EntityID = 10010, and then run the migration scripts.</p> </div> <p>The following script snippet is about updating the Std_LK_Text100_02 column to the 'NO_CODE' of MicrobiologicalOrganismStrainCode</p> <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p><b>MSSQL Server</b></p> <pre> UPDATE ArticleDomain     SET Std_LK_Text100_02 = 'NO_CODE'     WHERE EntityID = 10010 AND Std_LK_Text100_02 = 'DEFAULT'; </pre> </div> <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p><b>Oracle</b></p> <pre> UPDATE "ArticleDomain"     SET "Std_LK_Text100_02" = 'NO_CODE'     WHERE         "EntityID" = 10010 AND         "Std_LK_Text100_02" = 'DEFAULT'; </pre> </div>

### 1.8.3.2 Changes in data source export templates

#### New XSD file set

##### **GDSN 3.1.27**

A new XSD file set has been provided for GDSN version 3.1.27. All old XSD files used for export post steps have to be deleted, then all new XSD files must be uploaded. Export templates do not have to be adapted due to the changed XSD files.

##### **Item Management 8.48**

The only changed export template for IM is `CR_CatalogueRequest Item.ext`

##### **DSE export template**

The only changed export template for DSE is `CIN_CatalogItemNotification.ext`

#### DSE export template changes

The only changed export template for DSE is `CIN_CatalogItemNotification.ext`. The following module has been changed in that template.

##### **MicrobiologicalOrganismStrainCode**

DSE has not implemented the changes for the new attribute: organismStrainCode in Microbiological information module

#### Product information

Export template module: *Product information*

Changes: Module deleted

#### Cannabis information

Export template module: *Cannabis information*

Changes: New module

Items	
1	< <b>cannabisInformation</b> >
2	{?CreateXMLTagWithValue {?EnumerationKey {&Cannabis Information.Cannabis CBD type}}, "< <b>cannabisCBDTypeCode</b> >?</ <b>cannabisCBDTypeCode</b> >"}
3	{?CreateXMLTagWithValue {?EnumerationKey {&Cannabis Information.Cannabinoid type code}}, "< <b>cannabinoidTypeCode</b> >?</ <b>cannabinoidTypeCode</b> >"}
4	</ <b>cannabisInformation</b> >

Items

Export template module: *Items*

Changes: lines 103-108 have been updated

Items	
1	
2	{\$Delivery purchasing information: Collect distribution details}
3	{?CreateXMLTagWithContent {\$Delivery purchasing information},
4	" < <b>delivery_purchasing_information:deliveryPurchasingInformationModule</b> xsi:schemaLocation="urn:gs1:gdsn:delivery_purchasing_information:xsd:3 http://www.gdsregistry.org/3.1/schemas/gsl/gdsn/DeliveryPurchasingInformationModule.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:delivery_purchasing_information="urn:gs1:gdsn:delivery_purchasing_information:xsd:3">
5	< <b>deliveryPurchasingInformation</b> >?</ <b>deliveryPurchasingInformation</b> >
6	</ <b>delivery_purchasing_information:deliveryPurchasingInformationModule</b> >"
7	}
8	
9	
10	{?CreateXMLTagWithContent {?Concat {\$Claim details}, < <b>tobaccoCannabisInformation</b> >{\$Cannabis information}</ <b>tobaccoCannabisInformation</b> >},
11	< <b>product_information:productInformationModule</b> xsi:schemaLocation="urn:gs1:gdsn:product_information:xsd:3 http://www.gdsregistry.org/3.1/schemas/gsl/gdsn/ProductInformationModule.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:product_information="urn:gs1:gdsn:product_information:xsd:3">
12	< <b>productInformationDetail</b> >?</ <b>productInformationDetail</b> >
13	</ <b>product_information:productInformationModule</b> >
14	}
15	
16	{?CreateXMLTagWithContent {\$Place of item activity},


```

17         "<place_of_item_activity:placeOfItemActivityModule
xsi:schemaLocation="urn:gs1:gdsn:place_of_item_activity:xsd:3 http://
www.gdsregistry.org/3.1/schemas/gsl/gdsn/PlaceOfItemActivityModule.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:place_of_item_activity="urn:gs1:gdsn:place_of_item_activity:xsd:3">
18             <placeOfProductActivity>?</placeOfProductActivity>
19             </place_of_item_activity:placeOfItemActivityModule>"
20         }
21     }

```

### IM export template changes

The only changed export template for Item Management is `CR_CatalogueRequest Item.ext`. The following module has been changed in that template.

-  Product Information module added from GDSN Major Release 3.1.23 was not implemented by IM, and has been added this year, in addition to the other modules from this year

#### Modules from GDSN Major Release 3.1.23 implemented in 2024

- Product Information (New)
- Product Information → Claim Details (New)

#### Modules from GDSN Major Release 3.1.27

- Microbiological Information (Modified)
- Product Information → Cannabis Information (New)

### Microbiological information

Export template module: *Microbiological information*

Changes: line 4 has been added

#### Items

```

1  {?ValueSetLocal "tmpOrganismCode", {?EnumerationKeyStandard
2  {&Microbiological information.Organism code}}}
3  <row>
4  {?CreateXMLTagWithValue {?ValueGetLocal "tmpOrganismCode"}, <attr name="or
5  ganismCode">?</attr>}
6  {?CreateXMLTagWithValue {?EnumerationKeyStandard {&Microbiological
7  information.Microbiological Organism Strain Code}}, <attr name="organismSt
8  rainCode">?</attr>}
9  {?CreateXMLTagWithContent {?ValueGetLocal {?ValueGetLocal
10 "tmpOrganismCode"}_maxValue}, "<attrQualMany
11 name=\"organismMaximumValue\">?</attrQualMany>"}

```

```

6      {?CreateXMLTagWithContent {?ValueGetLocal {?ValueGetLocal
      "tmpOrganismCode"}_refValue}, "<attrQualMany
      name=\"organismReferenceValue\">?</attrQualMany>"}
7      {?CreateXMLTagWithContent {?ValueGetLocal {?ValueGetLocal
      "tmpOrganismCode"}_warnValue}, "<attrQualMany
      name=\"organismWarningValue\">?</attrQualMany>"}
8
9      </row>
10
11     {!Reset variables}
12     {?ValueSetLocal "tmpOrganismCode", ""}
13     {?ValueSetLocal {?ValueGetLocal "tmpOrganismCode"}_maxValue, ""}
14     {?ValueSetLocal {?ValueGetLocal "tmpOrganismCode"}_refValue, ""}
15     {?ValueSetLocal {?ValueGetLocal "tmpOrganismCode"}_warnValue, ""}

```

#### Claim details

Export template module: *Claim details*

Changes: line 1-10 have been added

Items	
1	{?ValueSetLocal "tmpProductInfoClaimDetails", "claim"_{&Claim details.Target market}}
2	{?ValueSetLocal {?ValueGetLocal "tmpProductInfoClaimDetails"}, {?ValueGetLocal {?ValueGetLocal "tmpProductInfoClaimDetails"}}
3	
4	<row>
5	{?CreateXMLTagWithValue {?EnumerationKey {&Claim details.Claim type code}}, "<attr name=\"claimTypeCode\">?</attr>"}
6	{?CreateXMLTagWithValue {?EnumerationKey {&Claim details.Claim element code}}, "<attr name=\"claimElementCode\">?</attr>"}
7	{?CreateXMLTagWithValue {?EnumerationKey {&Claim details.Claim marked on package}}, "<attr name=\"claimMarkedOnPackage\">?</attr>"}</row>
8	
9	
10	}

#### Cannabis information

Export template module: *Cannabis information*

Changes: line 1-7 have been added

Items	
1	{?ValueSetLocal "tmpProductInfoTobCannabisInfo", "tobacco"_{&Cannabis Information.Target market}}

```

2      {?ValueSetLocal {?ValueGetLocal "tmpProductInfoTobCannabisInfo"}, {?
3      ValueGetLocal {?ValueGetLocal "tmpProductInfoTobCannabisInfo"}}
4      <row>
5      {?CreateXMLTagWithValue {?EnumerationKey {&Cannabis
6      Information.Cannabinoid type code}}, "<attr name=\"cannabinoidTypeCode\">?
7      </attr>"}
8      {?CreateXMLTagWithValue {?EnumerationKey {&Cannabis Information.Cannabis
9      CBD type}}, "<attr name=\"cannabisCBDTypeCode\">?</attr>"}
10     </row>
11   }

```

Product information

Export template module: *Product information*

Changes: lines 1-10 have been added

Items	
1	{?ValueSetLocal "tmpProductInfoTobCannabisInfo", "tobacco"_{&Product
2	Information.Target market}}
3	{?CreateXMLTagWithContent {?ValueGetLocal {?ValueGetLocal
4	"tmpProductInfoTobCannabisInfo"}},
5	"<attrGroupMany name=\"tobaccoCannabisInformation\"><row><attrGroupMany
6	name=\"cannabisInformation\"></attrGroupMany></row></attrGroupMany>"}>
7	{!Reset variable}{?ValueSetLocal {?ValueGetLocal
8	"tmpProductInfoTobCannabisInfo"}, ""}
9	
10	{?ValueSetLocal "tmpProductInfoClaimDetails", "claim"_{&Product
11	Information.Target market}}
12	{?CreateXMLTagWithContent {?ValueGetLocal {?ValueGetLocal
13	"tmpProductInfoClaimDetails"}},
14	"<attrGroupMany name=\"claimDetails\"></attrGroupMany>"}>
15	{!Reset variable}{?ValueSetLocal {?ValueGetLocal
16	"tmpProductInfoClaimDetails"}, ""}

Items

Export template module: *Items*

Changes: lines 144-145 have been added

Items	
1	{\$Collect packaging material}
2	{\$Collect Ingredient statement}
3	{\$Collect Microbiological information}
4	{\$Collect allergens}
5	{\$Collect nutrients}

6	{Collect product yield information}
7	{Collect additional diet certification information}
8	{Collect diet certifications}
9	{Collect diets}
10	{Collect certificates}
11	{Collect certification organisations}
12	{Collect trade item handling instructions}
13	{Collect trade item handling stacking}
14	{Collect nutritional program ingredients}
15	{Collect nutritional programs}
16	{Collect fish meat poultry content}
17	{Collect marketing messages}
18	{Cannabis information}
19	{Claim details}

Changes: line 203 has been added

Items	
1	, {CreateXMLTagWithContent {\$Certifications}, "<attrGroupMany name='certificationInformation'>?</attrGroupMany>"}
2	, {IfEmptyThenNotEnc "", {\$Trade item lifespan information}}
3	, {CreateXMLTagWithContent {\$Product information}, "<attrGroupMany name='productInformation'><row>?</row></attrGroupMany>"}
4	, {IfEmptyThenNotEnc "", {\$Trade item handling}}
5	, {CreateXMLTagWithContent {\$Physiochemical information}, "<attrGroupMany name='physioChemicalProperties'>?</attrGroupMany>"}
6	, {IfEmptyThenNotEnc "", {\$Health related information}}

### 1.8.3.3 Compatibility

The relevant changes of GDSN Major Release 3.1.27 will be supported from version 10.5.0.03 upwards. All older versions won't have any of the changes mentioned in this migration guide so we highly recommend updating to the latest version.

**As there are no structural changes in the XSD files compared to version 3.1.23 all export templates are compatible to version 3.1.27.**

## 1.9 GDSN Accelerator FAQ

### 1.9.1 General

#### 1.9.1.1 Q: Can I also send items to the GDSN pool from a supplier catalog?

**A:** This is theoretically possible but not recommended. The master catalog should contain the "golden" data which is send to the GDSN pool.

#### 1.9.1.2 Q: What does a response message like "GDSN Numeric Rule ID 1281: The format of "Ingredient Sequence" must be 'dd.dd.dd...'. Where 'd' must be a digit, always ending in a 'dd' and never having a value of '00'." mean?

**A:** This means the sent data is not valid regarding the validation which is done by the data pool. Usually the error message describes the deficiency in the data. You could add such a data check by a data quality rule configuration or a repository adjustment to ensure that you always send correct data.

### 1.9.2 Export

#### 1.9.2.1 Q: The export fails due to the error "... One of '{document}' is expected."

Step	Type	Export post-process...	Validate XML file(s)	File 'Types.xsd' loaded
20	Note	Export post-process...	Validate XML file(s)	File 'Types.xsd' loaded
21	Note	Export post-process...	Validate XML file(s)	File 'AttrTypes.xsd' loaded
22	Mandato...	Export post-process...	catalogueRequest	File "ISync CatalogueRequest Item.xml": Error in line 15, column 22: cvc-complex-type.2.4.b: The content of element 'catalogueRequest' is not complete. One of '{document}' is expected.
23	Summary	Post-export step	Validate XML file(s)	Export canceled due to XML validation error(s)

**A:** It can happen that there are no items in the generated XML file because no item passed the data quality checks or no item was passed to the export. If this case there will be no <document> tag generated in the XML which is leading to the above shown validation error. Despite to the fact that this is not very usable or obvious for an user, everything works correctly and the are no bad side effects. This behavior will be improved in a future Product 360 version.

#### 1.9.2.2 Q: The export fails due to XSD error "...The value " of attribute..."

Data type:	Export post-p	ID:	startAvailabilityDate	Position:	
Message:	File "HF4EBF1_CatalogueItemRequest-MOD.xml": Error in line 1294, column 54: cvc-attribute.3: The value " of attribute 'dataRecipientGLN' on element 'startAvailabilityDate' is not valid with respect to its type, 'glnType'.				

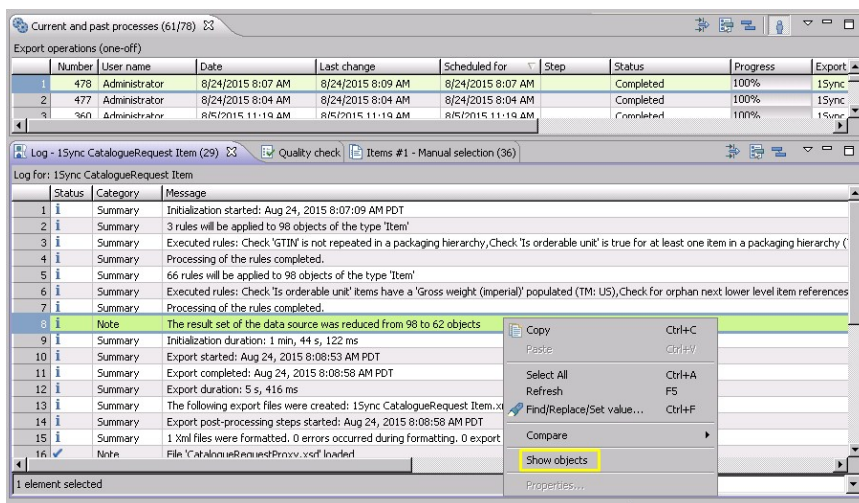
**A:** If a generated XML file doesn't fit to the according GDSN XSD schema, an XSD validation error will occur. The above shown message indicates that no value was given for the attribute 'dataRecipientGLN' which is



not valid regarding the 'glnType'. Therefore you need to open the "Customer" view and maintain the GLN for your customers.

### 1.9.2.3 Q: Not all of my items has been sent to the pool

**A:** Usually there is a data quality run before you send your data to the GDSN pool. If an item does not pass all data quality checks it will be filtered out and will not be sent to the GDSN pool. You can easily identify these items by navigating to the corresponding export log and select "Show objects" in the context menu of the corresponding log entry (see screenshot).



### Q: Why does my GTIN differ in the exported file?

**A:**

When your GTIN is less than 14 digits in Product360 while exporting, the GTIN will be automatically filled with leading 0's to match the GS1 criteria of 14 digit GTINs. **It is strongly recommended to have 14 digit GTINs while working with Product360 GSDN Accelerator.**

Otherwise there will be mapping issues between the CICs received from from the pool.

## Custom Implementations

### 1.9.3

#### 1.9.3.1 Q: How do we add missing units of measurement to the GDSN unit system?

**A:** Open a ticket at the Informatica support. Informatica will provide a database script which adds the needed units.

### 1.9.3.2 Q: Why does the deletion of a subentity of "Article" doesn't work via Service API when using a reserve logical key as qualification filter?

**A:** When deleting a subentity, the used `qualificationFilter` arguments must not only be unique in the entity path, it must be also unique of all subentities of the target entity. For example you want to delete all your `ArticleIngredient` objects which have a specific qualification, the statement could look like this:

```
http://localhost:1513/rest/V1.0/list/Article/ArticleIngredient/byCatalog?
catalog=Catalog_Test_RestAPI&qualificationFilter=res_LK_Text100_01("DEFAULT")
```

The problem is that several logical keys are using "res\_LK\_Text100\_01" as `Object Name`, to be exact the logical keys of entity `ArticleIngredient` and `ArticleIngredientComponent`. To prevent this define an `Alias` for your logical key. In case this is not solving your problem, please contact the Informatica support.

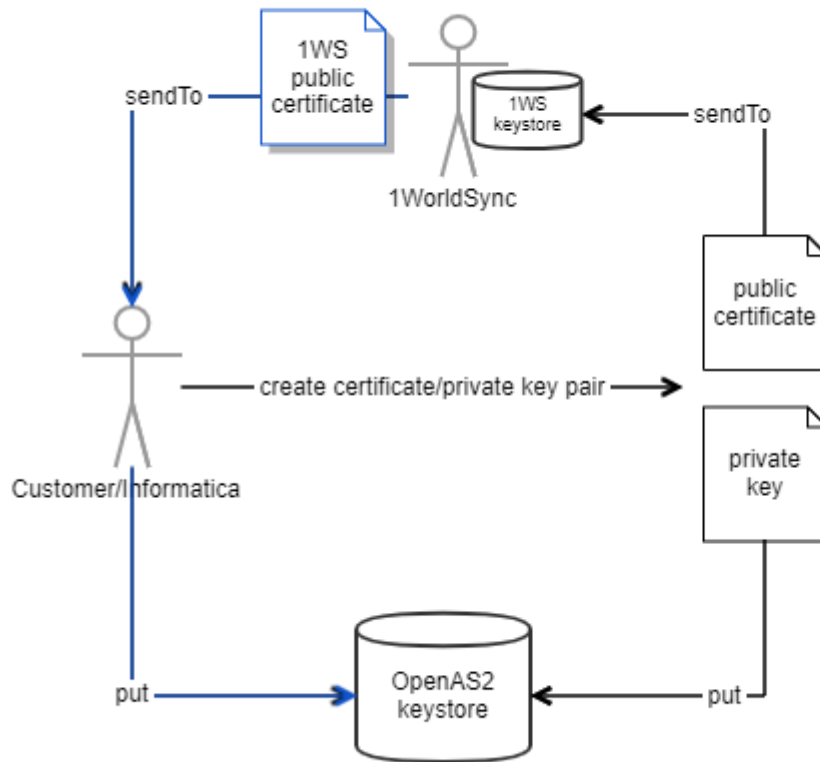
## 1.10 Appendix

### 1.10.1 Appendix A: Certificates (OpenAS2)

#### 1.10.1.1 Overview

To have a successfully and secured AS2 connection, certificates have to be exchanged between both trade partners. For on premise installations the customer has to do this, for hosted installation Informatica has to do this part. As the diagram below shows, the customer/Informatica has to create a public certificate / private key pair, send the public certificate to 1WorldSync or the respective data pool provider (usually via EMail) and put the private key to the key store of OpenAS2. Then the public certificate of 1WorldSync or the respective data pool provider should be retrieved (by EMail or download from the webpage) and also put to the OpenAS2 key store.

## Certification exchange



### 1.10.1.2 Create a private key / public certificate

OpenAS2 provides a command to create a private key / public certificate pair. The public certificate has to be sent to 1WorldSync or the respective data pool provider so they can encrypt their messages with this certificate. To create the pair go to the <OpenAS2>/config directory and execute (you can find the password in the config.xml file)

```

..\bin\gen_p12_key_par.bat <keyStore> <Customer>_<env>_to_1ws <sigAlg>
"CN=<CommonName>, O=<Organization>, OU=<OrganizationalUnit>, L=<Locality>, S=<State>,
C=<Country>"
  
```

or in a concrete example for Informatica

```

..\bin\gen_p12_key_par.bat as2_certs customer_dev_to_1ws sha256 "CN=informatica.com,
O=Informatica GmbH, OU=R&D, L=Stuttgart, S=BW, C=GER"
  
```

The result should look like:

```

PS C:\Informatica\AS2\testcertificate> ..\bin\gen_p12_key_par.bat as2_certs informatica_to_1ws sha256 "CN=informatica.com, O
=Informatica GmbH, OU=R&D, L=Stuttgart, S=BW, C=GER"
DNAM = "CN=informatica.com, O=Informatica GmbH, OU=R&D, L=Stuttgart, S=BW, C=GER"
Generate a certificate to a PKCS12 key store.
Generating certificate: using alias informatica_to_1ws to as2_certs.p12"
Enter password for keystore:testas2
Certificate stored in file <informatica_to_1ws.cer>

Generated files:
PKCS12 keystore: as2_certs.p12
Public Key File: informatica_to_1ws.cer

PS C:\Informatica\AS2\testcertificate> |

```

You now can send the public certificate (in this case informatica\_to\_1ws.cer) to 1WorldSync or the respective data pool provider and either use the generated key store or update an existing one.

### 1.10.1.3 How to import a new certificate

1. Open a console and navigate to the <OpenAS2>/config
2. Execute following command:

```

..\bin\import_public_cert.bat c:\Informatica\OpenAS2\config\<yourCertificate>
<keyStore> <alias> <action>

```

or in example:

```

..\bin\import_public_cert.bat c:
\Informatica\OpenAS2\config\as2_preprod_1worldsync_com_10Jan2020_der.cer c:
\Informatica\OpenAS2\config\as2_certs.p12 1ws replace

```

replace is optional and only needed if the certificate already exists.

3. The password for the keystore can be found in the config.xml of OpenAS2 at the XML element <certifications>.

### 1.10.1.4 How to list all certificates (check)

1. To list all certificates of your key store

```

keytool -list -keystore <keystore> -storepass <password> -storetype PKCS12

```

or in example:

```

keytool -list -keystore as2_certs.p12 -storepass testas2 -storetype PKCS12

```

2. Your keystore should look like this:

```
# keytool -list -keystore as2_certs.p12 -storepass testas2 -storetype PKCS12
Keystore type: PKCS12
Keystore provider: SunJSSE

Your keystore contains 4 entries

partnera, Sep 6, 2018, PrivateKeyEntry,
Certificate fingerprint (SHA1): 2D:4B:42:05:56:80:9B:5D:0E:63:4D:4A:23:3D:9A:39:C3:8D:51:21
mycompany, Sep 6, 2018, PrivateKeyEntry,
Certificate fingerprint (SHA1): 1E:16:65:9B:7A:F2:59:EA:B7:B7:4F:E5:EB:D3:CF:89:3A:0F:89:CA
infa_to_1ws, Mar 30, 2020, PrivateKeyEntry,
Certificate fingerprint (SHA1): C2:80:BD:CE:81:22:EE:FF:26:0B:41:FC:04:AB:94:A7:CB:50:3D:50
1ws, Mar 30, 2020, trustedCertEntry,
Certificate fingerprint (SHA1): B1:5C:18:25:05:08:BD:80:33:44:B3:04:9B:D5:58:3D:7F:7A:DD:91
```

### 1.10.1.5 How to delete a certificate

To delete a certificate of your key store

```
keytool -delete -keystore <keystore> -storepass <password> -alias <alias>
```

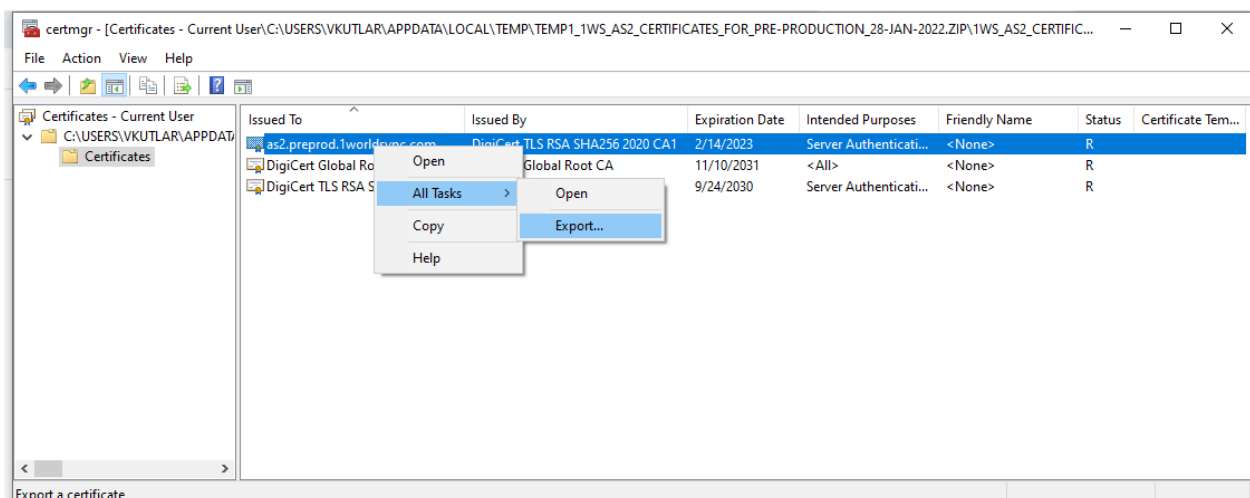
or in example:

```
keytool -delete -keystore as2_certs.p12 -storepass testas2 -alias 1ws
```

### 1.10.1.6 How to extract a certificate from a p7b file

When 1WorldSync or the respective data pool provider provide new certificates, you will not only get the required certificate itself but a p7b file. This file contains a collection of different certificate files. To extract and use the required certificate, you need to do the following steps:

On Windows, double click the p7b file, open the certificates path and right click the first file. Select All Tasks → Export...



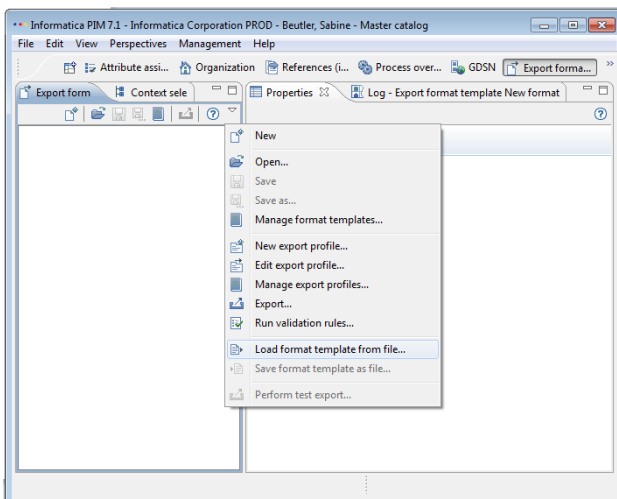
Now follow the steps of the export wizard and provide a valid file name. Preferably the same name as the original file name as the file extension will be replaced. So for example: as2\_preprod\_1worldsync\_com\_10Jan2020\_der. Finish the export and continue your required use case with the exported certificate.

## 1.10.2 Appendix B: Export templates

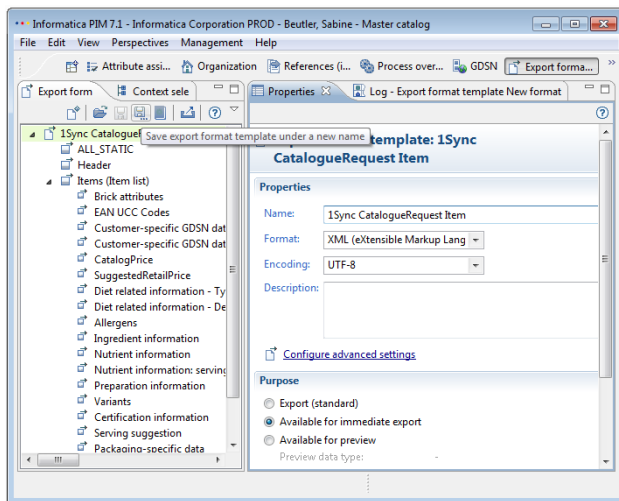
The GDSN accelerator package contains various export template files that you should load and then save within Product 360. This is necessary to be able to configure the export templates, to schedule repeating export jobs as well as to create one-click exports for immediate data transfer to the GDSN pool. Depending on the GDSN data pool the export templates are completely different. In the subchapters it is described which are needed for the corresponding Item Management or Standard GDSN data pool.

### 1.10.2.1 Load and save the export templates

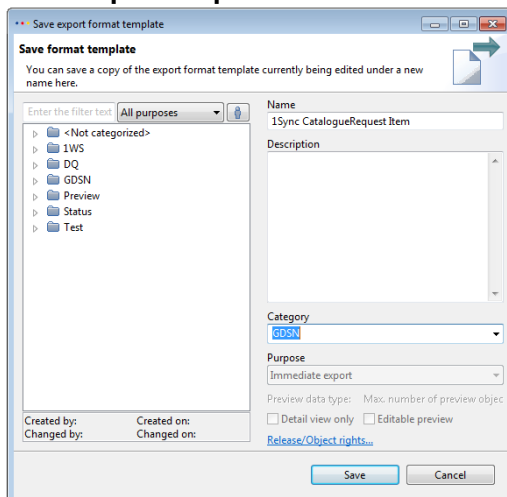
Open the "Export format templates" perspective, choose "Load format template from file...", select one export format template file, assign or create an appropriate category, for example "GDSN", and save the template into Product 360 and repeat those steps for all export format templates:



#### 6 Load export template from file



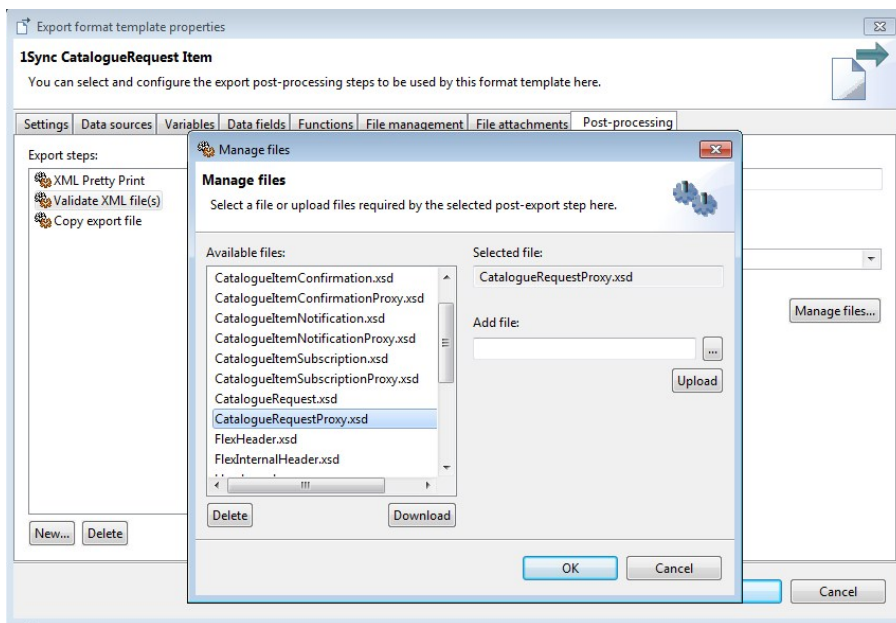
## 7 Save export template as..



## 8 Category for export template

### 1.10.2.2 Add XSD schema files

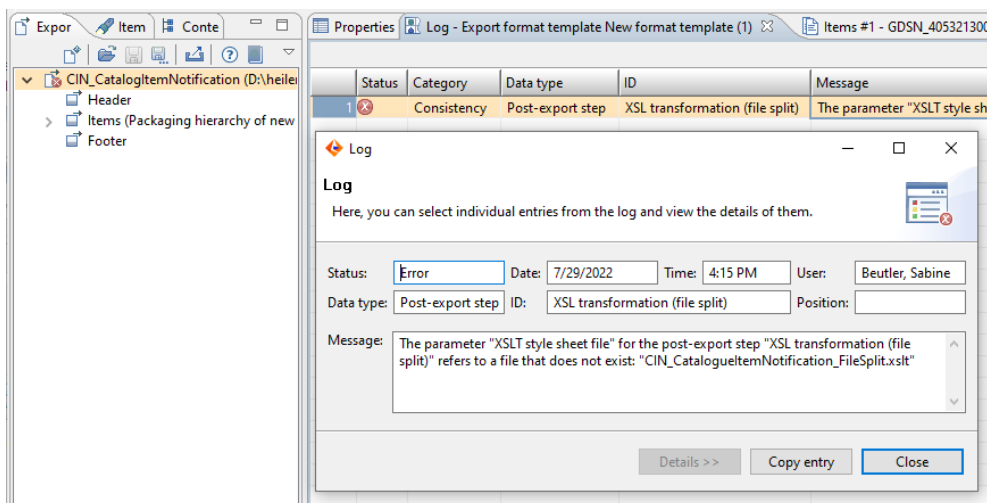
After saving the export templates, you need to make sure that the needed XSD files are already uploaded to the Product 360 system. Therefore go to the "Post-processing" tab in the property dialog of any export format template and select "Validate XML file(s)" export step. Press on "Manage files..." and examine if the delivered XSD files (e.g. CatalogueRequestProxy.xsd) are already in the "Available files" list. In case those are not uploaded yet, find the XSD files to upload in the GDSN accelerator package and upload them here. Again it is important to upload the XSD files which are appropriate for your GDSN data pool, locations can be found in the subchapters for Item Management and Standard GDSN.



The upload of the XSD files has only to be done once. They will be available for all export format templates.

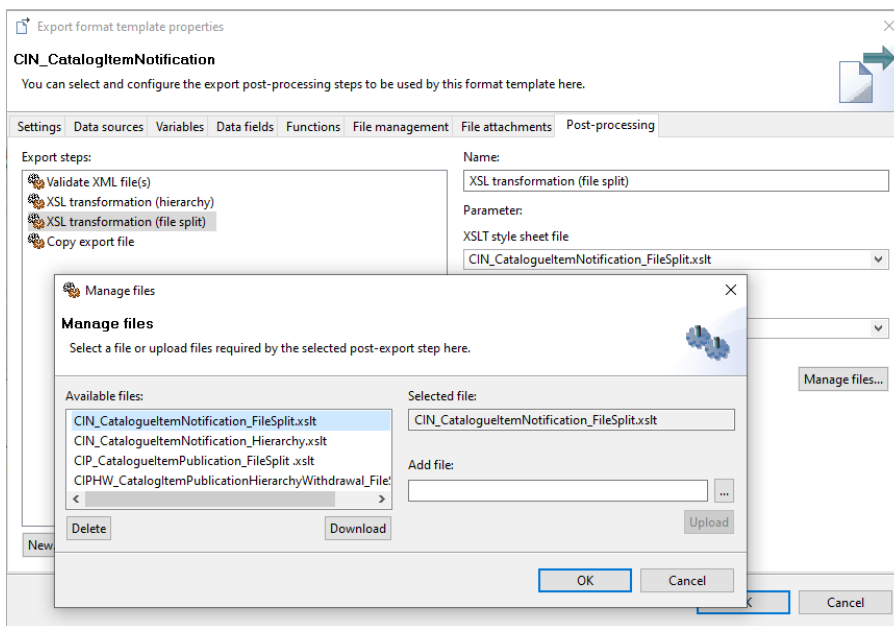
### 1.10.2.3 Add XSLT files

Some of the export templates contain another post-processing step to transform the exported xml file. The corresponding XSLT files have to be uploaded to the Product 360 system, too. If a XSLT file is missing, the corresponding export template cannot be used to export data and an error is shown.



For uploading the needed XSLT file, go to the "Post-processing" tab in the property dialog of the export template and select "XSL transformation" export step. The next steps are the same as for adding XSD schema files.



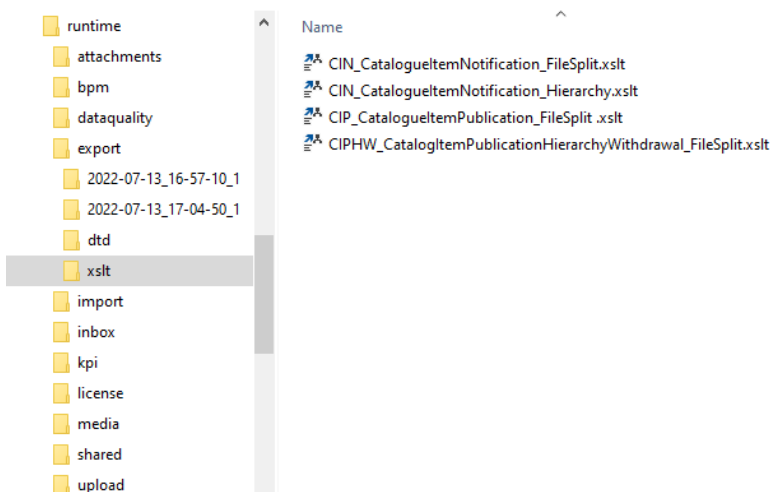


#### 1.10.2.4 Disabled file upload

For security reasons, the export file upload can be restricted. If this is the case in your system and you are not allowed to upload the required files, you have to contact an administrator to copy the files to the P360 server. XSD and XSLT export files are expected to be in a subdirectory of the export directory configured in the server.properties file.

These are the subdirectories used by post-processing export steps:

- Validate XML file(s): *dtd*
- XSL transformation: *xslt*



### 1.10.2.5 File split post export step

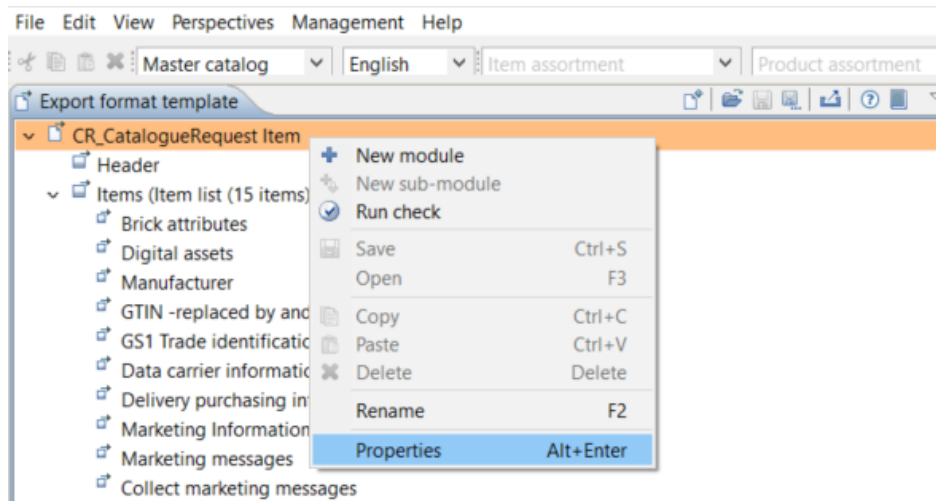
IM, like DSE, allows a file only to contain a certain number of documents (items respectively). Product 360 provides a standard post export step for XSL transformations. To fulfill GDSN requirements to restrict the file content to the allowed number of documents this post export step is used together with the provided XSLT files.

#### Configuration

If you create your own export templates or have adjusted one from prior versions, please add the xsl transformation to them.

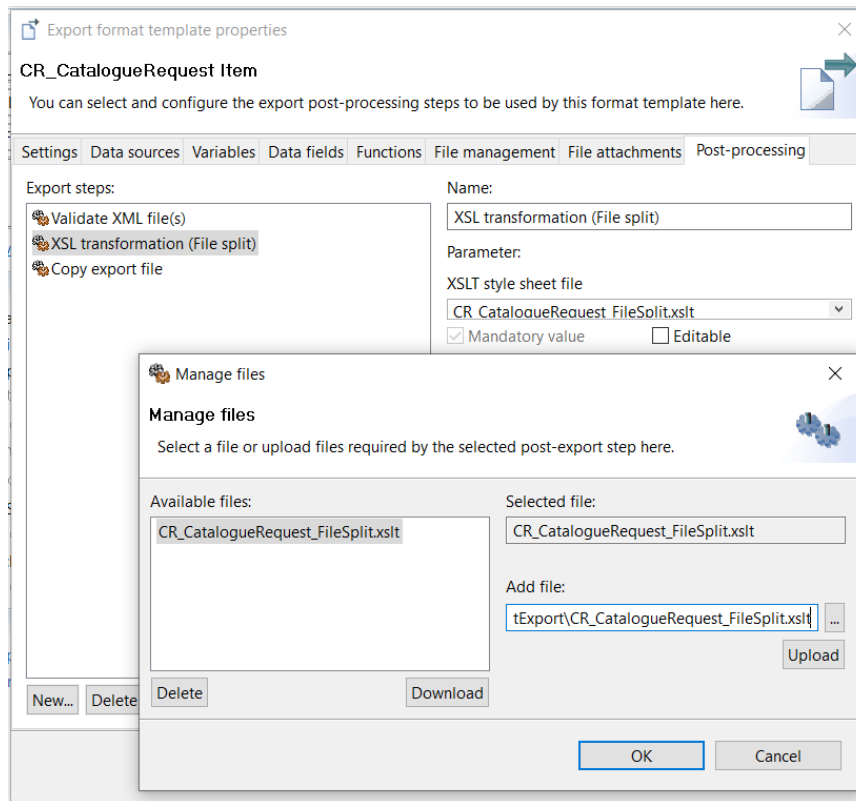
This is how to configure the export:

1. Open the export template in the *Export format templates* perspective
2. Use a right click on the template to open the *properties* and select the tab *Post-processing*



3. Add a new post export step by selecting *XSL transformation*  
You can change the name of the assigned post export step, e.g. XSL transformation (File split)

4. Click on Manage files... to add the stylesheet.



### 1.10.2.6 Customization

If you need to adjust the number of documents in a file, do the following:

1. Get the corresponding stylesheet from the resources package. It can be found in the folder DataSource\<GDSN pool>\ExportTemplates\PostExport. In the following example the stylesheet "CR\_CatalogueRequest\_FileSplit.xslt" is used.
2. Adjust the for-each-group tag to use the desired number of documents

Stylesheet	
1	<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:n="http://www.1worldsync.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.1worldsync.com http://schemas.preprod.1worldsync.com/schemas/item/2.0/CatalogueRequestProxy.xsd" version="2.0">
2	<xsl:param name="filename"/>
3	<xsl:template match="/n:envelope">
4	<xsl:for-each-group select="catalogueRequest/document" group-adjacent="(position()-1) idiv <Number-of-documents>">

Example	
1	<code>&lt;xsl:for-each-group select="catalogueRequest/document" group-adjacent="(position()-1) idiv 100"&gt;</code>

3. Add the changed stylesheet to your export template(s) as described above.

### 1.10.3 Appendix C: Packaging hierarchy data providers

The pools differentiate in how they accept items and their hierarchies. The case for the DSE pool you have to send the whole hierarchy of the item on the initial export. There is no way to add the hierarchy afterward like in the IM scenario.

To be able to support this case there are 2 additional hierarchy data providers available, which can be used for all GDSN scenarios.

#### 1.10.3.1 Complete packaging hierarchy

**Note:** In the GDSN scenario this data provider makes only sense when sending "Catalog Item Notification" messages to the DSE data pool.

##### Configuration

To configure the data provider you can look at the "Export data source configuration" on the page [GDSN Accelerator operation/Export templates \(see page 89\)](#).

##### Mode of operation

When using this data provider for an export template it will automatically export the whole hierarchy where the article is included. If the article is included in more than 1 hierarchy both hierarchies are exported.

This data provider only accepts valid hierarchies. A hierarchy is considered valid if there are no endless loops in it and if there're not more than 10 levels. (An invalid hierarchy is indicated in the packaging hierarchy view with a yellow triangle.) Otherwise this data provider will cancel the export immediately.

##### Data quality checks

When using a data quality run as a pre-export step, this data provider will re-evaluate the results of the data quality result.

It will only export items with the following criteria:

- The item is considered valid from DQ
- All items of the hierarchy are considered valid from DQ

All other items and hierarchies are not exported.

*Hint:* It is best practice to create an assortment with all items which only include the top level of the packaging hierarchies and then export the whole assortment.

## 1.10.4 Appendix D: Technical export information

### 1.10.4.1 Special export functions

#### GDSNDocumentId

The `GDSNDocumentId` and `GDSNParentDocumentId` export functions create unique identifiers for each document output in an export file. *Note:* these functions are only used and available for the IM pool.

#### EnumerationKey

Usually, GDSN messages contain codes for attributes that refer to a valid values list. By default, an exports output the labels for such attributes. The `EnumerationKey` export function is used to output the code instead of the label of enumeration values.

#### EnumerationKeyStandard

In contrast to the function `EnumerationKey`, this function can only be used for data having an enumeration assigned which is an extension of a standard enumeration. The function only returns a key if the value is contained in the standard enumeration, keys of the extended enumeration won't be returned.

This is needed for data fields that are logical keys in the data model but should be empty in the context of GDSN. Those fields got enumerations providing all valid GDSN values and a "<no code>" value.

#### Example

Enumeration of the "ArticlePackaging.PackagingType" field is `Enum.PackagingTypeCode.WithOptionalCode`. That enumeration is an extension of the `Enum.PackagingTypeCode` enumeration.

The function call with that field returns an empty string for the value "<No code>", and "BBG" for the field value "Bag in box".

#### GDSNEnumerationCode

Some GDSN attributes use other codes than the codes maintained in PIM. This function reads the GDSN codes from the corresponding configuration. It is possible to add additional codes, see [Repository configurations \(see page 29\)](#).

## GDSNDutyFeeTaxAgencyCode

The value for DutyFeeTaxAgencyCode depends on the target market and the tax type. The DSE specific export function `GDSNDutyFeeTaxAgencyCode` returns the configured value for "DutyFeeTaxAgencyCode" depending on target market and tax type.

Similar to the enumeration codes you can configure the values for DutyFeeTaxAgencyCode in the `plugin_customization.properties` file. The pattern of such entries is

```
com.heiler.ppm.gdsn.core/GDSN.dutyFeeTaxAgencyCode.<target market code>.<tax
type code> = <DutyFeeTaxAgencyCode>
```

```
# Germany
com.heiler.ppm.gdsn.core/GDSN.dutyFeeTaxAgencyCode.276.VAT = 246
# Austria
com.heiler.ppm.gdsn.core/GDSN.dutyFeeTaxAgencyCode.040.VAT = 294
```

### 1.10.4.2 General approach to output data

#### Empty values

All optional fields don't create an XML tag in case there is no value maintained. This is needed because an error will occur for empty tags. For this the export function `CreateXMLTagWithValue` respectively `CreateXMLTagWithContent` are used.

#### Format numbers and dates

It is important to format the numbers and dates in the needed format of the GDSN pool.

Therefore all fields with numbers must use the `FormatDecimal` export function. This function is responsible for formatting and transferring the given number according to the specified decimal separator and the number of decimal places. The decimal separator must be always "." for decimal values. All used date fields will use the `FormatDate` export function which is formatting the given date with a given pattern to match the needed format.

#### Encoding

There is need for defining or preventing the encoding of strings according to their encoding setting. There are possibilities to define this actively by the export functions `IfNotEmptyThenNotEnc` or `IfEmptyThenNotEnc` and inactively by using export functions that are handling the encoding already like `CreateXMLTagWithValue` which returns the transferred fragment itself as not encoded.

## Conditional output

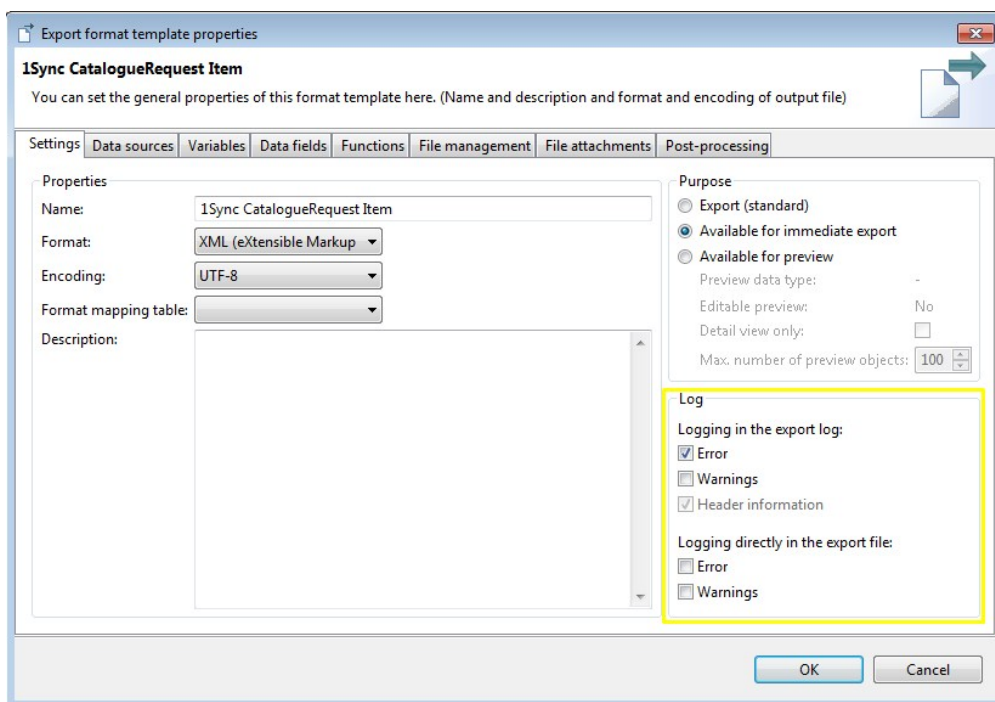
The data to output often depends on if any other specific data is set. Therefore you can define a conditional output by using export functions around the section that is to be output if the condition is fulfilled. This is possible in general by using the `IfNotEmptyThen` export function, or you can use specific export functions to generate conditional output by using a Boolean comparison with the `CompareBooleanValue` export function for example. Besides this it is also possible to define a default value for some export functions.

### 1.10.4.3 Extend/modify export templates

## Debug

There are two different possibilities to enable some debug information to be able to detect a problem.

## General export log

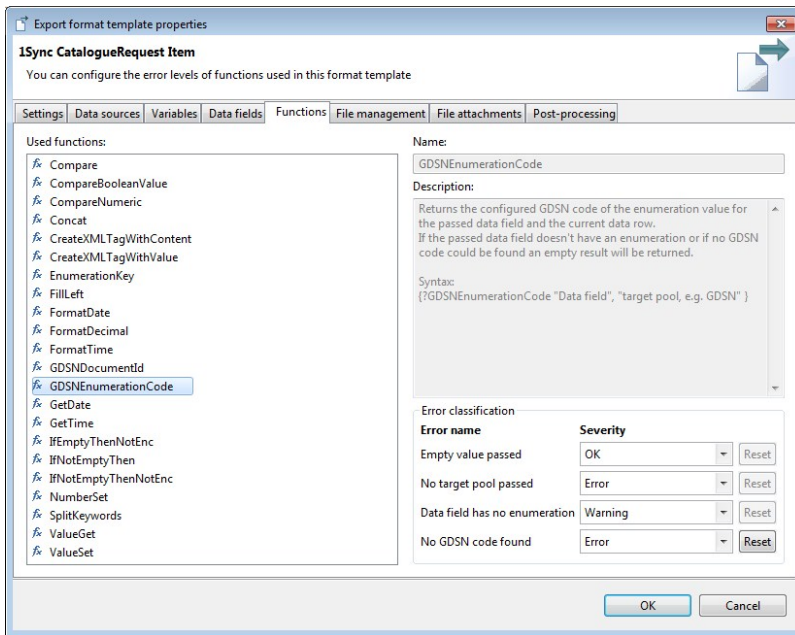


## 9 export log level

The logging in the export log defines the granularity of the logged information in the export template log which can be viewed in the process overview. By default it is only logging header information and errors. For debugging it is possible to enable the warning level as well but it is recommended to not having it active during casual business since this can change the export speed. You can also enable logging directly in the output file. You can enable the error and warning level to assist you in tracing the causes of errors during the development.

## Export function

Another possibility is to change the error levels of functions. You can define the error level for some functions in the export template properties dialog which can help you to figure out a problem. Those are also helpful if you want to define a specific behavior when for example an empty value is passed to the function. So this configuration can also be taken as standard modification depending on your data.



## Check output

There are different ways to provide the opportunity to inspect the output XML file when errors occur. Find two examples described in more detail below.

Add an additional "Copy export file" post-processing step before the "Validate XML file(s)" step. Define a specific target directory for this new export step. Whenever an export will be executed the export file will be copied to the target directory even if the XML validation will cause an error. But the target directory will always only contain the last executed export file.

Another way would be to disable the cancelling of the export within the "Validate XML file(s)" export step by setting the "Cancel export in case of error" to "No". Furthermore you need to change the "copy export file" target directory because you will probably send not valid XML files to the GDSN pool which will lead to errors.

With both of the mentioned examples it is possible to find out what the output is on a specific line in a XSD validation error. This makes it easier to figure out where a problem is in the data and/or template.

Besides this, it's always possible to manually transfer the export files to the pool. To put this into action the "Copy export file" export step can be deleted and after each export execution the export file can be downloaded in case of no error. This file has to be copied manually into the target directory.



### 1.10.5 Appendix E: Import mappings and hotfolder configuration

Almost every message transferred to or received from the data pool is tracked as status entry at the affected items. In the OpenAS2 solution, this happens by importing the corresponding message files.

The GDSN accelerator package contains various import mapping files that you should load and then save within Product 360. This is necessary to be able to setup the hotfolder configuration that manages the import of all the different message files. Depending on the used scenario, the import mappings are completely different. In the subchapters it is described which are needed for the corresponding Item Management or Standard GDSN data pool.

#### 1.10.5.1 Configure observed import hotfolders

1. Navigate to the configuration folder of your Product360 server and open the file server.properties.
2. Find and set the property inbox.hotfolders according to your OpenAS2 configuration, e.g.

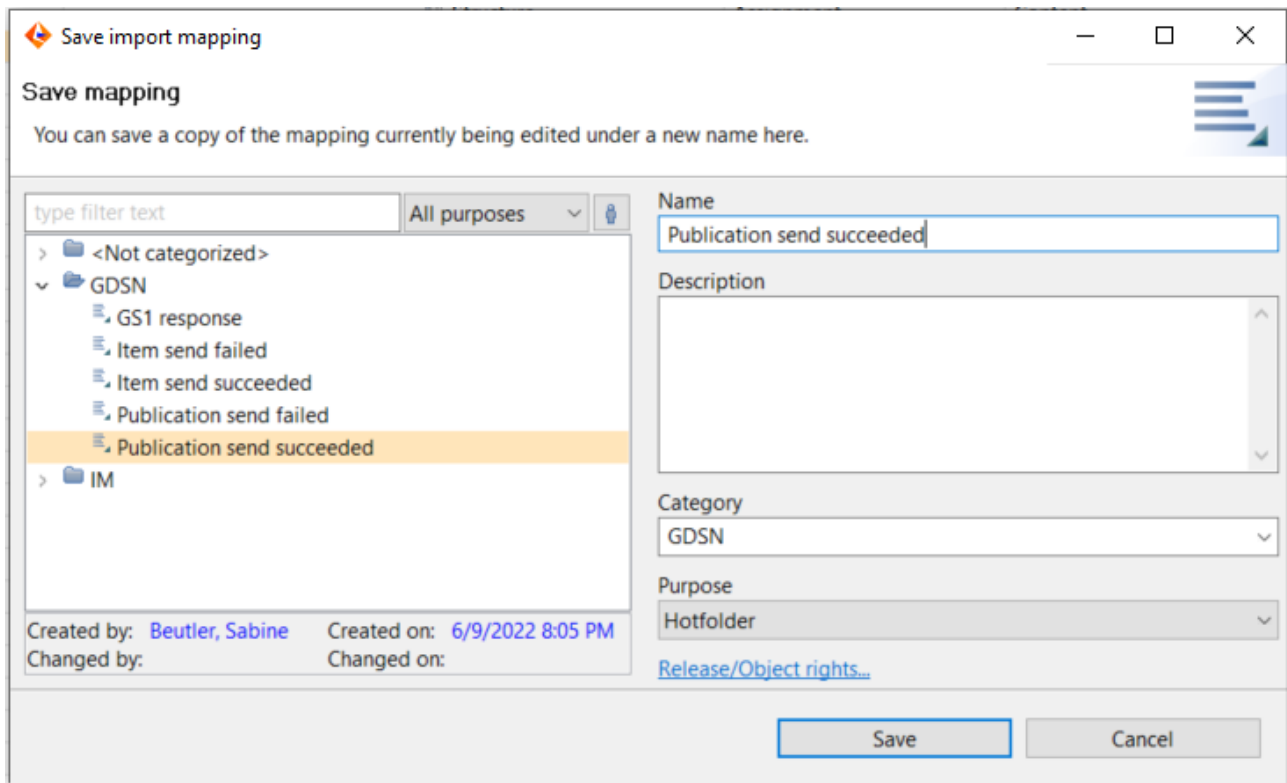
```
inbox.hotfolders = ${filestorage.dir.shared}/inbox/hotfolder;<OpenAS2>/data/
toGDSN/error;<OpenAS2>/data/toGDSN/sent;<OpenAS2>/data/fromGDSN/inbox
```

Please be sure to have no spaces between the configurations as this leads to an error during startup.

#### 1.10.5.2 Load and save the import mappings

Import the mappings to Product 360 by following the steps below:

- Go to "Import" perspective in the desktop client
- Click on "File" → "New import project..."
- In the dialog select "Load mapping from a file" and select your file. Click "OK" afterwards
- Click "File" → "Save mapping as..." and save the mapping. Set the *Category* to GDSN and please be sure that the *Purpose* is set to Hotfolder



### 1.10.5.3 Create and configure hotfolder

Depend on the used scenario, the specific values to configure the hotfolder may vary but the general approach is the same.

Hotfolder configuration

Go to the "Hotfolder" perspective and create a hotfolder configuration with the following settings:

- Identifier: for example "GDSN Communication Dispatcher"
- File pattern: for example "\*\*GDSN"
  - This pattern depends on the OpenAS2 configuration
  - Attention: files don't have a file extension (e.g. .xml)
- Preparatory import steps: select the available mapping dispatcher step

**Edit hotfolder configuration**

Hotfolder configuration  
Edit hotfolder configuration settings

Identifier: \*GDSN communication dispatcher

File pattern: \*GDSN  
Note: Placeholders, such as \*, # and ? can be used.

Mapping: Select...

Hotfolder group:

Structure: Heiler Standard

Catalog: Master catalog

Group assignment: Add (existing assignments are retained)

Lookup:

Import mode: Import or update all objects

Error mode: Tolerant (object is imported as far as possible)  
☐ Perform test run before import, cancel import on errors

Preparatory import steps: Mapping dispatcher for communication with GDSN data | Select...

Status: ☒ Hotfolder configuration is active

OK Cancel

### Preparatory import step

For each supported scenario using OpenAS2 we provide a pre-import step which dispatches the import mappings according to the specific file to import to ensure the proper processing of all messages.

**Select new preparatory import step**

Add new preparatory import step  
You can select a new preparatory import step here and add it to the hotfolder configuration

Import steps	Description
Mapping dispatcher for communication with GDSN data pool	This pre-import step analyzes all communication messages to and from the GDSN data pool and selects the corresponding mapping to import the right publication status.

OK Cancel

In the "Preparatory import steps" dialog, select the right mapping for each communication from the list of mappings as shown in the screenshot below. Usually, you can identify the respective mapping by its name.

Preparatory import steps

Preparatory import steps

Configuration of preparatory import steps

Preparatory import steps

Mapping dispatcher for communication with GDSN data pool

Name

Mapping dispatcher for communication with GDSN data pool

Item send failed\*: Item send failed

Item send succeeded\*: Item send succeeded

Publication send failed\*: Publication send failed

Publication send succeeded\*: Publication send succeeded

GS1 response from GDSN data pool\*: GS1 response

New...Delete

OK

Cancel

### 1.10.6 Appendix F: Field List

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleMicrobiologics.OrganismCode	Organism code	Microbiological information	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ foodAndBeveragePropertiesInformationModule/ microbiologicalInformation/ microbiologicalOrganismCode	FAB
ArticleMicrobiologicsUOM.Unit	Unit	UOM		FAB
ArticleMicrobiologicsUOM.OrganismMaximumValue	Organism maximum value ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	UOM	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ foodAndBeveragePropertiesInformationModule/ microbiologicalInformation/ microbiologicalOrganismMaximumValue	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleMicrobiologicsUOM.OrganismReferenceValue	Organism reference value ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	UOM	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ foodAndBeveragePropertiesInformationModule/ microbiologicalInformation/ microbiologicalOrganismReferenceValue	FAB
ArticleMicrobiologicsUOM.OrganismWarningValue	Organism warning value ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	UOM	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ foodAndBeveragePropertiesInformationModule/ microbiologicalInformation/ microbiologicalOrganismWarningValue	FAB
ArticleDietLang.DietTypeDescription	Description ({ArticleDomainLangType.LK.Language#Language selectable})	Language-specific data	dietInformationModule/ dietInformation/dietTypeDescription	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleDietList.DietTypeCode	Diet type	Diets	dietInformationModule/ dietInformation/dietTypeInformation/ dietTypeCode	FAB
ArticleDietList.DietSubtypeCode	Diet subtype ({ArticleSubDomainType.LK.Std_LK_T ext250_01#Diet type code selectable})	Diets	dietInformationModule/ dietInformation/dietTypeInformation/ dietTypeSubcode	FAB
ArticleDietList.Certification	Certification ({ArticleSubDomainType.LK.Std_LK_T ext250_01#Diet type code selectable})	Diets	Reference to certification entity with these fields: dietInformationModule/ dietInformation/dietTypeInformation/ dietCertification/ certificationOrganisationIdentifier, dietInformationModule/ dietInformation/dietTypeInformation/ dietCertification/ certificationAgencydietInformationMo dule/dietInformation/ dietTypeInformation/dietCertification/ certificationStandard,	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleAllergen.AllergenSpecificationAgency	Allergen specification agency	Allergen related information	allergenInformationModule/ allergenRelatedInformation/ allergenSpecificationAgency	FAB
ArticleAllergen.AllergenSpecificationName	Allergen specification name	Allergen related information	allergenInformationModule/ allergenRelatedInformation/ allergenSpecificationName	FAB
ArticleAllergenLang.AllergenStatement	Allergen statement ({ArticleDomainLangType.LK.Language#Language selectable})	Language-specific data	allergenInformationModule/ allergenRelatedInformation/ allergenStatement	FAB
ArticleAllergenList.AllergenTypeCode	Allergen type	Allergens	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ allergenInformationModule/ allergenRelatedInformation/allergen/ allergenTypeCode	FAB



Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleAllergenList.LevelOfContainmentCode	Level of containment ({ArticleSubDomainType.LK.Std_LK_Text250_01#Allergen type selectable})	Allergens	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ allergenInformationModule/ allergenRelatedInformation/allergen/ LevelOfContainmentCode	FAB
ArticlePreparationServing.PreparationType	Preparation type	Preparation serving information	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ foodAndBeveragePreparationServingModule/preparationServing/ preparationTypeCode	FAB
ArticlePreparationServing.ConvenienceLevel	Convenience level	Preparation serving information	foodAndBeveragePreparationServingModule/preparationServing/ convenienceLevelPercent	FAB
ArticlePreparationServingUOM.maximumOptimumConsumptionTemperature	Maximum optimum consumption temperature ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Preparation serving information UOM	foodAndBeveragePreparationServingModule/preparationServing/ maximumOptimumConsumptionTemperature	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticlePreparationServingUOM.maximumOptimumConsumptionTemperatureUOM	Maximum optimum consumption temperature UOM ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Preparation serving information UOM		FAB
ArticlePreparationServingUOM.minimumOptimumConsumptionTemperature	Minimum optimum consumption temperature ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Preparation serving information UOM	foodAndBeveragePreparationServingModule/preparationServing/minimumOptimumConsumptionTemperature	FAB
ArticlePreparationServingUOM.minimumOptimumConsumptionTemperatureUOM	Minimum optimum consumption temperature UOM ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Preparation serving information UOM		FAB
ArticlePreparationServingLang.Precautions	Precautions ({ArticleDomainLangType.LK.Language#Language selectable})	Language-specific data	foodAndBeveragePreparationServingModule/preparationServing/preparationConsumptionPrecautions	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticlePreparationServingLang.PreparationInstructions	Preparation instructions ({ArticleDomainLangType.LK.Language#Language selectable})	Language-specific data	foodAndBeveragePreparationServingModule/preparationServing/preparationInstructions	FAB
ArticlePreparationServingLang.ServingSuggestion	Serving suggestion ({ArticleDomainLangType.LK.Language#Language selectable})	Language-specific data	foodAndBeveragePreparationServingModule/preparationServing/servingSuggestion	FAB
ArticlePreparationServingProductYield.ProductYieldType	Product yield type	Product yield information	CatalogueItemNotification/CatalogueItem/tradeItem/tradeItemComponents/foodAndBeveragePreparationServingModule/preparationServing/productYieldInformation/productYieldTypeCode	FAB
ArticlePreparationServingProductYield.ProductYieldVariationPercentage	Product yield variation [%]	Product yield information	CatalogueItemNotification/CatalogueItem/tradeItem/tradeItemComponents/foodAndBeveragePreparationServingModule/preparationServing/productYieldInformation/productYieldVariationPercentage	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticlePreparationServingProductYieldUOM.ProductYield	Product yield ({ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Product yield UOM	foodAndBeveragePreparationServingModule/preparationServing/productYieldInformation/productYield	FAB
ArticlePreparationServingProductYieldUOM.ProductYieldUOM	Product yield UOM ({ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Product yield UOM		FAB
ArticleNutrient.PreparationState	Preparation state	Nutritional information	nutritionalInformationModule/nutrientHeader/preparationStateCode	FAB
ArticleNutrientLang.ServingSizeDescription	Serving size description ({ArticleDomainLangType.LK.Language#Language selectable})	Language-specific data	nutritionalInformationModule/nutrientHeader/servingSizeDescription	FAB
ArticleNutrientLang.DailyValueIntakeReference	Daily value intake reference ({ArticleDomainLangType.LK.Language#Language selectable})	Language-specific data	CatalogueItemNotification/CatalogueItem/tradeItem/tradeItemComponents/nutritionalInformationModule/nutrientHeader/dailyValueIntakeReference	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleNutrientBasisQuantity.NutrientBasisQuantity	Nutrient basis quantity ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Nutrient basis quantity	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ nutritionalInformationModule/ nutrientHeader/nutrientBasisQuantity	FAB
ArticleNutrientBasisQuantity.NutrientBasisQuantityUOM	Nutrient basis quantity UOM ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Nutrient basis quantity		FAB
ArticleNutrientBasisQuantity.ServingSize	Serving size ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Nutrient basis quantity	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ nutritionalInformationModule/ nutrientHeader/servingSize	FAB
ArticleNutrientBasisQuantity.ServingSizeUOM	Serving size UOM ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Nutrient basis quantity		FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleNutrientList.NutrientTypeCode	Nutrient type	Nutrients	nutritionalInformationModule/ nutrientHeader/nutrientDetail/ nutrientTypeCode	FAB
ArticleNutrientListQuantity.NutrientBasisQuantityType	Nutrient basis quantity type	Nutrient list quantity	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ nutritionalInformationModule/ nutrientHeader/ nutrientBasisQuantityTypeCode	FAB
ArticleNutrientListQuantity.QuantityContained	Quantity contained ({../ ArticleSubDomainType.LK.Std_LK_Text250_01#Nutrient type selectable}, {ArticleSubDomainUOMType.LK.Std_LK_Text100_01#Nutrient basis quantity type selectable}, {ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Nutrient list quantity	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ nutritionalInformationModule/ nutrientHeader/nutrientDetail/ quantityContained	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleNutrientListQuantity.QuantityContainedUOM	Quantity contained UOM ({../ArticleSubDomainType.LK.Std_LK_Text250_01#Nutrient type selectable}, {ArticleSubDomainUOMType.LK.Std_LK_Text100_01#Nutrient basis quantity type selectable}, {ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Nutrient list quantity		FAB
ArticleNutrientListQuantity.DailyValueIntakePercent	Percentage of daily intake ({../ArticleSubDomainType.LK.Std_LK_Text250_01#Nutrient type selectable}, {ArticleSubDomainUOMType.LK.Std_LK_Text100_01#Nutrient basis quantity type selectable}, {ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Nutrient list quantity	nutritionalInformationModule/ nutrientHeader/nutrientDetail/ avpList/ dailyValueIntakePercentMeasurement PrecisionCode	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleNutrientListQuantity.MeasurementPrecisionCode	Measurement precision ({../ArticleSubDomainType.LK.Std_LK_Text250_01#Nutrient type selectable}, {ArticleSubDomainUOMType.LK.Std_LK_Text100_01#Nutrient basis quantity type selectable}, {ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Nutrient list quantity	nutritionalInformationModule/nutrientHeader/nutrientDetail/measurementPrecisionCode	FAB
ArticleIngredient.IngredientOfConcernCode	Ingredient of concern code	Ingredient information	CatalogueItemNotification/CatalogueItem/tradeItem/tradeItemComponents/foodAndBeverageIngredientModule/ingredientOfConcernCode	FAB
ArticleIngredient.JuiceContentPercentage	Juice content [%]	Ingredient information	foodAndBeverageIngredientModule/juiceContentPercent	FAB
ArticleIngredientAdditive.AdditiveName	Additive name	Additives	foodAndBeverageIngredientModule/additiveInformation/additiveName	FAB



Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleIngredientAdditive.LevelOfContainmentCode	Level of containment ({ArticleSubDomainType.LK.Std_LK_T ext250_01#Additive name selectable})	Additives	foodAndBeverageIngredientModule/ additiveInformation/ LevelofContainmentCode	FAB
ArticleIngredientComponent.Sequence	Sequence	Ingredient component	foodAndBeverageIngredientModule/ foodAndBeverageIngredient/ ingredientSequence	FAB
ArticleIngredientComponent.Ingredient	Ingredient ({ArticleSubDomainType.LK.Std_LK_T ext250_01#Sequence selectable})	Ingredient component		FAB
ArticleIngredientComponent.Purpose	Purpose ({ArticleSubDomainType.LK.Std_LK_T ext250_01#Sequence selectable})	Ingredient component	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ foodAndBeverageIngredientModule/ foodAndBeverageIngredient/ ingredientPurpose	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleIngredientComponent.ContentPercentage	Content [%] ({ArticleSubDomainType.LK.Std_LK_Text250_01#Sequence selectable})	Ingredient component	foodAndBeverageIngredientModule/ foodAndBeverageIngredient/ ingredientContentPercentage	FAB
ArticleServingQuantity.MaximumNumberOfSmallestUnitsPerPackage	Maximum number of smallest units per package	Serving quantity information	foodAndBeveragePreparationServingModule/servingQuantityInformation/ maximumNumberOfSmallestUnitsPerPackage	FAB
ArticleServingQuantity.NumberOfServingsPerPackage	Number of servings per package	Serving quantity information	foodAndBeveragePreparationServingModule/servingQuantityInformation/ numberOfServingsPerPackageMeasurementPrecisionCode	FAB
ArticleServingQuantity.MeasurementPrecisionOfNumberOfServingsPerPackage	Measurement precision of number of servings per package	Serving quantity information	foodAndBeveragePreparationServingModule/servingQuantityInformation/ numberOfServingsPerPackageMeasurementPrecisionCode	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleServingQuantity.NumberOfSmallestUnitsPerPackage	Number of smallest units per package	Serving quantity information	foodAndBeveragePreparationServingModule/servingQuantityInformation/numberOfSmallestUnitsPerPackage	FAB
ArticlePreparation.ManufacturerPreparationTypeCode	Manufacturer preparation type code	Preparation Information	CatalogueItemNotification/CatalogueItem/tradeItem/tradeItemComponents/foodAndBeveragePreparationServingModule/manufacturerPreparationTypeCode	FAB
ArticleNutritionalClaim.NutritionalClaimTypeCode	Nutritional claim type code	Nutritional claims	CatalogueItemNotification/CatalogueItem/tradeItem/tradeItemComponents/nutritionalInformationModule/nutritionalClaimDetail/nutritionalClaimCode	FAB
ArticleNutritionalClaim.NutritionalClaimNutrientElementCode	Nutritional claim element code	Nutritional claims	nutritionalInformationModule/nutritionalClaimDetail/nutritionalClaimNutrientElementCode	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleCertifications.Certification	Certification	Certification information		FAB
ArticlePhysioChemical.PhysiochemicalCharacteristicCode	Physiochemical characteristic code	Physiochemical information	foodAndBeveragePropertiesInformationModule/physiochemicalCharacteristic/physiochemicalCharacteristicCode	FAB
ArticlePhysioChemicalUOM.PhysiochemicalCharacteristicValue	Physiochemical characteristic value ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Physiochemical information UOM	foodAndBeveragePropertiesInformationModule/physiochemicalCharacteristic/physiochemicalCharacteristicValue	FAB
ArticlePhysioChemicalUOM.PhysiochemicalCharacteristicValueUOM	Physiochemical characteristic value UOM ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Physiochemical information UOM		FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleHealthCare.CompositionIncludeLatex	Does composition include latex	Health related information	healthRelatedInformationModule/ healthRelatedInformation/ doesTradItemCompositionIncludeLatex	FAB
ArticleHealthCare.HealthClaimCode	Health claims	Health related information	healthRelatedInformationModule/ healthRelatedInformation/ healthClaimCode	FAB
ArticleHealthCare.IsNotIntendedForConsumption	Is not intended for consumption	Health related information	healthRelatedInformationModule/ healthRelatedInformation/ isTradItemChemicalNotIntendedForHumanConsumption	FAB
ArticleHealthCare.NutritionLabelTypeCode	Nutrition label type code	Health related information	healthRelatedInformationModule/ healthRelatedInformation/ nutritionalLabelTypeCode	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleHealthCare.NutritionalProgramCode	Nutritional program code	Health related information	healthRelatedInformationModule/ healthRelatedInformation/ nutritionalProgram/ nutritionalProgramCode	FAB
ArticleHealthCareLang.CompulsoryAdditivesLabel	Compulsory additives label ({ArticleDomainLangType.LK.Language#Language selectable})	Language-specific data	healthRelatedInformationModule/ healthRelatedInformation/ compulsoryAdditiveLabelInformation	FAB
ArticleHealthCareLang.HealthClaimDescription	Description ({ArticleDomainLangType.LK.Language#Language selectable})	Language-specific data	healthRelatedInformationModule/ healthRelatedInformation/ healthClaimDescription	FAB
ArticleDairyFishMeatPoultry.FatInMilkContent	Fat in milk content [%]	Dairy fish meat poultry	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ fatInMilkContent	FAB
ArticleDairyFishMeatPoultry.IsHomogenised	Is homogenised	Dairy fish meat poultry	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ isHomogenised/IsHomogenised	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleDairyFishMeatPoultryUOM.CasingTareWeight	Casing tare weight ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Dairy fish meat poultry UOM	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ casingTareWeight	FAB
ArticleDairyFishMeatPoultryUOM.CasingTareWeightUOM	Casing tare weight UOM ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Dairy fish meat poultry UOM		FAB
ArticleCheese.CheeseMaturationProcessContainerTypeCode	Cheese maturation container process type	Cheese information	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ cheeseMaturationProcessContainerTypeCode	FAB
ArticleCheese.FatPercentageInDryMatter	Fat in dry matter [%]	Cheese information	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ fatPercentageInDryMatter	FAB
ArticleCheese.IsRindEdible	Is rind edible	Cheese information	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ isRindEdible	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleCheese.RennetTypeCode	Rennet type code	Cheese information	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ rennetTypeCode	FAB
ArticleCheese.SurfaceOfCheeseAtEndOfRipeningCode	Surface of cheese at end of ripening	Cheese information	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ surfaceOfCheeseAtEndOfRipeningCode	FAB
ArticleCheeseUOM.RipeningTimePeriod	Ripening time period ({ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Cheese information UOM	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ ripeningTimePeriod	FAB
ArticleCheeseUOM.RipeningTimePeriodUOM	Ripening time period UOM ({ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Cheese information UOM		FAB
ArticleCheeseLang.CheeseMaturationPeriodDescription	Cheese maturation period description ({ArticleSubDomainLangType.LK.Language#Language selectable})	Language-specific data	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ cheeseMaturationPeriodDescription	FAB



Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleFishMeatPoultryContent.FishMeatPoultryTypeCode	Fish meat poultry type code	Fish meat poultry content	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ fishMeatPoultryContent/ fishMeatPoultryTypeCode	FAB
ArticleFishMeatPoultryContent.FishMeatPoultryTypeCodeListAgency	Fish meat poultry type code list agency	Fish meat poultry content	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ fishMeatPoultryContent/ fishMeatPoultryTypeCodeListAgency	FAB
ArticleFishMeatPoultryContent.FishMeatPoultryTypeCodeListIdentification	Fish meat poultry type code list identification	Fish meat poultry content	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ fishMeatPoultryContent/ fishMeatPoultryTypeCodeListIdentification	FAB
ArticleFishMeatPoultryContentUOM. MinimumFishMeatPoultryContent	Minimum fish meat poultry content ({../ ArticleSubDomainType.LK.Std_LK_Text250_01#Fish meat poultry type code selectable}, {ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Fish meat poultry content UOM	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ fishMeatPoultryContent/ minimumFishMeatPoultryContent	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleFishMeatPoultryContentUOM. MinimumFishMeatPoultryContentUOM	Minimum fish meat poultry content UOM ({../ArticleSubDomainType.LK.Std_LK_Text250_01#Fish meat poultry type code selectable}, {ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Fish meat poultry content UOM		FAB
GDSNCanadaExtension.ReinstatementDate	Reinstatement date	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.NotLegalToAdvertise	Not legal to advertise by province	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.NotLegalToDiscount	Not legal to discount by province	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.GSTHSTApplicable	GST/HST applicable	Canada-specific GDSN data	-	IM

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNCanadaExtension.PSTApplicableByProvince	PST applicable by province	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.EnvironmentalLevyApplicableByProvince	Environmental levy applicable by province	Canada-specific GDSN data	environmentalLevyApplicableByProvince	IM
GDSNCanadaExtension.JuiceContentPercentage	Juice content [%]	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.MarkedLotNumber	Is marked lot number	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.SpecialHandlingCodeTransportation	Special handling code - transportation	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.ChannelOfDistribution	Channel of distribution	Canada-specific GDSN data	-	IM

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNCanadaExtension.BrokerInvoiceIndicator	Broker invoice indicator	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.PercentageOfWaterContent	Percentage of water content	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.PercentageOfMoistureLoss	Percentage of moisture loss	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.ProductionType	Production type	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.ExpiryDateFormat	Expiry date format	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.ConsumerSupportNumber	Consumer support number	Canada-specific GDSN data	consumerSupportNumber	IM

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNCanadaExtension.IsCustomLabel	Is custom label	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.AverageServingsPerCase	Average servings per case	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.ActiveIngredientGroupNumber	Active ingredient group number	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.ActiveIngredientNameStrengthAndBasis	Active ingredient name strength and basis	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.DosageFormType	Dosage form type	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.DrugMerchandisingBtcProvince	Drug merchandising behind the counter by province	Canada-specific GDSN data	-	IM

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNCanadaExtension.DrugMerchandisingOtcProvince	Drug merchandising OTC by province	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.DrugMerchandisingRxProvince	Drug merchandising Rx by province	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.DrugMerchandisingUnscheduledProvince	Drug merchandising unscheduled by province	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.ShapeDescriptionFrench	Shape description french	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.TradeItemregulationTypeCodeControlledDrug	Trade item regulation type code controlled drug	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.TradeItemregulationTypeCodeNarcotic	Trade item regulation type code narcotic	Canada-specific GDSN data	-	IM

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNCanadaExtension.TradeItemregulationTypeCodePrecursor	Trade item regulation type code precursor	Canada-specific GDSN data	-	IM
GDSNCanadaExtension.TradeItemregulationTypeCodeTargetedSubstance	Trade item regulation type code targeted substance	Canada-specific GDSN data	-	IM
GDSNCanadaExtensionLang.OperatorDescription	Operator description (ArticleDomainLangType.LK.Language#Language selectable)	Language-specific data	-	IM
GDSNCanadaExtensionLang.ForMoreInfoAboutProduct	For more info about product (ArticleDomainLangType.LK.Language#Language selectable)	Language-specific data	-	IM
GDSNCanadaExtensionLang.PackageAndStorageAboutProduct	Package and storage about product (ArticleDomainLangType.LK.Language#Language selectable)	Language-specific data	packageAndStorageAboutProduct	IM

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNCanadaExtensionLang.WeightScaleDescription	Weight scale description (ArticleDomainLangType.LK.Language#Language selectable)	Language-specific data	-	IM
GDSNCanadaExtensionLang.TradeItemMarkingsDescription	Trade item markings description (ArticleDomainLangType.LK.Language#Language selectable)	Language-specific data	-	IM
GDSNCanadaExtensionUOM.SuggestedServingSize	Suggested serving size (ArticleDomainUOMType.LK.UOMType#UOM type selectable)	UOM	suggestedServingSize	IM
GDSNCanadaExtensionUOM.SuggestedServingSizeUOM	Suggested serving size UOM (ArticleDomainUOMType.LK.UOMType#UOM type selectable)	UOM		IM
GDSNCanadaExtensionCustomer.Customer	Customer	Customer-specific data	-	IM



Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNCanadaExtensionCustomer.DeliveryLeadTime	Delivery lead time ({ArticleDomainPartyType.LK.Party#Customer selectable})	Customer-specific data	-	IM
GDSNCanadaExtensionCustomer.MinimumTradeItemDaysInWarehouse	Minimum trade item days in warehouse ({ArticleDomainPartyType.LK.Party#Customer selectable})	Customer-specific data	-	IM
GDSNCustomerSpecificExtension.Party	Customer	Customer-specific GDSN data	-	GDSN
GDSNCustomerSpecificExtension.PublishItem	Publish item	Customer-specific GDSN data	-	GDSN
GDSNTargetMarketExtension.IsActiveInMarket	Is active in market	Target market specific GDSN data	-	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtension.IsOrderableUnit	Is orderable unit	Target market specific GDSN data	CatalogueItemNotification/ CatalogueItem/tradeItem/ isTradeItemAnOrderableUnit	GDSN
GDSNTargetMarketExtension.IsDispatchUnit	Is dispatch unit	Target market specific GDSN data	CatalogueItemNotification/ CatalogueItem/tradeItem/ isTradeItemADespatchUnit	GDSN
GDSNTargetMarketExtension.IsInvoiceUnit	Is invoice unit	Target market specific GDSN data	CatalogueItemNotification/ CatalogueItem/tradeItem/ isTradeItemAnInvoiceUnit	GDSN
GDSNTargetMarketExtension.IsPackagingMarkedReturnable	Is packaging marked returnable	Target market specific GDSN data	packagingMarkingModule/ packagingMarking/ isPackagingMarkedReturnable	GDSN
GDSNTargetMarketExtension.DiscontinuedDate	Discontinued date	Target market specific GDSN data	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemSynchronisationDates/ discontinuedDateTime	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtension.CountriesOfOrigin	Countries of origin	Target market specific GDSN data	placeOfItemActivityModule/ placeOfProductActivity/ countryOfOrigin/countryCode	GDSN
GDSNTargetMarketExtension.PrimaryDeliveryMethod	Primary delivery method	Target market specific GDSN data	-	GDSN
GDSNTargetMarketExtension.UnitsPerCase	Total units per case	Target market specific GDSN data	-	GDSN
ArticleMarketing.CouponFamilyCode	Coupon family code	Marketing information	marketingInformationModule/ marketingInformation/ couponFamilyCode	GDSN
GDSNTargetMarketExtension.IsPrivate	Is private	Target market specific GDSN data	isPrivate	GDSN
GDSNTargetMarketExtension.PricingOnProduct	Pricing on product	Target market specific GDSN data	packagingMarkingModule/ packagingMarking/isPriceOnPack	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtension.NumberOfLayersPerPallet	Number of layers per pallet	Target market specific GDSN data	tradeItemHierarchyModule/ tradeItemHierarchy/ quantityOfLayersPerPallet	GDSN
GDSNTargetMarketExtension.NumberOfItemsInPalletLayer	Number of items per pallet layer	Target market specific GDSN data	tradeItemHierarchyModule/ tradeItemHierarchy/ quantityOfTradeItemsPerPalletLayer	GDSN
GDSNTargetMarketExtension.NumberOfItemsPerPallet	Number of items per pallet	Target market specific GDSN data	tradeItemHierarchyModule/ tradeItemHierarchy/ quantityOfTradeItemsPerPallet	GDSN
GDSNTargetMarketExtension.ProductMarkedRecyclable	Product marked recyclable	Target market specific GDSN data	packagingMarkingModule/ packagingMarking/ isTradeItemMarkedAsRecyclable	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtension.Manufacturer	Manufacturer	Target market specific GDSN data	CatalogueItemNotification/ CatalogueItem/tradeItem/ manufacturer/gln; CatalogueItemNotification/ CatalogueItem/tradeItem/ manufacturer/partyName	GDSN
GDSNTargetMarketExtension.ReplacesGTIN	Replaces GTIN	Target market specific GDSN data	CatalogueItemNotificationType/ CatalogueItem/TradeItem/ dependentProprietaryTradeItem/ referencedTradeItemGTIN; CatalogueItemNotificationType/ CatalogueItem/TradeItem/ dependentProprietaryTradeItem/ referencedTradeItemTypeCode = REPLACED	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtension.ReplacedByGTIN	Replaced by GTIN	Target market specific GDSN data	CatalogueItemNotificationType/ CatalogueItem/TradeItem/ dependentProprietaryTradeItem/ referencedTradeItemGTIN; CatalogueItemNotificationType/ CatalogueItem/TradeItem/ dependentProprietaryTradeItem/ referencedTradeItemTypeCode = REPLACED_BY	GDSN
ArticleMarketing.SpecialItemCode	Special item code	Marketing information	marketingInformationModule/ marketingInformation/ specialItemCode	GDSN
GDSNTargetMarketExtension.VariableTradeItemTypeCode	Variable trade item type	Target market specific GDSN data	variableTradeItemInformationModule /variableTradeItemInformation/ variableTradeItemTypeCode	GDSN
GDSNTargetMarketExtension.ChildNutritionLabel	Child nutrition label (USDA)	Target market specific GDSN data	catalogueItemNotification/ catalogueItem/tradeItem/avpList/ doesTradeItemCarryUSDAChildNutritionLabel	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtension.IsDangerousSubstance	Is dangerous substance	Target market specific GDSN data	dangerousSubstanceInformationModule/dangerousSubstanceInformation/dangerousSubstanceProperties/isDangerousSubstance	GDSN
GDSNTargetMarketExtension.IsRegulatedForTransportation	Is regulated for transportation	Target market specific GDSN data	safetyDataSheetModule/safetyDataSheetInformation/isRegulatedForTransportation	GDSN
GDSNTargetMarketExtensionLang.FunctionalName	Functional name ({ArticleMarketExtensionLangType.LK.Language#Language selectable})	Language-specific data	CatalogueItemNotification/CatalogueItem/tradeItem/tradeItemComponents/tradeItemDescriptionModule/tradeItemDescriptionInformation/functionalName	GDSN
GDSNTargetMarketExtensionLang.ShortDescription	GDSN short description ({ArticleMarketExtensionLangType.LK.Language#Language selectable})	Language-specific data	CatalogueItemNotification/CatalogueItem/tradeItem/tradeItemComponents/tradeItemDescriptionModule/tradeItemDescriptionInformation/descriptionShort	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtensionLang.AdditionalDescription	Additional description ({ArticleMarketExtensionLangType.LK.Language#Language selectable})	Language-specific data	tradItemDescriptionModule/ tradItemDescriptionInformation/ additionalTradItemDescription	GDSN
ArticleMarketingLang.MarketingMessage	Marketing message ({ArticleDomainLangType.LK.Language#Language selectable}, {ArticleDomainLangType.LK.Std_LK_Int_01#Sequence selectable})	Language-specific data	marketingInformationModule/ marketingInformation/ tradItemMarketingMessage	GDSN
GDSNTargetMarketExtensionLang.Variant	Variant ({ArticleMarketExtensionLangType.LK.Language#Language selectable})	Language-specific data	tradItemDescriptionModule/ tradItemDescriptionInformation/ variantDescription	GDSN
GDSNTargetMarketExtensionLang.ProductDescription	Product description ({ArticleMarketExtensionLangType.LK.Language#Language selectable})	Language-specific data	tradItemDescriptionModule/ tradItemDescriptionInformation/ tradItemDescription	GDSN
GDSNTargetMarketExtensionLang.SubBrand	Sub brand ({ArticleMarketExtensionLangType.LK.Language#Language selectable})	Language-specific data	tradItemDescriptionModule/ tradItemDescriptionInformation/ brandNameInformation/subBrand	GDSN



Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtensionUOM.Height	Height (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM	tradeItemMeasurementsModule/ tradeItemMeasurements/height	GDSN
GDSNTargetMarketExtensionUOM.HeightUOM	Height UOM (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM		GDSN
GDSNTargetMarketExtensionUOM.Width	Width (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM	tradeItemMeasurementsModule/ tradeItemMeasurements/width	GDSN
GDSNTargetMarketExtensionUOM.WidthUOM	Width UOM (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM		GDSN
GDSNTargetMarketExtensionUOM.Depth	Depth (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM	tradeItemMeasurementsModule/ tradeItemMeasurements/depth	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtensionUOM.DepthUOM	Depth UOM (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM		GDSN
GDSNTargetMarketExtensionUOM.GrossWeight	Gross weight (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM	tradeItemMeasurementsModule/ tradeItemMeasurements/ tradeItemWeight/grossWeight	GDSN
GDSNTargetMarketExtensionUOM.GrossWeightUOM	Gross weight UOM (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM		GDSN
GDSNTargetMarketExtensionUOM.NetWeight	Net weight (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM	tradeItemMeasurementsModule/ tradeItemMeasurements/ tradeItemWeight/netWeight	GDSN
GDSNTargetMarketExtensionUOM.NetWeightUOM	Net weight UOM (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM		GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtensionUOM.DeliveryToDCTemperatureMaximum	Delivery to distribution center temperature maximum (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM	tradeItemTemperatureInformationModule/ tradeItemTemperatureInformation/ maximumTemperature; tradeItemTemperatureInformationModule/ tradeItemTemperatureInformation/ temperatureQualifierCode = DELIVERY_TO_DISTRIBUTION_CENTRE	GDSN
GDSNTargetMarketExtensionUOM.DeliveryToDCTemperatureMaximumUOM	Delivery to distribution center temperature maximum UOM (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM		GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtensionUOM.DeliveryToDCTemperatureMinimum	Delivery to distribution center temperature minimum ({ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable})	UOM	tradeltemTemperatureInformationModule/ tradeltemTemperatureInformation/ maximumTemperature; tradeltemTemperatureInformationModule/ tradeltemTemperatureInformation/ temperatureQualifierCode = DELIVERY_TO_DISTRIBUTION_CENTRE	GDSN
GDSNTargetMarketExtensionUOM.DeliveryToDCTemperatureMinimumUOM	Delivery to distribution center temperature minimum UOM ({ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable})	UOM		GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtensionUOM.DeliveryToMarketTemperatureMaximum	Delivery to market temperature maximum (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM	tradeltemTemperatureInformationModule/ tradeltemTemperatureInformation/ maximumTemperature; tradeltemTemperatureInformationModule/ tradeltemTemperatureInformation/ temperatureQualifierCode = DELIVERY_TO_MARKET	GDSN
GDSNTargetMarketExtensionUOM.DeliveryToMarketTemperatureMaximumUOM	Delivery to market temperature maximum UOM (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM		GDSN
GDSNTargetMarketExtensionUOM.DeliveryToMarketTemperatureMinimum	Delivery to market temperature minimum (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM	tradeltemTemperatureInformationModule/ tradeltemTemperatureInformation/ maximumTemperature; tradeltemTemperatureInformationModule/ tradeltemTemperatureInformation/ temperatureQualifierCode = DELIVERY_TO_MARKET	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtensionUOM.DeliveryToMarketTemperatureMinimumUOM	Delivery to market temperature minimum UOM (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM		GDSN
GDSNTargetMarketExtensionUOM.StorageHandlingTempMax	Storage handling temperature maximum (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM	tradeItemTemperatureInformationModule/ tradeItemTemperatureInformation/ maximumTemperature; tradeItemTemperatureInformationModule/ tradeItemTemperatureInformation/ temperatureQualifierCode = STORAGE_HANDLING	GDSN
GDSNTargetMarketExtensionUOM.StorageHandlingTempMaxUOM	Storage handling temperature maximum UOM (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM		GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNTargetMarketExtensionUOM.StorageHandlingTempMin	Storage handling temperature minimum (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM	tradeItemTemperatureInformationModule/ tradeItemTemperatureInformation/ maximumTemperature; tradeItemTemperatureInformationModule/ tradeItemTemperatureInformation/ temperatureQualifierCode = STORAGE_HANDLING	GDSN
GDSNTargetMarketExtensionUOM.StorageHandlingTempMinUOM	Storage handling temperature minimum UOM (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM		GDSN
GDSNTargetMarketExtensionUOM.Volume	Volume (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM	tradeItemMeasurementsModule/ tradeItemMeasurements/ inBoxCubeDimension	GDSN
GDSNTargetMarketExtensionUOM.VolumeUOM	Volume UOM (ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable)	UOM		GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNExtension.ProductType	Product type	GDSN	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemUnitDescriptorCode	GDSN
GDSNExtension.BrandName	Brand name	GDSN	tradeItemDescriptionModule/ tradeItemDescriptionInformation/ brandNameInformation/brandName	GDSN
GDSNExtension.BrandOwner	Brand owner	GDSN	CatalogueItemNotification/ CatalogueItem/tradeItem/ brandOwner/gln	GDSN
GDSNExtension.IsService	Is service	GDSN	CatalogueItemNotification/ CatalogueItem/tradeItem/ isTradeItemAService	GDSN
GDSNExtension.IsConsumerUnit	Is consumer unit	GDSN	CatalogueItemNotification/ CatalogueItem/tradeItem/ isTradeItemAConsumerUnit	GDSN



Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNExtension.IsBaseUnit	Is base unit	GDSN	CatalogueItemNotification/ CatalogueItem/tradeItem/ isTradeItemABaseUnit	GDSN
GDSNExtension.HasVariableWeight	Has variable weight	GDSN	variableTradeItemInformationModule /variableTradeItemInformation/ isTradeItemAVariableUnit	GDSN
GDSNExtension.QuantityOfNextLevelItems	Quantity of next level items	GDSN	CatalogueItemNotification/ CatalogueItem/tradeItem/ nextLowerLevelTradeItemInformation / totalQuantityOfNextLowerLevelTradeItem	IM, dataSource
GDSNExtension.NumberOfLayersContainedInItem	Number of complete layers contained in item	GDSN	tradeItemHierarchyModule/ tradeItemHierarchy/ quantityOfCompleteLayersContainedInATradeItem	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNExtension.NumberOfItemsInALayer	Number of items in a complete layer	GDSN	tradeItemHierarchyModule/ tradeItemHierarchy/ quantityOfTradeItemsContainedInACompleteLayer	GDSN
GDSNExtension.EffectiveDate	Effective date	GDSN	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemSynchronisationDates/ effectiveDateTime	GDSN
GDSNExtension.CanceledDate	Canceled date	GDSN	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemSynchronisationDates/ cancelledDateTime	GDSN
GDSNExtension.GDSNRegistrationDate	GDSN registration date	GDSN	-	dataRecipient
GDSNExtension.GDSNLastModifiedDate	Last modified date	GDSN	-	dataRecipient

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
GDSNExtension.ProductForm	Product form	GDSN	tradelItemDescriptionModule/ tradelItemDescriptionInformation/ tradelItemFormDescription	GDSN
GDSNExtensionLang.GTINName	GTIN name ({ArticleMarketExtensionLangType.LK .Language#Language selectable})	Language-specific GDSN data	-	GDSN
GDSNExtensionUOM.NetContent	Net content ({ArticleMarketExtensionUOMType.LK .UOMType#UOM type selectable})	UOM	tradelItemMeasurementsModule/ tradelItemMeasurements/netContent	GDSN
GDSNExtensionUOM.NetContentUOM	Net content UOM ({ArticleMarketExtensionUOMType.LK .UOMType#UOM type selectable})	UOM		GDSN
ArticlePackaging.PackagingType	Packaging type	Packaging information	packaginginformationModule/ packaging/packagingTypeCode	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticlePackaging.PackagingFeatureCode	Packaging feature code	Packaging information	packaginginformationModule/ packaging/packagingFeatureCode	GDSN
ArticlePackaging.PackagingFunctionCode	Packaging function code	Packaging information	packaginginformationModule/ packaging/packagingFunctionCode	GDSN
ArticlePackaging.PlatformTypeCode	Platform type code	Packaging information	packaginginformationModule/ packaging/platformTypeCode	GDSN
ArticlePackaging.PackagingShapeCode	Packaging shape code	Packaging information	packaginginformationModule/ packaging/packagingShapeCode	GDSN
ArticlePackaging.DoesPackagingHaveWheels	Does packaging have wheels	Packaging information	packaginginformationModule/ packaging/ doesPackagingHaveWheels	GDSN
ArticlePackaging.IsPackagingExemptFromRefuseObligation	Is packaging exempt from refuse obligation	Packaging information	packaginginformationModule/ packaging/ isPackagingExemptFromRefuseObligation	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticlePackaging.PackagingRefuseObligationName	Packaging refuse obligation name	Packaging information	packaginginformationModule/ packaging/ packagingRefuseObligationName	GDSN
ArticlePackaging.PlatformTermsAndConditionsCode	Platform terms and conditions	Packaging information	packaginginformationModule/ packaging/ platformTermsAndConditionsCode	GDSN
ArticlePackagingUOM.UsableProductVolume	Usable product volume ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Packaging information UOM	packaginginformationModule/ packaging/usableProductVolume	GDSN
ArticlePackagingUOM.UsableProductVolumeUOM	Usable product volume UOM ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Packaging information UOM		GDSN
ArticlePackagingUOM.PackagingWeight	Packaging weight ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Packaging information UOM	packaginginformationModule/ packaging/packagingWeight	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticlePackagingUOM.PackagingWeightUOM	Packaging weight UOM ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Packaging information UOM		GDSN
ArticlePackagingMaterial.PackagingMaterialTypeCode	Packaging material type code	Packaging material	packaginginformationModule/ packaging/ packagingPackagingMaterial/ packagingMaterialTypeCode	GDSN
ArticlePackagingMaterialUOM.PackagingMaterialCompositionQuantity	Packaging material composition quantity ({../ArticleSubDomainType.LK.Std_LK_Text250_01#Packaging material type code selectable}, {ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Packaging material UOM	packaginginformationModule/ packaging/ packagingPackagingMaterial/ compositeMaterialDetail/ packagingMaterialCompositionQuantity	GDSN
ArticlePackagingMaterialUOM.PackagingMaterialCompositionQuantityUOM	Packaging material composition quantity UOM ({ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Packaging material UOM		GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleGS1TradeItemIdentificationKey .GS1TradeItemIdentificationType	GS1 trade item identification type	GS1 trade item identification key	tradeItemDataCarrierAndIdentificationModule/ gs1TradeItemIdentificationKey/ gs1TradeItemIdentificationKeyCode	GDSN
ArticleGS1TradeItemIdentificationKey .GS1TradeItemIdentification	GS1 trade item identification	GS1 trade item identification key	tradeItemDataCarrierAndIdentificationModule/ gs1TradeItemIdentificationKey/ gs1TradeItemIdentificationKeyValue	GDSN
ArticleGS1TradeItemIdentificationKey .IsBarcodeSymbologyDerivable	Is barcode symbology derivable	GS1 trade item identification key	tradeItemDataCarrierAndIdentificationModule/ gs1TradeItemIdentificationKey/ isBarcodeDerivable	GDSN
ArticleDeliveryPurchasing.StartAvailabilityDate	Start availability date	Delivery purchasing information	deliveryPurchasingInformationModule /deliveryPurchasingInformation/ startAvailabilityDateTime	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleDeliveryPurchasing.EndAvailabilityDate	End availability date	Delivery purchasing information	deliveryPurchasingInformationModule /deliveryPurchasingInformation/ endAvailabilityDateTime	GDSN
ArticleDistributionDetail.IsDistributionMethodPrimary	Is distribution method primary ({ArticleSubDomainType.LK.Std_LK_Txt250_01#Distribution method code selectable})	Distribution details	deliveryPurchasingInformationModule /deliveryPurchasingInformation/ distributionDetails/ isDistributionMethodPrimary	GDSN
ArticleDistributionDetailUOM.OrderingLeadTime	Ordering lead time ({../ArticleSubDomainType.LK.Std_LK_Txt250_01#Distribution method code selectable})	Distribution details UOM	deliveryPurchasingInformationModule /deliveryPurchasingInformation/ distributionDetails/orderingLeadTime	GDSN
ArticleDistributionDetailUOM.OrderingLeadTimeUOM	Ordering lead time UOM ({../ArticleSubDomainType.LK.Std_LK_Txt250_01#Distribution method code selectable})	Distribution details UOM		GDSN



Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleDataCarrier.DataCarrierFamilyType	Data carrier family type	Data carrier information	tradeItemDataCarrierAndIdentificationModule/dataCarrier/dataCarrierFamilyTypeCode	GDSN
ArticleDataCarrier.DataCarrierType	Data carrier type	Data carrier information	tradeItemDataCarrierAndIdentificationModule/dataCarrier/dataCarrierTypeCode	GDSN
ArticleDataCarrier.ApplicationIdentifierType	Application identifier type	Data carrier information	tradeItemDataCarrierAndIdentificationModule/dataCarrier/applicationIdentifierTypeCode	GDSN
ArticleDataCarrier.DataCarrierPresence	Data carrier presence	Data carrier information	tradeItemDataCarrierAndIdentificationModule/dataCarrier/dataCarrierPresenceCode	GDSN
ArticleTradeItemLifespan.DoesTradeItemHaveAutoReaderTracker	Does trade item have auto reader tracker	Trade item lifespan	tradeItemLifespanModule/tradeItemLifespan/doesTradeItemHaveAutoReaderTracker	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleTradelItemLifespan.MinProductLifespanFromArrival	Minimum product lifespan from arrival	Trade item lifespan	tradelItemLifespanModule/ tradelItemLifespan/ minimumTradelItemLifespanFromTimeOfArrival	GDSN
ArticleTradelItemLifespan.MinProductLifespanFromProduction	Minimum product lifespan from production	Trade item lifespan	tradelItemLifespanModule/ tradelItemLifespan/ minimumTradelItemLifespanFromTimeOfProduction	GDSN
ArticleTradelItemLifespan.OpenedTradelItemLifespan	Opened trade item lifespan	Trade item lifespan	tradelItemLifespanModule/ tradelItemLifespan/ openedTradelItemLifespan	GDSN
ArticleTradelItemLifespan.SupplierSpecifiedMinimumConsumerStorage	Supplier specified minimum consumer storage	Trade item lifespan	tradelItemLifespanModule/ tradelItemLifespan/ supplierSpecifiedMinimumConsumerStorageDays	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleTradeItemHandlingUOM.ClampPressure	Clamp pressure ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Trade item handling UOM	tradeItemHandlingModule/ tradeItemHandlingInformation/ clampPressure	GDSN
ArticleTradeItemHandlingUOM.ClampPressureUOM	Clamp pressure UOM ({ArticleDomainUOMType.LK.UOMType#UOM type selectable})	Trade item handling UOM		GDSN
ArticleTradeItemHandlingLang.HandlingInstructionsDescription	Handling instructions description ({ArticleDomainLangType.LK.Language#Language selectable})	Language-specific data	tradeItemHandlingModule/ tradeItemHandlingInformation/ handlingInstructionsDescription	GDSN
ArticleTradeItemHandlingInstructions.HandlingInstructionCode	Handling instructions code	Handling instructions	tradeItemHandlingModule/ tradeItemHandlingInformation/ handlingInstructionsCodeReference	GDSN
ArticleTradeItemHandlingInstructions.HandlingInstructionCodeAgency	Handling instructions code agency ({ArticleSubDomainType.LK.Std_LK_Text250_01#Handling instruction code selectable})	Handling instructions	tradeItemHandlingModule/ tradeItemHandlingInformation/ handlingInstructionsCodeReferenceAgency	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleTradelItemHandlingStacking.StackingFactorTypeCode	Stacking factor type code	Stacking instructions	tradelItemHandlingModule/ tradeitemHandlingInformation/ tradeitemStacking/ stackingFactorTypeCode	GDSN
ArticleTradelItemHandlingStacking.StackingFactor	Stacking factor ({ArticleSubDomainType.LK.Std_LK_Text250_01#Stacking factor type code selectable})	Stacking instructions	tradelItemHandlingModule/ tradeitemHandlingInformation/ tradeitemStacking/stackingFactor	GDSN
ArticleTradelItemHandlingStackingUOM.StackingWeightMaximum	Stacking weight maximum ({../ ArticleSubDomainType.LK.Std_LK_Text250_01#Stacking factor type code selectable}, {ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Stacking UOM	tradelItemHandlingModule/ tradeitemHandlingInformation/ tradeitemStacking/ stackingWeightMaximum	GDSN
ArticleTradelItemHandlingStackingUOM.StackingWeightMaximumUOM	Stacking weight maximum UOM ({../ ArticleSubDomainType.LK.Std_LK_Text250_01#Stacking factor type code selectable}, {ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Stacking UOM		GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleSustainability.DoesTradItemContainPesticide	Is product a pesticide	Sustainability information	sustainabilityModule/ sustainabilityInformation/ doesTradItemContainPesticide	DSE
ArticleSustainability.IsTradItemRigidPlasticPackagingContainer	Is product RPPC compliant	Sustainability information	sustainabilityModule/ sustainabilityInformation/ isTradItemRigidPlasticPackagingContainer	DSE
ArticleSustainability.IsTradItemROHSCompliant	Is product RoHS compliant	Sustainability information	sustainabilityModule/ sustainabilityInformation/ isTradItemROHSCompliant	DSE
ArticleSustainability.PostConsumerRecycledContentPercentage	Post consumer recycled material [%]	Sustainability information	sustainabilityModule/ sustainabilityInformation/ postConsumerRecycledContentPercentage	DSE

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleSustainability.RenewablePlantBasedPlasticComponentsPercent	Plant based plastic components [%]	Sustainability information	sustainabilityModule/ sustainabilityInformation/ renewablePlantBasedPlasticComponentsPercent	DSE
ArticleSustainability.ROHSComplianceFailureMaterial	Materials that fail RoHS compliance	Sustainability information	sustainabilityModule/ sustainabilityInformation/ roHSComplianceFailureMaterial	DSE
ArticleSustainability.TotalRecyclableContentPercentage	Totaled recycled content [%]	Sustainability information	sustainabilityModule/ sustainabilityInformation/ totalRecyclableContentPercentage	DSE
ArticleSustainability.TradeItemSustainabilityFeatureCode	Sustainability feature code	Sustainability information	sustainabilityModule/ sustainabilityInformation/ tradeItemSustainabilityFeatureCode	DSE
ArticleDistributionDetail.DeliveryFrequencyCode	Delivery frequency code ({ArticleSubDomainType.LK.Std_LK_T ext250_01#Distribution method code selectable})	Distribution details	deliveryPurchasingInformationModule/ deliveryPurchasingInformation/ distributionDetails/ deliveryFrequencyCode	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleHealthCare.CannabisCBDType Code	Cannabis CBD Type Code	Health related information	healthRelatedInformationModule/ healthRelatedInformation/ cannabisCBDTypeCode	GDSN
ArticleDietList.IsDietTypeMarkedOnPackage	Is diet type marked on package ({ArticleSubDomainType.LK.Std_LK_Text250_01#Diet type code selectable})	Diets	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ dietInformationModule/ dietInformation/dietTypeInfoInformation/ isDietTypeMarkedOnPackage	FAB
ArticleNutrientListQuantity.ExpressedAsPartOf	Expressed as part of ({../ ArticleSubDomainType.LK.Std_LK_Text250_01#Nutrient type selectable}, {ArticleSubDomainUOMType.LK.Std_LK_Text100_01#Nutrient basis quantity type selectable}, {ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Nutrient list quantity	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ nutritionalInformationModule/ nutrientHeader/nutrientDetail/ expressedAsPartOf	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleCheese.CheeseMilkAcquisitionTypeCode	Cheese milk acquisition type	Cheese information	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ cheeseInformation/ cheeseMilkAcquisitionTypeCode	FAB
ArticleCheese.FatPercentageInDryMatterMeasurementPrecisionCode	Fat percentage in dry matter measurement precision code	Cheese information	dairyFishMeatPoultryItemModule/ dairyFishMeatPoultryInformation/ fatPercentageInDryMatterMeasurementPrecisionCode	FAB
ArticleIngredientStatement.IngredientStatement	Ingredient statement	Ingredient statements	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ foodAndBeverageIngredientModule/ ingredientStatement	FAB
ArticleIngredientStatement.Sequence	Sequence	Ingredient statements	CatalogueItemNotification/ CatalogueItem/tradeItem/ tradeItemComponents/ foodAndBeverageIngredientModule/ ingredientStatement@sequenceNumber	FAB



Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleNutritionalClaim.ClaimMarkedOnPackage	Claim marked on package	Nutritional claims	nutritionalInformationModule/ nutritionalClaimDetail/ claimMarkedOnPackage	FAB
ArticleNutritionalProgram.NutritionalProgramCode	Nutritional program	Nutritional program	healthRelatedInformationModule/ healthRelatedInformation/ nutritionalProgram/ nutritionalProgramCode	FAB
ArticleNutritionalProgram.NutritionalScore	Nutritional score ({ArticleSubDomainType.LK.Std_LK_T ext250_01#Nutritional program selectable})	Nutritional program	healthRelatedInformationModule/ healthRelatedInformation/ nutritionalProgram/nutritionalScore	FAB
ArticleNutritionalProgram.NutritionalValue	Nutritional value ({ArticleSubDomainType.LK.Std_LK_T ext250_01#Nutritional program selectable})	Nutritional program	healthRelatedInformationModule/ healthRelatedInformation/ nutritionalProgram/nutritionalValue	FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleNutritionalProgramLang.NutritionalProgramDetail	Nutritional program detail ({../ArticleSubDomainType.LK.Std_LK_Text250_01#Nutritional program selectable}, {ArticleSubDomainLangType.LK.Language#Language selectable})	Language-specific data		
ArticleNutritionalProgramUOM.NutritionalProgramIngredientTypeCode	Nutritional program ingredient type	Nutritional program UOM	healthRelatedInformationModule/ healthRelatedInformation/ nutritionalProgram/ nutritionalProgramIngredients/ nutritionalProgramIngredientTypeCode	FAB
ArticleNutritionalProgramUOM.NutritionalProgramIngredientUnit	Nutritional program ingredient unit ({../ArticleSubDomainType.LK.Std_LK_Text250_01#Nutritional program selectable}, {ArticleSubDomainUOMType.LK.Std_LK_Text100_01#Nutritional program ingredient type selectable}, {ArticleSubDomainUOMType.LK.UOMType#UOM type selectable})	Nutritional program UOM		FAB

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleNutritionalProgramUOM.NutritionalProgramIngredientValue	Nutritional program ingredient value ({../ ArticleSubDomainType.LK.Std_LK_Text250_01#Nutritional program selectable}, {ArticleSubDomainUOMType.LK.Std_LK_Text100_01#Nutritional program ingredient type selectable}, {ArticleSubDomainUOMType.LK.UOM Type#UOM type selectable})	Nutritional program UOM	healthRelatedInformationModule/ healthRelatedInformation/ nutritionalProgram/ nutritionalProgramIngredients/ nutritionalProgramIngredientMeasure ment	FAB
GDSNCustomerSpecificExtension.NumberOfItemsInPalletLayer	Number of items per pallet layer	Customer-specific GDSN data	tradeItemHierarchyModule/ tradeItemHierarchy/ quantityOfTradeItemsPerPalletLayer	GDSN
GDSNCustomerSpecificExtension.NumberOfItemsPerPallet	Number of items per pallet	Customer-specific GDSN data	tradeItemHierarchyModule/ tradeItemHierarchy/ quantityOfTradeItemsPerPallet	GDSN
GDSNCustomerSpecificExtension.NumberOfLayersPerPallet	Number of layers per pallet	Customer-specific GDSN data	tradeItemHierarchyModule/ tradeItemHierarchy/ quantityOfLayersPerPallet	GDSN

Product 360 Identifier	Display name (English)	Module (English)	GDSN XML name	Scope
ArticleIngredientLang.IngredientStatement	Ingredient statement, computed ({ArticleIngredientLangType.LK.Language#Language selectable})	Language-specific data	-	FAB
GDSNTargetMarketExtension.IsUnitOfUse	Is unit of use	Target market specific GDSN data	catalogueItemNotification/ catalogueItem/tradeItem/ isTradeItemUnitOfUse	GDSN
GDSNTargetMarketExtensionUOM.Diameter	Diameter ({ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable})	UOM	tradeItemMeasurementsModule/ tradeItemMeasurements/diameter	GDSN
GDSNTargetMarketExtensionUOM.DiameterUOM	Diameter UOM ({ArticleMarketExtensionUOMType.LK.UOMType#UOM type selectable})	UOM		GDSN

## 2 Informatica BPM Accelerator

The Informatica BPM Accelerator Package should give some examples helping to start working with BPM in Product 360.

- [Required Workflows](#) (see page 276)
- [Step Workflow](#) (see page 290)
  - [Installation](#) (see page 291)
  - [Configuration](#) (see page 292)
    - [URN mappings](#) (see page 292)
    - [StepWorkflowMap.xml](#) (see page 294)
    - [StepWorkflow.xml](#) (see page 295)
    - [Comments.xml](#) (see page 298)
    - [Email](#) (see page 300)
    - [Google Vision AIproject](#) (see page 302)
  - [Examples](#) (see page 302)
    - [Default workflows](#) (see page 305)
    - [Example 1 \(1 Task\)](#) (see page 312)
    - [Example 2 \(2 Tasks\)](#) (see page 316)
    - [Example 3 \(2 Tasks with DQ checks\)](#) (see page 320)
    - [Example 4 \(5 Tasks with DQ checks, parallel and sequential\)](#) (see page 324)
    - [Example 5 \(extending example 4 with an approval step\)](#) (see page 332)
    - [Example 6 \(2 tasks \(incl. 1 supplier task\) + modifying of fields\)](#) (see page 341)
    - [Example 7 \(2 Tasks + modifying of fields + approval step\)](#) (see page 351)
    - [Example 8 \(2 Tasks modifying of fields approval step merge\)](#) (see page 360)
    - [Example 9 \(Classic Tasks\)](#) (see page 366)
    - [Example 10 \(1 Task\)](#) (see page 375)
    - [Example 11 \(1 Task with DQ checks\)](#) (see page 380)
    - [Example 12 \(2 Triggers and Merge with Results\)](#) (see page 387)
    - [Example 13 \(Workflow starts another workflow\)](#) (see page 397)
    - [Example 14 \(Export items\)](#) (see page 402)
    - [Example 15 \(1 Task with DQ check and comments\)](#) (see page 408)
    - [Example 16 \(Approval task with comments\)](#) (see page 413)
    - [StepWorkflow Sanity](#) (see page 421)
  - [Steps](#) (see page 424)
    - [ExecuteDqBatch-Process](#) (see page 424)
    - [GVisionAi-Process](#) (see page 426)
    - [GVisionSetBooleanState-Process](#) (see page 428)
    - [GVisionTransferFieldValue-DQ](#) (see page 429)
    - [ItemsInWorkflowTasks-Process](#) (see page 431)
- [Creating MQ based workflows](#) (see page 432) — This guide describes how to design ActiveVOS workflows that leverage the new queue based communication mode between P360 and ActiveVOS
- [DAM Workflows](#) (see page 444)
  - [DAM Configurator](#) (see page 444)
  - [DAM Event Listener](#) (see page 459)
  - [DAM Hotfolder](#) (see page 471)
- "Required Workflows" contains workflows which are mandatory for working with BPM in the context of Product 360.
- "Step Workflow" contains a best practice example which allows you easily create new workflows for Product 360 based on XML files.

- The chapter "Creating MQ based workflows" describes how to design BPM (ActiveVOS) workflows that leverage the new queue based communication mode between Product 360 and BPM (ActiveVOS).

## 2.1 Resources

Filename	Belongs to	Description
P360_JMS_Core.bpr	Required Workflows	This BPM workflow is mandatory to support the Active MQ communication between Product 360 server and BPM server.
P360_JMS_Core.zip	Required Workflows	This zip file contains the project resources for the BPM Designer of the above workflow.
P360_BPM_Management.bpr	Required Workflows	This BPM workflow is mandatory to terminate workflows.
P360_BPM_Management.zip	Required Workflows	This zip file contains the project resources for the BPM Designer of the above workflow.
InfNextSteps.bpr	Step Workflow	This bpr file contains mandatory bpel workflows like DQchecks, Mergeltems,...
InfNextSteps.zip	Step Workflow	This zip file contains the project resources for the BPM Designer of the above workflow.
InfResources.bpr	Step Workflow	This bpr file contains mandatory resources for the Step Workflow like images etc.
InfResources.zip	Step Workflow	This zip file contains the project resources for the BPM Designer of the above workflow.

Filename	Belongs to	Description
StepWorkflow.bpr	Step Workflow	This BPM workflow contains the best practice workflow "StepWorkflow". For more information read the chapter "Step Workflow".
StepWorkflow.zip	Step Workflow	This zip file contains the project resources for the BPM Designer of the above workflow.
StepWorkflowExamples.bpr	Step Workflow	This BPM workflow contains example files for the StepWorkflow.
StepWorkflowExamples.zip	Step Workflow	This zip file contains the project resources for the BPM Designer of the above examples.
P360_JMS_Demo.zip	Creating MQ based workflows	This zip file contains a sample project for the BPM Designer to demonstrate the Queue communication between the servers.
P360_JMS_Demo_DataQuality.zip	Creating MQ based workflows	This zip file contains a sample project for the BPM Designer to demonstrate the Queue communication with DataQuality execution between the servers.
DAM_Hotfolder.bpr	<a href="#">DAM Hotfolder (see page 471)</a>	This BPM workflow contains a Digital Asset Management hotfolder for Product 360 Cloud Edition customers
DAM_Hotfolder.zip	<a href="#">DAM Hotfolder (see page 471)</a>	This zip file contains the project resources for the BPM Designer of the above workflow.

Filename	Belongs to	Description
DAM_Tools.bpr	<a href="#">DAM Event Listener (see page 459)</a> , <a href="#">DAM Configurator (see page 444)</a>	This BPM workflow contains tools for Digital Asset Management like creating derivatives and configuring DAM settings.
DAM_Tools.zip	<a href="#">DAM Event Listener (see page 459)</a> , <a href="#">DAM Configurator (see page 444)</a>	This zip file contains the project resources for the BPM Designer of the above workflow.
DAM_Resources.bpr	<a href="#">DAM Event Listener (see page 459)</a> , <a href="#">DAM Configurator (see page 444)</a>	This BPM workflow contains xml files to configure the DAM behaviour.
DAM_Resources.zip	<a href="#">DAM Event Listener (see page 459)</a> , <a href="#">DAM Configurator (see page 444)</a>	This zip file contains the project resources for the BPM Designer of the above workflow.

## 2.2 Required Workflows

It is mandatory to deploy these 2 workflows to ensure an error-free operation of the components Product 360 Server and BPM server.

- The Business Process Archive *P360\_BPM\_Management.bpr* contains a workflow to terminate workflow process instances provided as parameter. This workflow is needed for the Product 360 "Terminate workflow" functionality and therefore has to be deployed to the Informatica BPM Server. This action is available as workflow task action in the Product 360 web UI as well as in the Desktop client.
- The Business Process Archive *P360\_JMS\_Core.bpr* contains the workflow which consumes an Active Message Queue and routes the message to the corresponding workflow in the BPM server.

### Info

Since there is a dependency between these 2 workflows it is necessary that you the deploy the workflows in the following order:

1. *P360\_JMS\_Core.bpr*
2. *P360\_BPM\_Management.bp*

### 2.2.1 Configuration and Installation of BPM

- [Informatica BPM Installation \(see page 277\)](#)
  - [Installation of the Informatica BPM service \(see page 277\)](#)
  - [Webserver and Java \(see page 278\)](#)



- Adjusting the webserver to support non SSL port (see page 278)
- Integrated Security (see page 279)
  - Configuration during installation (see page 279)
  - Re-configuration of already installed server (see page 280)
- Installation of the required default workflows (see page 280)
- Configure Dispatch Services (see page 283)
- Preparing Informatica BPM service for JMS based communication (see page 286)
  - Download additional library (see page 286)
    - ActiveMQ client library (see page 286)
  - Setup messaging service within Informatica BPM (see page 286)
    - Detail configuration of the message manager (see page 287)
      - JMS Messaging Configuration (see page 287)
      - Initial Context Properties (see page 288)
      - Queues & Listeners (see page 289)
  - Verifying the configuration (see page 290)

## 2.2.2 Informatica BPM Installation

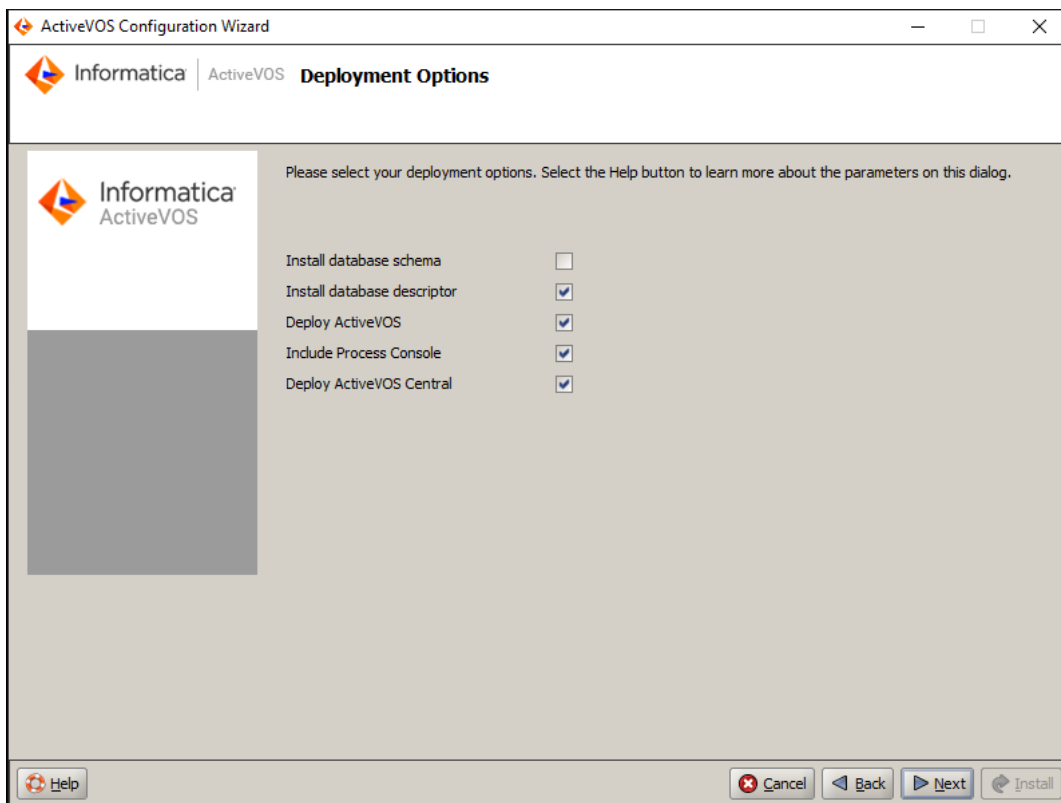
### 2.2.2.1 Installation of the Informatica BPM service

For the installation of the *Informatica BPM service* it is recommended to follow the official installation guide at <https://docs.informatica.com/process-automation/informatica-activevos/current-version.html>.

In the *Prerequisites* chapter <https://docs.informatica.com/process-automation/informatica-activevos/current-version/4-----server-installation--configuration--and-deployment/apache-tomcat/general-information/prerequisites.html> you can find the information about database servers supported for ActiveVOS and additional information about database drivers which are supported. The drivers themselves are not included in the installation package of ActiveVOS and should be get from database vendor.

In the past there were some issues occurred during installation of ActiveVOS if the database schemas have been already created before installation.

So please remember to uncheck "*Install database schema*" if empty DB is already created and to check the "*Install database schema*" option if database schema should be created during installation of ActiveVOS.



#### 2.2.2.2 Webserver and Java

A good start is to use the integration with Apache Tomcat as container where *Informatica BPM service* will be deployed to.

We recommend to use a container version equal or newer to Apache Tomcat 8.5.51.

**i** Make sure to use Java 8 as runtime for the *Informatica BPM service* by setting the necessary environment variables for example.

Adjusting the webserver to support non SSL port

The Informatica BPM web endpoint does enforce SSL by default. To disable that behavior it is possible to set a JVM property:

```
-Dae.web.filter.https.force=false
```

### 2.2.2.3 Integrated Security

It's possible to configure the "Informatica BPM" Server to use integrated Security for MS SQL Server connection. In this case the configuration files do not contain sensitive security data like database user and password of the database user.

The configuration can be made during the installation and initial configuration of "Informatica BPM" Server and as a post-configuration for already existing installations.

Configuration during installation

In this case it's better to use the silent installation and configuration mode

1. Create a Windows service user which will be used to execute BPM Server. (e.g. INFA\bpm-service)
2. Create the same user as MSSQL Server user and configure this user to use Windows Authentication (INFA\bpm-service)
3. Create manually a new ActiveVOS database and configure the owner of this database to newly created user (INFA\bpm-service).
4. Install the Webserver (in our case it's Apache Tomcat) and copy the SQL Server driver class and the additional dll (*sqljdbc42.jar* and *sqljdbc\_auth.dll*) in the *lib* folder of Tomcat
5. Adapt the *service.bat* to use tomcat lib folder additional to a *java.lib.path* like this: **--JvmOptions "...; ...;-Djava.library.path=%CATALINA\_HOME%\lib"**
6. Extract the installation and configuration tool for "Informatica BPM".  
Go to the `<installation_tool>\server-enterprise\tomcat_config\bin` folder and adapt the *install.properties* for server configuration. See example for properties relevant to integrated security. The content of username and password is mandatory but not relevant for the connection. So you can use any signs.

#### Properties

```
jdbc.database.driver.class=com.microsoft.sqlserver.jdbc.SQLServerDriver
jdbc.database.driver.jar=<tomcat_path>\lib\sqljdbc42.jar
jdbc.database.url=jdbc:sqlserver://
<sqlserver_host>;databaseName\=Active_VOS;integratedSecurity=true;
jdbc.database.name=Active_VOS
jdbc.database.password=xxxx
jdbc.database.username=xxxx
```

7. Go to the `<installation_tool>\server-enterprise\tomcat_config\bin` folder and adapt the *config\_deploy.bat* to use tomcat lib folder additional to a *java.lib.path* like this:

#### Configuration

```
"<jdk_path>\bin\java" -Xms128m -Xmx512m -Djava.library.path="<tomcat_path>\lib"
-jar config.jar %1
```

8. Open the command window and execute the *config\_deploy.bat* in **silent** mode.
9. Install the Tomcat service using *service.bat install*

10. Configure the "Log on" for this service to use the BPM service account (INFA\bpm-service)
11. Start the service and call the ActiveVOS Console.

Re-configuration of already installed server

Following steps can be made to re-configure the existing BPM Server installation to use the integrated security for database connection:

1. Stop and uninstall the Tomcat service for BPM Server. Use *service.bat uninstall [ tomcat service name ]*
2. Create a Windows service user which will be used to execute BPM Server. (e.g. INFA\bpm-service)
3. Create the same user as MSSQL Server user and configure this user to use Windows Authentication (INFA\bpm-service)
4. Configure the owner of the existing ActiveVOS database to newly created user (INFA\bpm-service).
5. Go to the webserver (Tomcat) installation ({install\_dir}/apache-tomcat/conf/Catalina) and to the BPM server ({install\_dir}/server-enterprise/tomcat\_config/conf) and adapt activevos.xml and active-bpel.xml in both places to use integrated security (s. example). The content of username and password is mandatory but not relevant for the connection. So you can use any signs like in example.

#### Configuration

```
<Context displayName="ActiveBPEL Enterprise Tomcat Database context" path="/
active-bpel">
  <Resource name="jdbc/ActiveVOS" auth="Container" type="javax.sql.DataSource"
    maxActive="100"
    maxIdle="10"
    maxWait="1000"
    username="xxxx"
    password="xxxx"
    driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
    url="jdbc:sqlserver://
server.informatica.com;databaseName=Active_VOS;integratedSecurity=true;"/>
</Context>
```

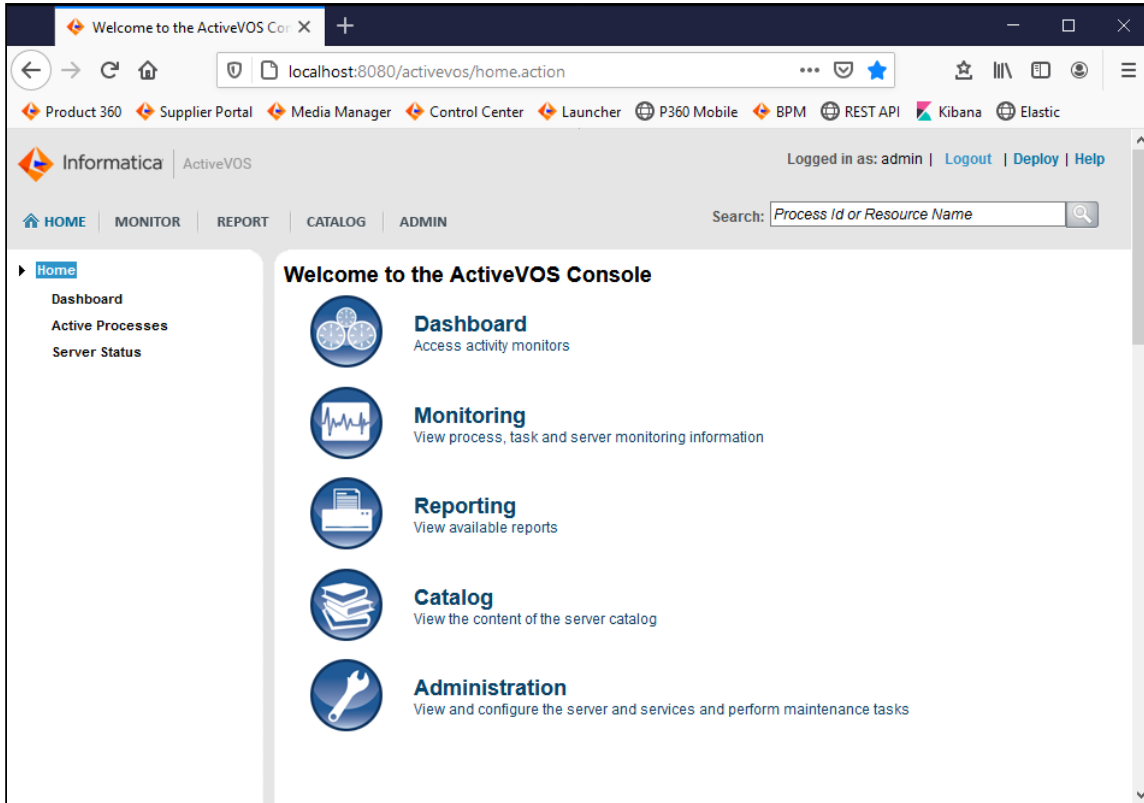
6. Copy the driver class and the additional dll (*sqljdbc42.jar* and *sqljdbc\_auth.dll*) in the lib folder of Tomcat
7. Adapt the *service.bat* to use tomcat lib folder additional to a java.lib.path like this: **--JvmOptions**  
"...; ...;-Djava.library.path=%CATALINA\_HOME%\lib"
8. Install the Tomcat service using *service.bat install*
9. Configure the "Log on" for this service to use the BPM service account (INFA\bpm-service)
10. Start the service and call the ActiveVOS Console.

## 2.2.3 Installation of the required default workflows

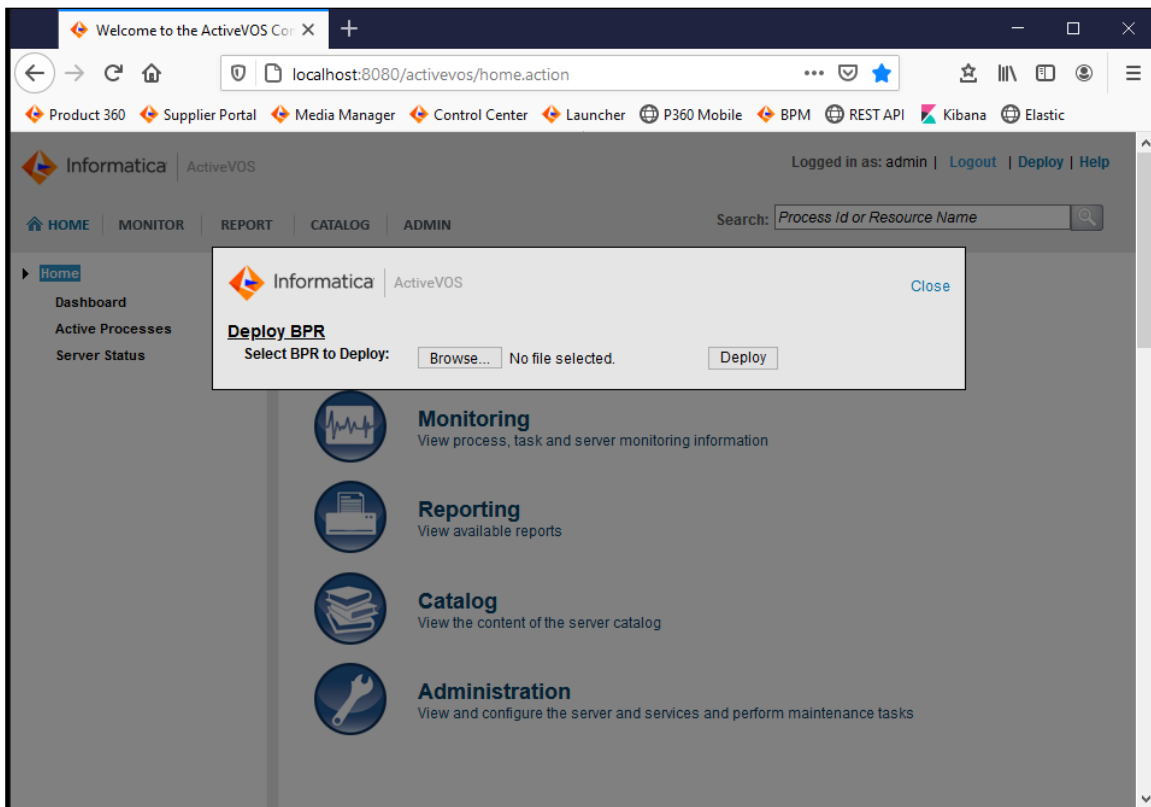
After installing the *Informatica BPM* service the required default workflows have to be deployed to the BPM instance.

The workflows are provided as deployable Business Process Archives: **P360\_JMS\_Core.bpr** and **P360\_BPM\_Management.bpr** in the Accelerator package <P360 version>\_InformaticaBPM of the P360 release.

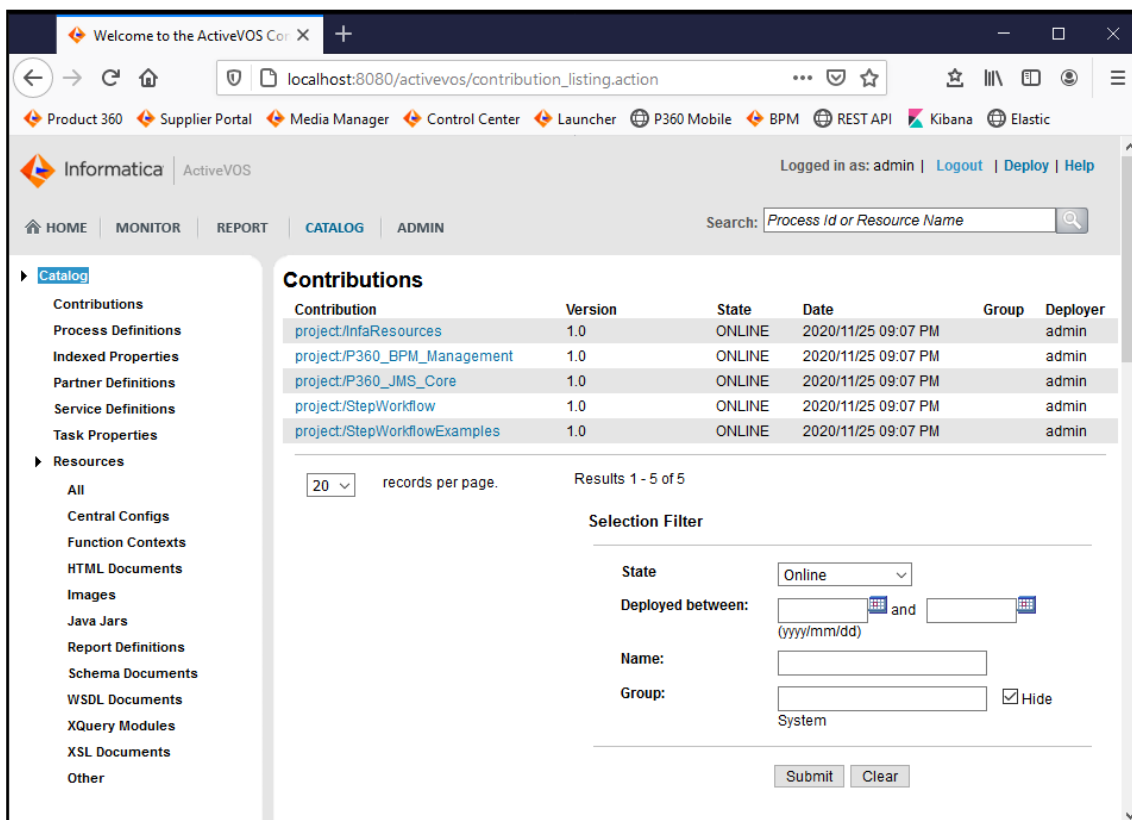
To deploy the workflows make sure the *Informatica BPM service* is up and running: Open your web browser and open the management console located at <http://your-bpm-server:8080/activevos>. You should see the start page of the management console.



To deploy the workflows open the deploy dialog by clicking on the "Deploy" button in the upper right corner and upload the Business Process Archives.



After the upload and deployment has been processed successfully you can check the availability of the P360 default workflows by navigating to *Catalog/Contributions* where you should now see the deployed projects containing the P360 default workflows in the list.



The screenshot shows the Informatica ActiveVOS web interface. The top navigation bar includes links for Product 360, Supplier Portal, Media Manager, Control Center, Launcher, P360 Mobile, BPM, REST API, Kibana, and Elastic. The user is logged in as 'admin'. The main menu on the left includes Catalog, Monitor, Report, and Admin. The 'Catalog' section is active, showing a list of contributions under the 'Contributions' tab. The table lists five contributions, all with a version of 1.0 and a state of 'ONLINE'. Below the table, there is a 'Selection Filter' section with dropdowns for State (set to 'Online'), Deployed between (with date pickers), Name, and Group (set to 'System'). A 'Hide' checkbox is also present. The bottom of the filter section has 'Submit' and 'Clear' buttons.

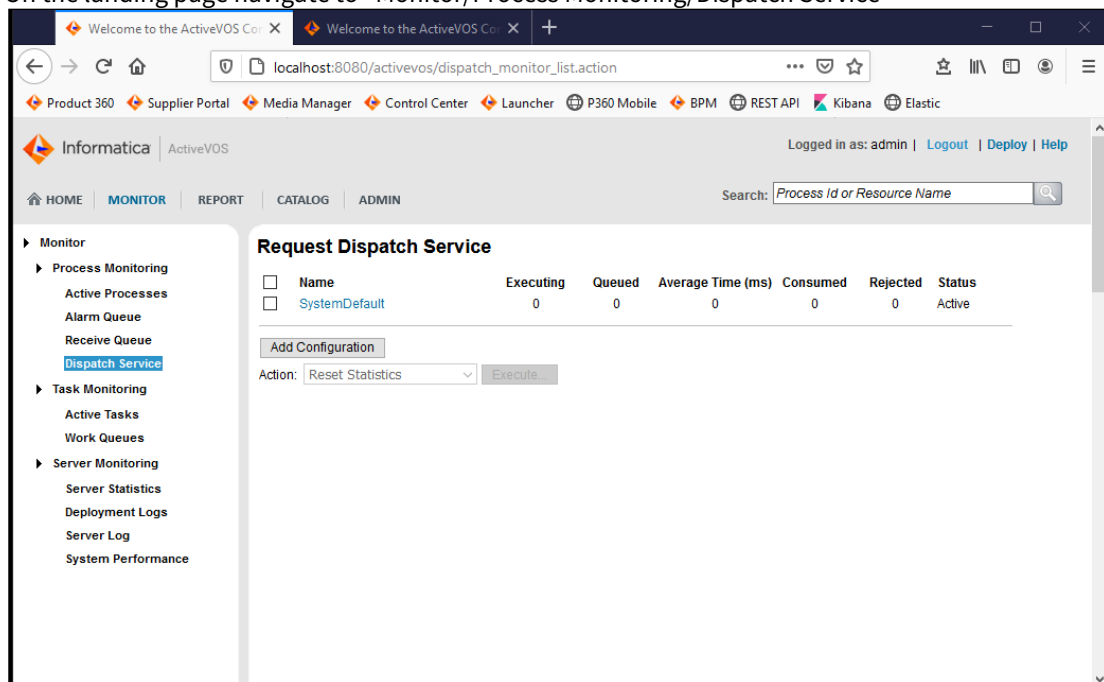
Contribution	Version	State	Date	Group	Deployer
project/InfResources	1.0	ONLINE	2020/11/25 09:07 PM		admin
project/P360_BPM_Management	1.0	ONLINE	2020/11/25 09:07 PM		admin
project/P360_JMS_Core	1.0	ONLINE	2020/11/25 09:07 PM		admin
project/StepWorkflow	1.0	ONLINE	2020/11/25 09:07 PM		admin
project/StepWorkflowExamples	1.0	ONLINE	2020/11/25 09:07 PM		admin

## 2.2.4 Configure Dispatch Services

The dispatch services have to be configured in the console of BPM:

1. Open a web browser and navigate to <http://your-bpm-server:8080/activevos> and login using your user credentials

- On the landing page navigate to "Monitor/Process Monitoring/Dispatch Service"



- Click on SystemDefault and update the values

**Dispatch Configuration**

Name: SystemDefault

Max Concurrent: 150

Max In-Memory: 20000

Max Queued: 20000

Timeout (seconds): 300

Persistent: ☒


At runtime, the dispatch configuration used for a particular request is chosen based on matching the configuration name in the following order of precedence: Service Name, Process Group, Tenant, System Default

Dispatch Configuration	
Name	SystemDefault
Max Concurrent	150
Max in Memory	20000



Dispatch Configuration	
Max Queued	20000
Timeout (seconds)	300
Persistent	true

4. Click on Update Configuration
5. Click on Add Configuration and set the values


Informatica | ActiveVOS

**Dispatch Configuration**

Name   
Max Concurrent   
Max In-Memory   
Max Queued   
Timeout (seconds)   
Persistent ☒

At runtime, the dispatch configuration used for a particular request is chosen based on matching the configuration name in the following order of precedence: Service Name, Process Group, Tenant, System Default

Dispatch Configuration	
Name	P360RouterService
Max Concurrent	100
Max in Memory	5000
Max Queued	5000
Timeout (seconds)	300

Dispatch Configuration	
Persistent	true

6. Click on Update Configuration

## 2.2.5 Preparing Informatica BPM service for JMS based communication

The communication between Informatica Product 360 and Informatica BPM service can be switched to an asynchronous message based communication. To enable this capability of Informatica BPM service some post installation steps have to be performed.

### 2.2.5.1 Download additional library

ActiveMQ client library

To make Informatica BPM service ready for working with messaging an additional library has to be installed.

- Download <https://repo1.maven.org/maven2/org/apache/activemq/activemq-all/5.15.9/activemq-all-5.15.9.jar>
- Put the downloaded file `activemq-all-5.15.9.jar` into the *lib* folder of the Tomcat container
- Ensure the JDK version is  $\geq 1.8$  as the active mq lib requires that

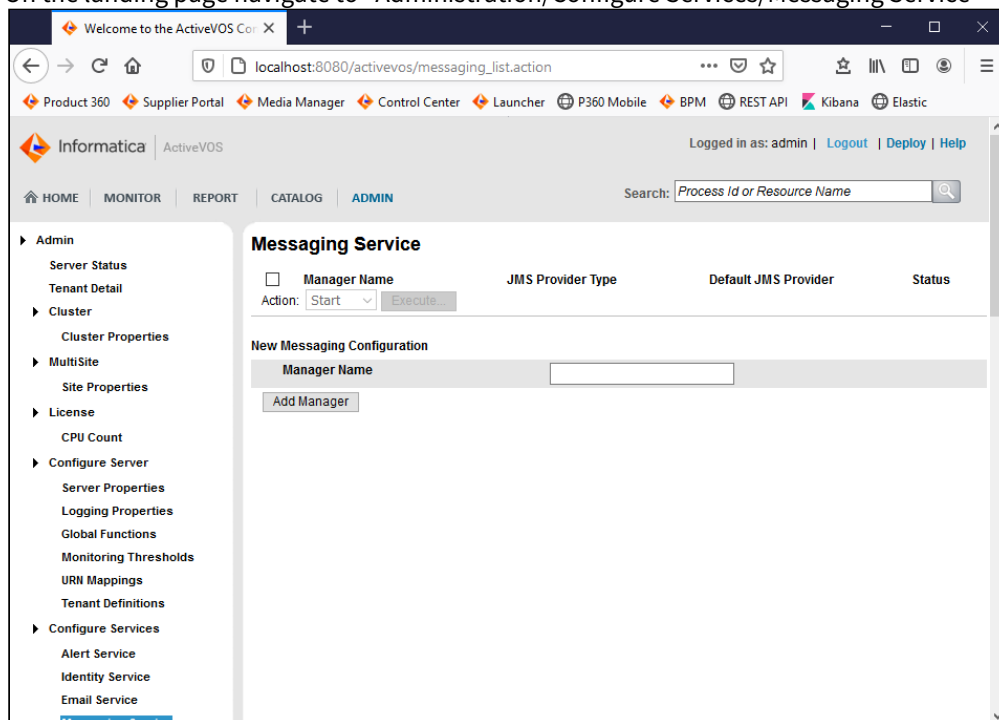
Make sure to restart the Informatica BPM service so that the changes take effect.

### 2.2.5.2 Setup messaging service within Informatica BPM

To be able to use the asynchronous message based communication between Informatica Product 360 and Informatica BPM service some basic configuration within Informatica BPM has to be set up. This configuration will be done using the Informatica BPM management console.

1. Open a web browser and navigate to `http://your-bpm-server:8080/activevos` and login using your user credentials

2. On the landing page navigate to "Administration/Configure Services/Messaging Service"



3. Create a new "Manager" entering "ActiveMQ" as name and clicking on the "Add Manager" button
4. A new entry with name "ActiveMQ" will appear in the list of manager, to configure the manager in detail click on the manager entry

Detail configuration of the message manager

To finish the configuration of the manager you have to add the following values as minimum configuration

JMS Messaging Configuration

Property name	Property value
Default JMS Provider	enabled
JMS Provider Type	Other JMS
Connection Factory Name	ConnectionFactory
Connection User	The username to connect to the ActiveMQ message broker

Property name	Property value
Connection Password	The password to connect to the ActiveMQ message broker
Send Empty Credentials	disabled
Maximum Total Connections	-1
Maximum Free Connections	15
Delivery Mode	Persistent
Time To Live (ms)	0
Priority (int)	0
Reconnect Interval (ms)	

#### Initial Context Properties

Property name	Property value
java.naming.provider.url	tcp://your-activemq-server:61616? jms.redeliveryPolicy.maximumRedeliveries=-1 <sup>1</sup>
java.naming.factory.initial	org.apache.activemq.jndi.ActiveMQInitialContextFactory
queue.JNDI_P360_BPM	P360_BPM
queue.JNDI_P360_SERVICE_API	P360_SERVICE_API

Property name	Property value
queue.JNDI_P360_BATCH_API	P360_BATCHAPI
queue.JNDI_P360_BPM_RESPONSE	P360_BPM_RESPONSE

<sup>1</sup> if using ssl connection make sure to import the corresponding certificates into your JREs trust store.

#### Queues & Listeners

Add a new entry in section Queues & Listeners, not mentioned properties need to stay with their default value

Property name	Property value
Name	P360_BPM_LISTENER
JNDI Location	JNDI_P360_BPM
Listener Class	com.activeee.rt.mom.jms.transport.AeJmsBpelListener
Listener Count	15
Default Service	P360RouterService

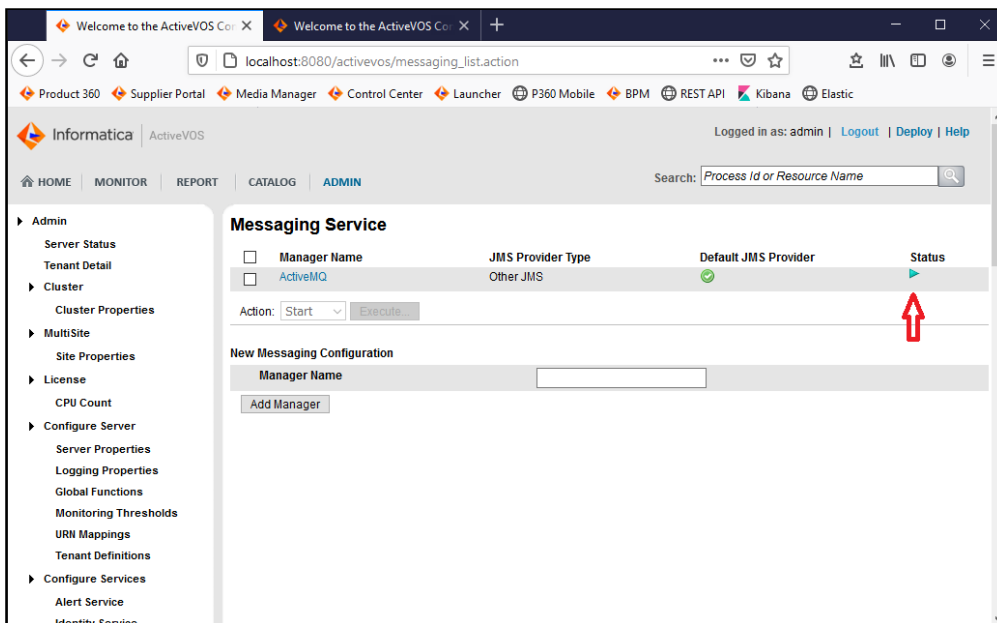
Add a new entry in section Queues & Listeners, not mentioned properties need to stay with their default value

Property name	Property value
Name	P360_BPM_RESPONSE_LISTENER
JNDI Location	JNDI_P360_BPM_RESPONSE

Property name	Property value
Listener Class	com.activee.rt.mom.jms.transport.AeJmsBpellListener
Listener Count	15
Default Service	P360RouterService

Verifying the configuration

After the manager configuration has been validated and saved you can go back to the Messaging Service overview page. Your manager should now appear in the list with status running.



## 2.3 Step Workflow

### 2.3.1 Preface

The StepWorkflow project is a workflow with the following objectives:

- You don't need to be a BPM (AVOS) expert to design workflows for MDM-Product 360 business use cases
- Development of new workflows is more structured (in comparison to previous developments)
- Easier to troubleshoot
- Less issues because of tested re-usable code

- Reduce the amount of running processes on the BPM server
- Enable batching

## 2.3.2 Table of Contents

- [Installation](#) (see page 291)
- [Configuration](#) (see page 292)
  - [URN mappings](#) (see page 292)
  - [StepWorkflowMap.xml](#) (see page 294)
  - [StepWorkflow.xml](#) (see page 295)
  - [Comments.xml](#) (see page 298)
  - [Email](#) (see page 300)
  - [Google Vision AIproject](#) (see page 302)
- [Examples](#) (see page 302)
  - [Default workflows](#) (see page 305)
  - [Example 1 \(1 Task\)](#) (see page 312)
  - [Example 2 \(2 Tasks\)](#) (see page 316)
  - [Example 3 \(2 Tasks with DQ checks\)](#) (see page 320)
  - [Example 4 \(5 Tasks with DQ checks, parallel and sequential\)](#) (see page 324)
  - [Example 5 \(extending example 4 with an approval step\)](#) (see page 332)
  - [Example 6 \(2 tasks \(incl. 1 supplier task\) + modifying of fields\)](#) (see page 341)
  - [Example 7 \(2 Tasks + modifying of fields + approval step\)](#) (see page 351)
  - [Example 8 \(2 Tasks modifying of fields approval step merge\)](#) (see page 360)
  - [Example 9 \(Classic Tasks\)](#) (see page 366)
  - [Example 10 \(1 Task\)](#) (see page 375)
  - [Example 11 \(1 Task with DQ checks\)](#) (see page 380)
  - [Example 12 \(2 Triggers and Merge with Results\)](#) (see page 387)
  - [Example 13 \(Workflow starts another workflow\)](#) (see page 397)
  - [Example 14 \(Export items\)](#) (see page 402)
  - [Example 15 \(1 Task with DQ check and comments\)](#) (see page 408)
  - [Example 16 \(Approval task with comments\)](#) (see page 413)
  - [StepWorkflow Sanity](#) (see page 421)
- [Steps](#) (see page 424)
  - [ExecuteDqBatch-Process](#) (see page 424)
  - [GVisionAi-Process](#) (see page 426)
  - [GVisionSetBooleanState-Process](#) (see page 428)
  - [GVisionTransferFieldValue-DQ](#) (see page 429)
  - [ItemsInWorkflowTasks-Process](#) (see page 431)

## 2.3.3 Installation

Before you can design your workflows based on the StepWorkflow you have to deploy these resources:

1. InfaResources.bpr
2. StepWorkflow.bpr
3. InfaNextSteps.bpr
4. StepWorkflowExamples.bpr (optional)
5. InfaNextStepsGVisionAI.bpr (optional)

with the BPM (AVOS) console.

Also it is possible that you deploy the workflows with your BPM designer. The resources of the projects are in the zip files:

1. InfaResources.zip
2. StepWorkflow.zip
3. InfaNextSteps.zip
4. StepWorkflowExamples.zip
5. InfaNextStepsGVAI.zip

## 2.3.4 Configuration

### 2.3.4.1 Prerequisites

The StepWorkflow is designed for the queue communication. Therefore Product 360 must also be configured for Queue communications. Please reach out to the Installation and Operation guide for the following areas:

- Configuration of queues in the Product 360 server
- Installation of ActiveVOS with ActiveMQ support and configuration of queues in ActiveVOS (*Note: ActiveVOS version 9.2.4.6 is a prerequisite for queue based interactions between Product 360 and ActiveVOS*)
- Message headers and message formats for the various interactions between Product 360 and ActiveVOS

### 2.3.4.2 StepWorkflow XML files

The StepWorkflow and the StepWorkflowExamples projects containing several xml files in the catalog resources.

- [URN mappings](#) (see page 292)
- [StepWorkflowMap.xml](#) (see page 294)
- [StepWorkflow.xml](#) (see page 295)
- [Comments.xml](#) (see page 298)
- [Email](#) (see page 300)
- [Google Vision AIproject](#) (see page 302)

### 2.3.4.3 URN mappings

The base configuration of the StepWorkflow is defined in the BPM URN mappings. Therefore you have to define the following mappings in the BPM console.

urn:p360.api.username	RestServiceUser	Name of the Product 360 user in which context the requests are fired
urn:p360.api.password	*****	Password of the user above



urn:p360.api.manager	ActiveMQ	Name of the configured messaging service of the BPM server
urn:p360.api.queue	JNDI_P360_SERVICE_API	JNDI name of the queue for the service calls
urn:p360.rest.url	http://p360server:1512	Url of the Product 360 rest services
urn:dq.timeout	PT4H	Timeout for DQ calls
urn:p360.api.data.quality.queue	JNDI_P360_DATA_QUALITY	JNDI name of the queue for the Data Quality calls
urn:p360.api.data.quality.response.queue	bpm_response	Identifier if the response queue for DQ calls (default: bpm)
urn:list.get.timeout	PT4H	Timeout for list calls
urn:merge.delay	PT30S	Delay before starting the merge process
urn:error.task.group	Superuser	Identifier of the usergroup for which a classic task will be created if an error happens within the StepWorkflow. (i.e. Step Id is not defined)
urn:workflow.resource.project	StepWorkflowExamples	Name of the project (AVOS catalog) where the Mapfile is stored. It is highly recommended that you create your own project to avoid overwrite with a new delivery.
urn:p360.web.url	https://mycompany.com	Url to the pim web application

### 2.3.4.4 StepWorkflowMap.xml

The StepWorkflowMap.xml file

This xml file defines the mapping between a Product 360 trigger configuration, Product 360 workflow identifier and a stepworkflow.xml file.

files	Within the files tag are the different file mappings that need to be defined
file	Each business workflow definition needs to be defined in this tag.
path	Path and name of the workflow. (Case-sensitive!)
triggerConfiguration	Name of the Product 360 trigger configuration.
workflowIdentifier	Identifier of the Product 360 workflow.

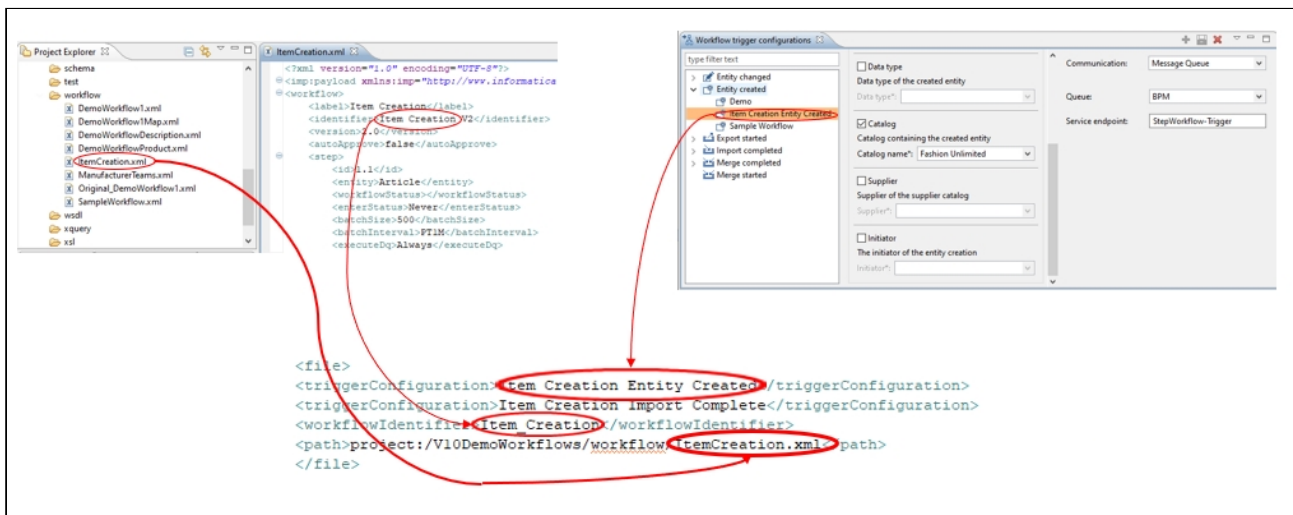
#### StepWorkflowMap

```
<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <files>
    <file>
      <workflowIdentifier>Default_Workflow_Article</workflowIdentifier>
      <path>project:/StepWorkflow/workflow/DefaultArticle.xml</path>
    </file>
    <file>
      <workflowIdentifier>Default_Workflow_Product2G</workflowIdentifier>
      <path>project:/StepWorkflow/workflow/DefaultProduct2G.xml</path>
    </file>
    <file>
      <workflowIdentifier>Default_Workflow_Variant</workflowIdentifier>
      <path>project:/StepWorkflow/workflow/DefaultVariant.xml</path>
    </file>
  </files>
</imp:payload>
```

```

</file>
<file>
  <triggerConfiguration>Example01_MergeCompleted</triggerConfiguration>
  <triggerConfiguration>Example01_EntityChange</triggerConfiguration>
  <triggerConfiguration>Example01_EntityCreate</triggerConfiguration>
  <triggerConfiguration>Example01_ExportStarted</triggerConfiguration>
  <triggerConfiguration>Example01_ImportComplete</triggerConfiguration>
  <workflowIdentifier>Workflow_01</workflowIdentifier>
  <path>project:/StepWorkflowExamples/workflow/Example_01.xml</path>
</file>
<!-- .....-->
</files>
</imp:payload>

```



### 2.3.4.5 StepWorkflow.xml

#### The StepWorkflow.xml file

The separate StepWorkflow.xml files define the different business workflows. In this chapter we will describe, and provide examples for different workflows. If you want to create new xml files within your workflow you have to open the StepWorkflow project in BPM designer and add a new xml file in the workflow folder with the tag `imp:payload` to ensure that this file will be deployed with your workflow. After that, you have to register this new xml file in the StepWorkflowMap.xml file.

#### Description of the StepWorkflow.xml file

```

<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">

```

<!-- This is a description and sample values of the fields in the Workflow Detail files and should not be used directly

You should not use xml files with a lot of comments for live processing. Please remove all comments in your

Workflow Detail files -->

<workflow>

<label>My Stepworkflow</label>

<!-- Name of the workflow that is visible in the UI -->

<identifier>My\_Stepworkflow\_V1</identifier>

<!--Name of the workflow not visible in the UI must be unique thru all workflows -->

<version>1.0</version>

<!--version number of the workflow needs to be incremented by one if a Status, Usergroup, description, decision or

UiTemplate has been changed -->

<updateWorkflowComments>true</updateWorkflowComments>

<!-- This tag decides whether events of this workflow will be logged -->

<step>

<!-- These are the steps to the workflow. They do not have to include entering into a task -->

<id>1.1</id>

<!-- This is the identifier of each step. They must be unique within a workflow.

While generally numeric in nature, can be any string value. -->

<entity>Article</entity>

<!-- This is the entity container that the items will be placed in within the workflow in Product 360. Valid values are:

Article

Variant

Product2G -->

<workflowStatus>StartCE01</workflowStatus>

<!-- This will be the status name within the Product 360 workflow. It will be the second part of the name of the task name in Product 360:

Product 360 task name = [Workflow] - [Status] - [Catalog] -->

<description>Desc 1.1</description>

<!-- Description of the status. It will be displayed in the task in the Product 360 UI -->

<enterStatus>Never</enterStatus>

<!-- The circumstances when an item should enter a status (task):

OnDqResults=Item(s) will enter the task/status on DQ failures or move to the next step on successuls.

Never=This is for steps that do not include a task

Always=Item(s) will always enter a task/status the first time the step is run. Subsequent times are based on the dq results. -->

<batchSize>500</batchSize>

<!-- This is the batch size to use with trigger batching. If the triggers coming from Product 360 have a large number of entities, this can break them down. Batch sizes can make Product 360 requests more efficient -->

<userType>userGroup</userType> <!-- Type of the default assignee valid values:

user

```

        userGroup
        supplier ==> mandatory field: <getField>Article.MainSupplier->Party.Name</
getField> has to be set!-->
        <userName>Standardusers</userName>
        <!--Default assignee of the task. Should be either a Product 360 user or
userGroup -->
        <uiTemplate>Item approve UI</uiTemplate>
        <!-- default UI of the task -->
        <singleChoice>true</singleChoice> <!-- This is for Rejects only and
determines if you can have more than one choice. Should be true or nothing as it
defaults to 'false' -->
        <rejectDecision>
            <id>p360.bpm.reject.TRIGGER:4.1</id> <!-- Always starts with
'p360.bpm.reject.' Can continue with:
                                STEP : Which step is rejected
ex. p360.bpm.reject.STEP:5.1
                                TRIGGER : Which parallel task is
rejected ex. p360.bpm.reject.TRIGGER:4.1
                                SERVICE : Which process is
generated with the reject ex. p360.bpm.reject.TaskByManufacturer-Workflow -->
            <label>Maintain texts</label> <!-- The label of the task being rejected.
-->
        </rejectDecision>
        <executeDq>Always</executeDq>
        <!-- The circumstances when a DQ should be run:
            Always=DQ will be executed whenever this step is run regardless of whether it
was run as a next step or from a trigger.
            OnTrigger=DQ will run only when the process was started from a trigger in
Product 360 and not from a previous step.
            Never=DQ is not used for this step. -->
        <dqService>ExecuteDqBatch-Process</dqService>
        <!-- The service name of the process that will be handle the processing of the
dq or any custom built business rules. -->
        <dqRule>Item image check</dqRule>
        <!-- List any Product 360 DQ rules you want to have run with a step. You can
have multiple entries. -->
        <dqRuleGroup>Item descriptions</dqRuleGroup>
        <!-- List any Product 360 DQ ruleGroups you want to have run with a step. You
can have multiple entries. -->
        <dqChannel>01 Classify</dqChannel>
        <!-- List any Product 360 DQ channels you want to have run with a step. You
can have multiple entries. -->
        <updateFieldDescriptor>Article.CurrentStatus</updateFieldDescriptor>
        <!-- This field allows the step to update a field in Product 360. The field
must be fully qualified. -->
        <updateFieldValue>600</updateFieldValue>
        <!-- This is the value for the update. -->
        <getField>Article.CurrentStatus</getField>
        <!-- List of any fields directly on the entity that will be required. These
entries must be fully qualified. -->
        <nextStep>STEP:3.1</nextStep>

```

```

    <!-- This is the step that the item(s) will move to if dq is successful or
there is no dq.
    It must start with STEP: to move to a step. Otherwise it will be interpreted
as a BPEL service that you can use for custom code.
    In both cases a the item(s) are sent in list using the ItemMap schema. -->
    <dqFailStep>TaskByManufacturer-Workflow</dqFailStep>
    <!-- This is the step that the item(s) move to if dq fails It must start with
STEP: to move to a step.
    Otherwise it will be interpreted as a BPEL service that you can use for
custom code.
    In both cases a the item(s) are sent in list using the ItemMap schema. -->
    <nextStep>StepWorkflow_ParallelTasks-Workflow</nextStep>
    <!-- StepWorkflow_ParallelTasks-Workflow should be used for steps that run in
parallel -->
    <dqFailStep>StepWorkflow_ParallelTasks-Workflow</dqFailStep>
    <!-- StepWorkflow_ParallelTasks-Workflow should be used for steps that run in
parallel -->
    <parallelStep>4.1</parallelStep>
    <!-- This is where you list parallel steps if a step should go to multiple
steps at the same time -->
    <parallelNextStep>5.1</parallelNextStep>
    <!-- If all the parallel steps finish, then the item will move to this step -->
    <dqFailTrigger>StepWorkflow_ParallelTasks-Finish-3</dqFailTrigger>
    <!-- Used to route to the Parallel steps after logic in StepWorkflow process is
finished. -->
    <nextTrigger>StepWorkflow_ParallelTasks-Finish-3</nextTrigger>
    <!-- Used to route to the Parallel steps after logic in StepWorkflow process is
finished. -->
    </step>
  </workflow>
</imp:payload>

```

### 2.3.4.6 Comments.xml

The Comments.xml file

This xml file defines how the the events will be logged when the tag *updateWorkflowComments* is set to true. [Default: false]

commentService	Service to be processed to create the comments.
commentDescriptor	Full qualified field where the comment will be persisted.

commentFormat	<p>Format of the comment. These variables can be used:</p> <ul style="list-style-type: none"> <li>• ##timestamp</li> <li>• ##user</li> <li>• ##action</li> <li>• ##workflow</li> <li>• ##step</li> <li>• ##comment</li> </ul> <p>Example: <b>##timestamp ##user - ##action - ##workflow - ##step -- ##comment</b></p>
timezoneOffset	Time offset between the server time and the time which should be logged.
maximumNumberOfCharacters	Maximum number of characters which can be stored in the field. If the maximum number is reached the oldest comments will be removed.
commentOrder	Order of the comments. Descending means that the newest event is at the top of the comment.

## Comments

```
<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <comments>
    <comment type="StepWorkflow">
      <commentService>UpdateWorkflowComments-Process</commentService>
      <commentDescriptor>ArticleLang.Remarks(9, "Default Channel")</commentDescriptor>
      <commentFormat>##timestamp ##user - ##action - ##workflow - ##step -- ##comment</commentFormat>
      <timezoneOffset>3</timezoneOffset>
      <maximumNumberOfCharacters>10000</maximumNumberOfCharacters>
      <commentOrder>descending</commentOrder>
    </comment>
  </comments>
</imp:payload>
```

The following events are traced:

Event
Task entered
Task finished
Task approved
Task rejected
Merged with results

**Exception: Tracing within parallel tasks**

The "Task enter" event in the parallel workflow steps(see example 4 will not be traced.

#### 2.3.4.7 Email

The Step Workflow framework supports as well sending of mails for example to all users which are assigned to tasks. A prerequisite to send emails is the correct configuration of the BPOM server. See the next screenshot.



The screenshot shows the Informatica ActiveVOS Admin console. The top navigation bar includes links for HOME, MONITOR, REPORT, CATALOG, and ADMIN. The left sidebar lists various administrative tasks under categories like Admin, Cluster, MultiSite, License, Configure Server, and Configure Services. The main content area is titled 'Email Service' and contains the 'Email SMTP Configuration' form. The form includes fields for Host (imap.gmail.com), Port (465), From Address (avos2024@avos.com), Username (mycompany@gmail.com), Password, and Confirm Password. There are also radio buttons for Security (None, SSL, TLS) and buttons for Update and Update and Test.

To send email you can use the step `NotifyUser-Process`. (See Example\_14). This process can be configured via the parameter *emailProfile*. These profiles are a combination of 2 files. First a xml file to configure the mail and a corresponding xsl file to adjust it to your company UX.

Sample configuration files are also part of the `StepWorkflowExamples` project. The xml files are stored in the folder `email-template` and the corresponding xsl files are stored in the folder `xsl`.

The following templates are included:

XML	XSL
/email-template/email_DqChannelFailed.xml	/xsl/DqChannelFailed.xsl
/email-template/email_exportStarted.xml	/xsl/exportStarted.xsl
/email-template/email_jobNotification.xml	/xsl/jobNotification.xsl
/email-template/email_notifyUser.xml	/xsl/notifyUser.xsl
/email-template/email_rejectTask.xml	/xsl/rejectTask.xsl
/email-template/email_sanity.xml	/xsl/sanity.xsl

### 2.3.4.8 Google Vision Alproject

The Step Workflow framework supports as well analysing images with Google Vision AI and persisting the results. To use this feature you have to adjust the resource GVisionAI.xml in the folder ai-settings of your resource project.

#### GVisionAi.xml

```
<aetgt:payload xmlns:aetgt="http://www.informatica.com/schema/ItemMap"
               xmlns:imp="http://www.informatica.com/schema/ItemMap"
               contentType="string">
  <gVisionAi>
    <gVisionAiUrl>https://vision.googleapis.com/v1/images:annotate</gVisionAiUrl>
    <gVisionAiToken>[YOURTOKEN]</gVisionAiToken>
    <escape4windows>0</escape4windows>
    <gVisionAiJsonRequest>{
      "requests": [
        {
          "image": {
            "source": {
              "imageUri": "P360publicUrl"
            }
          },
          "features": [
            P360FeatureBlock
          ]
        }
      ]
    }
  </gVisionAiJsonRequest>
</gVisionAi>
</aetgt:payload>
```

You have to adjust only 2 things inside this file:

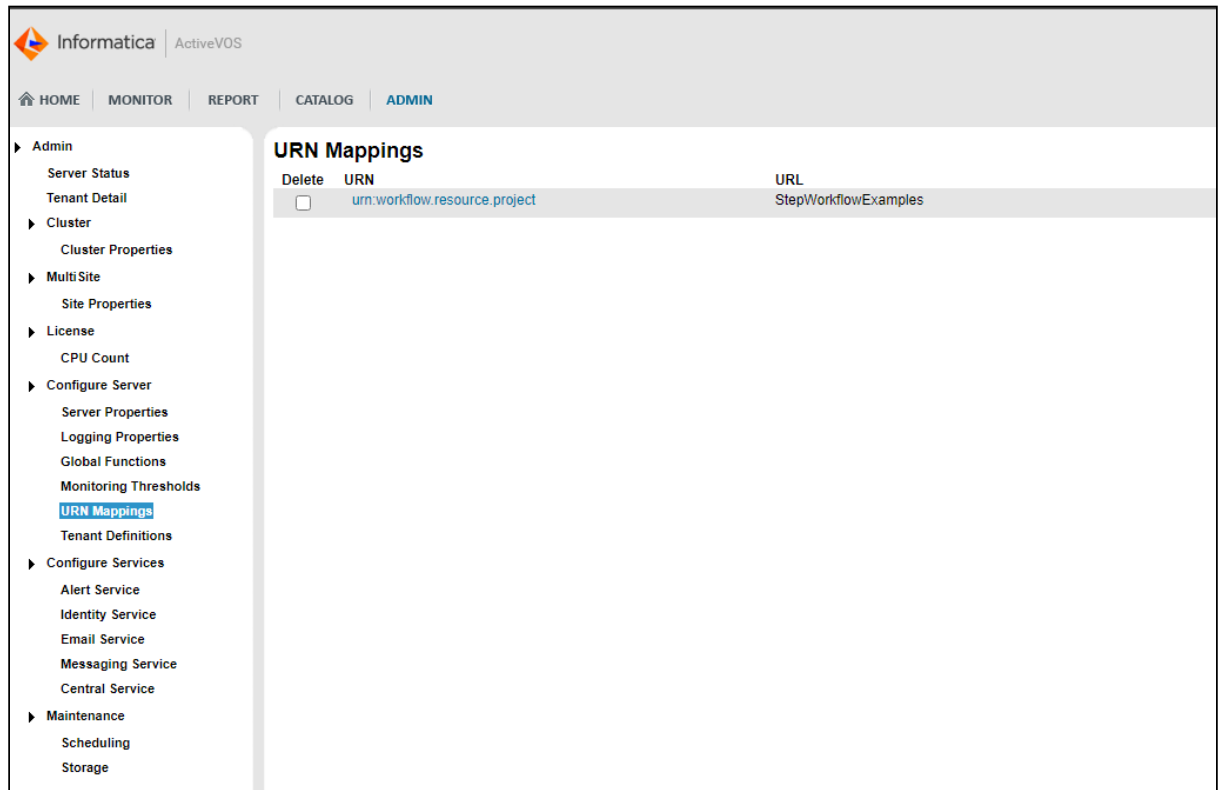
1. Replace [YOURTOKEN] with your Google Vision API key.
2. set the value of the tag *escape4windows* to 1, if your BPM server is running on Microsoft Windows.

## 2.3.5 Examples

### 2.3.5.1 Steps to run the examples

#### Option 1

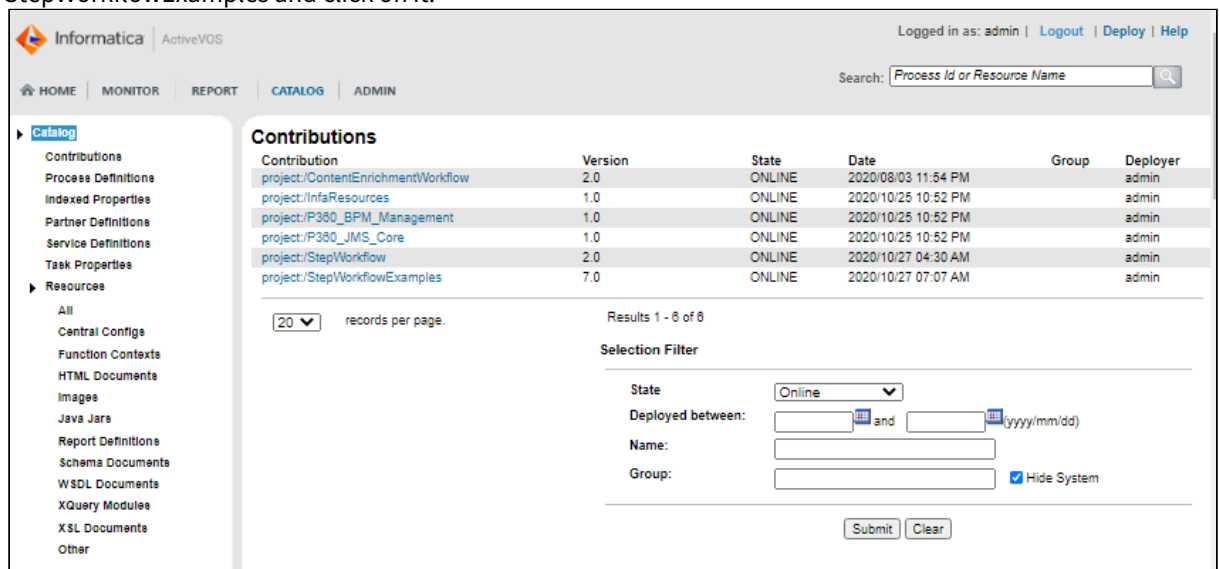
- Open the BPM console in your web browser and navigate to Admin/URN Mappings and create the mapping "urn:workflow.resource.project" and set the value to "StepWorkflowExamples".



- Define the trigger in the perspective "Business process management" in your MDM-Product 360 Desktop client. (refer to the StepWorkflowMap file in the examples project)

## Option 2

- Open the BPM console in your web browser and navigate to Catalog/Contributions/project:/StepWorkflowExamples and click on it.



- Select one of the png files and click on it to see the flow of the workflow.

The screenshot shows the Informatica ActiveVOS Catalog interface. The left sidebar contains a navigation menu with categories like Contributions, Process Definitions, Indexed Properties, Partner Definitions, Service Definitions, Task Properties, and Resources. The main area displays the 'Contribution Detail' for a specific contribution. Below this, there are two tables: 'Deployed Processes' and 'Contributed Resources'.

Contributed Resources					
Name	Version	Online	Target Namespace	Group	
Example_01 (1 Task).png	1.0.3	Yes			
Example_01.xml	1.0.3	Yes			
Example_02 (2 Tasks).png	1.0.3	Yes			
Example_02.xml	1.0.3	Yes			
Example_03 (2 Tasks with DQ checks).png	1.0.3	Yes			
Example_03.xml	1.0.3	Yes			
Example_04 (5 Tasks with DQ checks, parallel and sequential).png	1.0.3	Yes			
Example_04.xml	1.0.3	Yes			
Example_05 (extending example 4 with an approval step).png	1.0.2	Yes			
Example_05.xml	1.0.2	Yes			

- Select one of the xml files, i.e. Example\_01.xml click on it and copy the content into your clipboard.
- Now navigate to Catalog/Contributions/project:/StepWorkflow and click on it.
- Select one of the xml files, i.e. Workflow\_01.xml and click on it to enable the editing mode.
- Copy the content of your clipboard and paste it into the browser window.
- Click on update.
- Make the necessary adjustments in the file: StepWorkflowMap.xml. (It is recommended to use Workflow\_01.xml for the Example\_01.xml and so on.)
- Define the trigger in the perspective "Business process management" in your MDM-Product 360 Desktop client.
- Finished!

### 2.3.5.2 Table of contents

- [Default workflows](#) (see page 305)
- [Example 1 \(1 Task\)](#) (see page 312)
- [Example 2 \(2 Tasks\)](#) (see page 316)
- [Example 3 \(2 Tasks with DQ checks\)](#) (see page 320)
- [Example 4 \(5 Tasks with DQ checks, parallel and sequential\)](#) (see page 324)
- [Example 5 \(extending example 4 with an approval step\)](#) (see page 332)
- [Example 6 \(2 tasks \(incl. 1 supplier task\) + modifying of fields\)](#) (see page 341)
- [Example 7 \(2 Tasks + modifying of fields + approval step\)](#) (see page 351)
- [Example 8 \(2 Tasks modifying of fields approval step merge\)](#) (see page 360)
- [Example 9 \(Classic Tasks\)](#) (see page 366)
- [Example 10 \(1 Task\)](#) (see page 375)
- [Example 11 \(1 Task with DQ checks\)](#) (see page 380)
- [Example 12 \(2 Triggers and Merge with Results\)](#) (see page 387)
- [Example 13 \(Workflow starts another workflow\)](#) (see page 397)
- [Example 14 \(Export items\)](#) (see page 402)
- [Example 15 \(1 Task with DQ check and comments\)](#) (see page 408)
- [Example 16 \(Approval task with comments\)](#) (see page 413)
- [StepWorkflow Sanity](#) (see page 421)

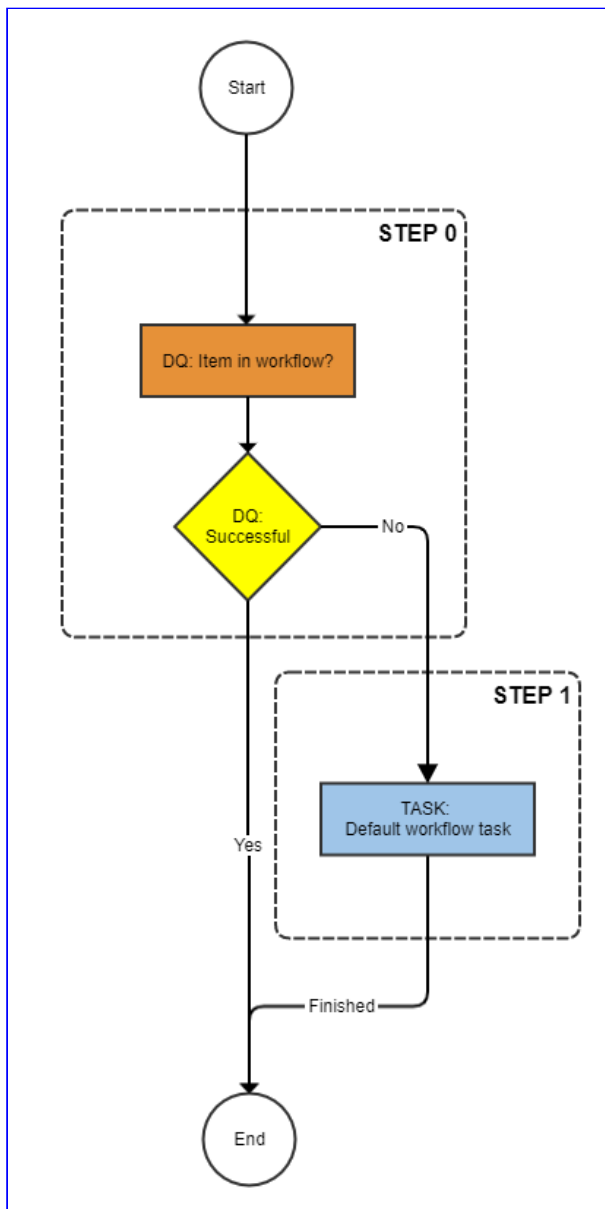
### 2.3.5.3 Default workflows

#### Defaults

The StepWorkflow package includes 3 default workflows. 1 workflow for items, 1 for products and an additional 1 for variants. These 3 workflows are only simple examples of defaults. Such a default workflow will be triggered if the trigger you defined in your Product 360 Desktop client does not match to an entry in the StepWorkflowMap.xml file.

#### DefaultArticle

#### **Diagram for default article**



**Explanation of the steps default article**

Step	Description
0	Check whether item is already in this workflow.
1	Put the item into the task "Default workflow task".

**Product 360 prerequisites for default article**

Product 360 entity	value
Usergroup	Standardusers

**StepWorkflow.xml file for defaultArticle.xml****Default Article**

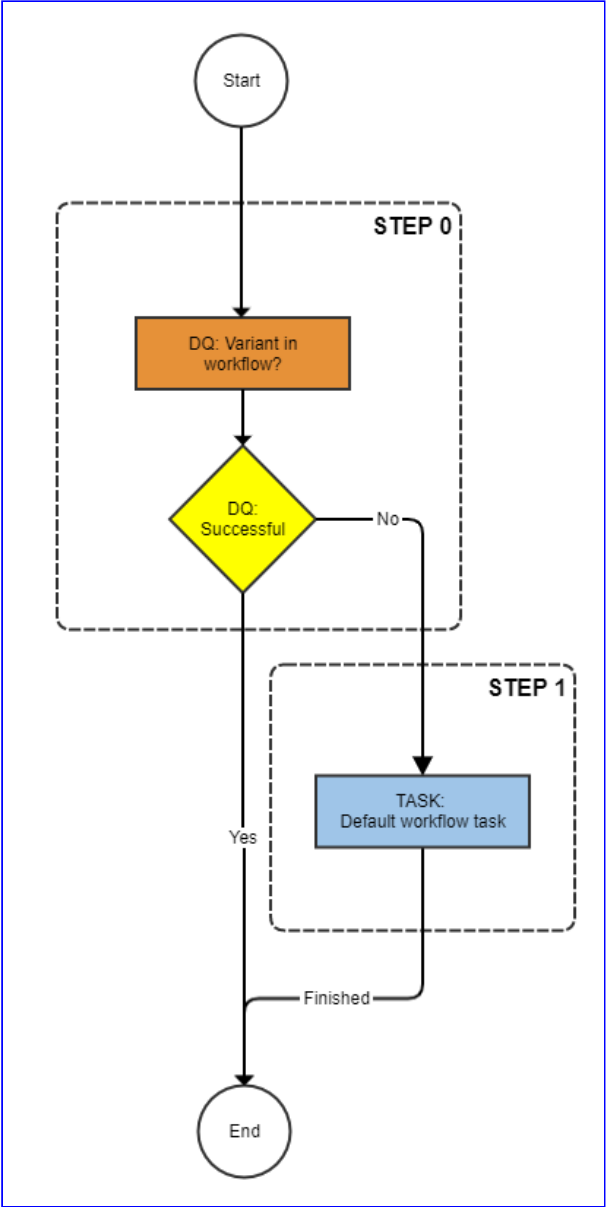
```

<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Item Workflow</label>
    <identifier>Default_Workflow_Article</identifier>
    <version>1.0</version>
    <step>
      <id>1</id>
      <entity>Article</entity>
      <batchSize>500</batchSize>
      <executeDq>Always</executeDq>
      <dqService>ItemsInWorkflowTasks-Process</dqService>
      <dqFailStep>STEP:2</dqFailStep>
    </step>
    <step>
      <id>2</id>
      <entity>Article</entity>
      <workflowStatus>Default workflow task</workflowStatus>
      <description>Default workflow task</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>Always</enterStatus>
      <batchSize>500</batchSize>
      <userType>userGroup</userType>
      <userName>Standardusers</userName>
      <uiTemplate></uiTemplate>
      <executeDq>Never</executeDq>
    </step>
  </workflow>
</imp:payload>

```

DefaultVariant

Diagram for default variant



Explanation of the steps default variant

Step	Description
0	Check whether item is already in this workflow.



Step	Description
1	Put the item into the task "Default workflow task".

**Product 360 prerequisites for default variant**

Product 360 entity	value
Usergroup	Standardusers

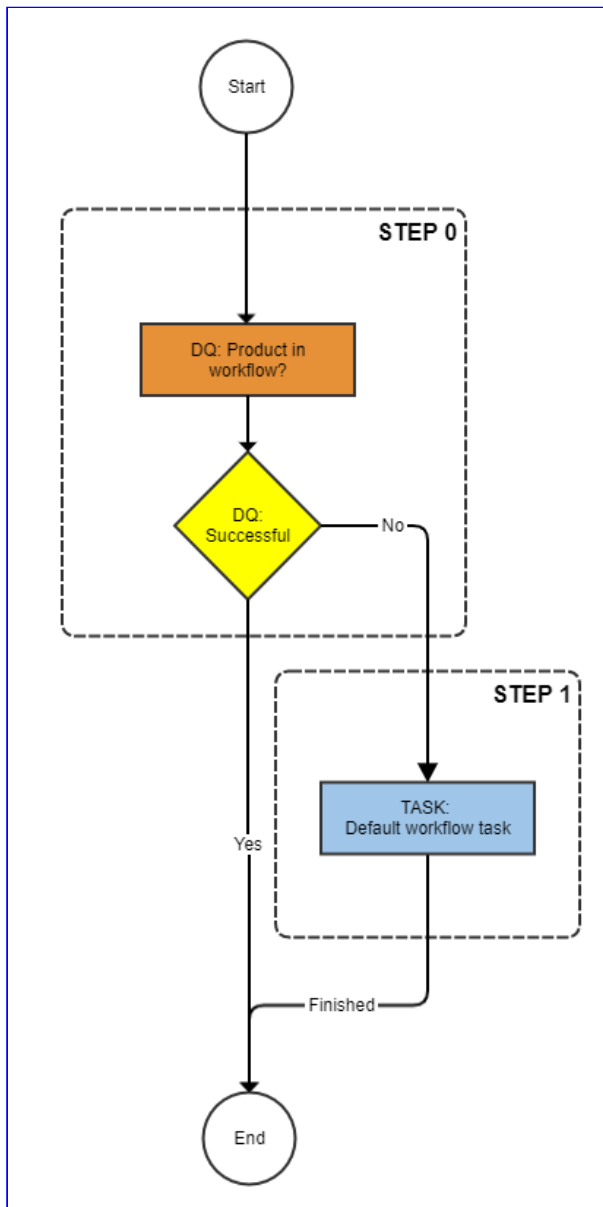
**StepWorkflow.xml file for defaultVariant.xml****Default Variant**

```
<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Variant Workflow</label>
    <identifier>Default_Workflow_Variant</identifier>
    <version>1.0</version>
    <step>
      <id>1</id>
      <entity>Variant</entity>
      <batchSize>500</batchSize>
      <executeDq>Always</executeDq>
      <dqService>ItemsInWorkflowTasks-Process</dqService>
      <dqFailStep>STEP:2</dqFailStep>
    </step>
    <step>
      <id>2</id>
      <entity>Variant</entity>
      <workflowStatus>Default workflow task</workflowStatus>
      <description>Default workflow task</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>Always</enterStatus>
      <batchSize>500</batchSize>
      <userType>userGroup</userType>
      <userName>Standardusers</userName>
      <uiTemplate></uiTemplate>
      <executeDq>Never</executeDq>
    </step>
  </workflow>
</imp:payload>
</payload>
```

```
</workflow>
</imp:payload>
```

DefaultProduct2G

Diagram for default product2g



Explanation of the steps default product2g

Step	Description
0	Check whether item is already in this workflow.
1	Put the item into the task "Default workflow task".

**Product 360 prerequisites for default product2g**

Product 360 entity	value
Usergroup	Standardusers

**StepWorkflow.xml file for defaultProduct2G.xml****Default Product2G**

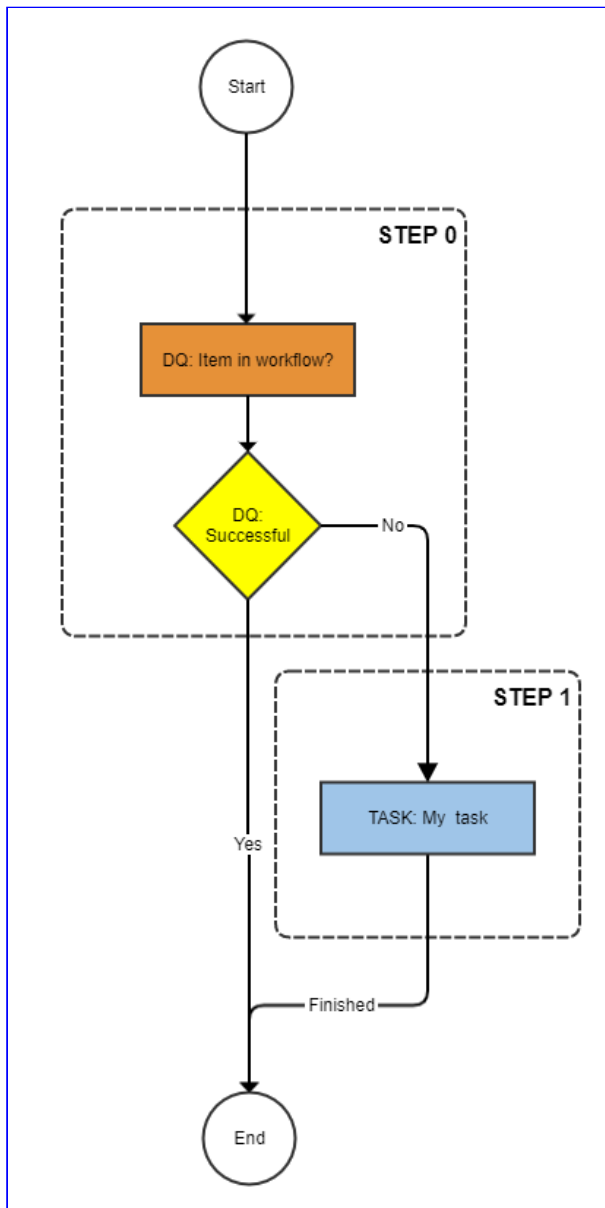
```
<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Product Workflow</label>
    <identifier>Default_Workflow_Product2G</identifier>
    <version>1.0</version>
    <step>
      <id>1</id>
      <entity>Product2G</entity>
      <batchSize>500</batchSize>
      <executeDq>Always</executeDq>
      <dqService>ItemsInWorkflowTasks-Process</dqService>
      <dqFailStep>STEP:2</dqFailStep>
    </step>
    <step>
      <id>2</id>
      <entity>Product2G</entity>
      <workflowStatus>Default workflow task</workflowStatus>
      <description>Default workflow task</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>Always</enterStatus>
      <batchSize>500</batchSize>
      <userType>userGroup</userType>
      <userName>Standardusers</userName>
    </step>
  </workflow>
</imp:payload>
```

```
<uiTemplate></uiTemplate>  
<executeDq>Never</executeDq>  
</step>  
</workflow>  
</imp:payload>
```

#### 2.3.5.4 Example 1 (1 Task)

The first workflow only puts items into a workflow task called “My first task” For this workflow task we need 2 steps. The first step checks whether the affected item is already in this workflow, if not it will be put into the task. The first step is mandatory and will be very useful in subsequent examples. With this approach you can ensure that the item is not in the same workflow more than one time. This is very important for typical approval workflows.

**Diagram for example 1**



**Explanation of the steps example 1**

Step	Description
0	Check whether item is already in this workflow.
1	Put the item into the task "My first task".

**Product 360 prerequisites for example 1**

Product 360 entity	value
Usergroup	Standardusers
Ui Template	Item approve UI

**Q Info**

Standarduser is the Identifier of the Usergroup!

StepWorkflow.xml file for example 1

**Example 1**

```
<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 1</label>
    <identifier>Workflow_01</identifier>
    <version>1.0</version>
    <step>
      <id>0</id>
      <entity>Article</entity>
      <enterStatus>Never</enterStatus>
      <batchSize>500</batchSize>
      <executeDq>Always</executeDq>
      <dqService>ItemsInWorkflowTasks-Process</dqService>
      <dqFailStep>STEP:1</dqFailStep>
    </step>
    <step>
      <id>1</id>
      <entity>Article</entity>
      <workflowStatus>My first task</workflowStatus>
      <description>My first workflow task</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>Always</enterStatus>
      <userType>userGroup</userType>
      <userName>Standardusers</userName>
    </step>
  </workflow>
</imp:payload>
</payload>
```

```

    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Never</executeDq>
  </step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

#### Step 0

This step checks only whether the item is already in this workflow. If not the executed next step is 1. If yes the workflow will end because there is no next step defined.

Key	Value	Description
id	0	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
batchSize	500	Batch size to use with trigger batching.
dqService	ItemsInWorkflowTasks-Process	This DQ service is an additional bpel inside the StepWorkflow which checks whether the item is already inside this workflow or not.
dqFailStep	STEP:1	The item will be moved to the step with the identifier 1 if the above dq fails.

#### Step 1

This step checks only whether the item is already in this workflow. If not the executed next step is 1. If yes the workflow will end because there is no next step defined.

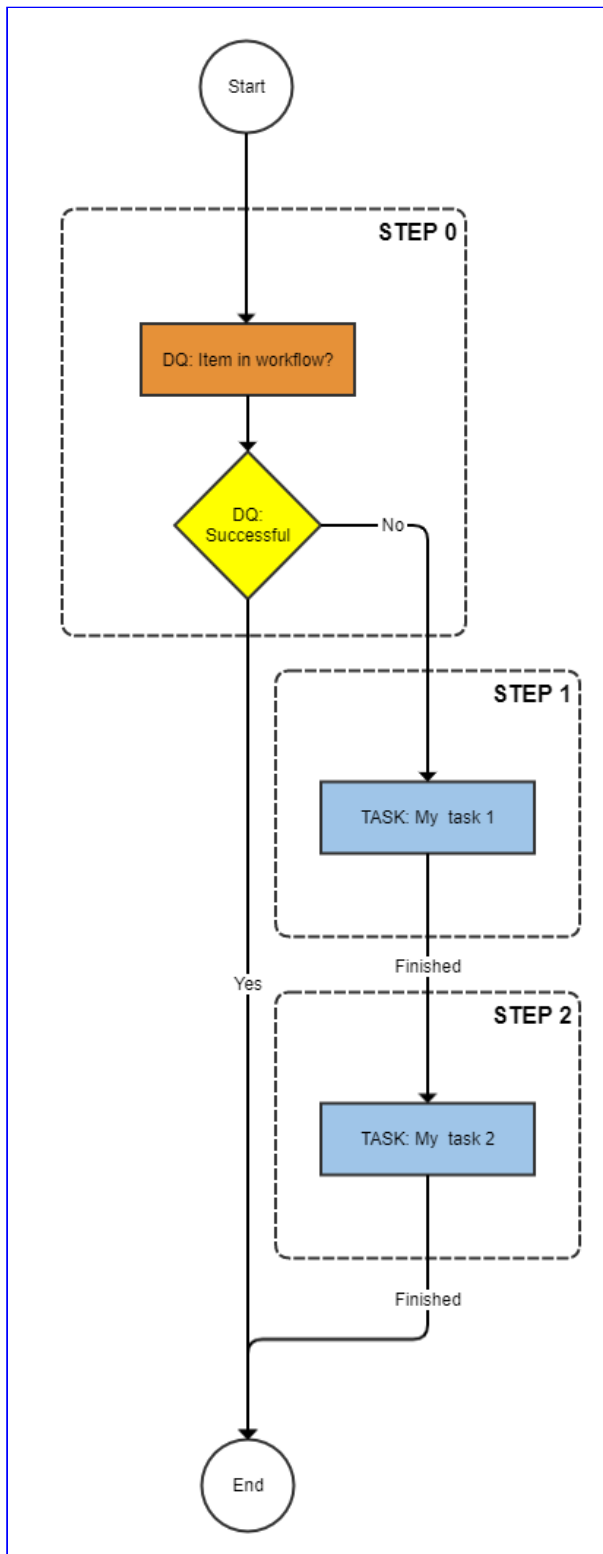
Key	Value	Description
id	1	Identifier of the step.
entity	Article	Entity container for this workflow.
workflow Status	My first task	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	My first workflow task	Description of the status.
workflow ServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStatus	Always	The circumstances when an item should enter a status (task). In this case always!
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.
userName	Standardusers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".
uiTemplate	Item approve UI	This is the default UI of the task.
executeDq	Never	The circumstances when a DQ should be run. In this case never!

### 2.3.5.5 Example 2 (2 Tasks)

The next example extends the first example with a further task after the first task.



Diagram for example 2



**Product 360 prerequisites for example 2**

Product 360 entity	value
Usergroup	Standardusers
Ui Template	Item approve UI

## Explanation of the steps example 2

Step	Description
0	Check whether item is already in this workflow.
1	Put the item into the task "My task 1".
2	Put the item into the task "My task 2".

## StepWorkflow.xml file for example 2

**Example 2**

```
<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 2</label>
    <identifier>Workflow_02</identifier>
    <version>1.0</version>
    <step>
      <id>0</id>
      <entity>Article</entity>
      <enterStatus>Never</enterStatus>
      <batchSize>500</batchSize>
      <executeDq>Always</executeDq>
      <dqService>ItemsInWorkflowTasks-Process</dqService>
      <dqFailStep>STEP:1</dqFailStep>
    </step>
  </workflow>
</imp:payload>
```

```

<step>
  <id>1</id>
  <entity>Article</entity>
  <workflowStatus>My task 1</workflowStatus>
  <description>My task 1</description>
  <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
  <enterStatus>Always</enterStatus>
  <batchSize>500</batchSize>
  <userType>userGroup</userType>
  <userName>Standardusers</userName>
  <uiTemplate>Item approve UI</uiTemplate>
  <executeDq>Never</executeDq>
  <nextStep>STEP:2</nextStep>
</step>
<step>
  <id>2</id>
  <entity>Article</entity>
  <workflowStatus>My task 2</workflowStatus>
  <description>My task 2</description>
  <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
  <enterStatus>Always</enterStatus>
  <batchSize>500</batchSize>
  <userType>userGroup</userType>
  <userName>Standardusers</userName>
  <uiTemplate>Item approve UI</uiTemplate>
  <executeDq>Never</executeDq>
</step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

Only the differences to the previous example will be explained.

Step 0

Step 0 is an exact copy of the step 0 of the example 1.

Step 1

Step 1 is nearly the same as the step 1 of the example above. Only the differences are shown in the next table.

Key	Value	Description
...	...	Same keys and values than in example 1.

Key	Value	Description
workflow Status	My task 1	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	My task 1	Description of the status.
nextStep	STEP:2	Identifier of the next step, The item will be moved to the next step, when this step is finished.

Step 2

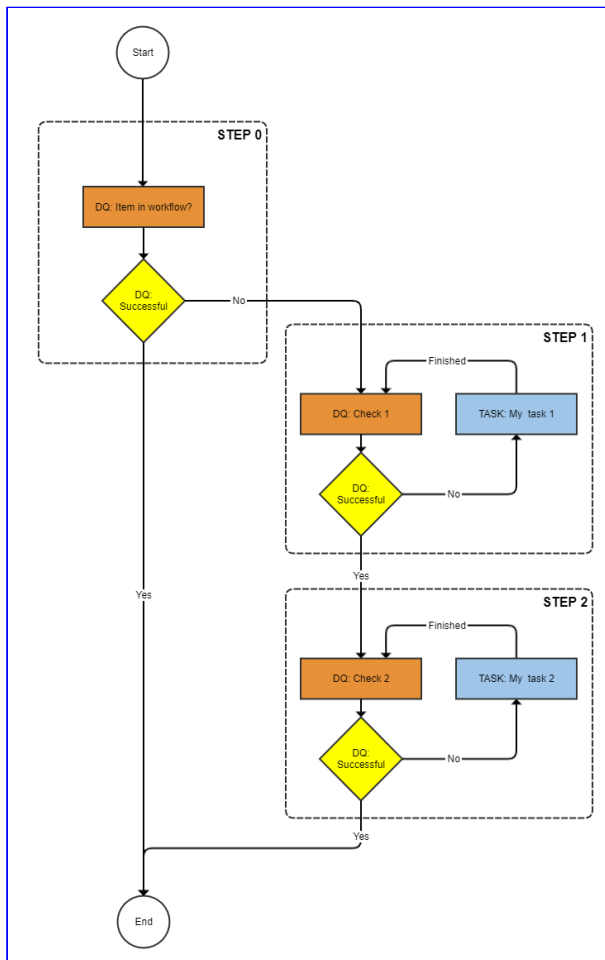
Step 2 is mostly a copy of step 1.

Key	Value	Description
...	...	Same keys and values than step 1.
id	2	Identifier of the step.
workflow Status	My task 2	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	My task 2	Description of the status.
nextStep		This step contains no next step, because the workflow have to end at this point.

#### 2.3.5.6 Example 3 (2 Tasks with DQ checks)

The next example extends the previous example so that the tasks will only be created when the DQ check fails.

**Diagram for example 3**



**Product 360 prerequisites for example 3**

Product 360 entity	value
Usergroup	Standardusers
Ui Template	Item approve UI
DQ Channel	Channel 1
DQ Channel	Channel 2

## Explanation of the steps example 3

Step	Description
0	Check whether item is already in this workflow.
1	Run DQ check "Channel 1" if fail ==>put the item into the task "My task 1"
2	Run DQ check "Channel 2" if fail ==>put the item into the task "My task 2"

## StepWorkflow.xml file for example 3

## Example 3

```

<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 3</label>
    <identifier>Workflow_03</identifier>
    <version>1.0</version>
    <step>
      <id>0</id>
      <entity>Article</entity>
      <enterStatus>Never</enterStatus>
      <batchSize>500</batchSize>
      <executeDq>Always</executeDq>
      <dqService>ItemsInWorkflowTasks-Process</dqService>
      <dqFailStep>STEP:1</dqFailStep>
    </step>
    <step>
      <id>1</id>
      <entity>Article</entity>
      <workflowStatus>My task 1</workflowStatus>
      <description>My task 1</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>OnDqResults</enterStatus>
      <batchSize>500</batchSize>
      <userType>userGroup</userType>
      <userName>Standardusers</userName>
      <uiTemplate>Item approve UI</uiTemplate>
      <executeDq>Always</executeDq>
      <dqService>ExecutedqBatch-Process</dqService>
    </step>
  </workflow>
</imp:payload>

```

```

    <dqChannel>Channel 1</dqChannel>
    <nextStep>STEP:2</nextStep>
  </step>
  <step>
    <id>2</id>
    <entity>Article</entity>
    <workflowStatus>My task 2</workflowStatus>
    <description>My task 2</description>
    <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
    <enterStatus>OnDqResults</enterStatus>
    <batchSize>500</batchSize>
    <userType>userGroup</userType>
    <userName>Standardusers</userName>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Always</executeDq>
    <dqService>ExecuteDqBatch-Process</dqService>
    <dqChannel>Channel 2</dqChannel>
  </step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

Only the differences to the previous example will be explained.

Step 0

Step 0 is an exact copy of the step 0 of the example 2.

Step 1

Step 1 is nearly the same as the step 1 of example 2. Only the differences are shown in the next table.

Key	Value	Description
...	...	Same keys and values than in example 2.
enterStatus	OnDqResults	The circumstances when an item should enter a status (task). In this case only when DQ check will fail!
executeDq	Always	The circumstances when a DQ should be run. In this case always!

Key	Value	Description
dqService	ExecuteDqBatch-Process	The service name of the process that will be handle the processing of the dq.
dqChannel	Channel 1	Name of the DQ channel which will be executed.

#### Step 2

Step 2 is nearly the same as the step 1 of the example above.

Only the differences are shown in the next table.

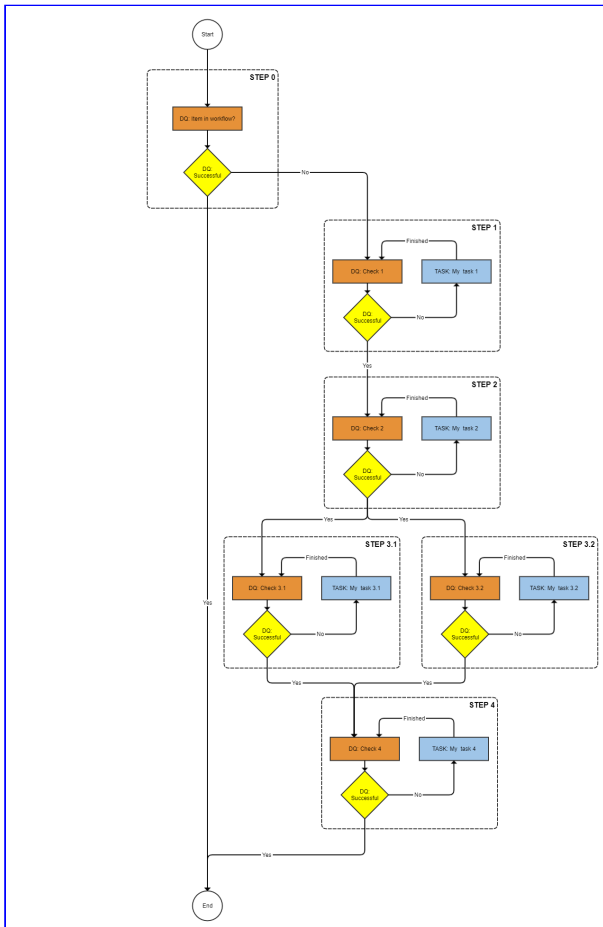
Key	Value	Description
...	...	Same keys and values than in example 2.
enterStatus	OnDqResults	The circumstances when an item should enter a status (task). In this case only when DQ check will fail!
executedDq	Always	The circumstances when a DQ should be run. In this case always!
dqService	ExecuteDqBatch-Process	The service name of the process that will be handle the processing of the DQ.
dqChannel	Channel 2	Name of the DQ channel which will be executed.

#### 2.3.5.7 Example 4 (5 Tasks with DQ checks, parallel and sequential)

Example 4 starts with 2 sequential tasks, after that there are 2 tasks which are running in parallel and after finishing those 2 parallel tasks the last task will start.



**Diagram for example 4**



**Product 360 prerequisites for example 4**

Product 360 entity	value
Usergroup	Standardusers
Ui Template	Item approve UI
DQ Channel	Channel 1
DQ Channel	Channel 2

Product 360 entity	value
DQ Channel	Channel 3.1
DQ Channel	Channel 3.2
DQ Channel	Channel 4

Explanation of the steps example 4

Step	Description
0	Check whether item is already in this workflow.
1	Run DQ check "Channel 1" if fail ==>put the item into the task "My task 1"
2	Run DQ check "Channel 2" if fail ==>put the item into the task "My task 2"
3.1	Run DQ check "Channel 3.1" if fail ==>put the item into the task "My task 3.1"
3.2	Run DQ check "Channel 3.2" if fail ==>put the item into the task "My task 3.2"
4	Run DQ check "Channel 4" if fail ==>put the item into the task "My task 4"

StepWorkflow.xml file for example 4

#### Example 4

```
<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 4</label>
```

```

<identifier>Workflow_04</identifier>
<version>1.0</version>
<step>
  <id>0</id>
  <entity>Article</entity>
  <enterStatus>Never</enterStatus>
  <batchSize>500</batchSize>
  <executeDq>Always</executeDq>
  <dqService>ItemsInWorkflowTasks-Process</dqService>
  <dqFailStep>STEP:1</dqFailStep>
</step>
<step>
  <id>1</id>
  <entity>Article</entity>
  <workflowStatus>My task 1</workflowStatus>
  <description>My task 1</description>
  <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
  <enterStatus>OnDqResults</enterStatus>
  <batchSize>500</batchSize>
  <userType>userGroup</userType>
  <userName>Standardusers</userName>
  <uiTemplate>Item approve UI</uiTemplate>
  <executeDq>Always</executeDq>
  <dqService>ExecuteDqBatch-Process</dqService>
  <dqChannel>Channel 1</dqChannel>
  <nextStep>STEP:2</nextStep>
</step>
<step>
  <id>2</id>
  <entity>Article</entity>
  <workflowStatus>My task 2</workflowStatus>
  <description>My task 2</description>
  <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
  <enterStatus>OnDqResults</enterStatus>
  <batchSize>500</batchSize>
  <userType>userGroup</userType>
  <userName>Standardusers</userName>
  <uiTemplate>Item approve UI</uiTemplate>
  <executeDq>Always</executeDq>
  <dqService>ExecuteDqBatch-Process</dqService>
  <dqChannel>Channel 2</dqChannel>
  <nextStep>StepWorkflow_ParallelTasks-Workflow</nextStep>
  <parallelStep>3.1</parallelStep>
  <parallelStep>3.2</parallelStep>
  <parallelNextStep>4</parallelNextStep>
</step>
<step>
  <id>3.1</id>
  <entity>Article</entity>
  <workflowStatus>My task 3.1</workflowStatus>
  <description>My task 3.1</description>
  <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>

```

```

    <enterStatus>OnDqResults</enterStatus>
    <userType>userGroup</userType>
    <userName>Standardusers</userName>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Always</executeDq>
    <dqService>ExecuteDqBatch-Process</dqService>
    <dqChannel>Channel 3.1</dqChannel>
    <dqFailTrigger>StepWorkflow_ParallelTasks-Finish-1</dqFailTrigger>
    <nextTrigger>StepWorkflow_ParallelTasks-Finish-1</nextTrigger>
  </step>
  <step>
    <id>3.2</id>
    <entity>Article</entity>
    <workflowStatus>My task 3.2</workflowStatus>
    <description>My task 3.2</description>
    <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
    <enterStatus>OnDqResults</enterStatus>
    <userType>userGroup</userType>
    <userName>Standardusers</userName>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Always</executeDq>
    <dqService>ExecuteDqBatch-Process</dqService>
    <dqChannel>Channel 3.2</dqChannel>
    <dqFailTrigger>StepWorkflow_ParallelTasks-Finish-2</dqFailTrigger>
    <nextTrigger>StepWorkflow_ParallelTasks-Finish-2</nextTrigger>
  </step>
  <step>
    <id>4</id>
    <entity>Article</entity>
    <workflowStatus>My task 4</workflowStatus>
    <description>My task 4</description>
    <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
    <enterStatus>OnDqResults</enterStatus>
    <batchSize>500</batchSize>
    <userType>userGroup</userType>
    <userName>Standardusers</userName>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Always</executeDq>
    <dqService>ExecuteDqBatch-Process</dqService>
    <dqChannel>Channel 4</dqChannel>
  </step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

Only the differences to the previous example will be explained.

Step 0

Step 0 is an exact copy of the step 0 of example 3.

### Step 1

Step 1 is an exact copy of the step 1 of example above 3.

### Step 2

Step 2 is nearly the same as the step 2 of the example above. Only the differences are shown in the next table.

Key	Value	Description
...	...	Same keys and values than in example 3.
nextStep	StepWorkflow_ParallelTasks-Workflow	StepWorkflow_ParallelTasks-Workflow defines that the next steps run in parallel
parallelStep	3.1	Identifier of the first parallel step, the item will be moved to this parallel step. <b>The maximum number of parallel steps is 6.</b>
parallelStep	3.2	Identifier of the second parallel step, the item will be moved also to this parallel step.
parallelNextStep	4	The item will be moved to this step when all parallel steps are finished.

### Step 3.1

Key	Value	Description
id	3.1	Identifier of the step.
entity	Article	Entity container for this workflow.
workflowStatus	My task 3.1	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]

Key	Value	Description
description	My task 3.1	Description of the status.
workflowServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStatus	OnDqResults	The circumstances when an item should enter a status (task). In this case only when DQ check will fail!
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.
userName	Standardusers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".
uiTemplate	Item approve UI	This is the default UI of the task.
executeDq	Always	The circumstances when a DQ should be run. In this case always!
dqService	ExecuteDqBatch-Process	The service name of the process that will be handle the processing of the DQ.
dqChannel	Channel 3.1	Name of the DQ channel which will be executed.
dqFailTrigger	StepWorkflow_ParallelTasks-Finish-1	This is the service name of the partner link of the parallel tasks. It will be called if DQ fails to ensure that the item will remain in the status (task).
nextTrigger	StepWorkflow_ParallelTasks-Finish-1	This is another service name of the partner link of the parallel tasks. It will be called if the item will leave this step.

### Step 3.2

Step 3.2 is nearly the same as the step 3.1. Only the differences are shown in the next table.

Key	Value	Description
id	3.1	Identifier of the step.
workflowStatus	My task 3.2	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	My task 3.2	Description of the status.
dqChannel	Channel 3.2	Name of the DQ channel which will be executed.
dqFailTrigger	StepWorkflow_ParallelTasks-Finish-2	This is the service name of the partner link of the parallel tasks. It will be called if DQ fails to ensure that the item will remain in the status (task).
nextTrigger	StepWorkflow_ParallelTasks-Finish-2	This is another service name of the partner link of the parallel tasks. It will be called if the item will leave this step.

### Step 4

Key	Value	Description
id	4	Identifier of the step.
entity	Article	Entity container for this workflow.
workflowStatus	My task 4	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	My task 4	Description of the status.

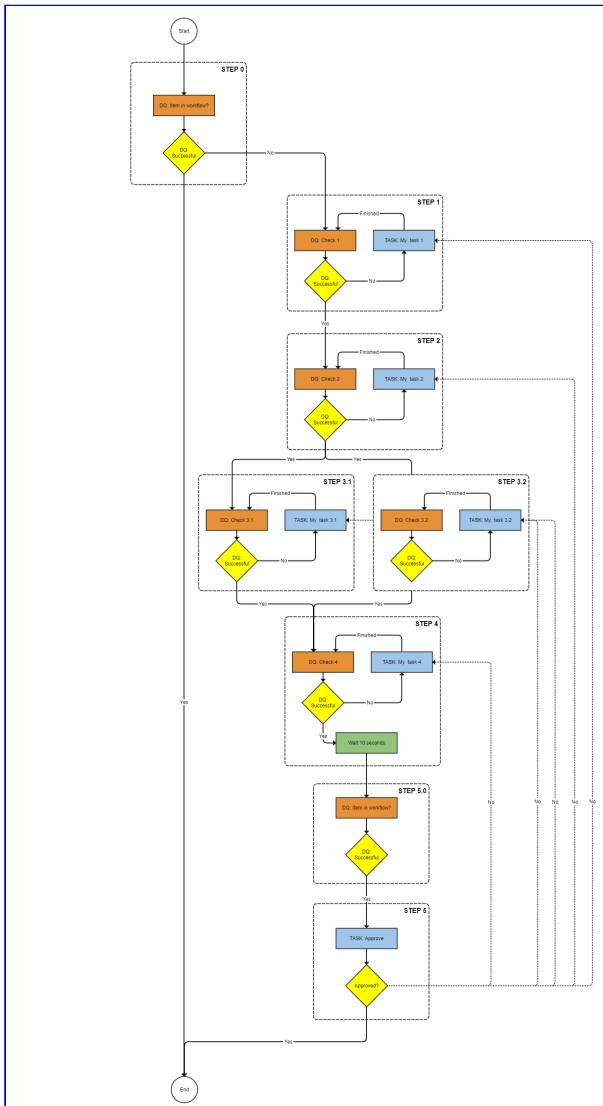
Key	Value	Description
workflowServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStatus	OnDqResults	The circumstances when an item should enter a status (task). In this case only when DQ check will fail!
batchSize	500	Batch size to use with trigger batching.
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.
userName	Standardusers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".
uiTemplate	Item approve UI	This is the default UI of the task.
executeDq	Always	The circumstances when a DQ should be run. In this case always!
dqService	ExecuteDqBatch-Process	The service name of the process that will be handle the processing of the DQ.
dqChannel	Channel 4	Name of the DQ channel which will be executed.

### 2.3.5.8 Example 5 (extending example 4 with an approval step)

The next example extends the previous example with an approval step at the end of the workflow, so the approver can reject the item to all of the previous steps.



Diagram for example 5



Product 360 prerequisites for example 5

Product 360 entity	value
Usergroup	Standardusers
Ui Template	Item approve UI
DQ Channel	Channel 1

Product 360 entity	value
DQ Channel	Channel 2
DQ Channel	Channel 3.1
DQ Channel	Channel 3.2
DQ Channel	Channel 4

Explanation of the steps example 5

Step	Description
0	Check whether item is already in this workflow
1	Run DQ check "Channel 1" if fail ==>put the item into the task "My task 1"
2	Run DQ check "Channel 2" if fail ==>put the item into the task "My task 2"
3.1	Run DQ check "Channel 3.1" if fail ==>put the item into the task "My task 3.1"
3.2	Run DQ check "Channel 3.2" if fail ==>put the item into the task "My task 3.2"
4	Run DQ check "Channel 4" if fail ==>put the item into the task "My task 4"
5.0	Check whether item is already in this workflow.
5	Approve task with rejection options to all previous tasks

StepWorkflow.xml file for example 5

### Example 5

```
<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 5</label>
    <identifier>Workflow_05</identifier>
    <version>1.0</version>
    <step>
      <id>0</id>
      <entity>Article</entity>
      <enterStatus>Never</enterStatus>
      <batchSize>500</batchSize>
      <executeDq>Always</executeDq>
      <dqService>ItemsInWorkflowTasks-Process</dqService>
      <dqFailStep>STEP:1</dqFailStep>
    </step>
    <step>
      <id>1</id>
      <entity>Article</entity>
      <workflowStatus>My task 1</workflowStatus>
      <description>My task 1</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>OnDqResults</enterStatus>
      <batchSize>500</batchSize>
      <userType>userGroup</userType>
      <userName>Standardusers</userName>
      <uiTemplate>Item approve UI</uiTemplate>
      <executeDq>Always</executeDq>
      <dqService>ExecuteDqBatch-Process</dqService>
      <dqChannel>Channel 1</dqChannel>
      <nextStep>STEP:2</nextStep>
    </step>
    <step>
      <id>2</id>
      <entity>Article</entity>
      <workflowStatus>My task 2</workflowStatus>
      <description>My task 2</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>OnDqResults</enterStatus>
      <batchSize>500</batchSize>
      <userType>userGroup</userType>
      <userName>Standardusers</userName>
      <uiTemplate>Item approve UI</uiTemplate>
      <executeDq>Always</executeDq>
      <dqService>ExecuteDqBatch-Process</dqService>
```

```

<dqChannel>Channel 2</dqChannel>
<nextStep>StepWorkflow_ParallelTasks-Workflow</nextStep>
<parallelStep>3.1</parallelStep>
<parallelStep>3.2</parallelStep>
<parallelNextStep>4</parallelNextStep>
</step>
<step>
  <id>3.1</id>
  <entity>Article</entity>
  <workflowStatus>My task 3.1</workflowStatus>
  <description>My task 3.1</description>
  <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
  <enterStatus>OnDqResults</enterStatus>
  <userType>userGroup</userType>
  <userName>Standardusers</userName>
  <uiTemplate>Item approve UI</uiTemplate>
  <executeDq>Always</executeDq>
  <dqService>ExecuteDqBatch-Process</dqService>
  <dqChannel>Channel 3.1</dqChannel>
  <dqFailTrigger>StepWorkflow_ParallelTasks-Finish-1</dqFailTrigger>
  <nextTrigger>StepWorkflow_ParallelTasks-Finish-1</nextTrigger>
</step>
<step>
  <id>3.2</id>
  <entity>Article</entity>
  <workflowStatus>My task 3.2</workflowStatus>
  <description>My task 3.2</description>
  <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
  <enterStatus>OnDqResults</enterStatus>
  <userType>userGroup</userType>
  <userName>Standardusers</userName>
  <uiTemplate>Item approve UI</uiTemplate>
  <executeDq>Always</executeDq>
  <dqService>ExecuteDqBatch-Process</dqService>
  <dqChannel>Channel 3.2</dqChannel>
  <dqFailTrigger>StepWorkflow_ParallelTasks-Finish-2</dqFailTrigger>
  <nextTrigger>StepWorkflow_ParallelTasks-Finish-2</nextTrigger>
</step>
<step>
  <id>4</id>
  <entity>Article</entity>
  <workflowStatus>My task 4</workflowStatus>
  <description>My task 4</description>
  <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
  <enterStatus>OnDqResults</enterStatus>
  <batchSize>500</batchSize>
  <userType>userGroup</userType>
  <userName>Standardusers</userName>
  <uiTemplate>Item approve UI</uiTemplate>
  <executeDq>Always</executeDq>
  <dqService>ExecuteDqBatch-Process</dqService>
  <dqChannel>Channel 4</dqChannel>

```

```

    <nextStep>WorkflowWait-Process</nextStep>
    <wait>PT10S</wait>
    <nextStep>STEP:5.0</nextStep>
</step>
<step>
    <id>5.0</id>
    <entity>Article</entity>
    <enterStatus>Never</enterStatus>
    <batchSize>500</batchSize>
    <executeDq>Always</executeDq>
    <dqService>ItemsInWorkflowTasks-Process</dqService>
    <dqFailStep>STEP:5</dqFailStep>
</step>
<step>
    <id>5</id>
    <entity>Article</entity>
    <workflowStatus>Approve</workflowStatus>
    <description>Approval</description>
    <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
    <enterStatus>Always</enterStatus>
    <batchSize>500</batchSize>
    <userType>userGroup</userType>
    <userName>Standardusers</userName>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Never</executeDq>
    <rejectTrigger>StepWorkflow_ParallelTasks-Reject</rejectTrigger>
    <singleChoice>>false</singleChoice>
    <rejectDecision>
        <id>p360.bpm.reject.STEP:1</id>
        <label>Reject to task 1</label>
    </rejectDecision>
    <rejectDecision>
        <id>p360.bpm.reject.STEP:2</id>
        <label>Reject to task 2</label>
    </rejectDecision>
    <rejectDecision>
        <id>p360.bpm.reject.TRIGGER:3.1</id>
        <label>Reject to task 3.1</label>
    </rejectDecision>
    <rejectDecision>
        <id>p360.bpm.reject.TRIGGER:3.2</id>
        <label>Reject to task 3.2</label>
    </rejectDecision>
    <rejectDecision>
        <id>p360.bpm.reject.STEP:4</id>
        <label>Reject to task 4</label>
    </rejectDecision>
</step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

Only the differences to the previous example will be explained.

Step 0

Step 0 is an exact copy of the step 0 of the example 4.

Step 1

Step 1 is an exact copy of the step 1 of the example 4.

Step 2

Step 2 is an exact copy of the step 2 of the example 4.

Step 3.1

Step 3.1 is an exact copy of the step 3.1 of the example 4.

Step 3.2

Step 3.2 is an exact copy of the step 3.2 of the example 4.

Step 4

Step 4 is nearly the same as the step 4 of the example 4. Only the differences are shown in the next table.

Key	Value	Description
...	...	Same keys and values than in the example 4.
nextStep	STEP:5.0	Identifier of the next step, The item will be moved to the next step, when this step is finished.

Step 5.0

This steps checks only whether the item is already in this workflow. If not the executed next step is 5. If yes the workflow will end to avoid that the item will go to the step 5 (Approve) until it is somewhere else in the workflow.

Key	Value	Description
id	5.0	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
batchSize	500	Batch size to use with trigger batching.
dqService	ItemsInWorkflowTasks-Process	This DQ service is an additional bpel inside the StepWorkflow which checks whether the item is already inside this workflow or not.
nextStep	WorkflowWait-Process	Wait process needed as step before continuing to avoid that the item is still in the workflow
wait	PT10S	Value to wait (10 seconds).
dqFailStep	STEP:5	The item will be moved to the step with the identifier 5 if the above DQ fails.

#### Step 5

This steps checks only whether the item is already in this workflow. If not the executed next step is 5. If yes the workflow will end to avoid that the item will go to the step 5 (Approve) until it is somewhere else in the workflow.

Key	Value	Description
id	5	Identifier of the step.

Key	Value	Description
entity	Article	Entity container for this workflow.
workflowStatus	Approve	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	Approval	Description of the status.
workflowServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStatus	Always	The circumstances when an item should enter a status (task). In this case always!
batchSize	500	Batch size to use with trigger batching.
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.
userName	Standardusers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".
uiTemplate	Item approve UI	This is the default UI of the task.
executeDq	Never	The circumstances when a DQ should be run. In this case never!
rejectTrigger	StepWorkflow_ParallelTasks-Reject	Endpoint for the reject trigger of parallel states (tasks).
singleChoice	false	This is for Rejects only and determines if you can have more than one choice. ==> The workflow can be rejected to all steps in one decision.



Key	Value	Description
<rejectDecision>id	p360.bpm.reject.STEP:1	Identifier of the 1 <sup>st</sup> reject step.
<rejectDecision>label	Reject to task 1	Label of the 1 <sup>st</sup> rejection. This label will show up in the UI.
<rejectDecision>id	p360.bpm.reject.STEP:2	Identifier of the 2 <sup>nd</sup> reject step.
<rejectDecision>label	Reject to task 2	Label of the 2 <sup>nd</sup> rejection. This label will show up in the UI.
<rejectDecision>id	p360.bpm.reject.TRIGGER:3.1	Identifier of the 3 <sup>rd</sup> reject step. (In this case you have to reject to a trigger!)
<rejectDecision>label	Reject to task 3.1	Label of the 3 <sup>rd</sup> rejection. This label will show up in the UI.
<rejectDecision>id	p360.bpm.reject.TRIGGER:3.2	Identifier of the 4th reject step. (In this case you have to reject to a trigger!)
<rejectDecision>label	Reject to task 3.2	Label of the 4th rejection. This label will show up in the UI.
<rejectDecision>id	p360.bpm.reject.STEP:4	Identifier of the 5th reject step.
<rejectDecision>label	Reject to task 4	Label of the 5th rejection. This label will show up in the UI.

#### 2.3.5.9 Example 6 (2 tasks (incl. 1 supplier task) + modifying of fields)

The next example extends the example 2 workflow with modifying of fields within the workflow after finishing the tasks,



## Q Info

Please consider to define your trigger in Product 360 carefully, because if the initiator "Service API" is not excluded for this workflow will run in a loop!

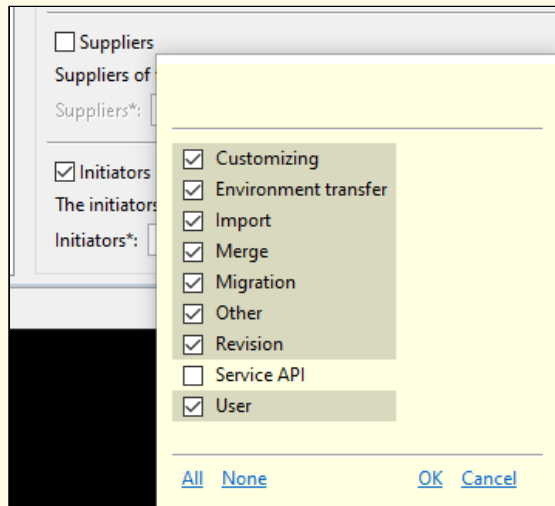
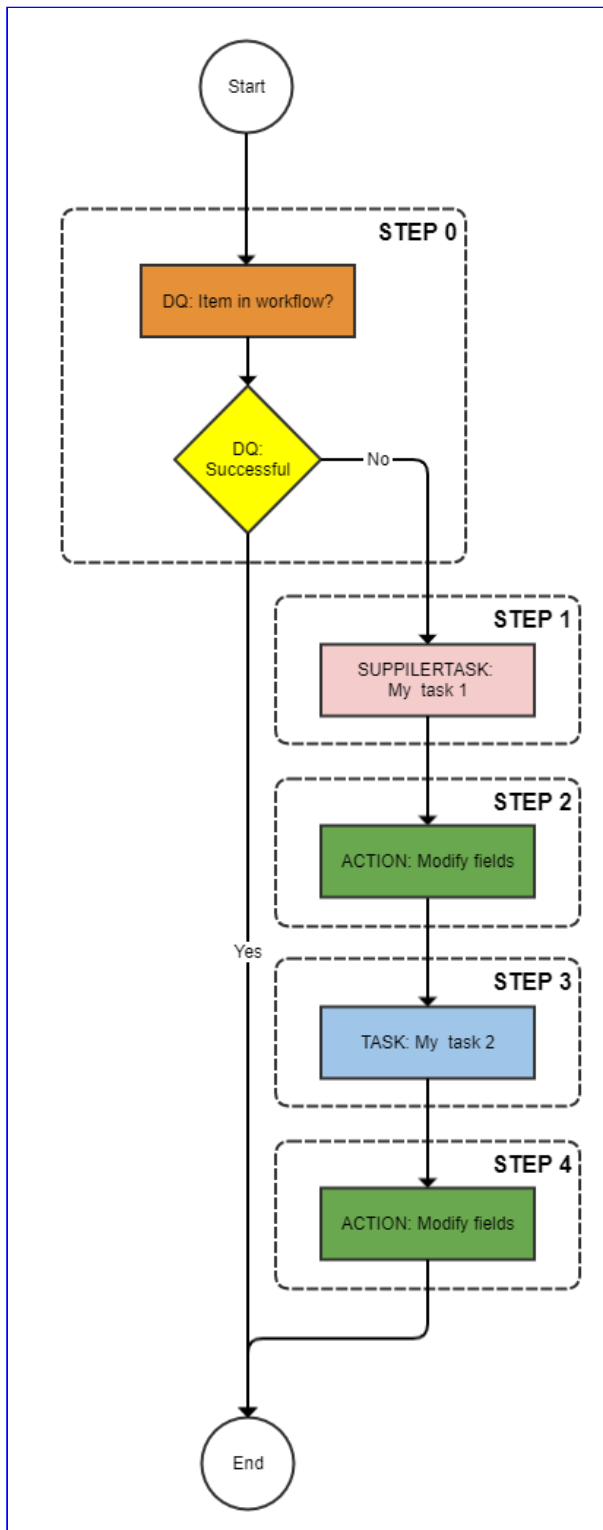


Diagram for example 6



**Product 360 prerequisites for example 6**

Product 360 entity	value
Usergroup	Standardusers
Ui Template	Item approve UI

## Explanation of the steps example 6

Step	Description
0	Check whether item is already in this workflow
1	Put the item into the supplier task "My task 1"
2	Modify the field Article.CurrentStatus to the value 200
3	Put the item into the task "My task 2"
4	Modify the field Article.CurrentStatus to the value 300

## StepWorkflow.xml file for example 6

**Example 6**

```
<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 6</label>
    <identifier>Workflow_06</identifier>
    <version>1.0</version>
    <step>
      <id>0</id>
      <entity>Article</entity>
```

```

    <enterStatus>Never</enterStatus>
    <batchSize>500</batchSize>
    <executeDq>Always</executeDq>
    <dqService>ItemsInWorkflowTasks-Process</dqService>
    <dqFailStep>STEP:1</dqFailStep>
    <getField>Article.MainSupplier->Party.Name</getField>
    <getField>Article.CatalogProxy->SupplierCatalog.Supplier->Party.Name</getField>
  </step>
  <step>
    <id>1</id>
    <entity>Article</entity>
    <workflowStatus>My task 1</workflowStatus>
    <description>My task 1</description>
    <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
    <enterStatus>Always</enterStatus>
    <batchSize>500</batchSize>
    <userType>supplier</userType>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Never</executeDq>
    <getField>Article.CurrentStatus</getField>
    <nextStep>STEP:2</nextStep>
    <nextStep>STEP:3</nextStep>
  </step>
  <step>
    <id>2</id>
    <entity>Article</entity>
    <updateFieldDescriptor>Article.CurrentStatus</updateFieldDescriptor>
    <updateFieldValue>200</updateFieldValue>
  </step>
  <step>
    <id>3</id>
    <entity>Article</entity>
    <workflowStatus>My task 2</workflowStatus>
    <description>My task 2</description>
    <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
    <enterStatus>Always</enterStatus>
    <batchSize>500</batchSize>
    <userType>userGroup</userType>
    <userName>Standardusers</userName>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Never</executeDq>
    <nextStep>STEP:4</nextStep>
  </step>
  <step>
    <id>4</id>
    <entity>Article</entity>
    <updateFieldDescriptor>Article.CurrentStatus</updateFieldDescriptor>
    <updateFieldValue>300</updateFieldValue>
  </step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

Step 0

Step 0 is an exact copy of the step 0 of the example 5 (incl. getField for supplier tasks).

Step 0

This steps checks only whether the item is already in this workflow. The difference to the previous step 0 steps is, that in this case the mandatory field "Article.MainSupplier->Party.Name " for supplier tasks is called. When you work with supplier tasks it is necessary that this getField command is part of the very first step of the StepWorkflow.

Key	Value	Description
id	0	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
batchSize	500	Batch size to use with trigger batching.
dqService	ItemsInWorkflowTasks-Process	This DQ service is an additional bpel inside the StepWorkflow which checks whether the item is already inside this workflow or not.
dqFailStep	STEP:1	The item will be moved to the step with the identifier 1 if the above dq fails.
getField	Article.MainSupplier->Party.Name	List of any fields directly on the entity that will be required. These entries must be fully qualified.

Key	Value	Description
getField	Article.CatalogProxy- >SupplierCatalog.Supplier- >Party.Name	List of any fields directly on the entity that will be required. These entries must be fully qualified.

### Step 1

This step puts the item into a workflow task and when leaving this task the steps 2 and 3 are executed.

Key	Value	Description
id	1	Identifier of the step.
entity	Article	Entity container for this workflow.
workflow Status	My task 1	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
descripti on	My task 1	Description of the status.
workflow ServiceE ndpoint	StepWorkflow- Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStat us	Always	The circumstances when an item should enter a status (task). In this case always!
batchSiz e	500	Batch size to use with trigger batching.
userType	supplier	Type of the default assignee of this task. In this case it will be assigned to the supplier of the cataslog.

Key	Value	Description
uiTemplate	Item approve UI	This is the default UI of the task.
executeDq	Never	The circumstances when a DQ should be run. In this case never!
getField	Article.CurrentStatus	List of any fields directly on the entity that will be required. These entries must be fully qualified.
nextStep	STEP:2	Identifier of the next step, The item will be moved to the next step, when this step is finished.
nextStep	STEP:3	Identifier of the next step, The item will be moved to the next step, when this step is finished.

## Step 2

Step 2 will manipulate data within the workflow. In this example we will change the field CurrentStatus of the entity Article.

Key	Value	Description
id	2	Same keys and values than in the example above.
entity	Article	Entity container for this workflow.
updateFieldDescriptor	Article.CurrentStatus	This field allows the step to update a field in Product 360. The field must be fully qualified.
updateFieldValue	200	This is the value for the update.

The next table shows you how to qualify field.



Key	Value
FieldDescriptor	Article.CurrentStatus
FieldDescriptor	Article.MainSupplier->Party.Name
FieldDescriptor	Article.ManufacturerName
FieldDescriptor	ArticleLang.DescriptionShort(9,"Default Channel")

### Step 3

This step puts the item again into a workflow task and when leaving this task the steps 4 is executed.

Key	Value	Description
id	3	Identifier of the step.
entity	Article	Entity container for this workflow.
workflow Status	My task 2	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	My task 2	Description of the status.
workflow ServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStatus	Always	The circumstances when an item should enter a status (task). In this case always!

Key	Value	Description
batchSize	500	Batch size to use with trigger batching.
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.
userName	Standardusers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".
uiTemplate	Item approve UI	This is the default UI of the task.
executeDq	Never	The circumstances when a DQ should be run. In this case never!
nextStep	STEP:4	Identifier of the next step, The item will be moved to the next step, when this step is finished.

#### Step 4

Step 4 is similar to step 2, but it will set another value to the field.

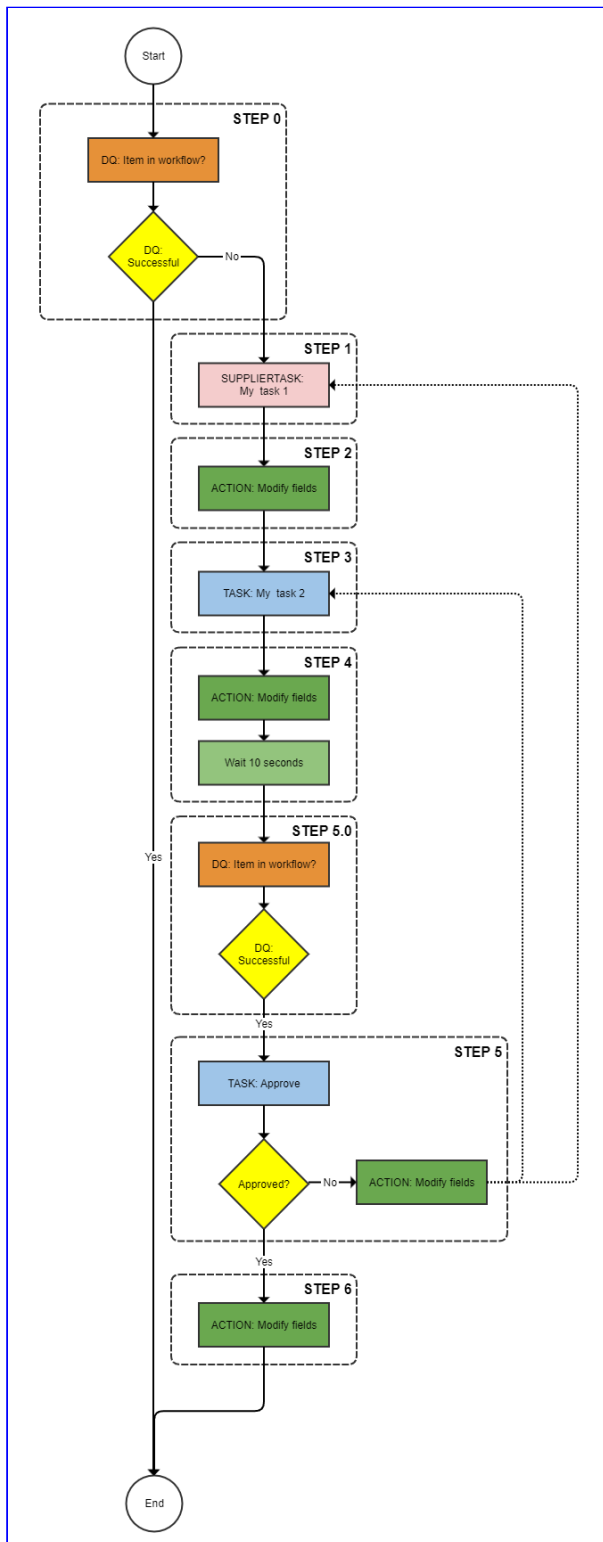
Key	Value	Description
id	4	Same keys and values than in the example above.
entity	Article	Entity container for this workflow.
updateFieldDescriptor	Article.CurrentStatus	This field allows the step to update a field in Product 360. The field must be fully qualified.

Key	Value	Description
updateFieldValue	300	This is the value for the update.

#### 2.3.5.10 Example 7 (2 Tasks + modifying of fields + approval step)

The next example extends the example 6 workflow with modifying of fields within the workflow after finishing the tasks followed by an approval task which also sets data if the task(s) are rejected.

Diagram for example 7



**Product 360 prerequisites for example 7**

Product 360 entity	value
Usergroup	Standardusers
Ui Template	Item approve UI

Explanation of the steps example 7

Step	Description
0	Check whether item is already in this workflow
1	Put the item into the supplier task "My task 1"
2	Modify the field Article.CurrentStatus to the value 200
3	Put the item into the task "My task 3"
4	Modify the field Article.CurrentStatus to the value 300 and wait 10 seconds
5.0	Check whether item is already in this workflow
5	Approve task with rejection options to all previous tasks. If rejected modify the field Article.CurrentStatus to the value 300
6	Modify the field Article.CurrentStatus to the value 400

StepWorkflow.xml file for example 7

### Example 7

```
<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 7</label>
    <identifier>Workflow_07</identifier>
    <version>1.0</version>
    <step>
      <id>0</id>
      <entity>Article</entity>
      <enterStatus>Never</enterStatus>
      <batchSize>500</batchSize>
      <executeDq>Always</executeDq>
      <dqService>ItemsInWorkflowTasks-Process</dqService>
      <dqFailStep>STEP:1</dqFailStep>
      <getField>Article.MainSupplier->Party.Name</getField>
    </step>
    <step>
      <id>1</id>
      <entity>Article</entity>
      <workflowStatus>My task 1</workflowStatus>
      <description>My task 1</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>Always</enterStatus>
      <batchSize>500</batchSize>
      <userType>supplier</userType>
      <uiTemplate>Item approve UI</uiTemplate>
      <executeDq>Never</executeDq>
      <getField>Article.CurrentStatus</getField>
      <nextStep>STEP:2</nextStep>
      <nextStep>STEP:3</nextStep>
    </step>
    <step>
      <id>2</id>
      <entity>Article</entity>
      <updateFieldDescriptor>Article.CurrentStatus</updateFieldDescriptor>
      <updateFieldValue>200</updateFieldValue>
    </step>
    <step>
      <id>3</id>
      <entity>Article</entity>
      <workflowStatus>My task 2</workflowStatus>
      <description>My task 2</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>Always</enterStatus>
```

```

    <batchSize>500</batchSize>
    <userType>userGroup</userType>
    <userName>Standardusers</userName>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Never</executeDq>
    <nextStep>STEP:4</nextStep>
  </step>
  <step>
    <id>4</id>
    <entity>Article</entity>
    <updateFieldDescriptor>Article.CurrentStatus</updateFieldDescriptor>
    <updateFieldValue>300</updateFieldValue>
    <nextStep>WorkflowWait-Process</nextStep>
    <wait>PT10S</wait>
    <nextStep>STEP:5.0</nextStep>
  </step>
  <step>
    <id>5.0</id>
    <entity>Article</entity>
    <enterStatus>Never</enterStatus>
    <batchSize>500</batchSize>
    <executeDq>Always</executeDq>
    <dqService>ItemsInWorkflowTasks-Process</dqService>
    <dqFailStep>STEP:5</dqFailStep>
  </step>
  <step>
    <id>5</id>
    <entity>Article</entity>
    <workflowStatus>Approve</workflowStatus>
    <description>Approval</description>
    <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
    <enterStatus>Always</enterStatus>
    <batchSize>500</batchSize>
    <userType>userGroup</userType>
    <userName>Standardusers</userName>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Never</executeDq>
    <rejectTrigger>StepWorkflow_ParallelTasks-Reject</rejectTrigger>
    <singleChoice>true</singleChoice>
    <rejectDecision>
      <id>p360.bpm.reject.STEP:1</id>
      <label>Reject to task 1</label>
    </rejectDecision>
    <rejectDecision>
      <id>p360.bpm.reject.STEP:3</id>
      <label>Reject to task 2</label>
    </rejectDecision>
    <rejectFieldDescriptor>Article.CurrentStatus</rejectFieldDescriptor>
    <rejectFieldValue>100</rejectFieldValue>
    <nextStep>STEP:6</nextStep>
  </step>
</step>

```

```

<id>6</id>
<workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
<entity>Article</entity>
<updateFieldDescriptor>Article.CurrentStatus</updateFieldDescriptor>
<updateFieldValue>400</updateFieldValue>
</step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

Step 0 - Step 3

These steps are exact copy of the example 6.

Step 4

Step 4 is nearly the same as the step 4 of the example above. Only the differences are shown in the next table.

Key	Value	Description
...	...	Same keys and values than in the example 6.
nextStep	WorkflowWait-Process	Wait process needed as step before continuing to avoid that the item is still in the workflow,
wait	PT10S	Value to wait (10 seconds).
nextStep	STEP:5.0	Identifier of the next step, The item will be moved to the next step, when this step is finished.

Step 5.0

This steps checks only whether the item is already in this workflow. If not the executed next step is 5. If yes the workflow will end to avoid that the item will go to the step 5 (Approve) until it is somewhere else in the workflow.



Key	Value	Description
id	5.0	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
batchSize	500	Batch size to use with trigger batching.
dqService	ItemsInWorkflowTasks-Process	This DQ service is an additional bpel inside the StepWorkflow which checks whether the item is already inside this workflow or not.
dqFailStep	STEP:5	The item will be moved to the step with the identifier 5 if the above DQ fails.

#### Step 5

This is the approval step with modifying a field while rejecting.

Key	Value	Description
id	5	Identifier of the step.
entity	Article	Entity container for this workflow.
workflowStatus	Approve	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	Approval	Description of the status.

Key	Value	Description
workflowServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStatus	Always	The circumstances when an item should enter a status (task). In this case always!
batchSize	500	Batch size to use with trigger batching.
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.
userName	Standardusers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".
uiTemplate	Item approve UI	This is the default UI of the task.
executeDq	Never	The circumstances when a DQ should be run. In this case never!
rejectTrigger	StepWorkflow_ParallelTasks-Reject	Endpoint for the reject trigger of parallel states (tasks).
singleChoice	true	This is for Rejects only and determines if you can have more than one choice. ==> The workflow can be rejected to only one step,
<rejectDecision>id	p360.bpm.reject.STEP:1	Identifier of the 1 <sup>st</sup> reject step.
<rejectDecision>label	Reject to task 1	Label of the 1 <sup>st</sup> rejection. This label will show up in the UI.

Key	Value	Description
<rejectDecision>id	p360.bpm.reject.STEP:3	Identifier of the 2 <sup>nd</sup> reject step.
<rejectDecision>label	Reject to task 2	Label of the 2 <sup>nd</sup> rejection. This label will show up in the UI.
rejectFieldDescriptor	Article.CurrentStatus	This is the field the will be modified if the task will be rected. The field must be fully qualified.
rejectFieldValue	100	This is the value for the update.
nextStep	STEP:6	Identifier of the next step, The item will be moved to the next step, when this step is approved.

#### Step 6

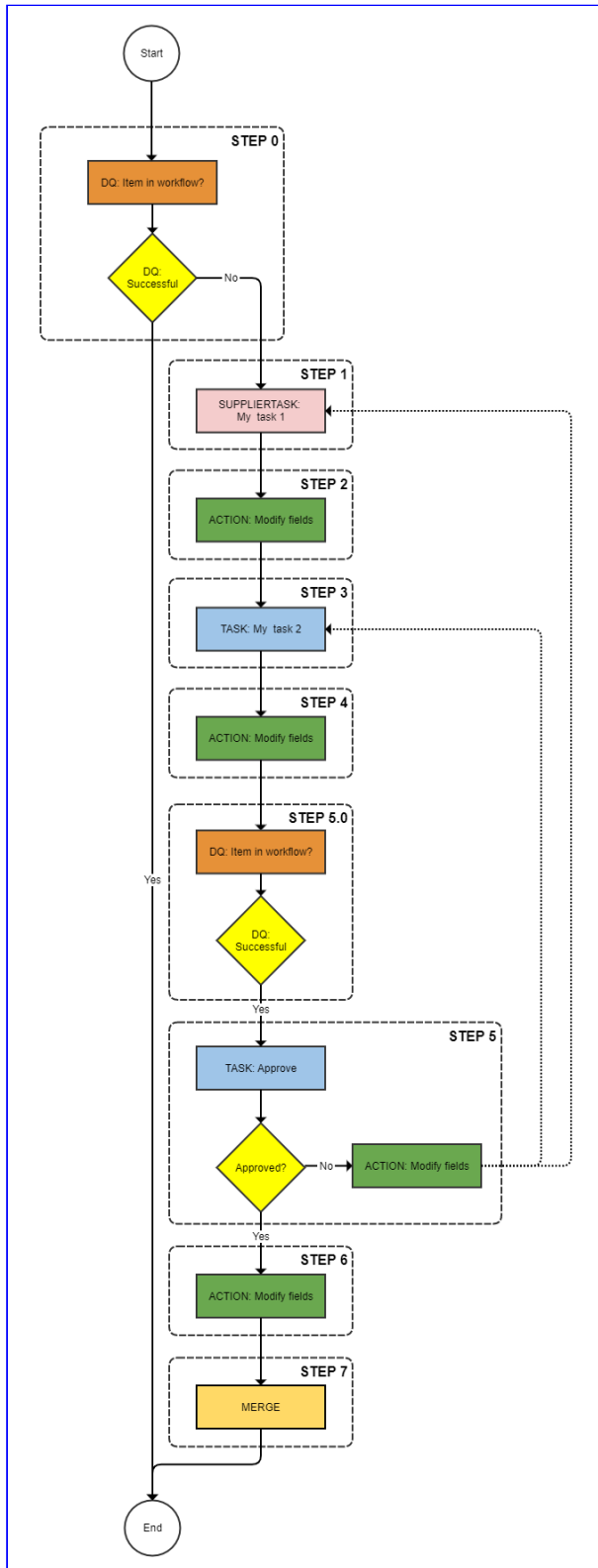
Step 2 will manipulate data within the workflow. In this example we will change the field CurrentStatus of the entity Article.

Key	Value	Description
id	6	Same keys and values than in the example 6.
entity	Article	Entity container for this workflow.
updateFieldDescriptor	Article.CurrentStatus	This field allows the step to update a field in Product 360. The field must be fully qualified.
updateFieldValue	400	This is the value for the update.

#### 2.3.5.11 Example 8 (2 Tasks modifying of fields approval step merge)

The next example extends the example 7 workflow with merging the item(s) after the approval.

Diagram for example 8



**Product 360 prerequisites for example 8**

Product 360 entity	value
Usergroup	Standardusers
Ui Template	Item approve UI

Explanation of the steps example 8

Step	Description
0	Check whether item is already in this workflow
1	Put the item into the supplier task "My task 1"
2	Modify the field Article.CurrentStatus to the value 200
3	Put the item into the task "My task 3"
4	Modify the field Article.CurrentStatus to the value 300
5.0	Check whether item is already in this workflow
5	Approve task with rejection options to all previous tasks. If rejected modify the field Article.CurrentStatus to the value 300
6	Modify the field Article.CurrentStatus to the value 400
7	Merge the item

StepWorkflow.xml file for example 8

### Example 8

```
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 8</label>
    <identifier>Workflow_08</identifier>
    <version>1.0</version>
    <step>
      <id>0</id>
      <entity>Article</entity>
      <enterStatus>Never</enterStatus>
      <batchSize>500</batchSize>
      <executeDq>Always</executeDq>
      <dqService>ItemsInWorkflowTasks-Process</dqService>
      <dqFailStep>STEP:1</dqFailStep>
      <getField>Article.MainSupplier->Party.Name</getField>
    </step>
    <step>
      <id>1</id>
      <entity>Article</entity>
      <workflowStatus>My task 1</workflowStatus>
      <description>My task 1</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>Always</enterStatus>
      <batchSize>500</batchSize>
      <userType>supplier</userType>
      <uiTemplate>Item approve UI</uiTemplate>
      <executeDq>Never</executeDq>
      <getField>Article.CurrentStatus</getField>
      <nextStep>STEP:2</nextStep>
      <nextStep>STEP:3</nextStep>
    </step>
    <step>
      <id>2</id>
      <entity>Article</entity>
      <updateFieldDescriptor>Article.CurrentStatus</updateFieldDescriptor>
      <updateFieldValue>200</updateFieldValue>
    </step>
    <step>
      <id>3</id>
      <entity>Article</entity>
      <workflowStatus>My task 2</workflowStatus>
      <description>My task 2</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>Always</enterStatus>
      <batchSize>500</batchSize>
```

```

    <userType>userGroup</userType>
    <userName>Standardusers</userName>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Never</executeDq>
    <nextStep>STEP:4</nextStep>
    <nextStep>STEP:5.0</nextStep>
  </step>
  <step>
    <id>4</id>
    <entity>Article</entity>
    <updateFieldDescriptor>Article.CurrentStatus</updateFieldDescriptor>
    <updateFieldValue>300</updateFieldValue>
  </step>
  <step>
    <id>5.0</id>
    <entity>Article</entity>
    <enterStatus>Never</enterStatus>
    <batchSize>500</batchSize>
    <executeDq>Always</executeDq>
    <dqService>ItemsInWorkflowTasks-Process</dqService>
    <dqFailStep>STEP:5</dqFailStep>
  </step>
  <step>
    <id>5</id>
    <entity>Article</entity>
    <workflowStatus>Approve</workflowStatus>
    <description>Approval</description>
    <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
    <enterStatus>Always</enterStatus>
    <batchSize>500</batchSize>
    <userType>userGroup</userType>
    <userName>Standardusers</userName>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Never</executeDq>
    <rejectTrigger>StepWorkflow_ParallelTasks-Reject</rejectTrigger>
    <singleChoice>true</singleChoice>
    <rejectDecision>
      <id>p360.bpm.reject.STEP:1</id>
      <label>Reject to task 1</label>
    </rejectDecision>
    <rejectDecision>
      <id>p360.bpm.reject.STEP:3</id>
      <label>Reject to task 2</label>
    </rejectDecision>
    <rejectFieldDescriptor>Article.CurrentStatus</rejectFieldDescriptor>
    <rejectFieldValue>100</rejectFieldValue>
    <nextStep>STEP:6</nextStep>
  </step>
  <step>
    <id>6</id>
    <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
    <entity>Article</entity>

```



```

    <updateFieldDescriptor>Article.CurrentStatus</updateFieldDescriptor>
    <updateFieldValue>400</updateFieldValue>
    <nextStep>STEP:7</nextStep>
  </step>
  <step>
    <id>7</id>
    <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
    <entity>Article</entity>
    <mergeProfile>profile_no_images</mergeProfile>
    <nextStep>MergeItems-Process</nextStep>
  </step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

Step 0 - Step 5

These steps are exact copy of the example 7.

Step 6

Step 6 is nearly the same as the step 5 of the example above. Only the differences are shown in the next table.

Key	Value	Description
...	...	Same keys and values than in the example above.
nextStep	STEP:7	Identifier of the next step, The item will be moved to the next step, when this step is finished.

Step 7

Step 7 will execute the merge.

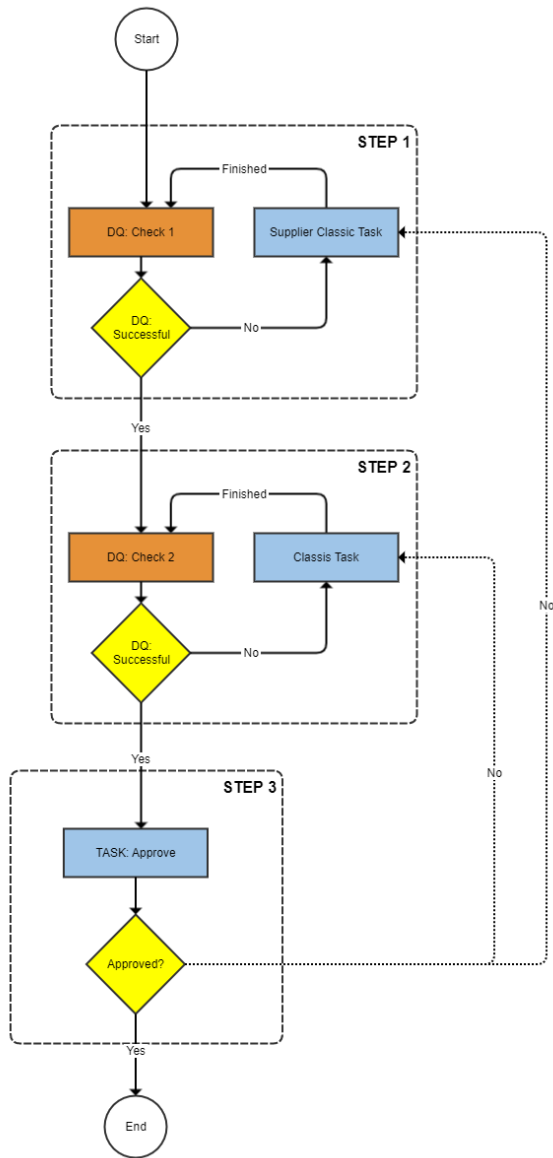
Key	Value	Description
id	7	

Key	Value	Description
workfl owServ iceEnd point	StepWorkflow- Trigger	
entity	Article	
merge Profile	profile_no_images	Profile of the merge process. There are 2 profiles within the standard workflow package. They are located in the folder profiles <b>merge-profile</b> of the StepWorkflow project. If you need additional profiles you have to add them to the project with your <b>BPM Designer</b> ,
nextSt ep	Mergeltems-Process	This bpel procoess allows you to merge items into the Master Catalog.

### 2.3.5.12 Example 9 (Classic Tasks)

In this example we are creating a classic (standard) tasks. The first classic task is a supplier task and the second one is a classic task for an user group. The last task will be again a workflow (approval) task, with the option to send the items back to a further step. The difference between a workflow and a classic task is when finishing a classic task all items (products or variants) within this task will be finished at once. It is highly recommended to use this approach only with report based events like export completed, import started etc., because there will be always a new task created. Independent whether the same task with the same name for the entire catalog already exists.

**Diagram for example 9**



Explanation of the steps example 9

Step	Description
1	Run DQ check "Channel 1" if fail ==>put the item into the supplier classic task
2	Run DQ check "Channel 2" if fail ==>put the item into the classic task

Step	Description
3	Approve task with rejection options to all previous tasks

**Product 360 prerequisites for example 9**

Product 360 entity	value
Usergroup	Standardusers
Ui Template	Item approve UI
DQ Channel	Channel 1
DQ Channel	Channel 2
user	jsmith
user	jsauer

StepWorkflow.xml file for example 9

**Example 9**

```

<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 9</label>
    <identifier>Workflow_09</identifier>
    <version>1.0</version>
    <step>
      <id>1</id>
      <entity>Article</entity>
      <batchSize>1000</batchSize>
      <getField>Article.MainSupplier->Party.Name</getField>
    
```

```

    <getField>Article.CatalogProxy->SupplierCatalog.Supplier->Party.Name</
getField>
    <uiTemplate>Item approve UI</uiTemplate>
    <classicTask>Supplier Classic Task</classicTask>
    <userType>supplier</userType>
    <escalation>PT12H</escalation>
    <substitute>jsmith</substitute>
    <deadline>PT24H</deadline>
    <responsible>jsauer</responsible>
    <description>Select each item and fix any rules that failed in the Quality
Status tab. Go back up to the task level, select Actions,Complete</description>
    <workflowServiceEndpoint>StepWorkflowClassic-Trigger</
workflowServiceEndpoint>
    <enterStatus>OnDqResults</enterStatus>
    <executeDq>Always</executeDq>
    <dqService>ExecuteDqOneChannel-Process</dqService>
    <dqChannel>Channel 1</dqChannel>
    <nextStep>STEP:2</nextStep>
</step>
<step>
    <id>2</id>
    <entity>Article</entity>
    <uiTemplate>Item approve UI</uiTemplate>
    <classicTask>Standard User Classic Task</classicTask>
    <userType>userGroup</userType>
    <userName>Standardusers</userName>
    <resetTask>true</resetTask>
    <escalation>P2D</escalation>
    <substitute>jsauer</substitute>
    <deadline>P4D</deadline>
    <responsible>pim</responsible>
    <description>This task is testing StepWorkflow using classic, or standard,
tasks.</description>
    <workflowServiceEndpoint>StepWorkflowClassic-Trigger</
workflowServiceEndpoint>
    <enterStatus>OnDqResults</enterStatus>
    <executeDq>Always</executeDq>
    <dqService>ExecuteDqOneChannel-Process</dqService>
    <dqChannel>Channel 2</dqChannel>
    <nextStep>STEP:3</nextStep>
</step>
<step>
    <id>3</id>
    <entity>Article</entity>
    <workflowStatus>Approve</workflowStatus>
    <description>Approval</description>
    <userType>userGroup</userType>
    <userName>Standardusers</userName>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Never</executeDq>
    <singleChoice>true</singleChoice>
    <rejectDecision>

```

```

        <id>p360.bpm.reject.STEP:1</id>
        <label>Send task back to supplier</label>
    </rejectDecision>
    <rejectDecision>
        <id>p360.bpm.reject.STEP:2</id>
        <label>Send task back to Standardusers</label>
    </rejectDecision>
    <rejectFieldDescriptor>Article.CurrentStatus</rejectFieldDescriptor>
    <rejectFieldValue>100</rejectFieldValue>
</step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

#### Step 1

This step will run a dq execution on channel "Chanel 1". If the result of this execution is false a classic task for the supplier will be created and the items will be put into this task.

Key	Value	Description
id	1	Identifier of the step.
entity	Article	Entity container for this workflow.
batchsize	1000	Batch size to use with trigger batching.
getField	Article.MainSupplier->Party.Name	List of any fields directly on the entity that will be required. These entries must be fully qualified.
getField	Article.CatalogProxy->SupplierCatalog.Supplier->Party.Name	List of any fields directly on the entity that will be required. These entries must be fully qualified.
uiTemplate	Item approve UI	This is the default UI of the task.
classicTask	Supplier Classic Task	Name of the classic task.

Key	Value	Description
userType	supplier	Type of the task. In this case a supplier task will be created.
escalation	PT12H	The escalation date of the task will be set to 12 hours after the creation data.
substitute	jsmith	Substitute user of this task.
deadline	PT24H	The deadline date of the task will be set to 1 day after the creation data.
responsible	jsauer	Responsible user for this task.
description	Select each item ....	Description of the task.
workflowServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStatus	OnDqResults	The circumstances when an item should enter the task. In this case only when dq fails!
executeDq	Always	The circumstances when a DQ should be run. In this case always!
dqService	ExecuteDqOneChannel-Process	This bpel service from InfaNextSteps executes a channel dq validation
dqChannel	Channel 1	Name of the DQ channel which will be executed.
nextStep	STEP:2	Identifier of the next step, The item will be moved to the next step, when this step is finished.

## Step 2

This step will run a dq execution on channel "Chanel 1 2". If the result of this execution is false a classic task for the usergroup "Standardusers" will be created and the items will be put into this task.

Key	Value	Description
...	...	Same keys and values than step 1.
id	2	Identifier of the step.
classicTask	Standard User Classic Task	Name of the classic task.
userType	userGroup	Type of the task. In this case a classic task for a user group will be created.
userName	Standardusers	Name of the user group.
resetTask		
escalation	P2D	The escalation date of the task will be set to 2 days after the creation data.
substitute	jsmith	Substitute user of this task.
deadline	P4D	The deadline date of the task will be set to 4 days after the creation data.
responsible	jsauer	Responsible user for this task.
description	Select each item ....	Description of the task.



Key	Value	Description
enterStatus	OnDqResults	The circumstances when an item should enter the task. In this case only when dq fails!
executeDq	Always	The circumstances when a DQ should be run. In this case always!
dqService	ExecuteDqOneChannel-Process	This bpel service from InfaNextSteps executes a channel dq validation
dqChannel	Channel 2	Name of the DQ channel which will be executed.
nextStep	STEP:3	Identifier of the next step, The item will be moved to the next step, when this step is finished.

### Step 3

Approve step with possibility to reject.

Key	Value	Description
id	3	Identifier of the step.
entity	Article	Entity container for this workflow.
workflowStatus	Approve	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	Approval	Description of the status.
workflowServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.

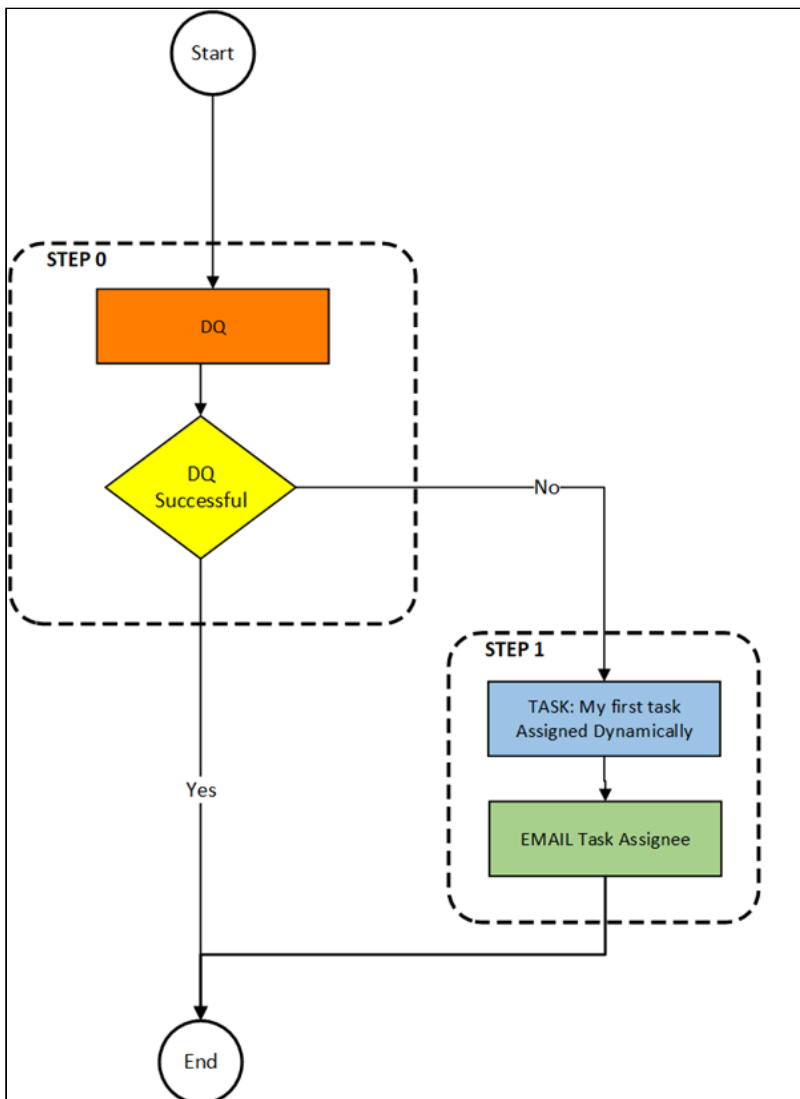
Key	Value	Description
enterStatus	Always	The circumstances when an item should enter a status (task). In this case always!
batchSize	500	Batch size to use with trigger batching.
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.
userName	Standardusers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".
uiTemplate	Item approve UI	This is the default UI of the task.
executeDq	Never	The circumstances when a DQ should be run. In this case never!
rejectTrigger	StepWorkflow_ParallelTasks-Reject	Endpoint for the reject trigger of parallel states (tasks).
singleChoice	true	This is for Rejects only and determines if you can have more than one choice. ==> The workflow can be rejected to a previous steps.
<rejectDecision>id	p360.bpm.reject.STEP:1	Identifier of the 1 <sup>st</sup> reject step.
<rejectDecision>label	Send task back to supplier	Label of the 1 <sup>st</sup> rejection. This label will show up in the UI.
<rejectDecision>id	p360.bpm.reject.STEP:2	Identifier of the 2 <sup>nd</sup> reject step.
<rejectDecision>label	Send task back to Standardusers	Label of the 2 <sup>nd</sup> rejection. This label will show up in the UI.

Key	Value	Description
rejectFieldDescriptor	Article.CurrentStatus	Field which will be modified if the item will be rejected.
rejectFieldValue	100	This value will be set during ejection.

#### 2.3.5.13 Example 10 (1 Task)

The tenth workflow puts items into a workflow task called "My first task" that is assigned to whatever user is in the field "Article.ManufacturerName". It also sends an email to that user. For this workflow task we need 2 steps. The first step checks whether the affected item is already in this workflow, if not it will move to the next step.. The second step puts the item in the task and sends the notification.

#### Diagram for example 10



Explanation of the steps example 10

Step	Description
0	Check whether item is already in this workflow.
1	Put the item into the task "My first task" and emails the user.

**Product 360 prerequisites for example 10**

Product 360 entity	value
User	The value in "Article.ManufacturerName" field.
Ui Template	Item approve UI

**Q Info**

Whatever user name is in "Article.ManufacturerName" is user that the task is assigned to. There is a new status added to the workflow with the user name appended to the task name. There is a delimiter "|" between the task name and the user name.

StepWorkflow.xml file for example 10

**Example 10**

```
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 10</label>
    <identifier>Workflow_10</identifier>
    <version>1.0</version>
    <taskUserDelimiter>|</taskUserDelimiter>
    <step>
      <id>0</id>
      <entity>Article</entity>
      <enterStatus>Never</enterStatus>
      <batchSize>500</batchSize>
      <executeDq>Always</executeDq>
      <dqService>ItemsInWorkflowTasks-Process</dqService>
      <dqFailStep>STEP:1</dqFailStep>
    </step>
    <step>
      <id>1</id>
      <entity>Article</entity>
      <workflowStatus>My first task</workflowStatus>
      <description>The description of my first workflow task.</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <getField>Article.SupplierAID</getField>
    </step>
  </workflow>
</imp:payload>
```

```

<getField>ArticleLang.DescriptionShort(9, "Default Channel")</getField>
<getField>ArticleLang.DescriptionLong(9, "Default Channel")</getField>
<getField>ArticleLog.ModificationDate(HPM)</getField>
<enterStatus>Always</enterStatus>
<notificationService>NotifyUser-Process</notificationService>
<batchSize>500</batchSize>
<userType>user</userType>
<userName>SERVICE:UserTask-Workflow</userName>
<userDescriptor>Article.ManufacturerName</userDescriptor>
<includeLabels>true</includeLabels>
<uiTemplate>Item approve UI</uiTemplate>
<executeDq>Never</executeDq>
</step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

#### Step 0

This step checks only whether the item is already in this workflow. If not the executed next step is 1. If yes the workflow will end because there is no next step defined.

Key	Value	Description
id	0	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
id	0	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
batchSize	500	Batch size to use with trigger batching.

Key	Value	Description
dqService	ItemsInWorkflowTasks-Process	This DQ service is an additional bpel inside the StepWorkflow which checks whether the item is already inside this workflow or not.
dqFailStep	STEP:1	The item will be moved to the step with the identifier 1 if the above dq succeeds.
taskDelimiter		The delimiter that goes between the original task name and the user name.

#### Step 1

This step enters the items in the task called “My first task” and emails the user this task is assigned to.

Key	Value	Description
id	1	Identifier of the step.
entity	Article	Entity container for this workflow.
workflow Status	My first task	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	My first workflow task	Description of the status.
workflow ServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStatus	Always	The circumstances when an item should enter a status (task). In this case always

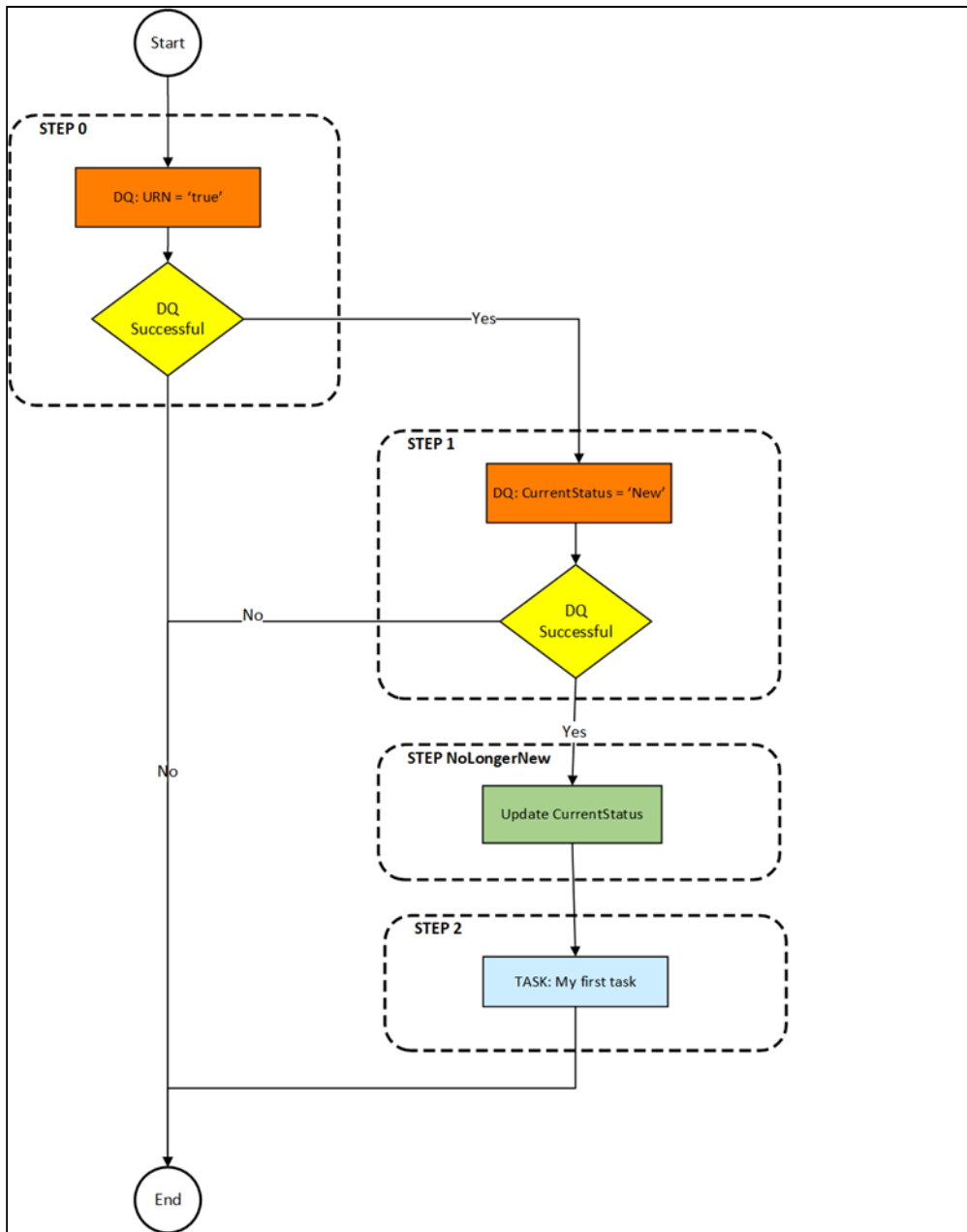
Key	Value	Description
userType	user	Type of the default assignee of this task. In this case it will be a user.
userName	SERVICE:UserTask-Workflow	The service for the bpel process that will determine the user or usergroup to assign the task to.
userDescriptor	Article.ManufacturerName	The field which stores the user name to assign the task to for dynamic users.
uiTemplate	Item approve UI	This is the default UI of the task.
executeDQ	Never	The circumstances when a DQ should be run. In this case never!
notificationService	NotifyUser-Process	The service for the bpel process that will perform the email service.
includeLabels	true	This will allow us to receive the item number in addition to the item ID.
getField	Several field names	This will get the field values for the items from P360. By default, for NotifyUser, these will be included in an html table in the email.

#### 2.3.5.14 Example 11 (1 Task with DQ checks)

The eleventh workflow puts items into a workflow task called "My first task". The first step checks whether the urn value for workflow\_11.active is true, if not, the workflow will end. If it is "true", it will move to the next step. The second step checks to see if the affected items have a Current Status of 100, or "01 New" on most systems. If it is not, then the items will not go into the task and the workflow will end. If so, it will move to the next step. The 3<sup>rd</sup> step updates the current status of the items to 200. The 4<sup>th</sup> step puts the item in the task.



Diagram for example 11



Explanation of the steps example 11

Step	Description
0	Check whether urn is set to true.

Step	Description
1	Check if the CurrentStatus of item is "01 New"
NoLongerNew	Update Current Status to something else/
2	Put the item into the task "My first task"

#### Product 360 prerequisites for example 11

Product 360 entity	value
UserGroup	Standardusers
Ui Template	Item approve UI
Article.CurrentStatus	100
ActiveVOS Console	value
Urn:workflow_11.active	true



#### Q Info

The DQ Service "dqValueCheck-Process" is used for both DQ's. There are 3 elements of the rule: [TYPE] | [KEY] | [VALUE] The types of comparisons are not case sensitive and are:

- \* Descriptor: <dqRule>Descriptor|Article.CurrentStatus|100</dqRule>
- \* Name: <dqRule>Name|Status|100</dqRule>
- \* URN value: <dqRule> URN|urn:skip.merge.task|true</dqRule>

StepWorkflow.xml file for example 11

### Example 11

```
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 11</label>
    <identifier>Workflow_11</identifier>
    <version>1.0</version>
    <step>
      <id>0</id>
      <entity>Article</entity>
      <enterStatus>Never</enterStatus>
      <batchSize>5000</batchSize>
      <executeDq>Always</executeDq>
      <getField>Article.CurrentStatus</getField>
      <dqService>dqValueCheck-Process</dqService>
      <dqRule>urn|urn:workflow_11.active|true</dqRule>
      <nextStep>STEP:1</nextStep>
    </step>
    <step>
      <id>1</id>
      <entity>Article</entity>
      <enterStatus>Never</enterStatus>
      <batchSize>5000</batchSize>
      <executeDq>Always</executeDq>
      <getField>Article.CurrentStatus</getField>
      <dqService>dqValueCheck-Process</dqService>
      <dqRule>descriptor|Article.CurrentStatus|100</dqRule>
      <nextStep>STEP:NoLongerNew</nextStep>
      <nextStep>STEP:2</nextStep>
    </step>
    <step>
      <id>NoLongerNew</id>
      <entity>Article</entity>
      <updateFieldDescriptor>Article.CurrentStatus</updateFieldDescriptor>
      <updateFieldValue>200</updateFieldValue>
    </step>
    <step>
      <id>2</id>
      <entity>Article</entity>
      <workflowStatus>My first task</workflowStatus>
      <description>My first workflow task</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>Always</enterStatus>
      <batchSize>500</batchSize>
      <userType>userGroup</userType>
```

```

    <userName>Standardusers</userName>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Never</executeDq>
  </step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

#### Step 0

This step checks only whether the item is already in this workflow. If not the executed next step is 1. If yes the workflow will end because there is no next step defined.

Key	Value	Description
id	0	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
dqService	dqValueCheck-Process	This DQ service is an additional bpel inside the StepWorkflow which checks whether the item is already inside this workflow or not.
dqRule	urn urn:workflow_11.active true	Checks to see if urn:workflow_11.active is "true"
nextStep	STEP:1	The item will be moved to the step with the identifier 1 if the above dq succeeds.

#### Step 1

This step checks to see if the Current Status is 100. If yes, the workflow will move to the next step2 defined as NoLongerNew and 2.

Key	Value	Description
id	1	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
dqService	dqValueCheck-Process	This DQ service is an additional bpel inside the StepWorkflow which checks whether the item is already inside this workflow or not.
dqRule	descriptor Article.CurrentStatus 100	Checks to see if Current Status is 100, or "01 New"
nextStep	STEP: NoLongerNew  STEP:2	The item will be moved to the step with the identifier 1 if the above dq succeeds.

#### Step NoLongerNew

This step updates the Current Status field.

Key	Value	Description
id	NoLongerNew	Identifier of the step.
entity	Article	Entity container for this workflow.
updateFieldDescriptor	Article.CurrentStatus	The descriptor for the field being updated.

Key	Value	Description
updateFieldValue	200	The value to update the field with.

## Step 2

This step puts the affected items in the "My first Task" task.

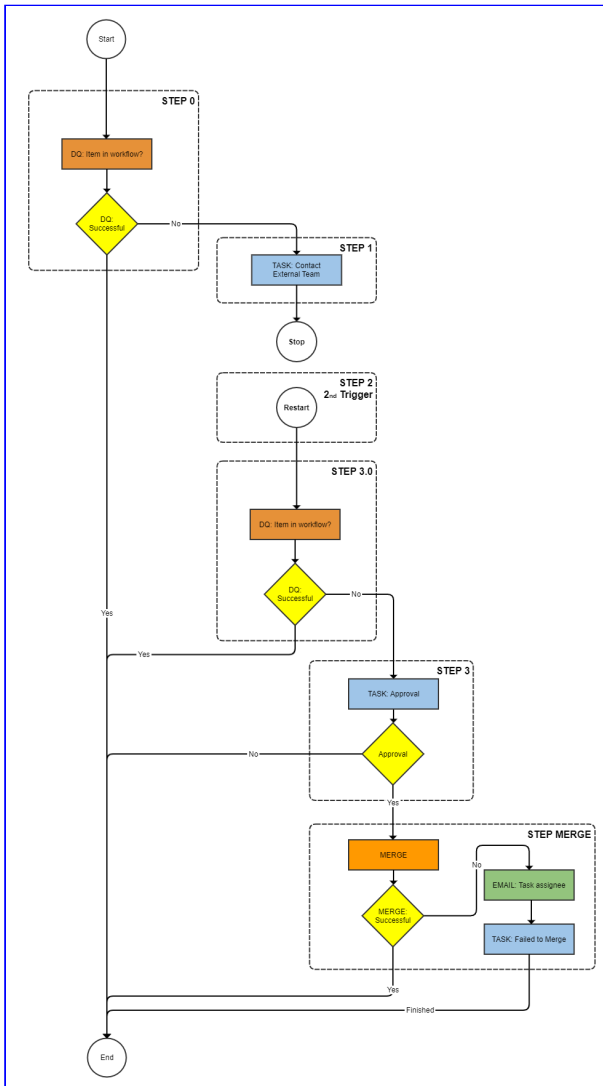
Key	Value	Description
id	2	Identifier of the step.
entity	Article	Entity container for this workflow.
workflow Status	My first task	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	My first workflow task	Description of the status.
workflow ServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStatus	Always	The circumstances when an item should enter a status (task). In this case always
userType	userGroup	Type of the default assignee of this task. In this case it will be a user.
userName	Standardusers	The service for the bpel process that will determine the user or usergroup to assign the task to.

Key	Value	Description
executeDq	Never	The circumstances when a DQ should be run. In this case never!

#### 2.3.5.15 Example 12 (2 Triggers and Merge with Results)

The twelfth workflow puts items into a workflow task called "Contact external team". The first step checks whether the affected item is already in this workflow, if not it will be put into the task. The second step puts the items in the "Contact external team" task. The workflow then stops until it is triggered again by an import, export, or entity change. Once the workflow is triggered again, it moves to an approval task. Approved items go to the next step and rejected items end the workflow. In the last step, the affected items merge. Because it is set up as a dq, any items that fail to merge go into a "Failed to Merge" task.

Diagram for example 12



Explanation of the steps example 12

Step	Description
0	Check whether item is already in this workflow.
1	Put the item into the task "Contact external team".
2 <sup>nd</sup> Trigger	This step is triggered by an import complete, export started, or an entity changed. It then moves to Step 3.



Step	Description
3.0	Check whether item is already in this workflow.
3	Put the item into an approval task. Items that are rejected end the workflow. Ones that are approved move to step Merge.
Merge	Merge the items as a DQ. if fail ==>put the item into the task "Failed to Merge" and send an email.

**Product 360 prerequisites for example 12**

Product 360 entity	value
Usergroup	Standardusers
Ui Template	Item approve UI

**Q Info**

Standarduser is the Identifier of the Usergroup. This workflow stops after step one until a new trigger is generated for the step "2<sup>nd</sup> Trigger". This workflow also features a merge that runs as a DQ. The items will merge, but if any fail, they will enter a merge failed task.

StepWorkflow.xml file for example 12

**Example 12**

```
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 12</label>
    <identifier>Workflow_12</identifier>
    <version>1.0</version>
    <taskUserDelimiter>|</taskUserDelimiter>
    <updateWorkflowComments>true</updateWorkflowComments>
  </workflow>
</imp:payload>
```

```

<commentDescriptor>ArticleLang.Remarks(9, "Default Channel")</commentDescriptor>
<step>
  <id>0</id>
  <entity>Article</entity>
  <enterStatus>Never</enterStatus>
  <batchSize>500</batchSize>
  <executeDq>Always</executeDq>
  <dqService>ItemsInWorkflowTasks-Process</dqService>
  <dqFailStep>STEP:1</dqFailStep>
</step>
<step>
  <id>1</id>
  <entity>Article</entity>
  <workflowStatus>Contact external team</workflowStatus>
  <description>Contact external team and finish the task. Once the import is
completed the workflow will pick up where it left off.</description>
  <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
  <enterStatus>Always</enterStatus>
  <userType>userGroup</userType>
  <userName>Standardusers</userName>
  <uiTemplate>Item approve UI</uiTemplate>
  <executeDq>Never</executeDq>
</step>
<step>
  <id>2nd Trigger</id>
  <entity>Article</entity>
  <triggerConfiguration>Example12_ImportCompleted_2nd</triggerConfiguration>
  <triggerConfiguration>Example12_ExportStarted_2nd</triggerConfiguration>
  <triggerConfiguration>Example12_EntityChange_2nd</triggerConfiguration>
  <nextStep>STEP:3.0</nextStep>
</step>
<step>
  <id>3.0</id>
  <entity>Article</entity>
  <enterStatus>Never</enterStatus>
  <batchSize>500</batchSize>
  <executeDq>Always</executeDq>
  <dqService>ItemsInWorkflowTasks-Process</dqService>
  <dqFailStep>STEP:3</dqFailStep>
</step>
<step>
  <id>3</id>
  <entity>Article</entity>
  <workflowStatus>Approve</workflowStatus>
  <description>Approval</description>
  <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
  <enterStatus>Always</enterStatus>
  <batchSize>500</batchSize>
  <userType>userGroup</userType>
  <userName>Standardusers</userName>
  <uiTemplate>Item approve UI</uiTemplate>
  <executeDq>Never</executeDq>

```

```

<singleChoice>>false</singleChoice>
<rejectDecision>
  <id>p360.bpm.reject</id>
  <label>Reject</label>
</rejectDecision>
<nextStep>WorkflowWait-Process</nextStep>
<wait>PT5S</wait>
<nextStep>STEP:Merge</nextStep>
</step>
<step>
  <id>Merge</id>
  <entity>Article</entity>
  <workflowStatus>Failed to Merge</workflowStatus>
  <description>These items failed to merge.</description>
  <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
  <enterStatus>OnDqResults</enterStatus>
  <userType>user</userType>
  <userName>pim</userName>
  <uiTemplate>Item approve UI</uiTemplate>
  <executeDq>Always</executeDq>
  <dqService>MergeItemsResults-Process</dqService>
  <mergeProfile>profile_no_images</mergeProfile>
  <dqFailStep>NotifyUser-Process</dqFailStep>
</step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

#### Step 0

This step checks only whether the item is already in this workflow. If not the executed next step is 1. If yes the workflow will end because there is no next step defined.

Key	Value	Description
id	0	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
batchSize	500	Batch size to use with trigger batching.

Key	Value	Description
dqService	ItemsInWorkflowTasks-Process	This DQ service is an additional bpel inside the StepWorkflow which checks whether the item is already inside this workflow or not.
dqFailStep	STEP:1	The item will be moved to the step with the identifier 1 if the above dq fails.

### Step 1

This step puts the affected items in a task. If the task is finished, the workflow will stop.

Key	Value	Description
id	1	Identifier of the step.
entity	Article	Entity container for this workflow.
workflow Status	Contact external team	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	Contact external team and finish the task. Once the import is completed the workflow will pick up where it left off.	Description of the status.
workflow ServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStatus	Always	The circumstances when an item should enter a status (task). In this case always!

Key	Value	Description
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.
userName	Standardusers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".
uiTemplate	Item approve UI	This is the default UI of the task.
executeDq	Never	The circumstances when a DQ should be run. In this case never!

#### Step 2nd Trigger

Once this step is triggered, it just moves to the next step.

Key	Value	Description
id	2nd Trigger	Identifier of the step.
entity	Article	Entity container for this workflow.
triggerConfiguration	Example12_ImportCompleted_2 <sup>nd</sup> Example12_ExportStarted_2 <sup>nd</sup> Example12_EntityChange_2nd	The 3 different trigger configurations that can trigger the step
nextStep	STEP:3.0	The item will be moved to the step.

## Step 0

This step checks only whether the item is already in this workflow. If not the executed next step is 1. If yes the workflow will end because there is no next step defined.

Key	Value	Description
id	3.0	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
dqService	ItemsInWorkflowTasks-Process	This DQ service is an additional bpel inside the StepWorkflow which checks whether the item is already inside this workflow or not.
dqFailStep	STEP:3	The item will be moved to the step with the identifier 1 if the above dq fails.

## Step 3

This is the approval step that moves to Merge if approved while workflow ends for rejected items.

Key	Value	Description
id	3	Identifier of the step.
entity	Article	Entity container for this workflow.
workflow Status	Approve	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]

Key	Value	Description
descripti on	Approval	Description of the status.
workflow ServiceE ndpoint	StepWorkflow- Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStat us	Always	The circumstances when an item should enter a status (task). In this case always!
batchSiz e	500	Batch size to use with trigger batching.
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.
userNam e	Standardusers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".
uiTempla te	Item approve UI	This is the default UI of the task.
executeD q	Never	The circumstances when a DQ should be run. In this case never!
singleCh oice	false	This is for Rejects only and determines if you can have more than one choice. ==> The workflow can be rejected to only one step,
<rejectDe cision>id	p360.bpm.reject	Identifier of the reject step. Since none is added, it will not go to a task.

Key	Value	Description
<rejectDecision>label	Reject	Label of the rejection. This label will show up in the UI.
nextStep	WorkflowWait-Process	This step will wait and then go to the next nextStep.
wait	PT5S	Standard duration notation. This is 5 seconds.
nextStep	STEP:Merge	Identifier of the next step, The item will be moved to the next step, when this step is approved.

#### Step Merge

This step does the merge through a DQ process. The items will merge, but if any fail, they will enter a merge failed task.

Key	Value	Description
id	Merge	Identifier of the step.
entity	Article	Entity container for this workflow.
workflow Status	Failed to Merge	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	These items failed to merge.	Description of the status.
workflow ServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.

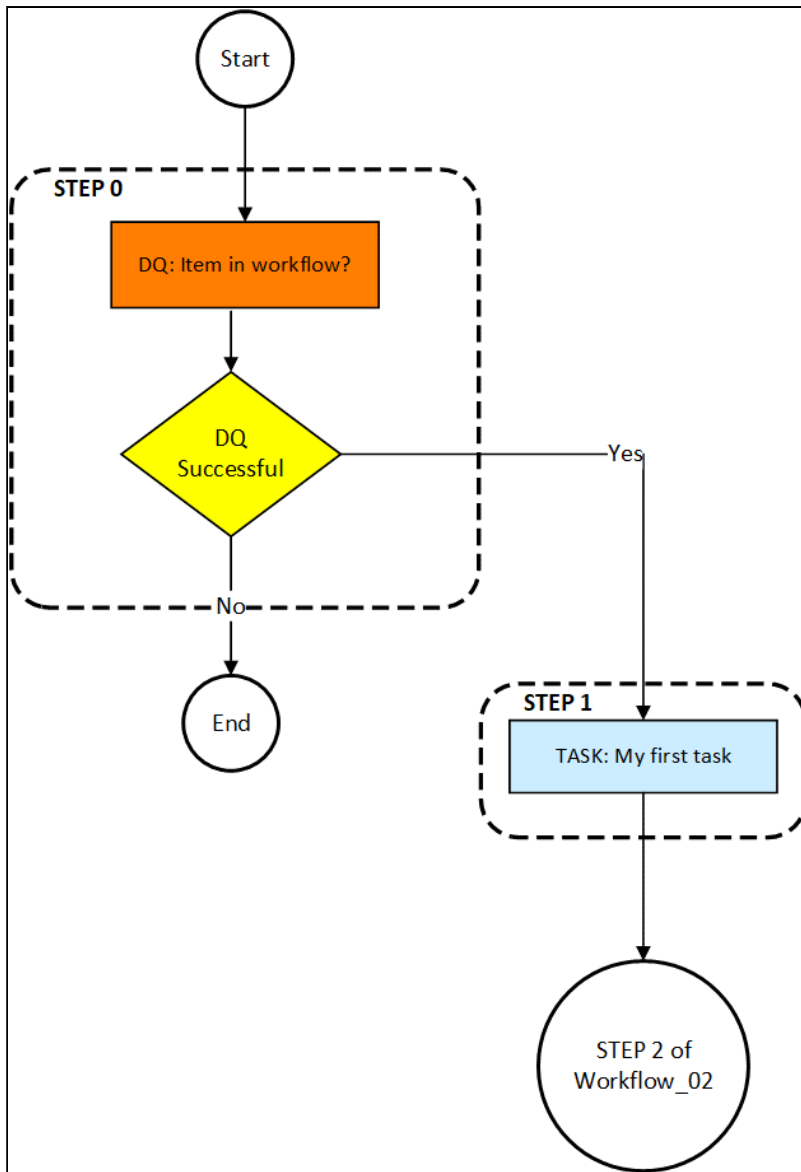


Key	Value	Description
enterStatus	OnDqResults	The circumstances when an item should enter a status (task). In this case only when DQ check will fail!
userType	user	Type of the default assignee of this task. In this case it will be a user.
userName	pim	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the user "pim".
uiTemplate	Item approve UI	This is the default UI of the task.
executeDq	Always	The circumstances when a DQ should be run. In this case always!
dqService	MergeItemsResults-Process	The service name of the process that will be handle the processing of the DQ. In this case, it will merge the items and return Success for those that do not have issues and Failed for those that do.
dqFailStep	NotifyUser-Process	This will send an email containing all items that failed to merge if any did.

#### 2.3.5.16 Example 13 (Workflow starts another workflow)

The thirteenth workflow puts items into a workflow task called "My first task" and then to another workflow. The first step checks whether the affected item is already in this workflow, if not it will be put into the task. The second step puts the items into the task. Once the items are finished in the task, they will move to another workflow at a specific step.

**Diagram for example 13**



Explanation of the steps example 13

Step	Description
0	Check whether item is already in this workflow.

Step	Description
1	Put the item into the task "My first task". When that task is finished, send the items to a step in another workflow

**Product 360 prerequisites for example 13**

Product 360 entity	value
Usergroup	Standardusers
Ui Template	Item approve UI
Workflow	Workflow_02

**Q Info**

Standarduser is the Identifier of the Usergroup!

StepWorkflow.xml file for example 13

**Example 13**

```
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 13</label>
    <identifier>Workflow_13</identifier>
    <version>1.0</version>
    <step>
      <id>0</id>
      <entity>Article</entity>
      <enterStatus>Never</enterStatus>
      <batchSize>500</batchSize>
      <executeDq>Always</executeDq>
      <dqService>ItemsInWorkflowTasks-Process</dqService>
    </step>
  </workflow>
</imp:payload>
```

```

    <dqFailStep>STEP:1</dqFailStep>
  </step>
  <step>
    <id>1</id>
    <entity>Article</entity>
    <workflowStatus>My first task</workflowStatus>
    <description>My first workflow task</description>
    <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
    <enterStatus>Always</enterStatus>
    <batchSize>500</batchSize>
    <userType>userGroup</userType>
    <userName>Standardusers</userName>
    <uiTemplate>Item approve UI</uiTemplate>
    <executeDq>Never</executeDq>
    <changeWorkflowLabel>Example Workflow 2</changeWorkflowLabel>
    <changeWorkflowIdentifier>Workflow_02</changeWorkflowIdentifier>
    <changeWorkflowStep>2</changeWorkflowStep>
    <nextStep>ChangeWorkflow-Process</nextStep>
  </step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

#### Step 0

This step checks only whether the item is already in this workflow. If not the executed next step is 1. If yes the workflow will end because there is no next step defined.

Key	Value	Description
id	0	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
batchSize	500	Batch size to use with trigger batching.

Key	Value	Description
dqService	ItemsInWorkflowTasks-Process	This DQ service is an additional bpel inside the StepWorkflow which checks whether the item is already inside this workflow or not.
dqFailStep	STEP:1	The item will be moved to the step with the identifier 1 if the above dq fails.

## Step 1

This step moves the items into a task called "My first task". When the items are finished by a user, they will move into "Example Workflow 2" step "2".

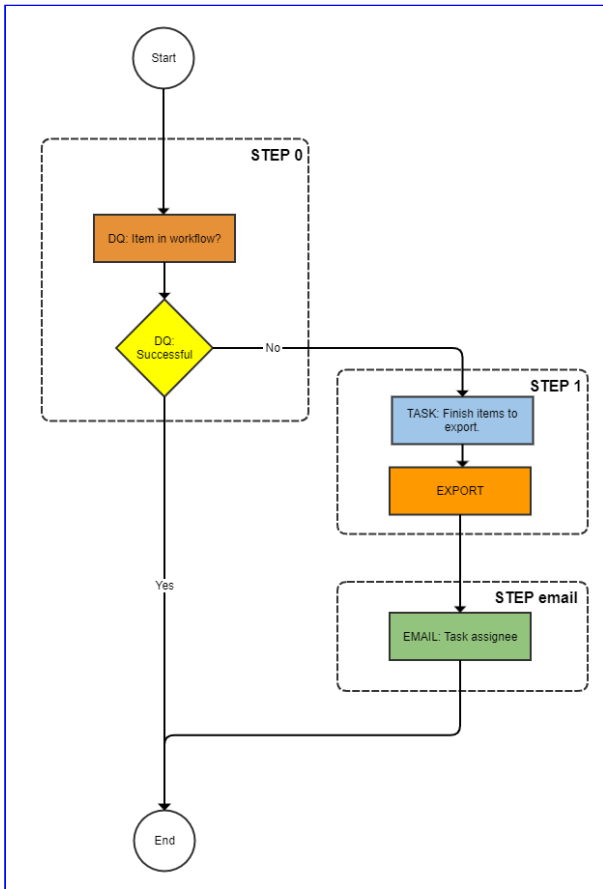
Key	Value	Description
id	1	Identifier of the step.
entity	Article	Entity container for this workflow.
workflow Status	My first task	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	My first workflow task	Description of the status.
workflow ServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStatus	Always	The circumstances when an item should enter a status (task). In this case always!
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.

Key	Value	Description
userName	Standardusers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".
uiTemplate	Item approve UI	This is the default UI of the task.
executeDq	Never	The circumstances when a DQ should be run. In this case never!
changeWorkflowLabel	Example Workflow 2	The label of the workflow the item is going to.
changeWorkflowIdentifier	Workflow_02	The identifier of the workflow the item is going to.
changeWorkflowStep	2	The step of the workflow the item is going into.
nextStep	ChangeWorkflow-Process	This process will send the item to another workflow and step based on the 3 previous parameters.

#### 2.3.5.17 Example 14 (Export items)

The 14 workflow puts items into a workflow task called "Finish items to export.". The first step checks whether the affected item is already in this workflow, if not it will be put into the task. The next step will schedule an immediate export with the profile "Item - Short, long descriptions" and the last step will send an email to the the user which finished the task.

**Diagram for example 14**



**Explanation of the steps example 14**

Step	Description
0	Check whether item is already in this workflow
1	Put the item into the task "Finish items to export." and exports it after task is finished.
email	Send email to the user

**Product 360 prerequisites for example 14**

Product 360 entity	value
UserGroup	Standardusers
Ui Template	Item approve UI
Export profile	Item - Short, long descriptions

StepWorkflow.xml file for example 14

**Example 14**

```
<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 14</label>
    <identifier>Workflow_14</identifier>
    <version>1.0</version>
    <taskUserDelimiter>|</taskUserDelimiter>
    <step>
      <id>0</id>
      <entity>Article</entity>
      <enterStatus>Never</enterStatus>
      <batchSize>500</batchSize>
      <executeDq>Always</executeDq>
      <dqService>ItemsInWorkflowTasks-Process</dqService>
      <dqFailStep>STEP:1</dqFailStep>
    </step>
    <step>
      <id>1</id>
      <entity>Article</entity>
      <workflowStatus>Export Items</workflowStatus>
      <description>Finish items to export.</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>Always</enterStatus>
      <batchSize>500</batchSize>
      <userType>userGroup</userType>
      <userName>Standardusers</userName>
    </step>
  </workflow>
</imp:payload>
</payload>
```



```

<uiTemplate>Item approve UI</uiTemplate>
<exportProfile>Item - Short, long descriptions</exportProfile>
<nextStep>ScheduleExport-Process</nextStep>
<nextStep>STEP:email</nextStep>
</step>
<step>
  <id>email</id>
  <entity>Article</entity>
  <enterStatus>Never</enterStatus>
  <batchSize>500</batchSize>
  <includeComment>true</includeComment>
  <userType>userGroup</userType>
  <userName>Standardusers</userName>
  <getField>Article.SupplierAID</getField>
  <getField>ArticleLang.DescriptionShort(9, "Default Channel")</getField>
  <getField>ArticleLang.DescriptionLong(9, "Default Channel")</getField>
  <emailProfile>exportStarted</emailProfile>
  <nextStep>NotifyUser-Process</nextStep>
</step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

#### Step 0

This steps checks only whether the item is already in this workflow. If not the executed next step is 1. If yes the workflow will end because there is no next step defined.

Key	Value	Description
id	0	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
batchSize	500	Batch size to use with trigger batching.

Key	Value	Description
dqService	ItemsInWorkflowTasks-Process	This DQ service is an additional bpel inside the StepWorkflow which checks whether the item is already inside this workflow or not.
dqFailStep	STEP:1	The item will be moved to the step with the identifier 1 if the above dq fails.

#### Step 1

Put the item into the task "Finish items to export." and exports it after task is finished.

Key	Value	Description
id	1	Identifier of the step.
entity	Article	Entity container for this workflow.
workflow Status	Export Items	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	Finish items to export.	Description of the status.
workflow ServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStatus	Always	The circumstances when an item should enter a status (task). In this case always!
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.

Key	Value	Description
userName	Standardusers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".
uiTemplate	Item approve UI	This is the default UI of the task.
exportProfile	Item - Short, long descriptions	Export profile
nextStep	ScheduleExport-Process	The service for the bpel process that will perform the export service.
nextStep	STEP:email	The item will be moved to the step.

#### Step email

This step puts the affected items in the "My first Task" task.

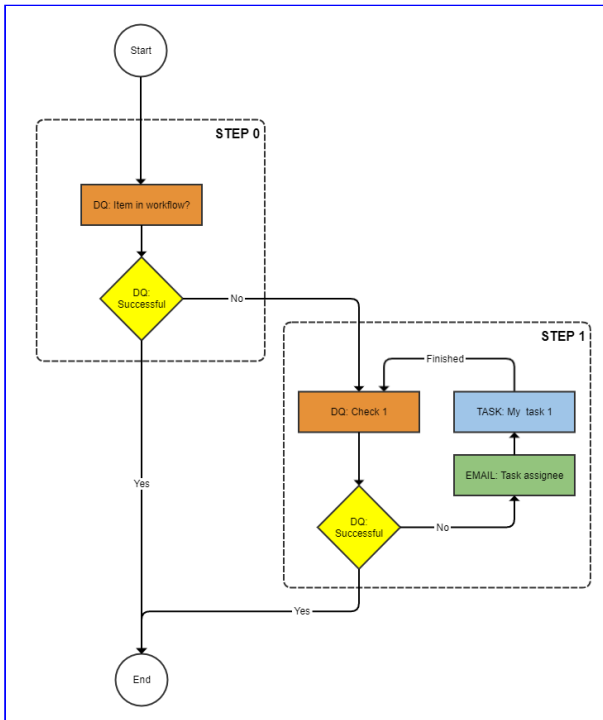
Key	Value	Description
id	email	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case always
userType	userGroup	Type of the default assignee of this task. In this case it will be a user.
userName	Standardusers	The service for the bpel process that will determine the user or usergroup to send the email.

Key	Value	Description
getField	Article.SupplierAI D	Additional info field for the email.
getField	ArticleLang.Descri ptionShort(9, "Default Channel")	Additional info field for the email.
getField	ArticleLang.Descri ptionShort(9, "Default Channel")	Additional info field for the email.
nextStep	NotifyUser- Process	The service for the bpel process that will perform the email service.

### 2.3.5.18 Example 15 (1 Task with DQ check and comments)

The 15th example extends the example 1 with a data quality check as well as enabling the comment functionality.

**Diagram for example 15**



**Product 360 prerequisites for example 15**

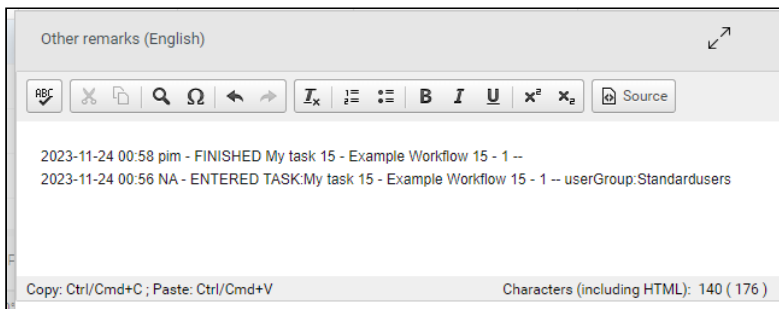
Product 360 entity	value
Usergroup	Standardusers
Ui Template	Item approve UI
DQ Channel	Channel 1

Explanation of the steps example 15

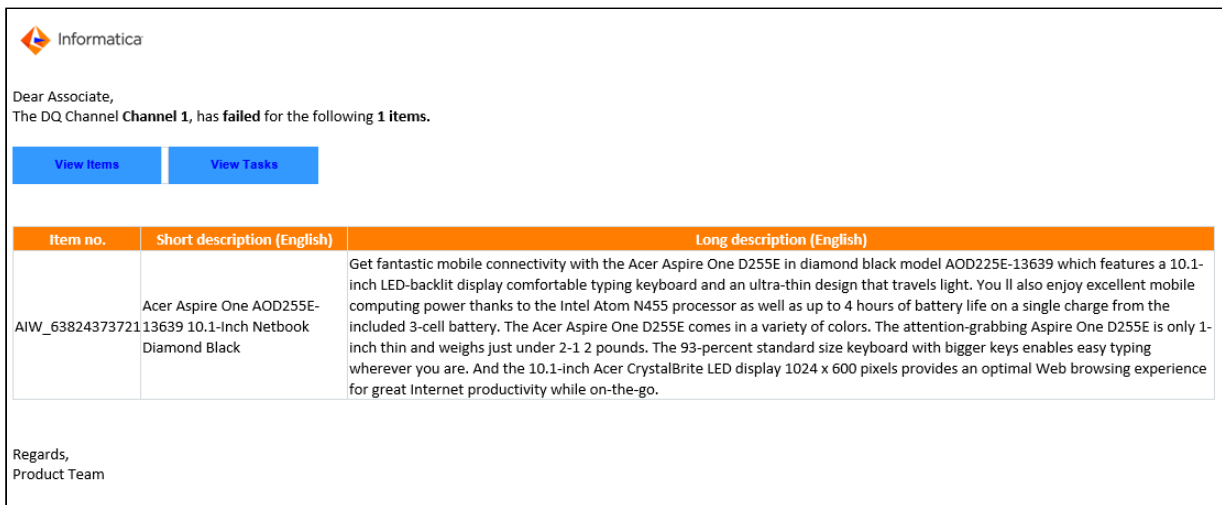
Step	Description
0	Check whether item is already in this workflow.

Step	Description
1	Run DQ check "Channel 1" if fail ==>put the item into the task "My task 1" and send a mail to the task assignee and persisting the comment.

### Example screenshot comments



### Example screenshot email



### StepWorkflow.xml file for example 15



```

<label>Example Workflow 15</label>
<identifier>Workflow_15</identifier>
<version>1.0</version>
<!-- the next line enables the comments-->
<updateWorkflowComments>true</updateWorkflowComments>
<step>
  <id>0</id>
  <entity>Article</entity>
  <enterStatus>Never</enterStatus>
  <batchSize>500</batchSize>
  <executeDq>Always</executeDq>
  <dqService>ItemsInWorkflowTasks-Process</dqService>
  <dqFailStep>STEP:1</dqFailStep>
</step>
<step>
  <id>1</id>
  <entity>Article</entity>
  <workflowStatus>My task 15</workflowStatus>
  <description>My task 15</description>
  <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
  <enterStatus>OnDqResults</enterStatus>
  <batchSize>500</batchSize>
  <userType>userGroup</userType>
  <userName>Standardusers</userName>
  <uiTemplate>Item approve UI</uiTemplate>
  <getField>Article.SupplierAID</getField>
  <getField>ArticleLang.DescriptionShort(9, "Default Channel")</getField>
  <getField>ArticleLang.DescriptionLong(9, "Default Channel")</getField>
  <executeDq>Always</executeDq>
  <dqService>ExecuteDqBatch-Process</dqService>
  <dqChannel>Channel 1</dqChannel>
  <emailProfile>DqChannelFailed</emailProfile>
  <dqFailStep>NotifyUser-Process</dqFailStep>
</step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

#### Step 0

This steps checks only whether the item is already in this workflow. If not the executed next step is 1. If yes the workflow will end because there is no next step defined.

Key	Value	Description
id	0	Identifier of the step.

Key	Value	Description
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
batchSize	500	Batch size to use with trigger batching.
dqService	ItemsInWorkflowTasks-Process	This DQ service is an additional bpel inside the StepWorkflow which checks whether the item is already inside this workflow or not.
dqFailStep	STEP:1	The item will be moved to the step with the identifier 1 if the above dq fails.

## Step 1

Key	Value	Description
id	1	Identifier of the step.
entity	Article	Entity container for this workflow.
workflow Status	My task 15	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	My task 15	Description of the status.
workflow ServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.

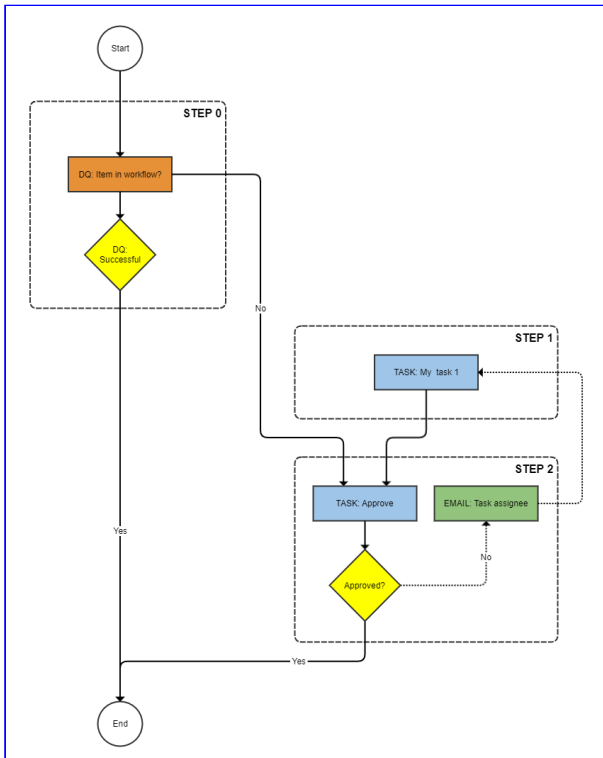


Key	Value	Description
enterStatus	OnDqResults	The circumstances when an item should enter a status (task). In this case only when DQ check will fail!
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.
userName	Standardusers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".
uiTemplate	Item approve UI	This is the default UI of the task.
executeDq	Always	The circumstances when a DQ should be run. In this case never!
dqService	ExecuteDqBatch-Process	The service name of the process that will be handle the processing of the dq.
dqChannel	Channel 1	Name of the DQ channel which will be executed.
getField	Several field names	This will get the field values for the items from P360. By default, for NotifyUser, these will be included in an html table in the email.
emailProfile	DqChannelFailed	This xsl profile will be used for creating the email.
notificationService	NotifyUser-Process	The service for the bpel process that will perform the email service.

### 2.3.5.19 Example 16 (Approval task with comments)

The sixteenth example shows how approval tasks will be logged.

**Diagram for example 16**



**Product 360 prerequisites for example 16**

Product 360 entity	value
Usergroup	Standardusers
Usergroup	Managers
Ui Template	Item approve UI

Explanation of the steps example 16

Step	Description
0	Check whether item is already in this workflow

Step	Description
1	Put the item into the task "My task 1"
2	Approve task with rejection options to the task "My task 1" incl. email to task assignee.

#### Example screenshot comments

Other remarks (English)

ABC

✂

📄

🔍

Ω

↶

↷

I<sub>x</sub>

≡

≡

B

I

U

x<sup>2</sup>

x<sub>2</sub>

Source

2023-11-24 01:32 pim - APPROVED Approve - Example Workflow 16 - 2 --  
2023-11-24 01:28 NA - ENTERED TASK:Approve - Example Workflow 16 - 2 -- userGroup:Managers  
2023-11-24 01:28 pim - FINISHED My task 1 - Example Workflow 16 - 1  
2023-11-24 01:19 NA - ENTERED TASK:My task 1 - Example Workflow 16 - 1 -- userGroup:Standardusers  
2023-11-24 01:19 pim - REJECTED Approve - Example Workflow 16 - 2 -- Not approved  
2023-11-24 01:16 NA - ENTERED TASK:Approve - Example Workflow 16 - 2 -- userGroup:Managers  
2023-11-24 00:58 pim - FINISHED My task 15 - Example Workflow 15 - 1 --  
2023-11-24 00:56 NA - ENTERED TASK:My task 15 - Example Workflow 15 - 1 -- userGroup:Standardusers

Copy: Ctrl/Cmd+C ; Paste: Ctrl/Cmd+V
Characters (including HTML): 553 ( 711 )

Save
Cancel

#### Example screenshot email

Dear Associate,  
The task **My task 1**, has been **rejected** for the following **1** items.

View Items

View Tasks

Item no.	Short description (English)	Long description (English)	userName	comment
AIW_63824373721	Acer Aspire One AOD255E-13639 10.1-Inch Netbook Diamond Black	Get fantastic mobile connectivity with the Acer Aspire One D255E in diamond black model AOD255E-13639 which features a 10.1-inch LED-backlit display comfortable typing keyboard and an ultra-thin design that travels light. You ll also enjoy excellent mobile computing power thanks to the Intel Atom N455 processor as well as up to 4 hours of battery life on a single charge from the included 3-cell battery. The Acer Aspire One D255E comes in a variety of colors. The attention-grabbing Aspire One D255E is only 1-inch thin and weighs just under 2-1 2 pounds. The 93-percent standard size keyboard with bigger keys enables easy typing wherever you are. And the 10.1-inch Acer CrystalBrite LED display 1024 x 600 pixels provides an optimal Web browsing experience for great Internet productivity while on-the-go.	pim	Not Approved please revisit and check the manufacturer item no again!

Regards,  
Product Team

StepWorkflow.xml file for example 16

### Example 16

```
<?xml version="1.0" encoding="UTF-8"?>
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>Example Workflow 16</label>
    <identifier>Workflow_16</identifier>
    <version>1.0</version>
    <updateWorkflowComments>true</updateWorkflowComments>
    <step>
      <id>0</id>
      <entity>Article</entity>
      <enterStatus>Never</enterStatus>
      <batchSize>500</batchSize>
      <executeDq>Always</executeDq>
      <dqService>ItemsInWorkflowTasks-Process</dqService>
      <dqFailStep>STEP:2</dqFailStep>
    </step>
    <step>
      <id>1</id>
      <entity>Article</entity>
      <workflowStatus>My task 1</workflowStatus>
      <description>My task 1</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>Always</enterStatus>
      <batchSize>500</batchSize>
      <userType>userGroup</userType>
      <userName>Standardusers</userName>
      <uiTemplate>Item approve UI</uiTemplate>
      <emailProfile>rejectTask</emailProfile>
      <includeComment>true</includeComment>
      <nextStep>STEP:2</nextStep>
    </step>
    <step>
      <id>2</id>
      <entity>Article</entity>
      <workflowStatus>Approve</workflowStatus>
      <description>Approval</description>
      <workflowServiceEndpoint>StepWorkflow-Trigger</workflowServiceEndpoint>
      <enterStatus>Always</enterStatus>
      <getField>Article.SupplierAID</getField>
      <getField>ArticleLang.DescriptionShort(9, "Default Channel")</getField>
      <getField>ArticleLang.DescriptionLong(9, "Default Channel")</getField>
      <batchSize>500</batchSize>
      <userType>userGroup</userType>
      <userName>Managers</userName>
    </step>
  </workflow>
</imp:payload>
```

```

<uiTemplate>Item approve UI</uiTemplate>
<executeDq>Never</executeDq>
<rejectTrigger>StepWorkflow_ParallelTasks-Reject</rejectTrigger>
<singleChoice>>false</singleChoice>
<rejectDecision>
  <id>p360.bpm.reject.STEP:1</id>
  <label>Reject to task 1</label>
  <service>NotifyUser-Process</service>
</rejectDecision>
</step>
</workflow>
</imp:payload>

```

Detailed explanation of the steps

#### Step 0

This step checks only whether the item is already in this workflow. If not the executed next step is 1. If yes the workflow will end because there is no next step defined.

Key	Value	Description
id	0	Identifier of the step.
entity	Article	Entity container for this workflow.
enterStatus	Never	The circumstances when an item should enter a status (task). In this case never!
batchSize	500	Batch size to use with trigger batching.
dqService	ItemsInWorkflowTasks-Process	This DQ service is an additional bpel inside the StepWorkflow which checks whether the item is already inside this workflow or not.

Key	Value	Description
dqFailStep	STEP:2	The item will be moved to the step with the identifier 1 if the above dq fails.

#### Step 1

This step puts the item into the task "My task 1"

Key	Value	Description
id	1	Identifier of the step.
entity	Article	Entity container for this workflow.
workflow Status	My task 1	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	My task 1	Description of the status.
workflow ServiceEndpoint	StepWorkflow-Trigger	This is the service name of the partner link. This name is defined in the <b>process deployment descriptor</b> of the workflow.
enterStatus	Always	The circumstances when an item should enter a status (task). In this case always!
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.
userName	Standardusers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".

Key	Value	Description
uiTemplate	Item approve UI	This is the default UI of the task.
emailProfile	Always	The circumstances when a DQ should be run. In this case never!
includeComment	true	Decides whether the comments on of workflowtask will be included in the mail
nextStep	STEP:2	Identifier of the next step, The item will be moved to the next step, when this step is finished.

## Step 2

This step is for the approval, if the task is rejected it will go to step 1 and send an email to task assignee including the reject decision.

Key	Value	Description
id	2	Identifier of the step.
entity	Article	Entity container for this workflow.
workflowStatus	Approve	Status name within the Product 360 workflow. ==> Product 360 task name = [Workflow] - <b>[Status]</b> - [Catalog]
description	Approval	Description of the status.
enterStatus	Always	The circumstances when an item should enter a status (task). In this case always!

Key	Value	Description
getField	Several field names	This will get the field values for the items from P360. By default, for NotifyUser, these will be included in an html table in the email.
batchSize	500	Batch size to use with trigger batching.
userType	userGroup	Type of the default assignee of this task. In this case it will be a usergroup.
userName	Managers	Name/Identifier of the Default assignee of the task. ==> The item will be assigned to the usergroup "Standardusers".
uiTemplate	Item approve UI	This is the default UI of the task.
executeDq	Never	The circumstances when a DQ should be run. In this case never!
rejectTrigger	StepWorkflow_ ParallelTasks- Reject	Endpoint for the reject trigger of parallel states (tasks).
singleChoice	false	This is for Rejects only and determines if you can have more than one choice. ==> The workflow can be rejected to all steps in one decision.
<rejectDecision>id	p360.bpm.reject.STEP:1	Identifier of the 1 <sup>st</sup> reject step.
<rejectDecision>label	Reject to task 1	Label of the 1 <sup>st</sup> rejection. This label will show up in the UI.
<rejectDecision>service	NotifyUser- Process	The service for the bpel process that will perform the email service.



### 2.3.5.20 StepWorkflow Sanity

The StepWorkflow package includes a Sanity process. It has a workflow file for configurations

#### Sanity

```
<imp:payload xmlns:imp="http://www.informatica.com/schema/ItemMap" contentType="string">
  <workflow>
    <label>StepWorkflowSanity</label>
    <identifier>StepWorkflowSanity</identifier>
    <version>1.0</version>
    <step>
      <id>Sanity</id>
      <urn>urn:p360.api.username</urn>
      <urn>urn:p360.api.password</urn>
      <urn>urn:p360.api.manager</urn>
      <urn>urn:p360.api.queue</urn>
      <urn>urn:p360.rest.url</urn>
      <urn>urn:dq.timeout</urn>
      <urn>urn:p360.api.data.quality.queue</urn>
      <urn>urn:p360.api.data.quality.response.queue</urn>
      <urn>urn:list.get.timeout</urn>
      <getField>Article.CurrentStatus</getField>
      <updateFieldDescriptor>Article.CurrentStatus</updateFieldDescriptor>
      <createTask>true</createTask>
      <sendNotification>true</sendNotification>
      <emailConfigs>sanity</emailConfigs>
    </step>
  </workflow>
</imp:payload>
```

It consists of 3 main sections:

- **URN check.** It verifies that urn values that are required for StepWorkflow do exist. It does this based on the values in the config file with the 'urn' tag. Users can add any urn's they want to include them in the sanity. If the urn exists and the test is considered SUCCESS if it does not, it is considered FAILED.
- **Workflow map files.** The sanity goes through the StepWorkflow.map file and verifies that it can access each workflow file in this file. It also verifies that the workflow identifier in the map file matches the one in each workflow xml file.
- **Access to P360.** The config file has get field and update field descriptor values to verify connection to P360. First, it does a get and verifies the response. Then it uses the same value to do a post. Using the same value makes sure nothing is changed for the sanity test but also verifies the connection.

The sanity test also allows for a task to be created and/or an email sent out. For the task and email, the sanity test is labeled a SUCCESS if no tests fail. Otherwise, it is considered FAILED.

Key	Value	Description
urn	urn:p360.api.queue	These are the urn values that the Sanity will verify that they do exist.
getField	Article.CurrentStatus	This is the field that the sanity will use to verify a get for P360
UpdateFieldDescriptor	Article.CurrentStatus	This is the fields used to verify that a post will work to P360. It must be one of the fields in the Update Descriptor field.
CreateTask	TRUE	This determines whether or not there will be a task for the Sanity results.
sendNotification	TRUE	This determines whether or not there will be an email sent for the Sanity results.
emailConfigs	sanity	The xls file that serves as the template for the email.

Sample email:

Subject: SANITY PROCESS ID 42431 FAILED			10:06
To: tina@pimdmo.org			
Dear Associate,			
SANITY PROCESS ID 42431 FAILED			
Here are the results of each test:			
Sanity Results			
Test	Details	Result	
urn:p360.api.username	RestServiceUser	SUCCESS	
urn:p360.api.password	#####	SUCCESS	
urn:p360.api.manager	ActiveMQ	SUCCESS	
urn:p360.api.queue	JNDI_P360_SERVICE_API	SUCCESS	
urn:This.is.a.test.urn	none	FAILED	
urn:p360.rest.url	http://localhost:1512	SUCCESS	
urn:dq.timeout	PT4M	SUCCESS	
urn:p360.api.data.quality.queue	JNDI_P360_DATA_QUALITY	SUCCESS	
urn:p360.api.data.quality.response.queue	bpm_response	SUCCESS	
urn:list.get.timeout	PT4M	SUCCESS	

**SanityResults****SanityResults**

```

<type>
  <test>
    <name>urn:p360.api.username</name>
    <details>RestServiceUser</details>
    <results>SUCCESS</results>
  </test>
  <test>
    <name>urn:p360.api.password</name>
    <details>#####</details>
    <results>SUCCESS</results>
  </test>
  <test>
    <name>urn:p360.api.manager</name>
    <details>ActiveMQ</details>
    <results>SUCCESS</results>
  </test>
  <test>
    <name>urn:This.is.a.test.urn</name>
    <details>none</details>
    <results>FAILED</results>
  </test>
  ...
  <test>
    <name>project:/StepWorkflowExamples/workflow/Example_01.xml</name>
    <details>Loaded file</details>
    <results>SUCCESS</results>
  </test>
  <test>
    <name>Workflow_01</name>
    <details>Workflow id matches map file</details>
    <results>SUCCESS</results>
  </test>
  <test>
    <name>project:/StepWorkflowExamples/workflow/Example_02.xml</name>
    <details>Loaded file</details>
    <results>SUCCESS</results>
  </test>
  <test>
    <name>Workflow_02</name>
    <details>Workflow id matches map file</details>
    <results>SUCCESS</results>
  </test>
  ...
  <test>
    <name>get(Article.CurrentStatus)</name>

```

```

    <details>0939699831001: 200</details>
    <results>SUCCESS</results>
  </test>
  <test>
    <name>update(Article.CurrentStatus)</name>
    <details>UPDATED</details>
    <results>SUCCESS</results>
  </test>
</type>

```

## 2.3.6 Steps

- [ExecuteDqBatch-Process](#) (see page 424)
- [GVisionAi-Process](#) (see page 426)
- [GVisionSetBooleanState-Process](#) (see page 428)
- [GVisionTransferFieldValue-DQ](#) (see page 429)
- [ItemsInWorkflowTasks-Process](#) (see page 431)

### 2.3.6.1 ExecuteDqBatch-Process

#### Dependencies

Dependency	Description
InfaNextSteps.bpr	Source project of this step
StepWorkflow.bpr	Framework to call this step

#### Description

This step checks whether the item(s) are already in this workflow.

Result	Description
True	All DQ results for item(s) true
False	Not all DQ results for item(s) true

## Parameters

Parameter	Description	Mandatory
<a href="#">entity</a>	Entity of the objects. Supported values: Article, Product, Variant	Yes
dqChannel	Identifiers of the channels which should be executed	No
dqRuleGroup	Identifiers of the rule groups which should be executed	No
dqRule	Identifiers of the groups which should be executed	No

## Example

Example step definition	
1	<code>&lt;step&gt;</code>
2	<code>  &lt;id&gt;1&lt;/id&gt;</code>
3	<code>  &lt;entity&gt;Article&lt;/entity&gt;</code>
4	<code>  &lt;workflowStatus&gt;My task 1&lt;/workflowStatus&gt;</code>
5	<code>  &lt;description&gt;My task 1&lt;/description&gt;</code>
6	<code>  &lt;workflowServiceEndpoint&gt;StepWorkflow-Trigger&lt;/workflowServiceEndpoint&gt;</code>
7	<code>  &lt;enterStatus&gt;OnDqResults&lt;/enterStatus&gt;</code>
8	<code>  &lt;batchSize&gt;500&lt;/batchSize&gt;</code>
9	<code>  &lt;userType&gt;userGroup&lt;/userType&gt;</code>
10	<code>  &lt;userName&gt;Standardusers&lt;/userName&gt;</code>
11	<code>  &lt;uiTemplate&gt;Item approve UI&lt;/uiTemplate&gt;</code>
12	<code>  &lt;executeDq&gt;Always&lt;/executeDq&gt;</code>
13	<code>  &lt;dqService&gt;ExecuteDqBatch-Process&lt;/dqService&gt;</code>
14	<code>  &lt;dqChannel&gt;Channel 1&lt;/dqChannel&gt;</code>
15	<code>  &lt;dqRule&gt;Image_Face_Detection&lt;/dqRule&gt;</code>
16	<code>  &lt;dqRule&gt;Image_Logo_Detection&lt;/dqRule&gt;</code>
17	<code>  &lt;nextStep&gt;STEP:2&lt;/nextStep&gt;</code>
18	<code>&lt;/step&gt;</code>

### 2.3.6.2 GVisionAi-Process

#### Dependencies

Dependency	Description
InfpNextStepsGVisionAI.bpr	Source project of this step
StepWorkflow.bpr	Framework to call this step
GVisionAi.xml	Resource to define Google Vision AI configuration

#### Description

This step requests Google Vision AI Rest calls with the given public available URLs of the images. The results of the response can be stored in given fields.

#### Parameters

Parameter	Description	Mandatory
<a href="#">entity</a>	Entity of the objects. Supported values: Article, Product, Variant	Yes
<a href="#">gVisionImageUrlField</a>	This field must contain the public available URL of the image. The value have to be requested in the Step Workflow framework with the <a href="#">getField</a> parameter.	Yes
<a href="#">gVisionLogoDetection</a>	This parameter defines whether the LogoDetection mode is on or off. Default value: False	No
<a href="#">gVisionFaceDetection</a>	This parameter defines whether the FaceDetection mode is on or off. Default value: False	No

Parameter	Description	Mandatory
<a href="#">gVisionTextDetection</a>	This parameter defines whether the TextDetection mode is on or off. Default value: False	No
<a href="#">gVisionLabelDetection</a>	This parameter defines whether the LabeDetection mode is on or off. Default value: False	No
<a href="#">gVisionWebDetection</a>	This parameter defines whether the WebDetection mode is on or off. Default value: False	No
<a href="#">gVisionLogoSetField</a>	This parameter defines the field where the result of the LogoDetection mode will be persisted, Note: Already existing values will be overwritten!	No
<a href="#">gVisionFaceSetField</a>	This parameter defines the field where the result of the FaceDetection mode will be persisted, Note: Already existing values will be overwritten!	No
<a href="#">gVisionTextSetField</a>	This parameter defines the field where the result of the TextDetection mode will be persisted, Note: Already existing values will be overwritten!	No
<a href="#">gVisionLabelSetField</a>	This parameter defines the field where the result of the LabeDetection mode will be persisted, Note: Already existing values will be overwritten!	No
<a href="#">gVisionWebSetField</a>	This parameter defines the field where the result of the WebDetection mode will be persisted, Note: Already existing values will be overwritten!	No

## Example

Example step definition	
1	<code>&lt;step&gt;</code>
2	<code>  &lt;id&gt;01-GVision-ExecuteGvision-01&lt;/id&gt;</code>
3	<code>  &lt;entity&gt;Article&lt;/entity&gt;</code>
4	<code>  &lt;nextStep&gt;GVisionAi-Process&lt;/nextStep&gt;</code>
5	<code>  &lt;gVisionLogoDetection&gt;true&lt;/gVisionLogoDetection&gt;</code>
6	<code>  &lt;gVisionFaceDetection&gt;false&lt;/gVisionFaceDetection&gt;</code>
7	<code>  &lt;gVisionTextDetection&gt;false&lt;/gVisionTextDetection&gt;</code>
8	<code>  &lt;gVisionLabelDetection&gt;false&lt;/gVisionLabelDetection&gt;</code>
9	<code>  &lt;gVisionWebDetection&gt;false&lt;/gVisionWebDetection&gt;</code>
10	<code>  &lt;gVisionImageUrlField&gt;ArticleMediaAssetMap.UniformResourceIdentif</code>
11	<code>  ier(normal)&lt;/gVisionImageUrlField&gt;</code>
12	<code>  &lt;gVisionLogoSetField&gt;Article.ManufacturerName&lt;/gVisionLogoSetField&gt;</code>
13	<code>  &lt;getField&gt;ArticleMediaAssetMap.UniformResourceIdentifier(normal)&lt;/getField&gt;</code>
14	<code>  &lt;workflowServiceEndpoint&gt;StepWorkflow-Trigger&lt;/workflowServiceEndpoint&gt;</code>
15	<code>  &lt;nextStep&gt;STEP:01-GVision-EnterTask-02&lt;/nextStep&gt;</code>
	<code>&lt;/step&gt;</code>

### 2.3.6.3 GVisionSetBooleanState-Process

#### Dependencies

Dependency	Description
InfpNextStepsGVisionAI.bpr	Source project of this step
StepWorkflow.bpr	Framework to call this step

#### Description

This step stores the state of a Google Vision AI result in a Boolean state field.



## Parameters

Parameter	Description	Mandatory
<a href="#">entity</a>	Entity of the objects. Supported values: Article, Product, Variant	Yes
<a href="#">gVisionStateGetField</a>	This field must contain the result of the Google Vision API request. The value have to be requested in the Step Workflow framework with the <a href="#">getField</a> parameter.	Yes
<a href="#">gVisionStateSetField</a>	This parameter defines the field to store the result state of the execution. Possible values: True, False. For example if a logo was detected and the sting is not empty the value will be true.	Yes

## Example

Example step definition	
1	<code>&lt;step&gt;</code>
2	<code>  &lt;id&gt;GVisionUpdateStateLogo-01&lt;/id&gt;</code>
3	<code>  &lt;entity&gt;Article&lt;/entity&gt;</code>
4	<code>  &lt;nextStep&gt;GVisionSetBooleanState-Process&lt;/nextStep&gt;</code>
5	<code>  &lt;gVisionStateSetField&gt;Article.GvLogoFound&lt;/gVisionStateSetField&gt;</code>
6	<code>  &lt;gVisionStateGetField&gt;Article.GvLogo&lt;/gVisionStateGetField&gt;</code>
7	<code>  &lt;getField&gt;Article.GvLogo&lt;/getField&gt;</code>
8	<code>  &lt;workflowServiceEndpoint&gt;StepWorkflow-Trigger&lt;/</code>
9	<code>  workflowServiceEndpoint&gt;</code>
	<code>&lt;/step&gt;</code>

### 2.3.6.4 GVisionTransferFieldValue-DQ

## Dependencies

Dependency	Description
InfaNextStepsGVisionAI.bpr	Source project of this step

Dependency	Description
StepWorkflow.bpr	Framework to call this step

#### Description

This step transfers field values from one field to another field.

#### Parameters

Parameter	Description	Mandatory
entity	Entity of the objects. Supported values: Article, Product, Variant	Yes
gVisionGetField	Source field. The value have to be loaded in the step with getField.	Yes
gVisionSetField	Target field. Important: The value will be overwritten if getField for the source field is not set or the value is empty.	Yes

#### Example

Example step definition	
1	<step>
2	<id>GVisionTransferFieldValue-DQ-06</id>
3	<entity>Article</entity>
4	<executeDq>Always</executeDq>
5	<dqService>GVisionTransferFieldValue-DQ</dqService>
6	<dqRule>GVisionTransferFieldValue-DQ-06</dqRule>
7	<gVisionSetField>Article.ManufacturerName</gVisionSetField>
8	<gVisionGetField>Article.GvLogo</gVisionGetField>
9	<getField>Article.ManufacturerName</getField>
10	<getField>Article.GvLogo</getField>
11	<workflowServiceEndpoint>StepWorkflow-Trigger</
12	workflowServiceEndpoint>
	</step>

### 2.3.6.5 ItemsInWorkflowTasks-Process

#### Dependencies

Dependency	Description
InfpNextSteps.bpr	Source project of this step
StepWorkflow.bpr	Framework to call this step

#### Description

This step checks whether the item(s) are already in this workflow.

Result	Description
True	Item(s) already in the workflow
False	Item(s) are not in workflow

#### Parameters

Parameter	Description	Mandatory
entity	Entity of the objects. Supported values: Article, Product, Variant	Yes

#### Example

Example step definition	
1	<step>
2	<id>0</id>
3	<entity>Article</entity>
4	<enterStatus>Never</enterStatus>
5	<batchSize>500</batchSize>

```

6      <executeDq>Always</executeDq>
7      <dqService>ItemsInWorkflowTasks-Process</dqService>
8      <dqFailStep>STEP:1</dqFailStep>
9      </step>

```

## 2.4 Creating MQ based workflows

This guide describes how to design ActiveVOS workflows that leverage the new queue based communication mode between P360 and ActiveVOS

### 2.4.1 Prerequisites

- See the Installation Manual for installation of ActiveVOS with ActiveMQ support and configuration of queues in ActiveVOS  
(Note: ActiveVOS version 9.2.4.6 is a prerequisite for queue based interactions between P360 and ActiveVOS)
- See the Message Queue API documentation
  - how to generally configure queues in P360 server
  - message headers and payloads for the various interactions between Product 360 and ActiveVOS

### 2.4.2 REST versus JMS based workflows

At its core, ActiveVOS is based on SOAP Style web services backed by WSDL definitions. The support for both REST based web services and for JMS interactions are internally mapped onto these concepts. The details are slightly different and they have an impact on how interactions work with Product 360 and workflows.

Area	REST based communication	Queue based communication
Service interface definitions	Leverages built-in generic REST service interface definitions.	Interface definitions need to be added as a WSDL in the ActiveVOS project.
Message header	Full access to REST headers through the built in interface definition.	Access to JMS header fields is available through "Partner" variables.
Message body	Free choice of format, can be JSON.	Message body needs to adhere to WSDL and therefore restricted to XML.

Area	REST based communication	Queue based communication
Incoming messages	Usually paired with an outgoing message to implement the HTTP response.	One way, there is no need to send a response message to the queue. P360 regards the message as successfully delivered as soon as it has been placed into the queue.
Outgoing messages	Outgoing messages may be "one-way" or paired with incoming messages that contain the response for the outgoing request,	Outgoing messages are written to a queue. Similarly to the REST case this may be paired with an incoming message for the response which also comes in via a queue. However from a workflow designer perspective they can be seen as two "one-way" communications.
Special case of outgoing message: Workflow tasks definition	Body can be either JSON or XML.  Workflow task definitions include ActiveVOS workflow endpoints that have to be invoked via REST.	Body has to be XML.  Workflow task definitions set the communication mode to "QUEUE", include the queue ID (as defined in P360 configuration) and the workflow endpoints to be invoked via JMS.

### 2.4.3 Mandatory Permission Required For Queue Based Communication

For BPM to have queue based communication with P360, "Queue Communication" named Action rights is required for the user group of the user used in BPM workflows. Without having this action right for the user group, BPM will get Unauthenticated error when it try to contact Product 360 via message queue.

Action rights (1/389) X					
Interface visibility					
Object rights (system)					
Field rights					
Qualified field rights					
All action rights of user group "Standardusers - New user group (2)"					
Grouping		Allowed	Permission	Rights group	Description
To gro... here.	1	<input checked="" type="checkbox"/>	Queue Communication	General	User can communicate with Product 360 over message queues

## 2.4.4 Types of interactions and their representation in the WSDL file

There are a couple of different message types that can be received from/sent to P360:

- Incoming event triggers
- Outgoing Service API requests (one-way or request-response)
- Incoming responses for Service API requests (result or error responses)
- Outgoing DQ execution requests
- Incoming responses for DQ executions (result or error responses)

To avoid the overhead of defining each of these messages down to the smallest detail within the WSDL file (or a separate XML Schema file) all of the messages have been defined to have `anyType` content within some root tag. The approach is different between incoming and outgoing messages:

- For *incoming* messages we can use a *single type* with an arbitrarily defined root tag as ActiveVOS is not strictly checking the root tag.
- For *outgoing* messages the defined root tag is what gets sent to P360 and needs to match P360 expectations. Therefore there are *a number of message types defined with different root tags*.

Based on these messages several port types have been defined:

Port type	Operation	Input message type	Remarks
Receive	<code>receive</code>	<code>p360RoutedMessage</code>	<p>This message type has two parts:</p> <ul style="list-style-type: none"> <li>• <code>p360Header</code> : this part for now only contains an optional element <code>correlationId</code></li> <li>• <code>p360MessageBody</code> : this part contains the original message sent from P360 (with an adjusted root tag)</li> </ul>
Send	<code>callWithoutParameters</code>	<code>emptyMessage</code>	<p>Used for sending requests to P360 that do not require any input parameters. Due to WSDL restrictions the operation still needs to specify a message type.</p>
	<code>startWorkflow</code>	<code>startWorkflowMessage</code>	<p>This operation and the following ones are used for requests that require different types of input messages.</p>

Port type	Operation	Input message type	Remarks
	enterWorkflowStatus	enterWorkflowStatusMessage	
	leaveWorkflowStatus	leaveWorkflowStatusMessage	
	executeQualityProfile	dataQualityProfileMessage	
	listWrite	entityItemTableMessage	
Router	route	p360Message	Only used by the P360Router process - see next section.

## 2.4.5

### Router process

In the REST based communication mode P360 invokes the exposed REST endpoint of the respective ActiveVOS process directly. In the queue based communication mode all P360 requests go to a single queue.

To deal with this a router process is set up on ActiveVOS which has essentially three responsibilities;

- Evaluate the JMS header property `P360TargetService` that is used to forward the request to the right endpoint.
- Copying the request message into a common base format ( `p360RoutedMessage` ) that all endpoints can deal with.
- Adding the correlation ID to the common base format message (see below for details).



When ActiveVOS monitors a queue for incoming messages it supports reading the target endpoint from the JMS header property `JmsTargetService` . However this feature does not work with endpoints that are based on `anyType` messages.

## 2.4.6 Core orchestration project

As described above, both the WSDL and the P360Router process are key components for setting up queue based communication between P360 and ActiveVOS. Therefore a ready-to-use orchestration project is made available as a ZIP file. It contains the WSDL and the BPEL and PDD files for the P360Router. Import this project into your ActiveVOS Designer workspace. While you can then add your workflow process definitions simply to this project, it is recommended to set them up separately:

- Create one or more separate projects for your workflow processes.
- Add the P360\_JMS\_Core project as a project reference in your project(s).
- Use the message, port and partner link types of the WSDL when defining variables and participants in your processes.
- The WSDL also includes a correlation definition that can be used in the correlation set definitions of your workflow processes (see below).

The P360\_JMS\_Core project needs to be deployed to your ActiveVOS as a prerequisite for the deployment of your own workflows. Before doing so note the endpoint name of the "Consumer" participant of the P360Router process in the PDD file. This name needs to match the default service set of the queue listener set up in the ActiveVOS Message Service configuration. By default both are called *P360RouterService*.

The listener monitors the queue which P360 uses to send requests to ActiveVOS. These messages need to flow through the P360Router process before they are forwarded to the actual target endpoint.

**The core orchestration project is included as a ZIP file in the BPM accelerator package.**

## 2.4.7 Correlation

The correlation concept assures that incoming messages are routed to the right process instance of a workflow. A good starting point is to use the process ID for this, however for more complex workflows with parallel execution it may be necessary to add additional information to the correlation IDs of specific endpoints within the workflow.

**i** ActiveVOS sets the correlation ID by picking it from a defined path of an outgoing message. It then checks a defined path in incoming messages to map the request to an existing process instance. For this to work the correlation ID must be part of both message bodies.

For backwards compatibility reasons many P360 calls expect the correlation ID to be sent in the `processId` field of a request and also return it in the same way in subsequent requests to ActiveVOS.

Newer APIs however support the use of the `JMSCorrelationID` header.

On the receiving side the P360Router process takes care of picking the correlation ID from the right location and puts it into the `p360Header` part of the `p360RoutedMessage`. So the individual workflows can define their correlation ID settings based on this without worrying how it was transmitted.

Unfortunately on the outgoing side these details cannot be hidden. Best practice here is to:

- Write the correlation ID to the `JMSCorrelationID` header of the outgoing message. This is where P360 expects it in general.



- Write it to a top level element called `processId` in the outgoing message body. This is for ActiveVOS correlation definition to be able to pick it up on the way out.  
Note that the name `processId` is used for backwards consistency. This is where the correlation ID is already passed in the REST based communication use case.

## 2.4.8 Defining participants

### 2.4.8.1 Consumers

Configure the Consumer participants in your workflow processes as follows

- Select `p360ReceivePortType` as its service interface.
- Select `p360ReceivePLT` as its partner link type
- Add a receive activity to your workflow, select the newly created Consumer, select the `receive` operation and use a variable of type `p360RoutedMessage` to store the input.
- After creating the deployment descriptor for your workflow set "Binding" to "Document literal" and the name of the endpoint to how it should be addressable from P360.

Note that such a Consumer is always expecting incoming messages at its defined endpoint. So usually you *cannot reuse* the same Consumer at different points in your workflow.

### 2.4.8.2 Producers

Configure a Producer participant as follows:

- Select `p360SendPortType` as its service interface.
- Select `p360SendPLT` as its partner link type
- After creating the deployment descriptor for your workflow set the Producer to be a "JMS Service" and "dynamic"
- Prepare the an input variable and the Producer in a script activity of your workflow
  - Setup a variable of the right type for the desired operation's input message body.
  - Setup the message header by assigning an appropriate XML structure to your newly created "Partner" (Choose "Partner" and your new participant as the target of a copy).  
See section below on how to create this "appropriate XML".
  - Add an invoke activity to your workflow, select the newly created participant, the desired operation and the variable as its input.

Note that with the "dynamic" setting the same Provider can be used for any outgoing calls from the workflow to P360. You just need to prepare the header and the input variable upfront.

In case of parallel execution threads you should still use separate Provider participants.

#### Setting up the outgoing message header

- Add a script activity to your workflow
- Add a copy assignment to the script activity
  - On the source side select "Expression" and set it to XQuery based expressions.
  - On the target side select "Partner" and the newly created Producer

- Go back to the PDD file and temporarily select "static" for the Producer. This will populate the text box below with an XML template. Copy this for further use.
- Go back to the copy assignment and paste the copied XML as the starting point for the expression. You need to make the following adjustments:

Section	Value
address	<p>The address needs to be set to the name of the target queue. Note that it needs to be the JNDI name configured for the queue on the ActiveVOS server side - not the actual queue name on the JMS server.</p> <p>There are two distinct queues that can be used here at the moment: one for DQ executions and one for Service API requests.</p>
jmsManagerId	The jmsManagerId needs to match the name of the "Message Service" configuration in the ActiveVOS server.
jmsMessageFormat jmsMessageType	Leave these at "xml", "text" respectively.
jmsTTL	<p>Set to "0" for most cases to avoid message expiration. Sometimes messages can become obsolete if they haven't been dealt with in a defined time frame.</p> <p>In such cases you can set this to the maximum number of milliseconds that this message should be kept on the queue. See the documentation of your JMS server on how expired messages are treated.</p>

Section	Value
ReferenceParameters	<p>This is essentially a list of key value pairs that will appear as header properties on the JMS message. Common properties for any type of message include:</p> <ul style="list-style-type: none"> <li>• <b>User</b> - P360 user name</li> <li>• <b>Password</b> - password for the P360 user</li> <li>• <b>JMSCorrelationID</b> for two way communication. P360 will include this as a header property in the response.</li> <li>• <b>SuccessTargetService</b> - for two way communication this specifies the value of the <b>P360TargetService</b> header property in the response.</li> <li>• <b>ErrorTargetService</b> - for two way communication this can be used optionally to specify a different for <b>P360TargetService</b> in error responses.</li> </ul> <p>Individual message types may require more properties.</p>



The WSDL includes partner link types. When adding new participants to a BPEL process and selecting a port type for its service interface ActiveVOS automatically creates a separate partner link type in a new WSDL. You can see these new WSDLs pop up under wsdl/bpel in your project structure. While it is not strictly necessary you can change the partner link types of your participants to use the types from the Core WSDL instead. Once you have done this the generated WSDLs should be empty as ActiveVOS Designer realizes you don't need the generated partner link types any more. However they are not removed completely and still appear as imports in your projects. To completely get rid of them, remove the imports first and then the WSDL files. Always make sure the files just contain an empty root tag before doing this. Also, as the behavior is the same for each new participant you might want to only do this once you have all participants defined.

## 2.4.9 Workflow interaction examples

Workflow examples are provided in two separate orchestration projects - **P360\_JMS\_Demo** and **P360\_JMS\_Demo\_DataQuality**. The later one contains a DQ execution specific example and the other one has all the other examples. Separation into two projects was arbitrary and just to show that workflow

processes can be defined in several projects. Both projects rely on the `P360_JMS_Core` project for reference to the WSDL and also at run time for the routing functionality.

#### 2.4.9.1 Workflow instance creation

The **NoOpReceiver** example workflow contains a minimal workflow. It basically contains a consumer, a variable and a receive activity that assigns the incoming message to the variable. Instances of it will appear in the ActiveVOS monitoring for each message that was successfully received.

Use this workflow for testing your setup for incoming messages:

- To test ActiveVOS Message Service setup and P360Router process deployment publish a message to the queue that is monitored by ActiveVOS
  - Message Body can be arbitrary XML.
  - Message header should include `P360TargetService` property set to `NoOp-Trigger`.
  - Instances of the workflow should appear in monitoring and the process variable should contain the message body sent.
- To test P360 to ActiveVOS communication set up a change trigger in P360 that is sending messages via queue to `NoOp-Trigger`.
  - Instances of the workflow should appear in monitoring and the process variable should contain the message sent from P360.

#### 2.4.9.2 P360 Service API oneway calls

**Note:** This example is using the LIST API to issue a write request to P360. The queue based communication support for the write direction of the LIST API is experimental in the current release.

The **ServiceApiCallOneway** example workflow shows how to invoke P360 from workflows when there is no need to receive a response from P360. Here is a short summary of its content:

- It consumes incoming trigger messages at the endpoint `ServiceApiCallOneway-Trigger`.
  - Again, for testing this workflow it should be hooked up with a change trigger definition on P360 side. E.g. whenever the short description of an item changes.
- Additionally it contains a Provider and an invoke activity using it for sending a message to the P360 Service API.
  - The concrete call is setting the `CurrentStatus` of the item extracted from the received trigger message to "07 Attributes OK"

Compared to the generic header property description above this call needs additional properties:

Property	Value	Description
<code>P360Url</code>	<code>rest/V2.0/list/Article</code>	Service API endpoint to invoke on P360 side.

Property	Value	Description
Method	POST	Does not strictly have to be set as this is the default. Don't be confused if this looks like REST/HTTP. The communication is still JMS based but it is mapped onto the existing REST based API on P360 side.

You can use this workflow to test the full roundtrip P360->ActiveVOS → P360. The trigger message is expected to be an entity change trigger containing the entity type and id. The workflow extracts this information from the incoming message and uses it within the message body of the outgoing request. Set up a change trigger on P360 that reacts on changes to the "short description" on items. Then create some items - they will have their initial "status" set to "01". Changing the short description should lead to the creation of a workflow instance and then to call to P360 that changes the status of the item to "07".

#### 2.4.9.3 P360 Service API calls with result handling

The **ServiceApiCall** example workflow shows how to invoke P360 Service API and process the response.

Compared to the one-way example we need two additional Consumer participants, one for the success case and one for the error case. The example uses the "Pick" activity to wait on two endpoints at the same time. Additionally it also has a "timeout" branch defined. There are three possible outcomes:

- Response is received on the endpoint waiting for "success" responses → workflow will be shown as completed in monitoring.
- Response is received on the endpoint waiting for "error" responses → workflow will be shown as faulted in monitoring.
- No response is received within the time set up on the "timeout" branch → workflow will be shown as faulted in monitoring.

This is a simple example. In the real world the workflow might continue in each of these cases but take different decisions based on the outcome of this call.

Note that for the correlating requests a correlation set has been defined using the definition provided in the WSDL. This correlation set is used on the outgoing "invoke" activity and the matching "receive" activities.

The outgoing message has the following header properties set:

Property	Value	Description
P360Url	rest/V2.0/ list/Article/ byCatalog	Service API endpoint for listing the items in the master catalog.

Property	Value	Description
Method	GET	We want to the functionality associated with the GET method in this endpoint.
ResponseQueueID	bpm	This is the ID of the message queue to which the response should be written as configured in P360 service properties (not the JNDI name used in ActiveVOS!)
SuccessTargetService	ServiceApiCall-Response	The endpoint to invoke in case of successful execution. The response message will use this as the value for the <code>P360TargetService</code> property.
ErrorTargetService	ServiceApiCall-Error	The endpoint to invoke in case of a failure during execution. The response message will use this as the value for the <code>P360TargetService</code> property.
JMSCorrelationID	...	Correlation id to match up response with the right workflow instance in ActiveVOS.

### Special handling of GET requests

REST/HTTP GET calls do not require/support an input message body. However the request and response are mapped here onto two distinct operations defined in the WSDL and each operation needs an input message:

- The request is using the `callWithoutParameters` and an `emptyMessage` as input.
- The response is using the standard `receive` operation and the `p360RoutedMessage`.

So the `emptyMessage` is essentially a workaround for having to provide a message type for every operation. But there is also a different reason for having it. As described in the section on correlation handling the ActiveVOS engine needs the correlation ID to be part of the outgoing message. So in cases like this we still have to add it to the `emptyMessage`.

#### 2.4.9.4 P360 DQ execution with result handling

The **CallDataQuality** workflow example shows how to execute DQ rule configurations. From a structural perspective it is very similar to the previous example that calls the Service API and waits for the response:

- A total of three consumers: initial trigger, success response, error response
- A provider for invoking P360 endpoint
- A "Pick" activity for reacting on the response from P360
- Correlation ID present in both the JMSCorrelationID header property and outgoing message body. Correlation set defined and used on "invoke" and matching "receives".

There are however also some differences:

- P360 listens for these requests on a separate queue, so the "address" of the Provider participant needs to be set to the correct JNDI name (JNDI\_P360\_DATA\_QUALITY).
- The Service API specific parameters are not needed: P360Url, Method

Note that this workflow assumes that there is a DQ rule configuration present on P360 which is named `IsEmpty`. It should be defined so that it checks whether the short description of an item is filled or not. For testing add a change trigger on short description changes and invoke `CallDataQuality-Trigger` endpoint. Add the short description and the DQ status of this rule configuration to the table in the P360 rich client and observe how filling/emptying the short description of an item changes the DQ status.

#### 2.4.9.5 P360 workflow tasks

The **WorkflowTasks** workflow example shows how to set up workflow tasks on P360. P360 workflow tasks can be used to include manual tasks by users into the overall workflow.

Workflow task management calls are part of the Service API, so the explanations in the previous examples around Service API calls apply here as well.

The overall structure of the workflow is as follows:

1. Receive a trigger to start the workflow - expected to be an entity create/change trigger.
2. Invoke Service API to define the workflow task structure - our case the definition includes two steps/statuses: `status01` and `status02`
  - a. Note that this is done for each instance of the workflow process. If P360 already knows about the workflow task definition it will ignore this. However the workflow could be sending a newer version of the definition which will cause an update on the P360 side.
3. Tell P360 that the item extracted from the trigger message should "enter" `status01`.
4. Wait for message from P360 that states that some user has marked `status01` as finished for this item.
5. Tell P360 that the item extracted from the trigger message should "leave" `status01`.
6. Tell P360 that the item extracted from the trigger message should "enter" `status02`.
7. Wait for message from P360 that states that some user has marked `status01` as finished for this item.
8. Tell P360 that the item extracted from the trigger message should "leave" `status02`.

Note that steps 4 and 7 are waiting for messages at certain endpoints: `WorkflowTasks-Finish01` and `WorkflowTasks-Finish02`. P360 knows to use these endpoints in the `P360TargetService` header because they are included in the workflow task definition sent to P360 in step 2.

Correlation is needed between steps 4 and 5 and again between 7 and 8. For this purpose a correlation set is set up similarly to previous examples.

## 2.5 DAM Workflows

- [DAM Configurator](#) (see page 444)
- [DAM Event Listener](#) (see page 459)
- [DAM Hotfolder](#) (see page 471)

### 2.5.1 DAM Configurator

#### **Cloud Edition only**

This project is designed for Product 360 Cloud Edition customers only. (On premise customers can use this project and modify it to their needs, but it is not under Informatica's support! )

#### 2.5.1.1 Preface

This BPM (AVOS) project enables Product 360 Cloud Edition customer's to define Digital Asset Management (DAM) settings without the help of Informatica's Operational teams . This functionality works only for the Asset Provider HMM, which is the default in the Cloud Edition settings.

#### **Table of Contents**

- [Preface](#) (see page 444)
- [Prerequisites](#) (see page 444)
- [Installation](#) (see page 445)
- [Configuration](#) (see page 445)
  - [URN mappings](#) (see page 445)
  - [Context properties for the messaging service](#) (see page 446)
  - [Database JNDI](#) (see page 446)
  - [Schedule the maintenance job](#) (see page 447)
- [Defining Main settings](#) (see page 448)
  - [The configuration file DAM\\_DefineMainSettings.xml](#) (see page 449)
- [Defining Property fields for digital assets](#) (see page 450)
  - [The configuration file DAM\\_DefinePropFields.xml](#) (see page 451)
  - [Example define 3 different property fields](#) (see page 453)
- [Defining Derivatives](#) (see page 454)
  - [The configuration file DAM\\_DefineDerivatives.xml](#) (see page 455)
  - [Example define a 70 pixel RGB Jpeg as derivative](#) (see page 457)
- [Apply DAM configurations](#) (see page 458)

#### 2.5.1.2 Prerequisites

This workflow also supports the messaging functionality over the message queue communication. Therefore the messaging service of ActiveVOS must be configured. Please reach out to the Installation and Operation guide for the following areas:

- Installation of ActiveVOS with ActiveMQ support and configuration of queues in ActiveVOS (*Note: ActiveVOS version 9.2.4.6 is a prerequisite for queue based interactions between Product 360 and ActiveVOS*)



### 2.5.1.3 Installation

You have to deploy these resources:

1. InfaResources.bpr
2. DAM\_Tools.bpr
3. DAM\_Resources.bpr

with the BPM (AVOS) console.

Also it is possible that you deploy the workflows with your BPM designer. The resources of the projects are in the zip files:

1. InfaResources.zip
2. DAM\_Tools.zip
3. DAM\_Resources.zip

### 2.5.1.4 Configuration

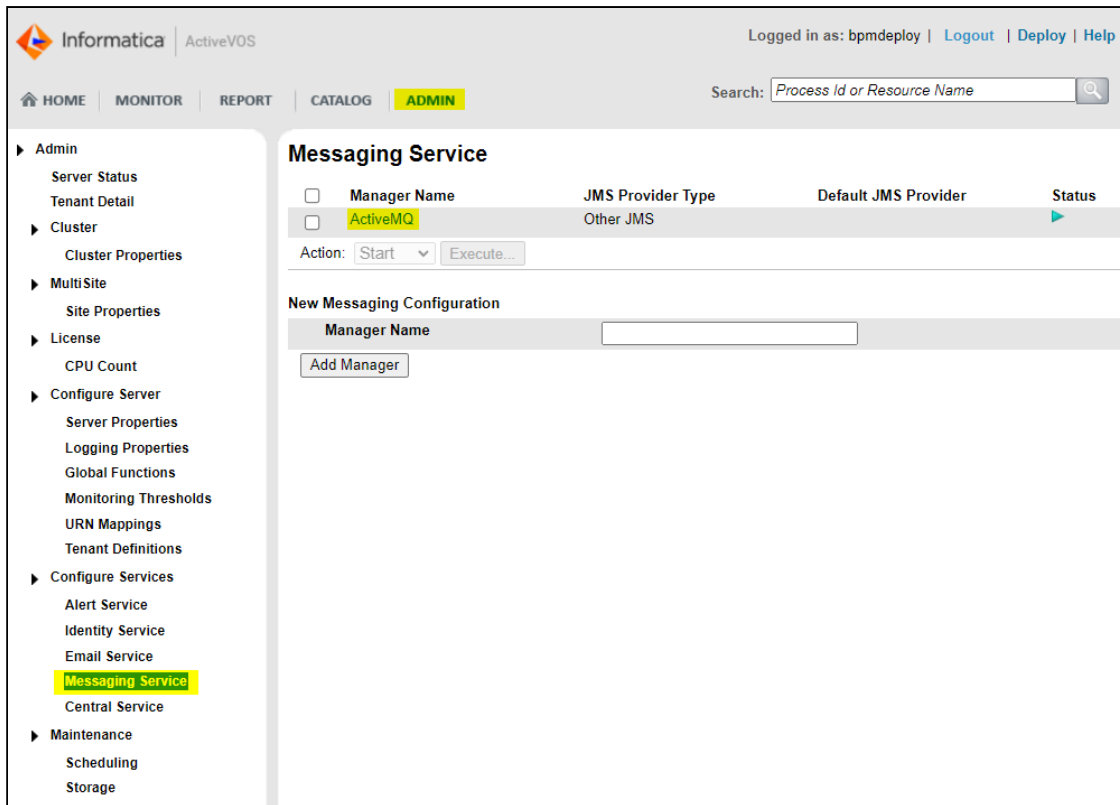
The base configuration of the DAM Event Listener, which is the base for creating derivatives, must be defined in the BPM URN mappings. Therefore you have to define the following mappings in the BPM console. These settings will be done from Informatica OPS team when the customer request it.

URN mappings

URN	URL	M a n d .	Description
urn:p360.dam .resource.proj ect	DAM_Resources	N o	Name of the resource project where the customer can define the derivatives, property fields, ...

Context properties for the messaging service

These 3 required context properties in the messaging service ActiveMQ have to be created. You can navigate to the service in the ActiveVOS console.



Name	Value
queue.JNDI_P360_DAM_ASSIGNMENT	heiler.hmm.backend.event.assignment
queue.JNDI_P360_DAM	heiler.hmm.backend.event
queue.JNDI_P360_DAM_MODIFIED	heiler.hmm.backend.event.assetModified

Database JNDI

ActiveVOS must be able to connect to the DAM related tables on the database, therefore you have to create an additional JNDI entry "**<Resource name="jdbc/DAM" auth="Container" type="javax.sql.DataSource"**" to the HMM tables, in the configuration file active-bpel.xml.

Schedule the maintenance job

DAM configuration changes can be applied via executing a maintenance job in the ActiveVOS console. Therefore you have to define a maintenance job. The workflow has to be started every time when you want to apply any configuration changes.

The screenshot shows the Informatica ActiveVOS Admin console. The left sidebar contains a navigation menu with categories like Admin, Cluster, Multi Site, License, Configure Server, Configure Services, Maintenance, and Storage. The 'Maintenance' category is expanded, and 'Scheduling' is selected. The main panel displays the 'Execution Schedule' configuration for a job named 'DAM Configurator'. The configuration includes fields for Name, Frequency, Start Date on Server, Start Time on Server, Run Options, Service Name, and Input Document. The 'Run Options' section has three radio buttons: 'Skip if running' (selected), 'Terminate if running', and 'Create new process'. The 'Input Document' field contains an XML snippet for a DAM initialization request.

**Execution Schedule Configuration:**

- Name:** DAM Configurator
- Frequency:** Once
- Start Date on Server:** 2022-06-08 (yyyy-mm-dd)
- Start Time on Server:** 00:00:00 (hh:mm:ss Coordinated Universal Time)
- Run Options:**
  - ☒ Skip if running
  - ☐ Terminate if running
  - ☐ Create new process
- Service Name:** DAM\_Init\_Values [Open Service Definitions](#)
- Input Document:**

```
<dam:damInitValuesRequest xmlns:dam="http://p360.dam/activevos/process/dam">
  <dam:updateValues>true</dam:updateValues>
</dam:damInitValuesRequest>
```

Buttons: OK, Cancel

Job settings

Key	Value
Name	DAM Configurator
Frequency	Once
Start Date on Server	Do not modify
Start Time on Server	Do not modify

Key	Value
Run Options	Skip if running
Service Name	DAM_Init_Values
Input Document	<pre>&lt;dam:damInitValuesRequest xmlns:dam="http:// p360.dam/activevos/process/damInitValues" xmlns:soap="http://www.w3.org/2003/05/soap- envelope"&gt;   &lt;dam:updateValues&gt;true&lt;/dam:updateValues&gt; &lt;/dam:damInitValuesRequest&gt;</pre>

### 2.5.1.5 Defining Main settings

Main settings must be defined with the help of a xml definition, which is located in the resource project which is defined in the URN mappings of ActiveVOS.

HOME

MONITOR

REPORT

CATALOG

ADMIN

Search:

Process Id or Resource Name

Catalog

Contributions

Process Definitions

Indexed Properties

Partner Definitions

Service Definitions

Task Properties

Resources

All

Central Configs

Function Contexts

HTML Documents

Images

Contributions

Contribution	Version	State	Date	Group	Deployer
project/DAM_Hotfolder	3.0	ONLINE	2022/06/02 01:44 PM		bpmdeploy
project/DAM_Resources	1.0	ONLINE	2022/06/02 02:03 PM		bpmdeploy
project/DAM_Tools	12.0	ONLINE	2022/06/03 06:11 AM		bpmdeploy
project/DemoVMResources	1.0	ONLINE	2022/03/21 11:33 AM		bpmdeploy
project/IW2022_Resources	11.0	ONLINE	2022/05/04 09:26 AM		bpmdeploy
project/InfNextSteps	3.0	ONLINE	2022/04/11 01:37 PM		bpmdeploy
project/InfResources	2.0	ONLINE	2022/03/25 08:15 PM		bpmdeploy
project/P360_BPM_Management	1.0	ONLINE	2022/03/25 08:14 PM		bpmdeploy
project/P360_JMS_Core	2.0	ONLINE	2022/03/25 08:14 PM		bpmdeploy
project/StepWorkflow	2.0	ONLINE	2022/04/12 07:20 PM		bpmdeploy
project/StepWorkflowExamples	2.0	ONLINE	2022/05/18 01:36 PM		bpmdeploy

**Contribution Detail**

Id: 113  
Contribution: project/DAM\_Resources  
Version: 1.0  
State: ONLINE  
Group:  
Description:

[Set to Offline](#) [Delete...](#)

Deployed Processes				
Name	Target Namespace	Version	State	Group
<b>Contributed Resources</b>				
Name	Version	Online	Target Namespace	Group
DAM_DefineDerivatives.xml	1.0	Yes		
DAM_DefineMainSettings.xml	1.0	Yes		
DAM_DefinePropFields.xml	1.0	Yes		

**Exported Namespaces**

Target Namespace

**Imported Namespaces**

Target Namespace	Contribution	Version

**Dependent Contributions:**

Contribution	Version	State	Date	Group	Deployer

[View Deployment Log](#)

If you want to make some modifications you have to click on the link, make your modifications and click on update.

The configuration file DAM\_DefineMainSettings.xml

### DAM\_DefineMainSettings.xml

```
<das:dataAccessResponse xmlns:das="http://schemas.active-endpoints.com/data-access/2010/04/data-access.xsd"
    statementId="string">
    <!-- Description of defining DAM main setting
        Tag: MODE: Defines the versioning behavior
        Allowed values:
            VERSIONING (A new version will be created when a file with
the same filename was uploaded. This new asset will inherit the values of the
previous version
            OVERWRITE (The current version will be replaced when a
file with the same filename was uploaded.)
-->
    <!-- Example
        <das:row>
            <MODE>OVERWRITE</MODE>
        </das:row>
    -->
</das:dataAccessResponse>
```



If you want to make some modifications you have to click on the link, make your modifications and click on update.

The configuration file DAM\_DefinePropFields.xml

#### DAM\_DefinePropFields.xml

```
<das:dataAccessResponse xmlns:das="http://schemas.active-endpoints.com/data-access/2010/04/data-access.xsd" statementId="string">
<!-- Description of defining DAM property fields
    Tag: IMF_LFD_NR: Mandatory number of the property field.
                        Allowed values between 1 and 110
    Tag: IMF_BEZ:      Description of the property field 50 characters at maximum.
                        Mandatory for all formats except format INVISIBLE
    Tag: IMF_FORMAT:   Mandatory format of the property field:
                        Allowed values for property field 1 to 100:
                            TEXT (Characters with maximum length of 3000 chars)
                            INVISIBLE (disables the property field)
                            BOOLEAN
                            DATE
                            INTEGER
                            NUMBER (Floating point number with 2 decimals at maximum)
                            SELECTIONLIST (List with selectable entries)
                            MULTISELECTIONLIST (List with multiple selectable entries)
                        Allowed values for property field 101 to 110:
                            LONGTEXT (Characters with no maximum length clob)
                            INVISIBLE (disables the property field)
    Tag: IMF_FORMATSIZE: Maximum size field. Only mandatory for these formats:
                        TEXT valid values from 1 to 3000
                        NUMBER (valid values from 0.2 up to 8.2)
    Tag: IMF_LISTVALUES: Values of the list. Each entry is separated by a
linefeed.Only mandatory for these formats:
                        SELECTIONLIST
                        MULTISELECTIONLIST
-->
<!-- Example TEXT field
    <das:row>
        <IMF_LFD_NR>88</IMF_LFD_NR>
        <IMF_BEZ>Number 88</IMF_BEZ>
        <IMF_FORMAT>TEXT</IMF_FORMAT>
        <IMF_FORMATSIZE>250</IMF_FORMATSIZE>
    </das:row>
-->
<!-- Example INIVISIBLE field
    <das:row>
        <IMF_LFD_NR>88</IMF_LFD_NR>
        <IMF_FORMAT>INVISIBLE</IMF_FORMAT>
    </das:row>
-->
<!-- Example BOOLEAN field
```

```

<das:row>
  <IMF_LFD_NR>88</IMF_LFD_NR>
  <IMF_BEZ>Number 88</IMF_BEZ>
  <IMF_FORMAT>BOOLEAN</IMF_FORMAT>
</das:row>
-->
<!-- Example DATE field
<das:row>
  <IMF_LFD_NR>88</IMF_LFD_NR>
  <IMF_BEZ>Number 88</IMF_BEZ>
  <IMF_FORMAT>DATE</IMF_FORMAT>
</das:row>
-->
<!-- Example INTEGER field
<das:row>
  <IMF_LFD_NR>88</IMF_LFD_NR>
  <IMF_BEZ>Number 88</IMF_BEZ>
  <IMF_FORMAT>INTEGER</IMF_FORMAT>
</das:row>
-->
<!-- Example NUMBER field
<das:row>
  <IMF_LFD_NR>88</IMF_LFD_NR>
  <IMF_BEZ>Number 88</IMF_BEZ>
  <IMF_FORMAT>NUMBER</IMF_FORMAT>
  <IMF_FORMATSIZE>3.1</IMF_FORMATSIZE>
</das:row>
-->
<!-- Example SELECTIONLIST field
<das:row>
  <IMF_LFD_NR>88</IMF_LFD_NR>
  <IMF_BEZ>Number 88</IMF_BEZ>
  <IMF_FORMAT>SELECTIONLIST</IMF_FORMAT>
  <IMF_FORMATSIZE></IMF_FORMATSIZE>
  <IMF_LISTVALUES>2020
2021
2022
2023</IMF_LISTVALUES>
</das:row>
-->
<!-- Example MULTISELECTIONLIST field
<das:row>
  <IMF_LFD_NR>88</IMF_LFD_NR>
  <IMF_BEZ>Number 88</IMF_BEZ>
  <IMF_FORMAT>MULTISELECTIONLIST</IMF_FORMAT>
  <IMF_FORMATSIZE></IMF_FORMATSIZE>
  <IMF_LISTVALUES>2020
2021
2022
2023</IMF_LISTVALUES>
</das:row>
-->

```



```

<!-- Example LONGTEXT field
  <das:row>
    <IMF_LFD_NR>105</IMF_LFD_NR>
    <IMF_BEZ>Number 105</IMF_BEZ>
    <IMF_FORMAT>LONGTEXT</IMF_FORMAT>
  </das:row>
-->
</das:dataAccessResponse>

```

Example define 3 different property fields

### Property fields

All property fields have to be defined in one xml definition!

Example requirement:

Type	Description
Text	Text field with a maximum length of 250 characters
List	List of selectable entries. I.e. publication dates (2018, 2019, 2020, 2021, 2022)
Text	Text field with unlimited characters

### EXAMPLE\_DAM\_DefinePropFields.xml

```

<das:dataAccessResponse xmlns:das="http://schemas.active-endpoints.com/data-access/2010/04/data-access.xsd" statementId="string">

  <das:row>
    <IMF_LFD_NR>1</IMF_LFD_NR>
    <IMF_BEZ>Field 1</IMF_BEZ>
    <IMF_FORMAT>TEXT</IMF_FORMAT>
    <IMF_FORMATSIZE>250</IMF_FORMATSIZE>
  </das:row>

  <das:row>
    <IMF_LFD_NR>2</IMF_LFD_NR>
    <IMF_BEZ>Field 2</IMF_BEZ>
    <IMF_FORMAT>SELECTIONLIST</IMF_FORMAT>
    <IMF_FORMATSIZE></IMF_FORMATSIZE>
  </das:row>

```

```

    <IMF_LISTVALUES>2018
2019
2020
2021
2022</IMF_LISTVALUES>
  </das:row>

  <das:row>
    <IMF_LFD_NR>101</IMF_LFD_NR>
    <IMF_BEZ>Field 3</IMF_BEZ>
    <IMF_FORMAT>LONGTEXT</IMF_FORMAT>
  </das:row>

</das:dataAccessResponse>

```

### 2.5.1.7 Defining Derivatives

Derivatives must be defined with the help of a xml definition, which is located in the resource project which is defined in the URN mappings of ActiveVOS.

#### ImageMagick

Only conversions for images with the command line tool convert from ImageMagick are supported!

HOME

MONITOR

REPORT

CATALOG

ADMIN

Search:

Catalog

Contributions

Process Definitions

Indexed Properties

Partner Definitions

Service Definitions

Task Properties

Resources

All

Central Configs

Function Contexts

HTML Documents

Images

Contributions

Contribution	Version	State	Date	Group	Deployer
project/DAM_Hotfolder	3.0	ONLINE	2022/06/02 01:44 PM		bpmdeploy
project/DAM_Resources	1.0	ONLINE	2022/06/02 02:03 PM		bpmdeploy
project/DAM_Tools	12.0	ONLINE	2022/06/03 06:11 AM		bpmdeploy
project/DemoVMResources	1.0	ONLINE	2022/03/21 11:33 AM		bpmdeploy
project/IW2022_Resources	11.0	ONLINE	2022/05/04 09:26 AM		bpmdeploy
project/InfNextSteps	3.0	ONLINE	2022/04/11 01:37 PM		bpmdeploy
project/InfResources	2.0	ONLINE	2022/03/25 08:15 PM		bpmdeploy
project/P360_BPM_Management	1.0	ONLINE	2022/03/25 08:14 PM		bpmdeploy
project/P360_JMS_Core	2.0	ONLINE	2022/03/25 08:14 PM		bpmdeploy
project/StepWorkflow	2.0	ONLINE	2022/04/12 07:20 PM		bpmdeploy
project/StepWorkflowExamples	2.0	ONLINE	2022/05/18 01:36 PM		bpmdeploy

Informatica | ActiveVOS Logged in as: bpmdeploy | [Logout](#) | [Deploy](#) | [Help](#)

HOME | MONITOR | REPORT | CATALOG | ADMIN Search:

**Catalog**

- Contributions
- Process Definitions
- Indexed Properties
- Partner Definitions
- Service Definitions
- Task Properties
- Resources
  - All
  - Central Configs
  - Function Contexts
  - HTML Documents
  - Images
  - Java Jars
  - Report Definitions
  - Schema Documents
  - WSDL Documents
  - XQuery Modules
  - XSL Documents
  - Other

### Contribution Detail

Id: 113  
 Contribution: project/DAM\_Resources  
 Version: 1.0  
 State: ONLINE  
 Group:  
 Description:

[Set to Offline](#) [Delete...](#)

Deployed Processes		Version	State	Group
Name	Target Namespace			
Contributed Resources				
Name	Version	Online	Target Namespace	Group
DAM_DefineDerivatives.xml	1.0	Yes		
DAM_DefineMainSettings.xml	1.0	Yes		
DAM_DefinePropFields.xml	1.0	Yes		
Exported Namespaces				
Target Namespace				
Imported Namespaces				
Target Namespace		Contribution	Version	
Dependent Contributions:				
Contribution	Version	State	Date	Group
Deployment Log				
<a href="#">View Deployment Log</a>				

If you want to make some modifications you have to click on the link, make your modifications and click on update.

The configuration file DAM\_DefineDerivatives.xml

#### DAM\_DefineDerivatives.xml

```
<das:dataAccessResponse xmlns:das="http://schemas.active-endpoints.com/data-access/2010/04/data-access.xsd" statementId="string">
<!-- Description of defining DAM property fields
    Tag: DEV_ID:      Mandatory number of the derivate.
                     Allowed values between 1 and 100
    Tag: DEV_NAME:    Mandatory description of the derivate 50 characters at
maximum.
                     If this value is empty the derivative with this id will be
deleted!
    Tag: DEV_EXT:     Mandatory extension of the derivate 10 characters at maximum.
    Tag: DEV_CONVERT_PARAMS:
                     Optional parameters for the ImageMagick command convert. If
this value is set a derivative will be created otherwise not!
                     ONLY_LINK (Create a symbolic link as derivative) (Only
available on linux!)
                     ONLY_COPY_WIN (Create a copy windows)
                     ONLY_COPY_UX (Create a copy linux)
    Tag: DEV_PUBLIC:
                     Optional parameter which defines whether the derivative will
be copied into the public space. Valid values: Yes, No, 1,0 (Beta for Azure!)
    Tag: DEV_PROP_FIELD_ID:
```

Optional parameter defines whether the public path will be pushed into a property field. Allowed values between 1 and 100

Tag: DEV\_FNAME\_START:

Optional parameter defines starting string of derivative file name

Tag: DEV\_FNAME\_END:

Optional parameter defines ending string of derivative file name

-->

<!-- Examples

```
<das:row>
  <DEV_ID>1</DEV_ID>
  <DEV_NAME>Jpeg 70x70</DEV_NAME>
  <DEV_EXT>.jpg</DEV_EXT>
  <DEV_CONVERT_PARAMS>-alpha off -flatten -strip -antialias -quality 80 -resize
70x70 -background white -gravity center -extent 70x70 -density 72x72 -colorspace sRGB
"%1"[0] "%2"</DEV_CONVERT_PARAMS>
  <DEV_PUBLIC>YES</DEV_PUBLIC>
  <DEV_PUBLIC_PROP_FIELD_ID>81</DEV_PUBLIC_PROP_FIELD_ID>
  <DEV_FNAME_START>JP70_</DEV_FNAME_START>
  <DEV_FNAME_END></DEV_FNAME_END>
</das:row>
<das:row>
  <DEV_ID>2</DEV_ID>
  <DEV_NAME>Jpeg 300x300</DEV_NAME>
  <DEV_EXT>.jpg</DEV_EXT>
  <DEV_CONVERT_PARAMS>-alpha off -flatten -strip -antialias -quality 80 -resize
300x300 -background white -gravity center -extent 300x300 -density 72x72 -colorspace
sRGB "%1"[0] "%2"</DEV_CONVERT_PARAMS>
  <DEV_PUBLIC>YES</DEV_PUBLIC>
  <DEV_PUBLIC_PROP_FIELD_ID>82</DEV_PUBLIC_PROP_FIELD_ID>
  <DEV_FNAME_START></DEV_FNAME_START>
  <DEV_FNAME_END></DEV_FNAME_END>
</das:row>
<das:row>
  <DEV_ID>3</DEV_ID>
  <DEV_NAME>PNG 1024x1024</DEV_NAME>
  <DEV_EXT>.png</DEV_EXT>
  <DEV_CONVERT_PARAMS>-alpha off -flatten -strip -antialias -quality 80 -resize
1024x1024 -background white -gravity center -extent 1024x1024 -density 72x72
-colorspace sRGB "%1"[0] "%2"</DEV_CONVERT_PARAMS>
  <DEV_PUBLIC>YES</DEV_PUBLIC>
  <DEV_PUBLIC_PROP_FIELD_ID>83</DEV_PUBLIC_PROP_FIELD_ID>
  <DEV_FNAME_START></DEV_FNAME_START>
  <DEV_FNAME_END></DEV_FNAME_END>
</das:row>
<das:row>
  <DEV_ID>4</DEV_ID>
  <DEV_NAME>Only Copy</DEV_NAME>
  <DEV_EXT>.jpg</DEV_EXT>
  <DEV_CONVERT_PARAMS>ONLY_LINK "%1" "%2"</DEV_CONVERT_PARAMS>
  <DEV_PUBLIC>YES</DEV_PUBLIC>
```

```

    <DEV_PUBLIC_PROP_FIELD_ID>84</DEV_PUBLIC_PROP_FIELD_ID>
    <DEV_FNAME_START></DEV_FNAME_START>
    <DEV_FNAME_END></DEV_FNAME_END>
  </das:row>
-->
</das:dataAccessResponse>

```

Example define a 70 pixel RGB Jpeg as derivative

#### Example 70x70

```

<das:dataAccessResponse xmlns:das="http://schemas.active-endpoints.com/data-access/
2010/04/data-access.xsd" statementId="string">
  <das:row>
    <DEV_ID>1</DEV_ID>
    <DEV_NAME>Jpeg 70x70</DEV_NAME>
    <DEV_EXT>.jpg</DEV_EXT>
    <DEV_CONVERT_PARAMS>-alpha off -flatten -strip -antialias -quality 80 -resize
70x70 -background white -gravity center -extent 70x70 -density 72x72 -colorspace sRGB
"%1"[0] "%2"</DEV_CONVERT_PARAMS>
    <DEV_PUBLIC>NO</DEV_PUBLIC>
    <DEV_PUBLIC_PROP_FIELD_ID></DEV_PUBLIC_PROP_FIELD_ID>
  </das:row>
</das:dataAccessResponse>

```

| Key                            | Value  | Description  |
|--------------------------------|--|--|
| DEV_ID                         | 1  | Unique identifier of the derivate  |
| DEV_N<br>AME                   | Jpeg 70x70   | Name of the derivative   |
| DEV_EX<br>T                    | .jpg   | New extension of the converted image   |
| DEV_C<br>ONVER<br>T_PAR<br>AMS | -alpha off -flatten -strip -antialias -quality 80 -resize 70x70 -background white -gravity center -extent 70x70 -density 72x72 -colorspace sRGB "%1"[0] "%2" | Command line parameters for the command convert to create a jpeg derivative with the dimension 70x70 in the colorspace RGB |

| Key                                  | Value | Description  |
|--------------------------------------|-------|--|
| DEV_P<br>UBLIC                       | NO    | This tag defines whether the derivative will also pushed to a public accessible blob storage. Please create a CTP ticket to create this storage. |
| DEV_P<br>UBLIC_<br>PROP_F<br>IELD_ID | empty | The public URL to the derivative will be stored in this property field, when DEV_PUBLIC is YES.  |

### 2.5.1.8 Apply DAM configurations

As mentioned above, every time you made some configuration changes you have to run the workflow again. The workflow can be started in the ActiveVOS console.

Informatica ActiveVOS

Logged in as: bpmdeploy | [Logout](#) | [Deploy](#) | [Help](#)

Search:

ADMIN

Admin

- Server Status
- Tenant Detail
- Cluster
  - Cluster Properties
- Multi Site
  - Site Properties
- License
  - CPU Count
- Configure Server
  - Server Properties
  - Logging Properties
  - Global Functions
  - Monitoring Thresholds
  - URN Mappings
  - Tenant Definitions
- Configure Services
  - Alert Service
  - Identity Service
  - Email Service
  - Messaging Service
  - Central Service
- Maintenance
  - Scheduling**

**Scheduling**

[Create Schedule](#)

Current Server Time: 2022-06-08 07:43:01  
Schedule Server: P360-ActiveVOS-us-west-2a

| <input type="checkbox"/>            | Enabled | Name               | Service Name       | Last Status | Last Execution      | Next Execution |
|-------------------------------------|---------|--------------------|--------------------|-------------|---------------------|----------------|
| <input checked="" type="checkbox"/> |         | DAM Configurator   | DAM_Init_Values    | Completed   | 2022-06-08 07:37:39 | -              |
| <input type="checkbox"/>            |         | DAM Event Listener | DAM_Event_Listener | Running     | 2022-06-07 13:43:14 | -              |
| <input type="checkbox"/>            |         | DAM Hotfolder      | Hotfolder_aws      | Running     | 2022-06-08 07:43:01 | -              |

☐ Select All / None

[Run Now](#) [Duplicate](#) [Delete](#) [Enable](#) [Disable](#)

After clicking on Run Now you should see the following processes

The screenshot shows the Informatica ActiveVOS Monitor interface. The top navigation bar includes 'HOME', 'MONITOR' (selected), 'REPORT', 'CATALOG', and 'ADMIN'. A search bar is present with the text 'Process Id or Resource Name'. The left sidebar shows a tree view with 'Monitor' expanded, containing 'Process Monitoring' and 'Task Monitoring'. Under 'Process Monitoring', 'Active Processes' is selected. The main area displays a table of active processes.

| ID                             | Title                        | Ver. | Start Date       | End Date         | State     | Group              |
|--------------------------------|------------------------------|------|------------------|------------------|-----------|--------------------|
| <input type="checkbox"/> 66216 | Prepare Derivatives [OK]     | 12.0 | 2022-06-08 07:47 | 2022-06-08 07:47 | Completed | DAM_Tools          |
| <input type="checkbox"/> 66215 | Prepare Property Fields [OK] | 8.0  | 2022-06-08 07:47 | 2022-06-08 07:47 | Completed | DAM_Tools          |
| <input type="checkbox"/> 66214 | Prepare Main Settings [OK]   | 8.0  | 2022-06-08 07:47 | 2022-06-08 07:47 | Completed | DAM_Tools          |
| <input type="checkbox"/> 66213 | DAM_InitValues               | 12.0 | 2022-06-08 07:47 | 2022-06-08 07:47 | Completed | DAM_Tools          |
| <input type="checkbox"/> 66208 | DAM-Hotfolder AWS            | 3.0  | 2022-06-08 07:46 |                  | Running   | P360_DAM_Hotfolder |
| <input type="checkbox"/> 66207 | DAM-Event Listener           | 12.0 | 2022-06-08 07:46 |                  | Running   | DAM_Tools          |

## 2.5.2 DAM Event Listener

### Cloud Edition only

This project is designed for Product 360 Cloud Edition customers only. (On premise customers can use this project and modify it to their needs, but it is not under Informatica's support!)

### 2.5.2.1 Preface

This BPM (AVOS) project enables Product 360 Cloud Edition customer's to create derivatives for Digital Assets image files via converting them with ImageMagick. This functionality works only for the Asset Provider HMM, which is the default in the Cloud Edition settings.

#### Table of Contents

- [Preface](#) (see page 459)
- [Prerequisites](#) (see page 459)
- [Installation](#) (see page 460)
- [Configuration](#) (see page 460)
  - [URN mappings for AWS](#) (see page 460)
  - [URN mappings for Azure](#) (see page 461)
  - [Context properties for the messaging service](#) (see page 463)
  - [Database JNDI](#) (see page 463)
  - [Schedule the maintenance job](#) (see page 464)
- [Defining Derivatives](#) (see page 465)
  - [The configuration file DAM\\_DefineDerivatives.xml](#) (see page 466)
  - [Example define a 70 pixel RGB Jpeg as derivative](#) (see page 467)
  - [Default states](#) (see page 468)
- [Start the DAM Event Listener](#) (see page 469)
- [Verification](#) (see page 470)

### 2.5.2.2 Prerequisites

The DAM-event-listener also supports the auto assignment functionality over the message queue communication. Therefore the messaging service of ActiveVOS must be configured. Please reach out to the Installation and Operation guide for the following areas:

- Installation of ActiveVOS with ActiveMQ support and configuration of queues in ActiveVOS (*Note: ActiveVOS version 9.2.4.6 is a prerequisite for queue based interactions between Product 360 and ActiveVOS*)

### 2.5.2.3 Installation

You have to deploy these resources:

1. InfaResources.bpr
2. DAM\_Tools.bpr
3. DAM\_Resources.bpr

with the BPM (AVOS) console.

Also it is possible that you deploy the workflows with your BPM designer. The resources of the projects are in the zip files:

1. InfaResources.zip
2. DAM\_Tools.zip
3. DAM\_Resources.zip

### 2.5.2.4 Configuration

The base configuration of the DAM Event Listener, which is the base for creating derivatives, must be defined in the BPM URN mappings. Therefore you have to define the following mappings in the BPM console. These settings will be done from Informatica OPS team when the customer request it.

URN mappings for AWS

| URN                           | URL                | Ma<br>nd. | Description  |
|-------------------------------|--------------------|-----------|--|
| urn:p360.aws.s3.bucket.name   | pim-example-bucket | Yes       | S3 bucket name of the customer's environment                               |
| urn:p360.dam.max.threads      | 2                  | No        | Maximum number of parallel ActiveVOS threads while creating new assets     |
| urn:p360.dam.resource.project | DAM_Resources      | No        | Name of the resource project where the customer can define the derivatives |



| URN                                      | URL  | Mand. | Description   |
|--|--|-------|---|
| urn:p360.dam.aws.public.path             | /qa/P360Server/public/   | No    | Public path on the S3 bucket, where the derivatives can be pushed                         |
| urn:p360.dam.aws.public.url              | https://pim-iw2022-bucket.s3.amazonaws.com/qa/P360Server/public/             | No    | Url to the public S3 bucket (can be used instead of urn:p360.dam.aws.public.path)         |
| urn:p360.dam.azure.bloburl               | https://mycompany.blob.core.windows.net/pim-mycompany-asset-container-public | No    | Url to the public Azure Blob storage container  |
| urn:p360.dam.azure.sas.token             | sv=2020-10-02&st=2023-06-20T07%3A00%3A00Z&se=.....                           | No    | SAS token for the Azure Blob storage container  |
| urn:p360.dam.event.listener.max.attempts | 7200   | No    | Number of processed events until the process restarts automatically. [Default value 7200] |

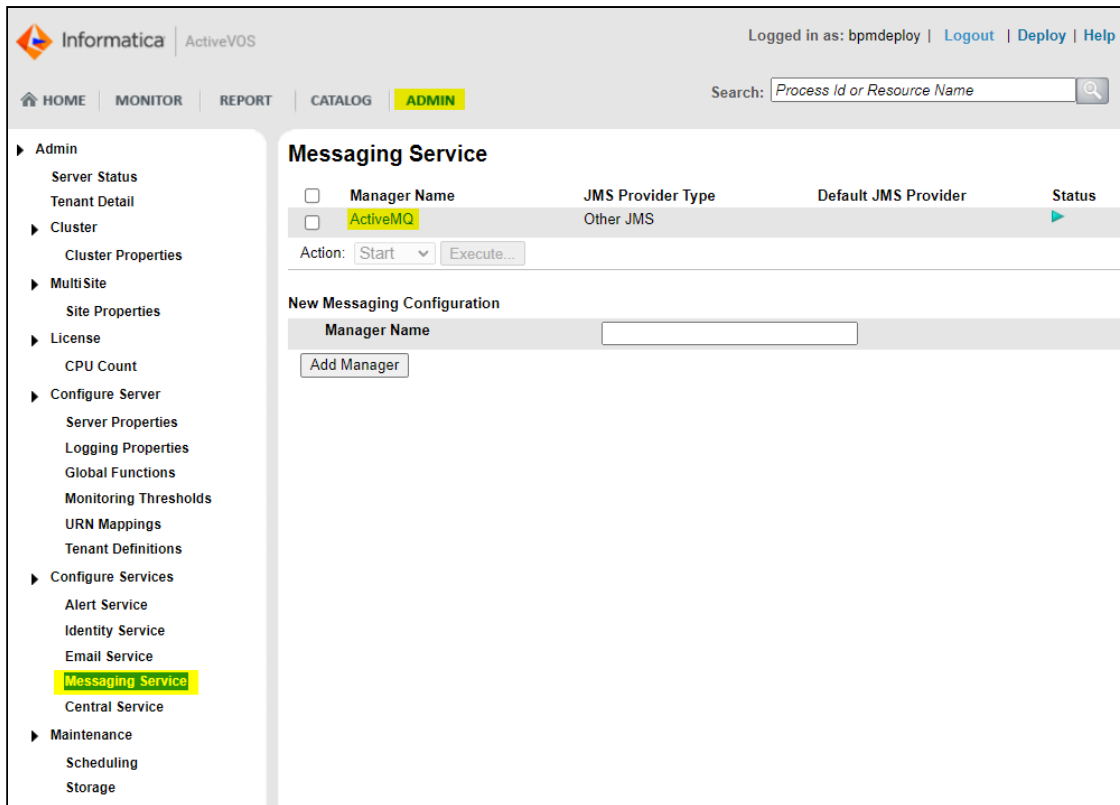
#### URN mappings for Azure

| URN                            | URL                                | Mand. | Description   |
|--------------------------------|------------------------------------|-------|---|
| urn:p360.dam.max.threads       | 2                                  | No    | Maximum number of parallel ActiveVOS threads while creating new assets      |
| urn:p360.dam.resource.project  | DAM_Resources                      | No    | Name of the resource project where the customer can define the derivatives  |
| urn:p360.dam.azure.public.path | /opt/informatica/shares/S3_PUBLIC/ | No    | Public path on the public blob storage, where the derivatives can be pushed |

| URN                                      | URL   | M<br>a<br>n<br>d<br>. | Description   |
|--|---|-----------------------|---|
| urn:p360.dam.azure.public.url            | <a href="https://voldemononprodsa.blob.core.windows.net/pim-voldemo-customer-container-nonprod-public/">https://voldemononprodsa.blob.core.windows.net/pim-voldemo-customer-container-nonprod-public/</a> | N<br>o                | Url to the public blob storage  |
| urn:p360.dam.event.listener.max.attempts | 7200  | N<br>o                | Number of processed events until the process restarts automatically. [Default value 7200] |
| urn:p360.dam.volume0.path                | /data/opt/informatica/shares/S3/dev/efs/immvolumes/volume0  | N<br>o                | Needed for version 10.1   |

Context properties for the messaging service

These 3 required context properties in the messaging service ActiveMQ have to be created. You can navigate to the service in the ActiveVOS console.



| Name                           | Value                                  |
|--------------------------------|--|
| queue.JNDI_P360_DAM_ASSIGNMENT | heiler.hmm.backend.event.assignment    |
| queue.JNDI_P360_DAM            | heiler.hmm.backend.event               |
| queue.JNDI_P360_DAM_MODIFIED   | heiler.hmm.backend.event.assetModified |

Database JNDI

ActiveVOS must be able to connect to the DAM related tables on the database, therefore you have to create an additional JNDI entry "**<Resource name="jdbc/DAM" auth="Container" type="javax.sql.DataSource"**" to the HMM tables, in the configuration file active-bpel.xml.

Schedule the maintenance job

The DAM-Event-Listener is implemented as an endless loop. The workflow has to be started one time via creating a scheduled maintenance job via the ActiveVOS console.

The screenshot shows the Informatica ActiveVOS Scheduling console. The left sidebar contains a navigation tree with categories like Admin, Cluster, MultiSite, License, Configure Server, Configure Services, Maintenance, and Storage. The 'Maintenance' category is expanded, and 'Scheduling' is selected. The main panel is titled 'Scheduling' and contains a 'Create Schedule' button. Below this, there are checkboxes for 'Enabled' and 'Select / Run Now'. The 'Execution Schedule' section is visible, showing the following configuration:

- Name:** DAM Event Listener
- Frequency:** Once
- Start Date on Server:** 2022-06-02 (with a calendar icon and format yyyy-mm-dd)
- Start Time on Server:** 00:00:00 (with a clock icon and format hh:mm:ss Coordinated Universal Time)
- Run Options:**
  - ☒ Skip if running
  - ☐ Terminate if running
  - ☐ Create new process
- Service Name:** DAM\_Event\_Listener (with a link 'Open Service Definitions')
- Input Document:**

```
<dam:startDAMEventListenerRequest xmlns:dam="http://p360.dam/activevos/pr
<dam:listencreateevent>true</dam:listencreateevent>
<dam:listenmodifyevent>true</dam:listenmodifyevent>
</dam:startDAMEventListenerRequest>
```

At the bottom of the console are 'OK' and 'Cancel' buttons.

Job settings

| Key                  | Value              |
|----------------------|--------------------|
| Name                 | DAM_Event_Listener |
| Frequency            | Once               |
| Start Date on Server | Do not modify      |
| Start Time on Server | Do not modify      |

| Key            | Value   |
|----------------|---|
| Run Options    | Skip if running   |
| Service Name   | DAM_Event_Listener  |
| Input Document | <pre>&lt;dam:startDAMEventListenerRequest xmlns:dam="http://p360.dam/activevos/process/DamEventListener" xmlns:soap="http://www.w3.org/2003/05/soap-envelope"&gt;   &lt;dam:listencreateevent&gt;true&lt;/dam:listencreateevent&gt;   &lt;dam:listenmodifyevent&gt;true&lt;/dam:listenmodifyevent&gt; &lt;/dam:startDAMEventListenerRequest&gt;</pre> |

### 2.5.2.5 Defining Derivatives

Derivatives must be defined with the help of a xml definition, which is located in the resource project which is defined in the URN mappings of ActiveVOS.

#### ImageMagick

Only conversions for images with the command line tool convert from ImageMagick are supported!

| <div> <div>HOME</div> <div>MONITOR</div> <div>REPORT</div> <div>CATALOG</div> <div>ADMIN</div> </div> <div>Search: <input type="text" value="Process Id or Resource Name"/></div>  |         |        |                     |       |           |  |
|--|---------|--------|---------------------|-------|-----------|--|
| <div> <div>Catalog</div> <div>Contributions</div> <div>Process Definitions</div> <div>Indexed Properties</div> <div>Partner Definitions</div> <div>Service Definitions</div> <div>Task Properties</div> <div>Resources</div> <div>All</div> <div>Central Configs</div> <div>Function Contexts</div> <div>HTML Documents</div> <div>Images</div> <div>Java Tools</div> </div> |         |        |                     |       |           |  |
| Contributions  |         |        |                     |       |           |  |
| Contribution   | Version | State  | Date                | Group | Deployer  |  |
| <a href="#">project/DAM_Hotfolder</a>  | 3.0     | ONLINE | 2022/06/02 01:44 PM |       | bpmdeploy |  |
| <a href="#">project/DAM_Resources</a>  | 1.0     | ONLINE | 2022/06/02 02:03 PM |       | bpmdeploy |  |
| <a href="#">project/DAM_Tools</a>  | 12.0    | ONLINE | 2022/06/03 06:11 AM |       | bpmdeploy |  |
| <a href="#">project/DemoVMResources</a>  | 1.0     | ONLINE | 2022/03/21 11:33 AM |       | bpmdeploy |  |
| <a href="#">project/IW2022_Resources</a>   | 11.0    | ONLINE | 2022/05/04 09:26 AM |       | bpmdeploy |  |
| <a href="#">project/InfNextSteps</a>   | 3.0     | ONLINE | 2022/04/11 01:37 PM |       | bpmdeploy |  |
| <a href="#">project/InfResources</a>   | 2.0     | ONLINE | 2022/03/25 08:15 PM |       | bpmdeploy |  |
| <a href="#">project/P360_BPM_Management</a>  | 1.0     | ONLINE | 2022/03/25 08:14 PM |       | bpmdeploy |  |
| <a href="#">project/P360_JMS_Core</a>  | 2.0     | ONLINE | 2022/03/25 08:14 PM |       | bpmdeploy |  |
| <a href="#">project/StepWorkflow</a>   | 2.0     | ONLINE | 2022/04/12 07:20 PM |       | bpmdeploy |  |
| <a href="#">project/StepWorkflowExamples</a>   | 2.0     | ONLINE | 2022/05/18 01:36 PM |       | bpmdeploy |  |

Informatica ActiveVOS

Logged in as: bpmdeploy | [Logout](#) | [Deploy](#) | [Help](#)

HOME | MONITOR | REPORT | CATALOG | ADMIN

Search:

**Contribution Detail**

Id: 113  
 Contribution: project/DAM\_Resources  
 Version: 1.0  
 State: ONLINE  
 Group:  
 Description:

[Set to Offline](#) [Delete...](#)

| Deployed Processes                  |                  | Version      | State            | Group |
|-------------------------------------|------------------|--------------|------------------|-------|
| Name                                | Target Namespace |              |                  |       |
| Contributed Resources               |                  |              |                  |       |
| Name                                | Version          | Online       | Target Namespace | Group |
| DAM_DefineDerivatives.xml           | 1.0              | Yes          |                  |       |
| DAM_DefineMainSettings.xml          | 1.0              | Yes          |                  |       |
| DAM_DefinePropFields.xml            | 1.0              | Yes          |                  |       |
| Exported Namespaces                 |                  |              |                  |       |
| Target Namespace                    |                  |              |                  |       |
| Imported Namespaces                 |                  |              |                  |       |
| Target Namespace                    |                  | Contribution | Version          |       |
| Dependent Contributions:            |                  |              |                  |       |
| Contribution                        | Version          | State        | Date             | Group |
| Deployment Log                      |                  |              |                  |       |
| <a href="#">View Deployment Log</a> |                  |              |                  |       |

If you want to make some modifications you have to click on the link, make your modifications and click on update.

The configuration file DAM\_DefineDerivatives.xml

### DAM\_DefineDerivatives.xml

```
<das:dataAccessResponse xmlns:das="http://schemas.active-endpoints.com/data-access/2010/04/data-access.xsd" statementId="string">
<!-- Description of defining DAM property fields
    Tag: DEV_ID:      Mandatory number of the derivate.
                     Allowed values between 1 and 100
    Tag: DEV_NAME:    Mandatory description of the derivate 50 characters at
maximum.
                     If this value is empty the derivative with this id will be
deleted!
    Tag: DEV_EXT:     Mandatory extension of the derivate 10 characters at maximum.
    Tag: DEV_CONVERT_PARAMS:
                     Optional parameters for the ImageMagick command convert. If
this value is set a derivative will be created otherwise not!
    Tag: DEV_PUBLIC:
                     Optional parameter which defines whether the derivative will
be copied into the public space. Valid values: Yes, No, 1,0
    Tag: DEV_PROP_FIELD_ID:
                     Optional parameter defines whether the public path will be
pushed into a property field. Allowed values between 1 and 100
-->
```

```

<!-- Examples
  <das:row>
    <DEV_ID>1</DEV_ID>
    <DEV_NAME>Jpeg 70x70</DEV_NAME>
    <DEV_EXT>.jpg</DEV_EXT>
    <DEV_CONVERT_PARAMS>-alpha off -flatten -strip -antialias -quality 80 -resize
70x70 -background white -gravity center -extent 70x70 -density 72x72 -colorspace sRGB
"%1"[0] "%2"</DEV_CONVERT_PARAMS>
    <DEV_PUBLIC>YES</DEV_PUBLIC>
    <DEV_PUBLIC_PROP_FIELD_ID>81</DEV_PUBLIC_PROP_FIELD_ID>
  </das:row>
  <das:row>
    <DEV_ID>2</DEV_ID>
    <DEV_NAME>Jpeg 300x300</DEV_NAME>
    <DEV_EXT>.jpg</DEV_EXT>
    <DEV_CONVERT_PARAMS>-alpha off -flatten -strip -antialias -quality 80 -resize
300x300 -background white -gravity center -extent 300x300 -density 72x72 -colorspace
sRGB "%1"[0] "%2"</DEV_CONVERT_PARAMS>
    <DEV_PUBLIC>YES</DEV_PUBLIC>
    <DEV_PUBLIC_PROP_FIELD_ID>82</DEV_PUBLIC_PROP_FIELD_ID>
  </das:row>
  <das:row>
    <DEV_ID>3</DEV_ID>
    <DEV_NAME>PNG 1024x1024</DEV_NAME>
    <DEV_EXT>.png</DEV_EXT>
    <DEV_CONVERT_PARAMS>-alpha off -flatten -strip -antialias -quality 80 -resize
1024x1024 -background white -gravity center -extent 1024x1024 -density 72x72
-colorspace sRGB "%1"[0] "%2"</DEV_CONVERT_PARAMS>
    <DEV_PUBLIC>YES</DEV_PUBLIC>
    <DEV_PUBLIC_PROP_FIELD_ID>83</DEV_PUBLIC_PROP_FIELD_ID>
  </das:row>
-->
</das:dataAccessResponse>

```

Example define a 70 pixel RGB Jpeg as derivative

### Example 70x70

```

<das:dataAccessResponse xmlns:das="http://schemas.active-endpoints.com/data-access/
2010/04/data-access.xsd" statementId="string">
  <das:row>
    <DEV_ID>1</DEV_ID>
    <DEV_NAME>Jpeg 70x70</DEV_NAME>
    <DEV_EXT>.jpg</DEV_EXT>
    <DEV_CONVERT_PARAMS>-alpha off -flatten -strip -antialias -quality 80 -resize
70x70 -background white -gravity center -extent 70x70 -density 72x72 -colorspace sRGB
"%1"[0] "%2"</DEV_CONVERT_PARAMS>
    <DEV_PUBLIC>NO</DEV_PUBLIC>
    <DEV_PUBLIC_PROP_FIELD_ID></DEV_PUBLIC_PROP_FIELD_ID>
  </das:row>
</das:dataAccessResponse>

```

```
</das:row>
</das:dataAccessResponse>
```

| Key                          | Value  | Description  |
|------------------------------|--|--|
| DEV_ID                       | 1  | Unique identifier of the derivate  |
| DEV_NAME                     | Jpeg 70x70   | Name of the derivative   |
| DEV_EXTENSION                | .jpg   | New extension of the converted image   |
| DEV_CONVERT_PARAMETERS       | -alpha off -flatten -strip -antialias -quality 80 -resize 70x70 -background white -gravity center -extent 70x70 -density 72x72 -colorspace sRGB "%1"[0] "%2" | Command line parameters for the command convert to create a jpeg derivative with the dimension 70x70 in the colorspace RGB                       |
| DEV_PUBLIC                   | NO   | This tag defines whether the derivative will also pushed to a public accessible blob storage. Please create a CTP ticket to create this storage. |
| DEV_PUBLIC_PROPERTY_FIELD_ID | empty  | The public URL to the derivative will be stored in this property field, when DEV_PUBLIC is YES.  |

Default states

By default the DAM-Event-Listener workflow will create 5 states.

| Id | Name                     |
|----|--------------------------|
| 0  | *No status defined*      |
| 1  | Create derivatives again |




| Id | Name                                   |
|----|--|
| 2  | Derivatives successfully created       |
| 3  | Error while creating derivatives       |
| 4  | No derivative definition for this type |

This state will be stored directly on the Media Asset.

TestHotfolder.jpg

Technical information
General information
Embedded meta data
Usage list
Derivatives
Image preview



Document identifier:  
D12000127205

Media asset file language-specific attribute languages
English

Name (English):  
Test Document

Memo (English):  
No content

\*No status defined\*
Create derivatives again
Derivatives successfully created
Error while creating derivatives
No derivative definition for this type

Derivatives successfully created

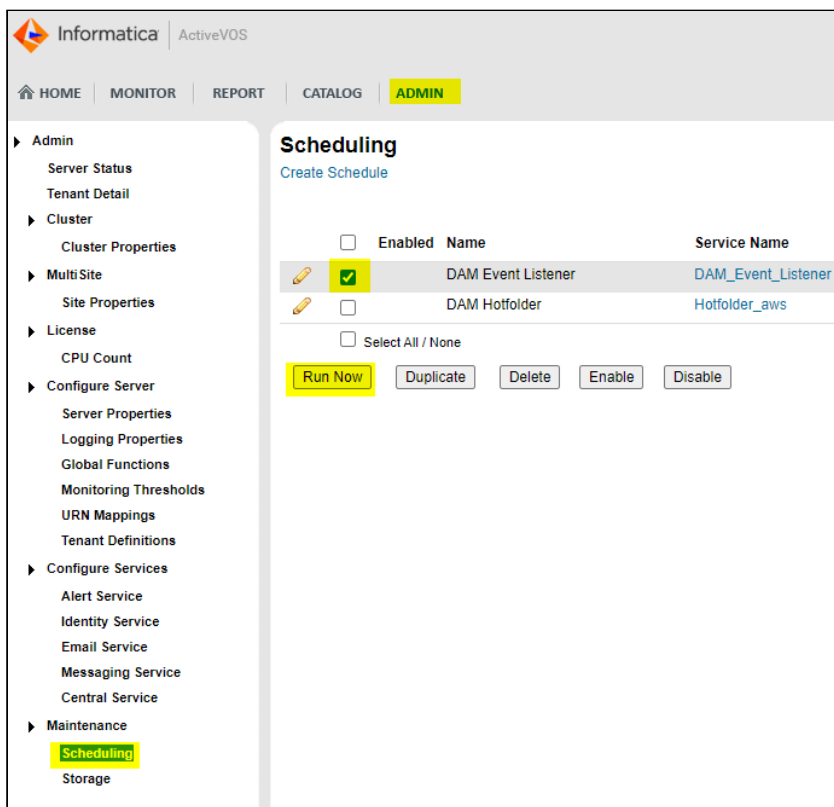
State:

Status:

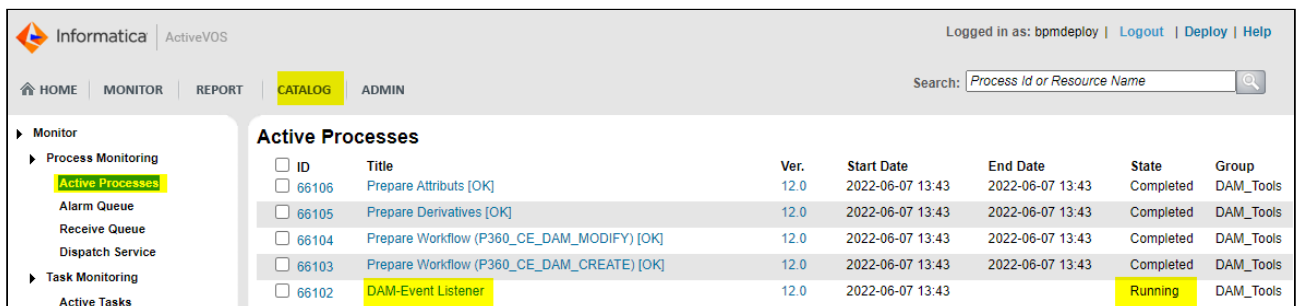
Free

### 2.5.2.6 Start the DAM Event Listener

The DAM Event Listener can be started in the ActiveVOS console.



Now the process should run. You can check this as well in the ActiveVOS console.



### Processes

You should see these processes.

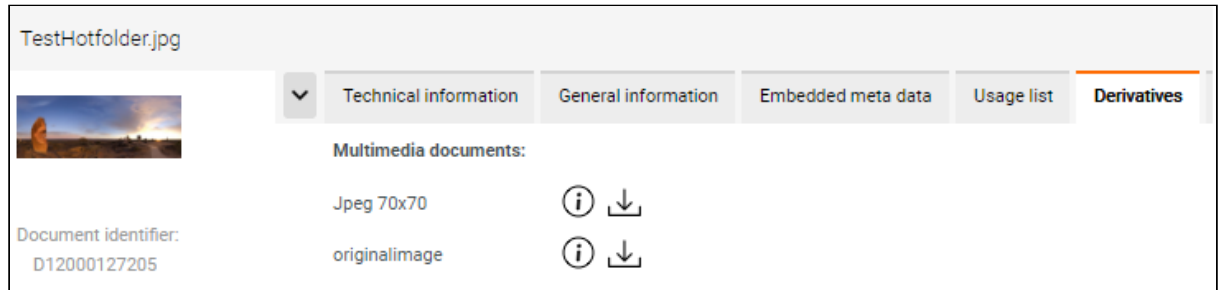
## 2.5.2.7 Verification

Let's assume you defined the derivative with the Identifier 1 in your DAM\_DefineDerivative.xml definition.

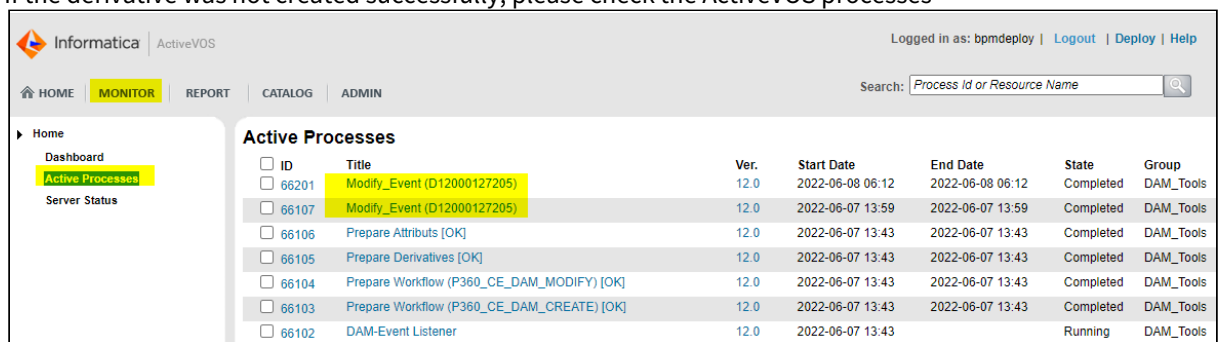
1. Login into you Product 360 web application
2. Click on Media



3. Search an asset
4. Go to the **General information** tab in the details section and set the state to **Create derivatives again**
5. Go to the **Derivatives** tab and refresh the page. If everything works as expected you should see the derivative



6. If the derivative was not created successfully, please check the ActiveVOS processes



## 2.5.3 DAM Hotfolder

### **Cloud Edition only**

This project is designed for Product 360 Cloud Edition customers only. (On premise customers can use this project and modify it to their needs, but it is not under Informatica's support! )

### 2.5.3.1 Preface

This BPM (ActiveVOS) project enables Product 360 Cloud Edition customer's to ingest Digital Assets via uploading the files into the blob storage. This hotfolder functionality works only for the Asset Provider HMM, which is the default in the Cloud Edition settings.

#### **Table of Contents**

- [Preface](#) (see page 471)
- [Prerequisites](#) (see page 472)
- [Installation](#) (see page 472)
- [Configuration](#) (see page 472)
  - [URN mappings for AWS](#) (see page 472)
  - [URN mappings for Azure](#) (see page 475)
  - [Context properties for the messaging service](#) (see page 477)
  - [Schedule the maintenance job](#) (see page 477)
- [Start the DAM hotfolder](#) (see page 479)

- [Throughput](#) (see page 480)
- [Limitations](#) (see page 480)
- [Verification](#) (see page 481)
  - [AWS](#) (see page 481)
  - [Azure](#) (see page 482)

### 2.5.3.2 Prerequisites

The DAM hotfolder also supports the auto assignment functionality over the message queue communication. Therefore the messaging service of ActiveVOS must be configured. Please reach out to the Installation and Operation guide for the following areas:

- Installation of ActiveVOS with ActiveMQ support and configuration of queues in ActiveVOS (*Note: ActiveVOS version 9.2.4.6 is a prerequisite for queue based interactions between Product 360 and ActiveVOS*)

### 2.5.3.3 Installation

Before you can use the DAM hotfolder you have to deploy these resources:

1. InfaResources.bpr
2. DAM\_Hotfolder.bpr

with the BPM (ActiveVOS) console.

Also it is possible that you deploy the workflows with your BPM designer. The resources of the projects are in the zip files:

1. InfaResources.zip
2. DAM\_Hotfolder.zip

### 2.5.3.4 Configuration

The base configuration of the DAM hotfolder is defined in the BPM URN mappings. Therefore you have to define the following mappings in the BPM console.

These settings will be done from Informatica OPS team when the customer request it.

URN mappings for AWS

| URN                   | URL             | Mand. | Description  |
|-----------------------|-----------------|-------|--|
| urn:p360.api.username | RestServiceUser | Yes   | Name of the Product 360 user in which context the requests are fired |

| URN                              | URL   | Ma<br>nd. | Description  |
|----------------------------------|---|-----------|--|
| urn:p360.api.password            | *****   | Yes       | Password of the user above   |
| urn:p360.rest.url                | https://<br>p360server.p360.internal:1543                             | Yes       | Url of the Product 360 rest services   |
| urn:p360.aws.s3.bucket.name      | pim-example-bucket  | Yes       | S3 bucket name of the customer's environment   |
| urn:p360.dam.aws.sqs.rest.url    | https://sqs.us-west-2.amazonaws.com/1234/<br>pim-example-qa-sqs_queue | Yes       | Url to the SQS queue, where new objects will be notified   |
| urn:p360.dam.default.category.id | 00010000000 .....<br>00000000000000000000                             | Yes       | Identifier of the category where new assets will be automatically assigned to (must be 120 characters!)    |
| urn:p360.dam.hotfolder.path      | /data/opt/informatica/shares/<br>efs/P360Server/daminbox/             | Yes       | Path where the files will be copied from the bucket before creating the asset within Product 360           |
| urn:p360.bpm.rest.url            | https://<br>p360bpm.p360.internal:443/<br>active-bpel/services/REST   | Yes       | Url to the ActiveVOS cluster   |
| urn:p360.dam.s3.error.path       | /dev/P360BPM/hotfolder/Error/   | Yes       | Path to the folder of the S3 bucket where files will be copied when the creation process run into an error |

| URN                            | URL   | Ma<br>nd. | Description   |
|--------------------------------|---|-----------|---|
| urn:p360.dam.s3.processed.path | /dev/P360BPM/hotfolder/<br>Archived/                    | Yes       | Path to the folder of the S3 bucket where files will be copied when the creation process was successful. It is recommended to leave this entry empty or remove the key, because otherwise a lot of space on the blob storage can be allocated, which can create additional costs)             |
| urn:p360.dam.upload.path       | /data/opt/informatica/shares/<br>efs/P360Server/upload/ | Yes       | Path to the folder where the assets will be uploaded during the "/manage/file" Rest call  |
| urn:p360.dam.max.threads       | 2   | Yes       | Maximum number of parallel ActiveVOS threads while creating new assets. Value has to be between 1 and 10.   |
| urn:p360.dam.filename.pattern  | .+\. (tif jpg pdf)                                      | Yes       | Regular expression for allowed filenames. (*. * ==> all files are supported)  |
| urn:p360.dam.assigndoc.mode    | 0   | Yes       | <p>0 = No AssignDocMessage<br/>1 = AssignDocMessage (only for Master Catalog)</p> <p>Note: only together with the corresponding settings in hmm.properties of P360 server, the auto assignment can be correctly triggered by a message whose sending mode is defined with this parameter.</p> |

| URN  | URL  | Ma<br>nd. | Description  |
|--|------|-----------|--|
| urn:p360.dam.hotfolder.list<br>ener.max.attempts | 7200 | No        | Number of processed events<br>until the process restarts<br>automatically. [Default value<br>7200] |

## URN mappings for Azure

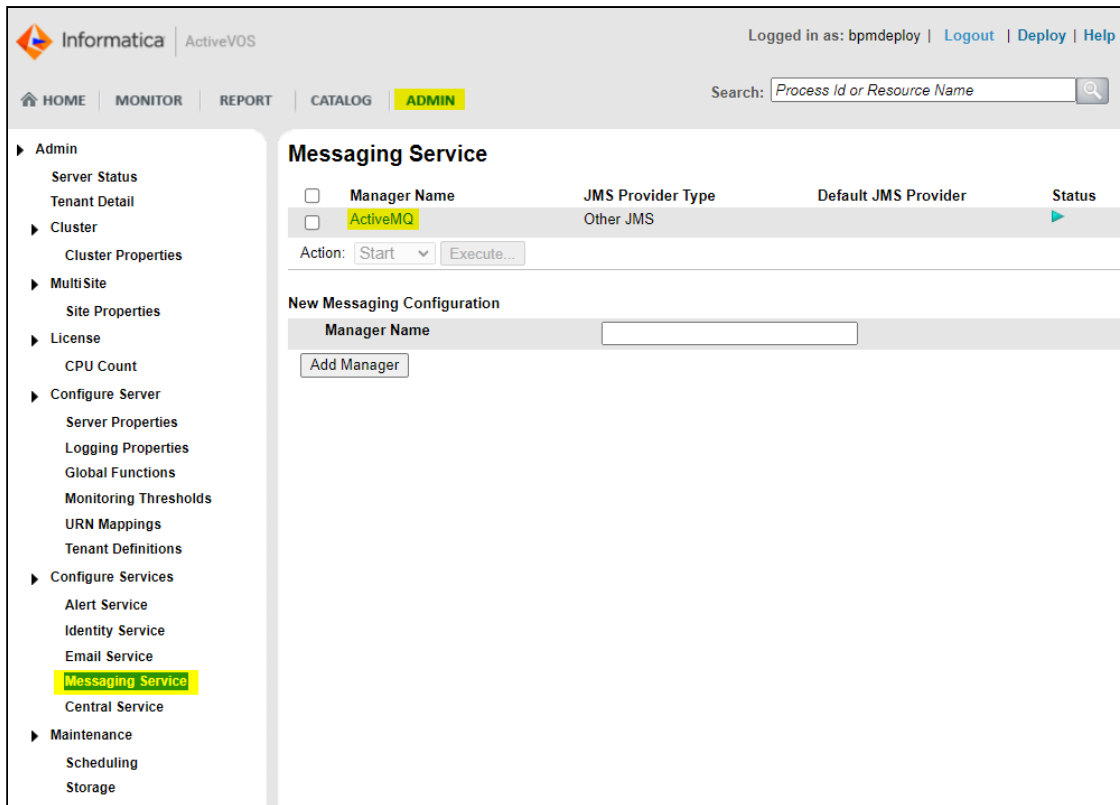
| URN                                      | URL  | M<br>a<br>n<br>d<br>. | Description   |
|--|--|-----------------------|---|
| urn:p360.api.<br>username                | RestServiceUser  | Y<br>e<br>s           | Name of the Product 360 user in which context<br>the requests are fired   |
| urn:p360.api.<br>password                | *****  | Y<br>e<br>s           | Password of the user above  |
| urn:p360.rest.<br>url                    | https://<br>p360server.p360.internal:1543  | Y<br>e<br>s           | Url of the Product 360 rest services  |
| urn:p360.dam<br>.default.categ<br>ory.id | 00010000000 .....<br>00000000000000000000  | Y<br>e<br>s           | Identifier of the category where new assets will<br>be automatically assigned to (must be 120<br>characters!)       |
| urn:p360.dam<br>.error.path              | /opt/informatica/shares/<br>S3_CUSTOMER/dev/<br>P360Server/assethotfolder/<br>Error/ | Y<br>e<br>s           | Path to the folder of the blob storage where files<br>will be copied when the creation process run<br>into an error |

| URN                                       | URL   | M<br>a<br>n<br>d<br>. | Description   |
|---|---|-----------------------|---|
| urn:p360.dam<br>.processed.p<br>ath       | /opt/informatica/shares/<br>S3_CUSTOMER/dev/<br>P360Server/assethotfolder/<br>Archived/ | Y<br>e<br>s           | Path to the folder of the blob storage where files will be copied when the creation process was successful. It is recommended to leave this entry empty or remove the key, because otherwise a lot of space on the blob storage can be allocated, which can create additional costs)  |
| urn:p360.dam<br>.upload.path              | /data/opt/informatica/shares/<br>efs/P360Server/upload/                                 | Y<br>e<br>s           | Path to the folder where the assets will be uploaded during the "/manage/file" Rest call  |
| urn:p360.dam<br>.max.threads              | 2   | Y<br>e<br>s           | Maximum number of parallel ActiveVOS threads while creating new assets. Value has to be between 1 and 10.   |
| urn:p360.dam<br>.filename.pat<br>tern     | .+\. (tif jpg pdf)  | Y<br>e<br>s           | Regular expression for allowed filenames. (. * ==> all files are supported)   |
| urn:p360.dam<br>.assigndoc.m<br>ode       | 0   | Y<br>e<br>s           | 0 = No AssignDocMessage<br>1 = AssignDocMessage (only for Master Catalog)<br><br>Note: only together with the corresponding settings in hmm.properties of P360 server, the auto assignment can be correctly triggered by a message whose sending mode is defined with this parameter. |
| urn:p360.dam<br>.update.old.d<br>ocuments | 0   | Y<br>e<br>s           | Deactivated! This is an experimental setting. It is not supported with version 10.5   |



Context properties for the messaging service

These 3 required context properties in the messaging service ActiveMQ have to be created. You can navigate to the service in the ActiveVOS console.



| Name                           | Value                                  |
|--------------------------------|--|
| queue.JNDI_P360_DAM_ASSIGNMENT | heiler.hmm.backend.event.assignment    |
| queue.JNDI_P360_DAM            | heiler.hmm.backend.event               |
| queue.JNDI_P360_DAM_MODIFIED   | heiler.hmm.backend.event.assetModified |

Schedule the maintenance job

The DAM hotfolder is implemented as an endless loop. The workflow has to be started one time via creating a scheduled maintenance job via the ActiveVOS console.

The screenshot displays the Informatica ActiveVOS Scheduling interface. The left sidebar contains the 'ADMIN' menu with 'Scheduling' highlighted. The main area shows the 'Execution Schedule' form with the following fields:

- Name:** Untitled Schedule
- Frequency:** Once
- Start Date on Server:** 2022-06-01 (yyyy-mm-dd)
- Start Time on Server:** 00:00:00 (hh:mm:ss Coordinated Universal Time)
- Run Options:**
  - ☒ Skip if running
  - ☐ Terminate if running
  - ☐ Create new process
- Service Name:** (Empty field with a link to 'Open Service Definitions')
- Input Document:** (Empty text area)

At the bottom of the form are 'OK' and 'Cancel' buttons.

## Job settings for AWS

| Key                  | Value           |
|----------------------|-----------------|
| Name                 | DAM Hotfolder   |
| Frequency            | Once            |
| Start Date on Server | Do not modify   |
| Start Time on Server | Do not modify   |
| Run Options          | Skip if running |
| Service Name         | Hotfolder_aws   |

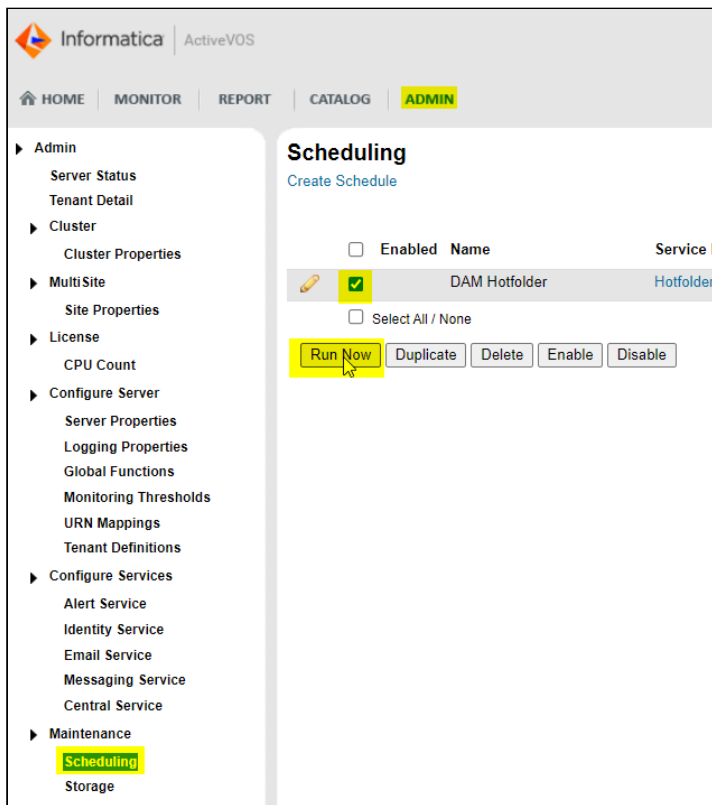
| Key            | Value   |
|----------------|---|
| Input Document | <pre>&lt;hot:startDAMHotfolderRequest xmlns:hot="http:// p360.dam/activevos/process/Hotfolder" xmlns:soap="http:// www.w3.org/2003/05/soap-envelope"&gt;   &lt;hot:hotfolderpath&gt;/data/opt/informatica/shares/efs/ P360Server/daminbox&lt;/hot:hotfolderpath&gt; &lt;/hot:startDAMHotfolderRequest&gt;</pre> |

#### Job settings for Azure

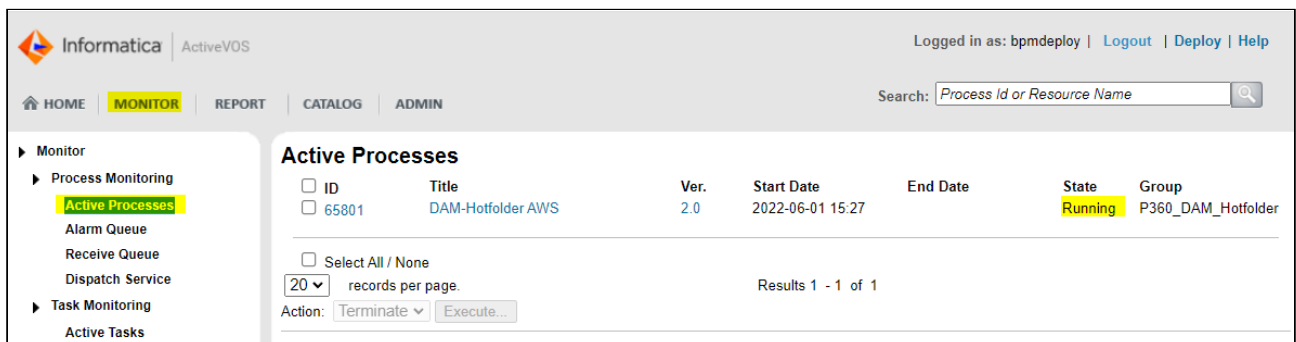
| Key                  | Value  |
|----------------------|--|
| Name                 | DAM Hotfolder  |
| Frequency            | Once   |
| Start Date on Server | Do not modify  |
| Start Time on Server | Do not modify  |
| Run Options          | Skip if running  |
| Service Name         | Hotfolder_azure  |
| Input Document       | <pre>&lt;hot:startDAMHotfolderRequest xmlns:hot="http://p360.dam/ activevos/process/Hotfolder" xmlns:soap="http:// www.w3.org/2003/05/soap-envelope"&gt;   &lt;hot:hotfolderpath&gt;/opt/informatica/shares/ S3_CUSTOMER/dev/P360Server/assethotfolder/&lt;/ hot:hotfolderpath&gt; &lt;/hot:startDAMHotfolderRequest&gt;</pre> |

#### 2.5.3.5 Start the DAM hotfolder

The DAM hotfolder can be started in the ActiveVOS console.



Now the process should run. You can check this as well in the ActiveVOS console.



### 2.5.3.6 Throughput

On AWS a throughput of about 40.000 files per 24h is observed. This has been measured with files of 3.3 MB average file size and ActiveVOS is running on a m5.large instance.

The throughput depends on the load of ActiveVOS. If other workflows requires resources, the file throughput will be lower.

### 2.5.3.7 Limitations

Upload of huge files can lead to an internal timeout while processing.

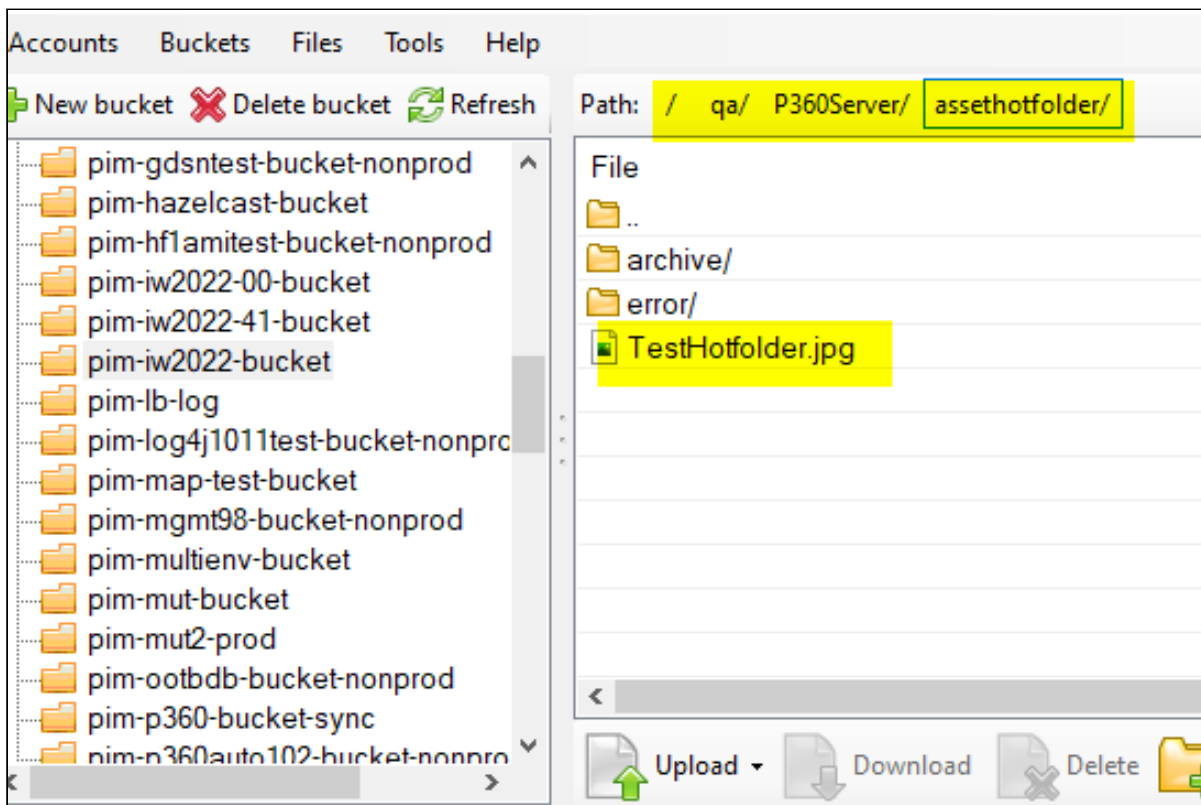
10 GB files are processed successfully.

### 2.5.3.8 Verification

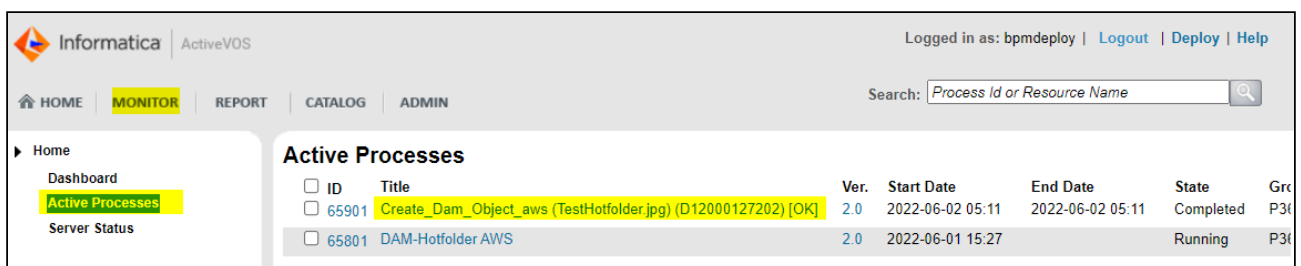
To verify whether the DAM hotfolder is working as expected you have to upload files into your blob storage.

AWS

Upload a file into your blob storage hotfolder. I.e. /qa/P360Server/assethotfolder/. (You can do this for example with the S3 Browser)



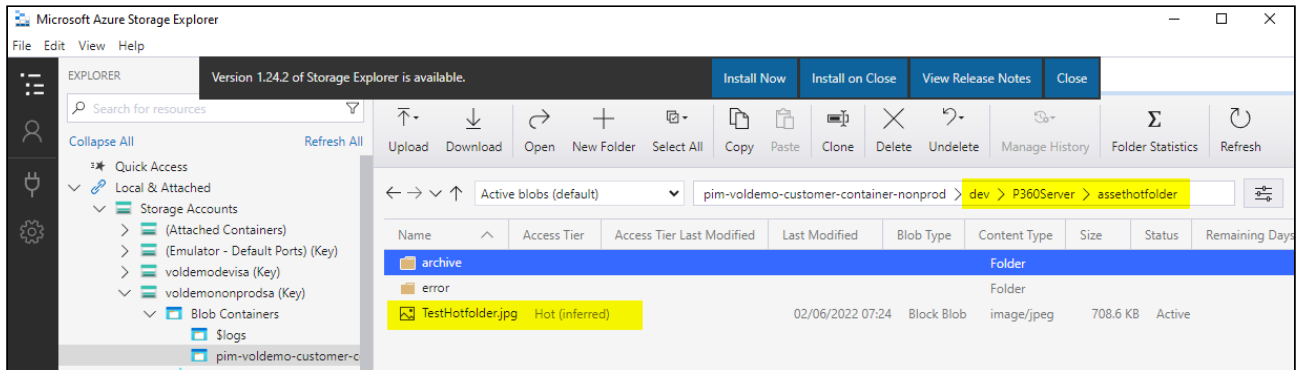
Check the processes in you ActiveVOS console



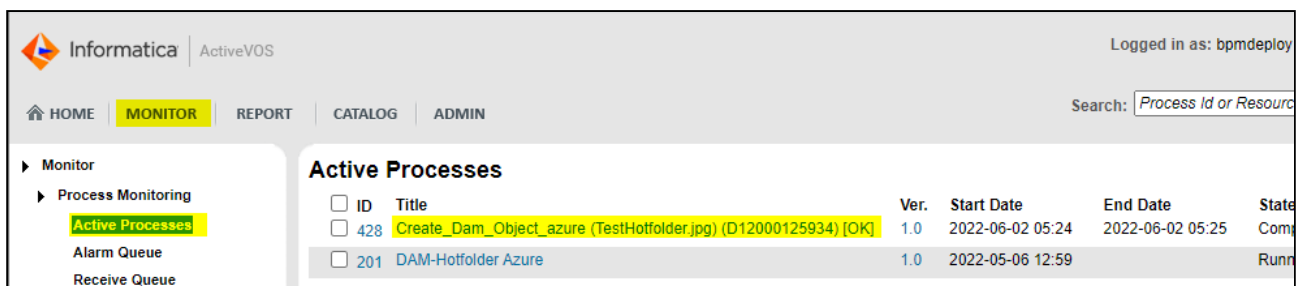
You can also check whether you see the new DAM asset in your Product 360 clients

Azure

Upload a file into your blob storage hotfolder. I.e. /qa/P360Server/assethotfolder/ (You can do this for example with the Microsoft Azure Storage Explorer)



Check the processes in you ActiveVOS console



You can also check whether you see the new DAM asset in your Product 360 clients

## 3 User Interface Templates

### 3.1 Content

This chapter describes how you can import the different standard User Interface Templates (UI Templates). The examples are located in the accelerator package in the sub folder "UI\_Templates".

### 3.2 Import

- Start the application "PIM Desktop client".
- Select the menu entry "Manage UI templates" in the menu "Management".
- Click on the button "Load UI templates from files..."
- Select a xml file and click ok.

After a successful import you can make also some changes for your system, for example which classification system will be used in the classification UI. For further information please read the configuration manual.

### 3.3 Approval UI

The Approval UI allows to quickly check the details of a list of objects in order to approve them in context of a specific work step. The object detail view on the right can be tailored to showcase the specific information needed for the approval of a given work step.

| Filename               | Location                  | Entity  |
|------------------------|---------------------------|---------|
| Item approve UI.xml    | ....\Examples\Flexible UI | item    |
| Product approve UI.xml | ....\Examples\Flexible UI | product |
| Variant approve UI.xml | ....\Examples\Flexible UI | variant |

Task: Approve item commercial data (59 Item(s) - Accepted)

Items (59)

| Image | Item no.                 | Status        | Short description                         |
|-------|--------------------------|---------------|---|
|       | Article_2918766499071457 | 01 New        | Remoty Remote Control                     |
|       | Article_2918766499071461 | 01 New        | DIGIT ISIO S1, black                      |
|       | Article_2918766499071464 | 01 New        | TechniControl, black                      |
|       | Article_2918766499071467 | 01 New        | DigiCorder HD K2, 160 GB, black           |
|       | Article_2918766499071470 | 01 New        | TechniStar S2+, black with HD+ Smartcard  |
|       | Article_2918766499071473 | 01 New        | TechniBox HD VA                           |
|       | Article_2918766499071476 | 01 New        | Comfort remote control (HD-Vision), black |
|       | Article_2918766499071481 | 01 New        | DIGIT ISIO S1, silver                     |
|       | Article_2918766499071484 | 01 New        | TechniControl Plus, silver                |
|       | Article_2918766499071487 | 01 New        | HDT 4, bicoloured                         |
|       | Article_2918766499071490 | 03 Selling pr | TechniStar K1, black                      |
|       | Article_2918766499071493 | 01 New        | DIGIT ISIO S, black                       |
|       | Article_2918766499071496 | 01 New        | TechniBox K1, black                       |
|       | Article_2918766499071499 | 01 New        | DIGYBOXX HD C+, black                     |
|       | Article_2918766499071502 | 01 New        | DIGIT ISIO S, silver                      |
|       | Article_2918766499071505 | 01 New        | TechniControl Plus, silver (for Receiver) |
|       | Article_2918766499071508 | 01 New        | DigiCorder HD K2, 160 GB, silver          |
|       | Article_2918766499071511 | 01 New        | DIGIT ISIO C, black                       |
|       | Article_2918766499071514 | 01 New        | TechniBox HD VAC, silver                  |
|       | Article_2918766499071517 | 01 New        | DIGYBOXX HD CX, black                     |
|       | Article_2918766499071520 | 01 New        | Connection cable (HDMI)                   |
|       | Article_2918766499071525 | 01 New        | Remoty Plus, silver                       |
|       | Article_2918766499071528 | 01 New        | DigiCorder HD S3                          |
|       | Article_2918766499071531 | 01 New        | TechniBox K1, black                       |

Detail (Item)

Short description:

TechniStar K1, black

Long description:

Digital HDTV cable receiver (MPEG-2/MPEG-4) with integrated CONAX conditional access system, smartcard reader and Ci+

Item no.:

Article\_2918766499071490

GTIN:

00083250028171

Manufacturer:

TechniSat

Net customer price(\$):

199.99

Non-binding price recommendation(\$):

249.95

Gross list price(\$):

159.99

Net list price(\$):

199.96

Set Status:

03 Selling prices OK

01 New

02 Purchase prices OK

03 Selling prices OK

04 Commercial data OK

05 Initial classification OK

06 Long description OK

07 Attributes OK

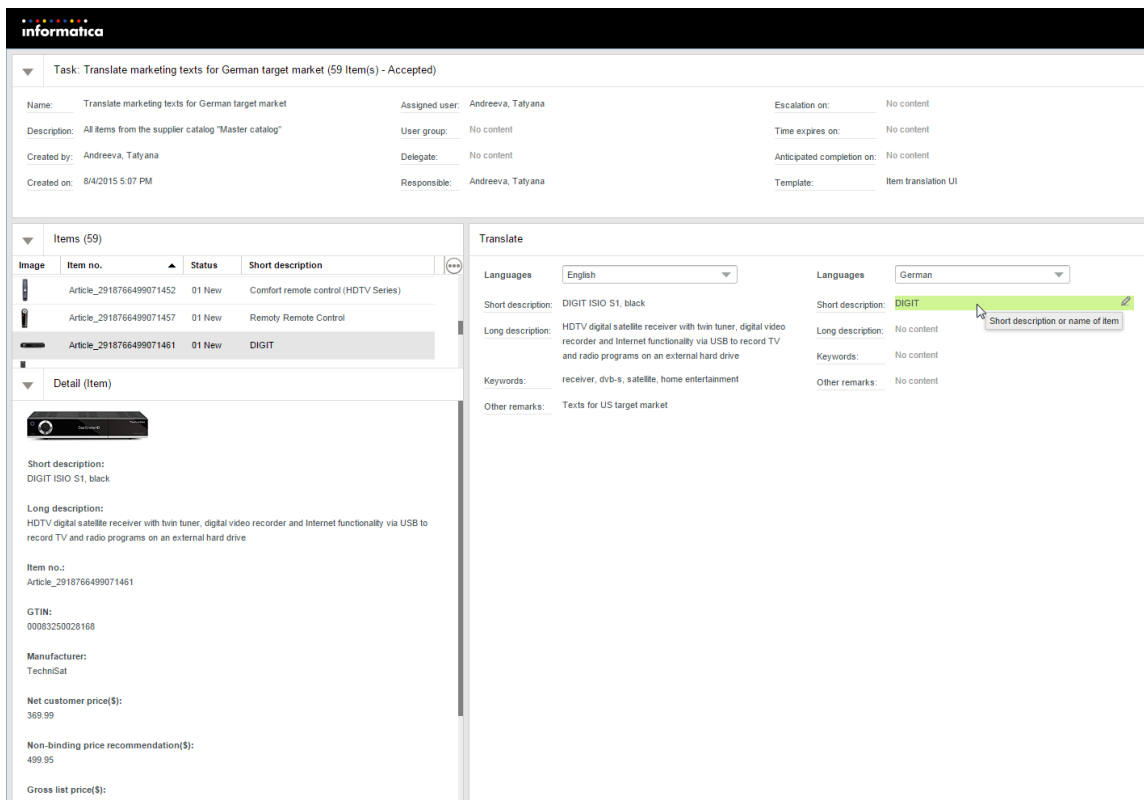
08 Internet image OK

09 Print image OK

## 3.4 Text Mastering UI

| Filename                           | Location                 | Entity         |
|------------------------------------|--------------------------|----------------|
| Item translation UI.xml            | ...\Examples\Flexible UI | item           |
| Product translation UI.xml         | ...\Examples\Flexible UI | product        |
| Variant translation UI.xml         | ...\Examples\Flexible UI | variant        |
| Product with Items translation.xml | ...\Examples\Flexible UI | product + item |

The text mastering UI allows for objects assigned to a task to select them and easily get an overview of all marketing text relevant information assigned. Furthermore it is possible to compare and edit text values for different languages or target markets directly within the details section.






**informatica**


Task: Translate marketing texts for German target market (59 Item(s) - Accepted)

Name: Translate marketing texts for German target market  
Assigned user: Andreeva, Tatyana  
Escalation on: No content  
Description: All items from the supplier catalog "Master catalog"  
User group: No content  
Time expires on: No content  
Created by: Andreeva, Tatyana  
Delegate: No content  
Anticipated completion on: No content  
Created on: 8/4/2015 5:07 PM  
Responsible: Andreeva, Tatyana  
Template: Item translation UI

**Items (59)**

| Image   | Item no.                 | Status | Short description                    |
|---|--------------------------|--------|--------------------------------------|
|  | Article_2918766499071452 | 01 New | Comfort remote control (HDTV Series) |
|  | Article_2918766499071457 | 01 New | Remoty Remote Control                |
|  | Article_2918766499071461 | 01 New | DIGIT                                |

**Detail (Item)**



Short description:  
DIGIT ISIO S1, black

Long description:  
HDTV digital satellite receiver with twin tuner, digital video recorder and Internet functionality via USB to record TV and radio programs on an external hard drive

Item no.:  
Article\_2918766499071461

GTIN:  
00083250028168

Manufacturer:  
TechniSat

Net customer price(\$):  
369.99

Non-binding price recommendation(\$):  
499.95

Gross list price(\$):  
599.99

**Translate**

Languages: English

Short description: DIGIT ISIO S1, black

Long description: HDTV digital satellite receiver with twin tuner, digital video recorder and Internet functionality via USB to record TV and radio programs on an external hard drive

Keywords: receiver, dvb-s, satellite, home entertainment

Other remarks: Texts for US target market

Languages: German

Short description: DIGIT

Long description: No content

Keywords: No content

Other remarks: No content

Short description or name of item



### 3.5 Media Assignment UI

The media assignment UI makes assigning media assets to objects of a task more easy than ever before. The template combines a list of objects with the document categories tree and allows the user to drag and drop media assets directly on each of the objects from the list.

| Filename                             | Location                  | Entity  |
|--------------------------------------|---------------------------|---------|
| Item multimedia assignment UI.xml    | ....\Examples\Flexible UI | item    |
| Product multimedia assignment UI.xml | ....\Examples\Flexible UI | product |
| Variant multimedia assignment UI.xml | ....\Examples\Flexible UI | variant |

The screenshot displays the Informatica MDM Media Assignment UI. At the top, a task bar shows 'Task: Assign media assets (59 Item(s))' with a checkmark. Below the task bar, a form contains details for the assignment, including 'Name: Assign media assets', 'Assigned user: Reinhardt, Stefan', 'Escalation on: 8/13/2015 5:58 PM', 'Description: All items from the supplier catalog "Master catalog"', 'User group: No content', 'Time expires on: 8/17/2015 5:58 PM', 'Created by: Andreeva, Tatyana', 'Delegate: Andreeva, Tatyana', 'Anticipated completion on: 8/16/2015 5:58 PM', 'Created on: 8/4/2015 5:07 PM', 'Responsible: Andreeva, Tatyana', and 'Template: Item Multimedia UI'. Below the form, there's a list of items with columns for 'Item no.', 'Status', and 'Short description'. One item, 'Article\_2918766499071378', is selected, showing its details including 'Short description: TechniBox S1, black', 'Long description: HDTV digital satellite receiver (MPEG-2/MPEG-4) with CI+', 'Item no.: Article\_2918766499071378', 'GTIN: 00083250028163', 'Manufacturer: TechniSat', 'Net customer price(\$): 149.99', 'Non-binding price recommendation(\$): 199.95', and 'Gross list price(\$): 119.99'. To the right, there's a 'Multimedia attachments' section with a tree view showing categories like 'bda\_system', 'Tools', 'Workshop Equipment', and 'Home Improvement categories'. Further right, a grid of product images is displayed, with one image highlighted.

### 3.6 Classification UI

The classification UI allows a very effective way of classifying product data. The user may select one or more objects assigned to the task to simply drag and drop them onto the structure group they should be assigned to.

| Filename                      | Location                 | Entity  |
|-------------------------------|--------------------------|---------|
| Item classification UI.xml    | ...\Examples\Flexible UI | item    |
| Product classification UI.xml | ...\Examples\Flexible UI | product |
| Variant classification UI.xml | ...\Examples\Flexible UI | variant |

**Task: Classify items to structure system (59 Item(s) - Accepted)**

Items (59)

| Image | Item no.                 | Status | Short description                               |
|-------|--------------------------|--------|---|
|       | Article_2018766499071317 | 01 New | Remote control suitable for digital receivers   |
|       | Article_2018766499071318 | 01 New | Remote control suitable for digital receivers   |
|       | Article_2018766499071326 | 01 New | Remoty Remote Control                           |
|       | Article_2018766499071330 | 01 New | RS 232 Update Cable                             |
|       | Article_2018766499071335 | 01 New | Remote control suitable for digital receivers   |
|       | Article_2018766499071340 | 01 New | External IR-receiver, for receiver with IR-port |
|       | Article_2018766499071346 | 01 New | TechniStar S1, black                            |
|       | Article_2018766499071349 | 01 New | DigiCorder HD S3                                |
|       | Article_2018766499071352 | 01 New | ISOControl Keyboard, black                      |
|       | Article_2018766499071355 | 01 New | Remote control suitable for digital receivers   |
|       | Article_2018766499071358 | 01 New | USB Ethernet adapter                            |
|       | Article_2018766499071364 | 01 New | SCART cable                                     |
|       | Article_2018766499071370 | 01 New | Comfort remote control (TechniLine Series)      |
|       | Article_2018766499071375 | 01 New | TechniBox S1+, black                            |
|       | Article_2018766499071378 | 01 New | TechniBox S1, black                             |
|       | Article_2018766499071381 | 01 New | Remote control suitable for digital receivers   |
|       | Article_2018766499071386 | 01 New | TechniControl, black                            |
|       | Article_2018766499071389 | 01 New | LCD panel cleaner                               |
|       | Article_2018766499071392 | 01 New | TechniSat TELTRONIC USB-WLAN                    |
|       | Article_2018766499071395 | 01 New | TechniStar S2, black                            |
|       | Article_2018766499071398 | 01 New | Remote control suitable for digital receivers   |
|       | Article_2018766499071403 | 01 New | HDT 4, black                                    |
|       | Article_2018766499071406 | 01 New | TechniStar IR, black                            |

Detail (Item)

Structure

- enter
- ECLASS-5.0.1
- Construction technology
- Electric engineering, automation, process control engineering
- Information, communication and media technology
- Entertainment electronics
  - Entertainment electronics (maintenance, inspection)
  - Entertainment electronics (other)
    - Entertainment electronics (other, unclassified)
    - Entertainment electronics furniture (other)
  - Entertainment electronics (parts)
  - Entertainment electronics (repair)
- Entertainment electronics furniture
- Machine, apparatus
- Machine, device (for special applications)
- Marketing
- Office products, facilities and technics, papeterie
- Service
- Tool

Classified in Structure (3)

| Image | Item no.                 | Short description                             |
|-------|--------------------------|---|
|       | Article_2018766499071317 | Remote control suitable for digital receivers |
|       | Article_2018766499071318 | Remote control suitable for digital receivers |
|       | Article_2018766499071326 | Remoty Remote Control                         |

Detail (Item)

### 3.7 Attribute Mastering UI

| Filename                  | Location                 | Entity  |
|---------------------------|--------------------------|---------|
| Item attributes UI.xml    | ...\Examples\Flexible UI | item    |
| Product attributes UI.xml | ...\Examples\Flexible UI | product |
| Variant attributes UI.xml | ...\Examples\Flexible UI | variant |

The attribute mastering UI focuses on attribute values of objects attached to a task. By selecting an object the detail view gives an overview of all its attributes and their values so that a user can easily approve or update them.

**Task:** Complete missing attribute values (59 Item(s) - Accepted)

**Items (59)**

| Image | Item no.                 | Status | Short description                               |
|-------|--------------------------|--------|---|
|       | Article_2918766499071340 | 01 New | External IR-receiver, for receiver with IR-port |
|       | Article_2918766499071346 | 01 New | TechniStar S1, black                            |
|       | Article_2918766499071349 | 01 New | DigitCorder HD S3                               |
|       | Article_2918766499071352 | 01 New | ISIOControl Keyboard, black                     |

**Detail (Item)**

Short description:  
TechniStar S1, black

Long description:  
HDTV digital satellite receiver (MPEG-2/MPEG-4) with integrated CONAX decoding system, Smartcard reader and CI+

Item no.:  
Article\_2918766499071346

GTIN:  
00063250028161

Manufacturer:  
TechniSat

Net customer price(\$):  
189.99

Non-binding price recommendation(\$):  
249.95

Gross list price(\$):  
151.99

Net list price(\$):  
199.96

Colour:  
black

**Attributes**

Language: English

2 x Common interface: No content

3 year guarantee: No

5 year guarantee: Yes

Adjustable headphone connection: Yes

Automatic picture brightness adjustment: Yes

Brightness: 500

Contrast ratio: 3.000:1

DVB-T Tuner: 2

DVR - Digital video recorder: Yes

Dolby Digital: Yes

Dolby Digital+: No

Dual HDTV multituner: Yes

Editing function: No

Energy class: E

Favourites programme manager: 2000

Fireproofed housing: Yes

## 4 Web Search Examples

There are six templates provided with the package as examples. These templates can be loaded in the export module of Product 360 Desktop client and used for exporting data to Elasticsearch for indexing. These templates provide examples of each type of field that can be indexed in Elasticsearch.

### 4.1 Items-only.ext

This template can be used when user wants to create an index for items from any one catalog. Catalog can be selected while exporting.

### 4.2 Products-only.ext

This template can be used when user wants to create an index for products only of master catalog in 1PPD , 2PPD or 3PPD system.

### 4.3 Variant.only.ext

This template can be used when user wants to create an index for variant only of master catalog in 1PPD , 2PPD or 3PPD system.

### 4.4 All Supplier Catalogs.ext

This template can be used when user wants to create an index for items from any number of catalogs (suppliers or master). Any number of catalogs can be selected while exporting.

### 4.5 2PPD.ext

This template can be used when user wants to create an index in 2PPD system, this template provide support for new, updated, deleted and orphan records in 2PPD system, here orphan stands for records not part of hierarchy of 2PPD.

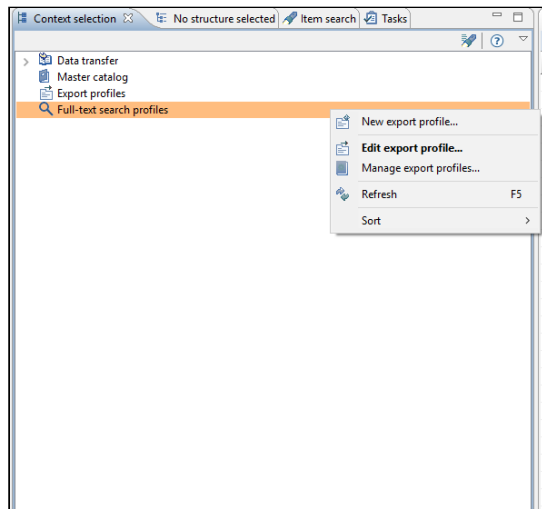
### 4.6 3PPD.ext

This template can be used when user wants to create an index for 3PPD system, this template provide support for new, updated, deleted and orphan records in 3PPD system, here orphan stands for records not part of hierarchy of 3PPD.

### 4.7 Steps to use it:

- Open Export Format Template perspective in Product 360 Desktop Client
- Select the option "Load format template from file.. "
- As per requirement, load the template into the system
- Save it on central storage

- Create a new Export profile as shown below from option "Full-text search profile"



- Now trigger the export or schedule it from the profile
- Search index should be created with the exported data

After the index creation is finished, following index will appear on web client as shown below in the Search view.

The screenshot shows the Informatica MDM web client interface in the Search view. The left sidebar contains navigation options: Dashboard, Structures, Catalogs, Media, Tasks, Queries, and Search. The main area displays a table of search results for 'Items-onl' (9999). The table has columns for Relevance, Item no., Status, GTIN, and Short description (English). The results are numbered 1 through 14, each with a 5-star relevance rating and a status of '01 New'. Below the table, there is a 'Detail view' section.

|    | Relevance | Item no.            | Status | GTIN | Short description (English)   |
|----|-----------|---------------------|--------|------|---|
| 1  | ★★★★★     | Article_POSTMAN3333 | 01 New |      | Fugiat iusto qui perspiciatis rerum voluptas aut quia voluptates rerum. |
| 2  | ★★★★★     | Article_POSTMAN3336 | 01 New |      | Dolorem nam temporibus qui sint.  |
| 3  | ★★★★★     | Article_POSTMAN3339 | 01 New |      | Omnis aliquid inventore earum ex fuga exercitationem.                   |
| 4  | ★★★★★     | Article_POSTMAN3342 | 01 New |      | Eum corrupti dolorem at consequatur.                                    |
| 5  | ★★★★★     | Article_POSTMAN3333 | 01 New |      | Quo sunt laborum earum autem consequatur vero quis.                     |
| 6  | ★★★★★     | Article_POSTMAN3345 | 01 New |      | Qui dolor animi veniam quas occaecati repudiandae doloribus sit et.     |
| 7  | ★★★★★     | Article_POSTMAN3348 | 01 New |      | Velit expedita cupiditate iusto enim quia dolorum est aut aliquam.      |
| 8  | ★★★★★     | Article_POSTMAN3351 | 01 New |      | Sunt velit aut reprehenderit iure eligendi laborum rerum adipisci hic.  |
| 9  | ★★★★★     | Article_POSTMAN3354 | 01 New |      | In nihil rerum veniam dolores sunt omnis.                               |
| 10 | ★★★★★     | Article_POSTMAN3357 | 01 New |      | Inventore officia et occaecati.   |
| 11 | ★★★★★     | Article_POSTMAN3360 | 01 New |      | Sequi qui vero alias.   |
| 12 | ★★★★★     | Article_POSTMAN3363 | 01 New |      | Quisquam eligendi saepe enim.   |
| 13 | ★★★★★     | Article_POSTMAN3366 | 01 New |      | Non vero quas.  |
| 14 | ★★★★★     | Article_POSTMAN3369 | 01 New |      | Eius omnis accusantium optio omnis in ratione officia vel.              |

## 5 CLAIRE Accelerator

### 5.1 CLAIRE Server Installation for ML Model Driven Use Cases

#### 5.1.1 Hardware Requirements

- At least 4 CPU cores
- At least 10 GB of RAM, free to use by the CLAIRE recommendation service

#### 5.1.2 Windows Installation

To install the CLAIRE recommendation service, an installation script is provided. It's located in the CLAIRE accelerator package under *server.ai/installation*. It will automatically install Python and all the required machine learning libraries in the needed versions for the CLAIRE recommendation service.

All you need to do for this is to follow below steps:

- Open *File Explorer*
- Locate the installation folder with the script
- Run the script *installation-script-windows.bat* by right clicking the batch file and selecting the *Run as administrator* option (some of the commands require administrator privileges)

After you complete the steps, the batch will run each command in sequence displaying the results in the command window.

##### 5.1.2.1 Folder creation

In addition, you have to create 4 folders

- **c:/informatica/ai/data/product\_classification** here is where the data used for the product classification training process will be saved
- **c:/informatica/ai/model/product\_classification** here is where the product classification trained models used for the classification process will be stored
- **c:/informatica/ai/data/attribute\_extraction** here is where the data used for the brand extraction training process will be saved
- **c:/informatica/ai/model/attribute\_extraction** here is where the brand extraction trained models used for the brand extraction process will be stored

The folders can be placed in a different location, and you can configure the path within the *server.ai* configuration file (*config.ini*). See chapters [Configuration and Operation of CLAIRE Product Classification](#) (see page 495) under the section *Configuration of CLAIRE product classification service* and [Configuration and Operation of CLAIRE Brand Extraction](#) (see page 509) under the section *Configuration of CLAIRE brand extraction service* for details.

### 5.1.2.2 CLAIRE server start up

Finally, you can start the CLAIRE server. A start up script *Start-CLAIRE.bat* is located in the CLAIRE accelerator package under *server.ai*. Navigate there via your Command Prompt and start the script by typing in *Start-CLAIRE.bat*.

In a few seconds, you should see the server up and running as shown in the screenshot below.

```
2022-06-09 07:35:41 DEBUG Natural language processing models [] found
2022-06-09 07:35:41 DEBUG Deep learning models [] found
2022-06-09 07:35:44 DEBUG Load pre-trained extraction models
2022-06-09 07:35:44 DEBUG There is no pre-trained model to load.
2022-06-09 07:35:44 INFO Serving on 5000 ...
```

### 5.1.3 Linux Installation

To install the CLAIRE recommendation service on a Linux machine, an installation script *installation-script-linux.sh* is provided. It's located in the CLAIRE accelerator package under *server.ai/installation*. It will automatically install Python and all the required machine learning libraries in the needed versions for the CLAIRE recommendation service.

All you need to do for this is to follow below steps:

- Navigate via your shell to the *server.ai/installation* folder
- Run `chmod +x installation-script-linux.sh` to make the installation script executable
- Run `./installation-script-linux.sh`

After you complete the steps, the script will run each command in sequence displaying the results in the shell window.

#### 5.1.3.1 Folder creation

You have to create 4 folders

- **/tmp/informatica/ai/data/product\_classification** here is where the data used for the product classification training process will be saved
- **/tmp/informatica/ai/model/product\_classification** here is where the product classification trained models used for the classification process will be stored
- **/tmp/informatica/ai/data/attribute\_extraction** here is where the data used for the brand extraction training process will be saved
- **/tmp/informatica/ai/model/attribute\_extraction** here is where the brand extraction trained models used for the brand extraction process will be stored

The folders can be placed in a different location and you can configure the path within the *server.ai* configuration file (*config.ini*). See chapters [Configuration and Operation of CLAIRE Product Classification](#) (see page 495) under the section *Configuration of CLAIRE product classification service* and [Configuration and Operation of CLAIRE Brand Extraction](#) (see page 509) under the section *Configuration of CLAIRE brand extraction service* for details

### 5.1.3.2 CLAIRE server start up

Finally, you can start the CLAIRE server. A start-up script *Start-CLAIRE-linux.sh* is located in the CLAIRE accelerator package under *server.ai*. Navigate there via your shell and start the script.

In a few seconds, you should see the server up and running as shown in the screenshot below.

```
2022-06-09 09:46:10 DEBUG Natural language processing models [] found
2022-06-09 09:46:10 DEBUG Deep learning models [] found
2022-06-09 09:46:12 DEBUG Load pre-trained extraction models
2022-06-09 09:46:12 DEBUG There is no pre-trained model to load.
2022-06-09 09:46:12 INFO Serving on 5000 ...
```

### 5.1.4 Claire Server Health Check

To check whether the python server is up and running and the CLAIRE application can respond to HTTP requests, we provide a REST API endpoint that you can test against:

|                  |   |
|------------------|---|
| URL pattern      | /health   |
| Method           | GET   |
| Success response | <ul style="list-style-type: none"> <li>• Code: HTTP 200</li> <li>• Content: OK</li> </ul> |

Congrats, this should have set you up properly to work with the Informatica MDM - Product 360 CLAIRE recommendation service. Enjoy!

## 5.2 CLAIRE Services Setup

The configuration for product classification and brand extraction services is specified in the *config.ini* file in the CLAIRE accelerator package under the *server.ai* folder.

#### config.ini

```
[DEFAULT]
# Username and password for basic authentication for ai server. Please ensure that
# you configured Product360 CLAIRE accelerator accordingly.
user = admin
password = admin
```



```

# Folders path where to store the data and models
upload_folder = c:/informatica/ai/data/product_classification
model_path = c:/informatica/ai/model/product_classification

# Python logging level. Possible values are: NOTSET, DEBUG, INFO, WARNING, ERROR,
CRITICAL
logging_level = DEBUG

# Port number which the ai server uses
port_number = 5000

[ATTRIBUTE_EXTRACTION]
# path to data and models folders
data_path_extraction = c:/informatica/ai/data/attribute_extraction
model_path_extraction = c:/informatica/ai/model/attribute_extraction

# NER model
model_extraction_spacy = S1

```

The parameters are mandatory and described here:

| Parameter name | Description  | Valid values                           | Default  |
|----------------|--|--|--|
| user           | Username for basic authentication against the server                                       | <user_name>                            | admin  |
| password       | Password for basic authentication against the server                                       | <user_password>                        | admin  |
| upload_folder  | Path to the folder where data (.csv files) used for the training process will be saved     | any valid path to an existing folder   | c:/informatica/ai/data/product_classification  |
| model_path     | Path to the folder where trained models used for the classification process will be stored | any valid path to an existing folder   | c:/informatica/ai/model/product_classification |
| logging_level  | Python logging level   | CRITICAL, ERROR, WARNING, INFO, NOTSET | DEBUG  |

| Parameter name        | Description   | Valid values                         | Default                                      |
|-----------------------|---|--------------------------------------|--|
| port_number           | Port number used by the ai server for communication   | <user_name>                          | 5000   |
| data_path_extraction  | Path to the folder where data (.csv files) used for the training process will be saved  | any valid path to an existing folder | c:/informatica/ai/data/attribute_extraction  |
| model_path_extraction | Path to the folder where trained models used for the classification process will be stored  | any valid path to an existing folder | c:/informatica/ai/model/attribute_extraction |
| model_extraction_spy  | The model to be used in training. Model S1 is faster to train with lower efficiency while model S2 is slower to train with higher accuracy. | S1, S2                               | S1   |

### 5.2.1 Product360 configuration

In the file *claire.properties* which is located in the *conf* folder of the Product 360 server the connection to the CLAIRE server should be configured:

#### claire.properties

```
#####
###                                     ###
### Claire server settings             ###
### =====                         ###
### These settings contain all configurations needed for CLAIRE features    ###
### like auto-classification or translation.                                ###
###                                     ###
#####

#####
# General Claire server settings
#
# These settings describe the connection with the claire server
#
claire.server.url = http://localhost:5000
claire.server.user = admin
```

```

claure.server.password = admin

#####
# Classification
#
# Connection settings for claure server used for classification training and auto-
classification
# If these values are empty, the classification feature is considered to be inactive.
#
claure.classification.server.url = ${claure.server.url}
claure.classification.server.user = ${claure.server.user}
claure.classification.server.password = ${claure.server.password}

#####
# Brand Extraction
#
# Connection settings for claure server used for brand extraction training and
prediction
# If these values are empty, the brand extraction feature is considered to be
inactive.
#
claure.brand.extraction.server.url = ${claure.server.url}
claure.brand.extraction.server.user = ${claure.server.user}
claure.brand.extraction.server.password = ${claure.server.password}

```

## 5.3 CLAIRE Product Classification

### 5.3.1 Configuration and Operation

#### 5.3.1.1 AI Training for Auto Classification

The classification of product data is supervised learning, and the first step is to train a model based on existing data. To generate a model the CLAIRE accelerator leverages the power of Product 360's export. To train a model, an export template has to be used in order to define the data for the training. It usually consists of the assignment to a structure group and one or more text values per product record and we ship examples to use. In addition, the export template must configure the post processing step "*Classification training*". This post processing step will send the created file to the CLAIRE recommendation service which will eventually train a model based on the data.



#### Supported content languages for model generation

Machine learning models allowing to support auto classification use cases can be sensitive to the exact language being used for their training. Every individual model needs to be trained on the same content language for all records used. We have extensively tested English and German during development and received very good results with it.

Besides these two here is the overall list of languages which we expect to work well with the accelerator:

- Dutch
- English
- Finnish
- French
- German
- Italian
- Norwegian
- Portuguese
- Spanish
- Swedish

**i** Note that any content language not listed above is likely to not work with the accelerator. Also note that we have not done full test cycles with each of the languages listed above.

#### Standard export templates for AI training

With the CLAIRE accelerator package 3 pre-built export templates are provided under *Resources/Export Templates* and can be used as is or as template for your own training exports.

- *AI Classification Training Items.ext*
- *AI Classification Training Products.ext*
- *AI Classification Training Variants.ext*

### Training data

Please ensure that there is enough training data to create your machine learning model. A training based on a few thousand records only might not bring best results. Also, the labels should be meaningful and more than just a short description like "Blue T-Shirt". Generally, the prediction results are heavily dependent on the quantity and quality of the data you feed the training with.

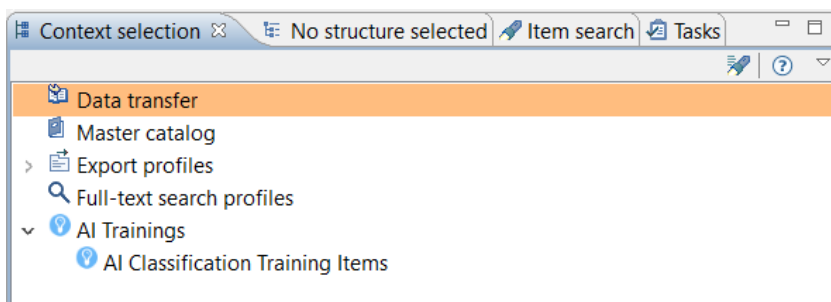
Create your own export templates for AI classification training

It is possible to create custom export templates for AI training, e.g., if the data you want to train on is in custom fields. Some preconditions to keep in mind are:

- It must be a csv file
- The purpose of the export template must be "AI Training"
- There must be a column having the header title "category". This column needs to contain the structure group assignment within the training structure.
- All label fields (can be multiple) must have the column header "label"
- The export template must have the post processing step "Classification training" attached

Create AI classification training export profiles

After creating an export template or importing the examples you can create AI classification trainings in the context selection view. Although the "AI Classification Trainings" are basically exports we separated them from the "regular" export profiles for a better user experience.



Also, in the process overview the AI trainings are separated from the "regular" export job executions for better process traceability



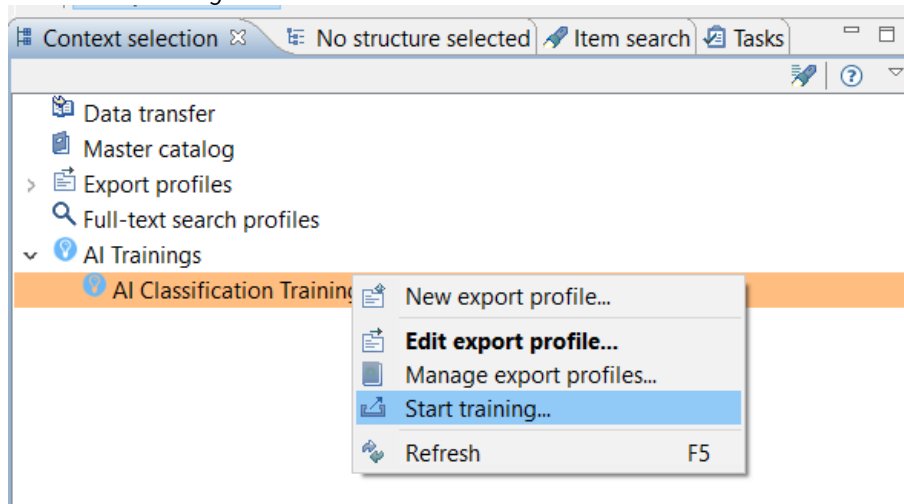
As we are using the export functionality of Product 360 you have the full power of configuring and scheduling export profiles to manage your AI trainings.

**i** Please note that we currently don't have any delta training. That means you have to retrain the model including exporting all of your data for every training.

Start AI classification training

To start an AI classification training, follow the steps bellow:

1. Right click on the export profile
2. Select *Start training...*



- The export parameter configuration dialog will open up. At this stage you set the export parameters for executing the training.

Export export profile - AI Training Items

**Export parameter configuration**

Set the export parameters for executing this export.

Data source "Item list"

Catalog:\* Master catalog

Assortment:

Update assortment: Yes

Version:\* Working version

Variables

Training ML approach:\* Deep learning

Training duration (only deep learning): 10

Training language:\* English

Training structure:\* FashionUnlimited

Update: before exporting

< Back Next > Finish Cancel

- Click on *Finish* and wait until the training has been successfully completed.

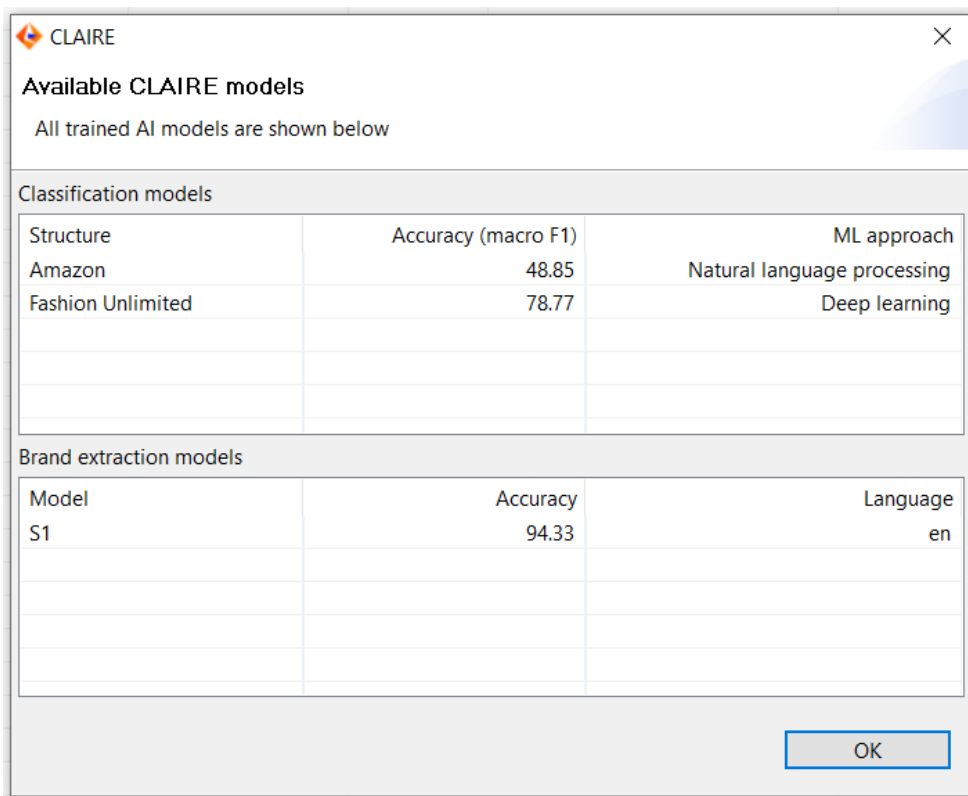
#### Export variables

| Variable name                          | Description  |
|--|--|
| Training ML approach                   | The ML approach used for the training process. Possible values are "Deep learning" or "Natural language processing".<br><br>See section <i>Natural language processing and deep learning explained</i> for details.  |
| Training duration (only deep learning) | The training duration (in hours) for deep learning that should be used to train the model. By default, it's set to 10 hours which should give reasonable results according to our tests. In case of natural language processing trainings this parameter won't be taken into consideration and the generation of a model will be a lot faster. |
| Training language                      | The language of text fields exported to train model.   |

| Variable name      | Description                                    |
|--------------------|--|
| Training structure | The structure the training should be based on. |

#### Display available models

There is an entry in the "Management" menu of the Desktop UI called "Show CLAIRE models" where you can display all trained classification models, the algorithm they are based on, and their accuracy based on macro F1 (see chapter [Best Practices and Recommendations](#) (see page 506) under section *Model accuracy measurement* for details).



#### Natural language processing and deep learning explained

##### Deep learning:

Convolutional neural network, specifically tailored for text classification tasks. Such models are based on decades of research into both AI and biology and are the technology of choice for many modern applications, including autonomous driving and speech recognition. Such networks consist of a large number of computational units that vaguely resemble some properties of neurons in human brain. Ordinarily, anyone who trains such models needs to know a lot of low-level details, to get the best performance. For this



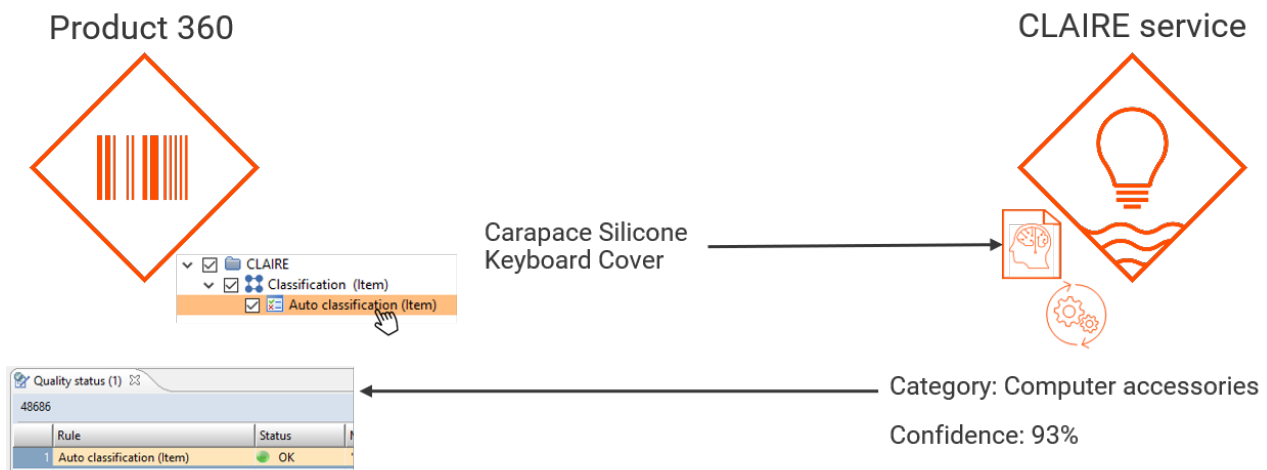
accelerator however everything is handled automatically "under the hood" for your convenience. Tests have shown that we need around 10h of training to get a deep learning model of good accuracy.

### Natural language processing:

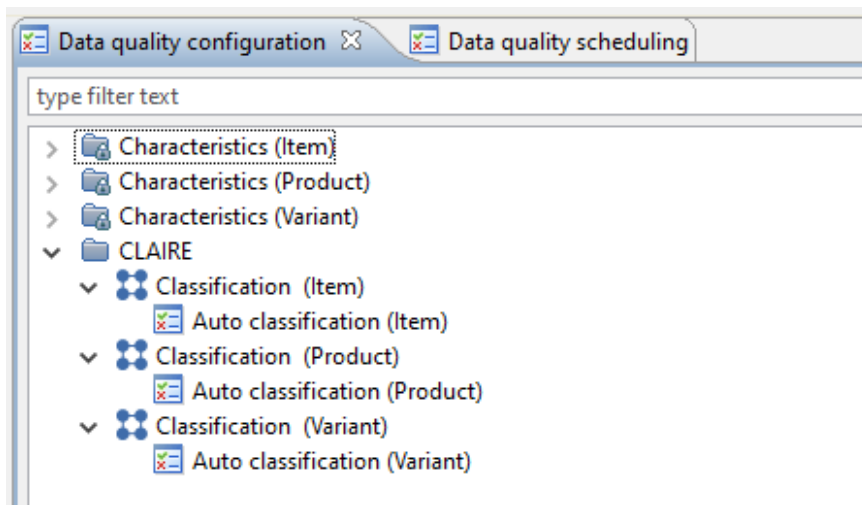
This approach produces a model rooted in classic techniques such as bag-of-words and "traditional" classifiers, such as a support vector machine or a random forest. These models are simpler, compared to deep learning, but train much faster (in our tests it took less than an hour). It might seem that such techniques are getting completely obsoleted by deep learning, but the reality is that they're still useful, although usually tend to produce models of lower accuracy.

#### 5.3.1.2 Batch Classification

In order to provide an auto classification in a batch process setup (for example after the import of data) the accelerator comes with the following capabilities:



After installation of the accelerator there is a new data quality category named "CLAIRE" which looks like this:



#### How to use the auto classification rule configuration

**i** It is highly recommended to **clone** one of the existing auto classification rule configurations and then modify the input ports according to your needs. The "default" rule configuration is just a template and should not be used as is. Even if you delete this rule configuration, category or group, it will come up on next server start again.

After the data quality rule configuration has been cloned and the parameter adjusted, it can be used for triggers or direct execution just as every other data quality rule configuration. The parameters are mandatory and described here:

| Parameter            | Description   |
|----------------------|---|
| Source Field         | The value which will be taken as input for the prediction.                                  |
| Threshold            | Only recommendations with a confidence equal or above this value (in %) will be considered. |
| Structure identifier | Only recommendations for this structure will be considered.                                 |
| Mapping mode         | Set "MOVE" to replace existing mapping(s) or "ADD" to add a mapping.                        |


| Parameter                   | Description  |
|-----------------------------|--|
| ML approach                 | Set "DL" for using the deep learning ML approach or "NLP" for using natural language processing. See section <i>Natural language processing and deep learning explained</i> for details. |
| Retain existing assignments | If true, objects which already have an assignment to a group of the defined structure system will be skipped.  |

It is possible to optionally configure a direct link within the data quality status detail tab of the Web UI which allows a user not only to spot a possible failed batch execution on item level but also to open a flex UI for fixing the classification problem right away. For that it is required to edit `DQNavigationDefinition.xml` which can be found in the `webdefinitions` folder of the Product 360 server. The following lines have to be added to it to link an erroneous status with a flex UI:

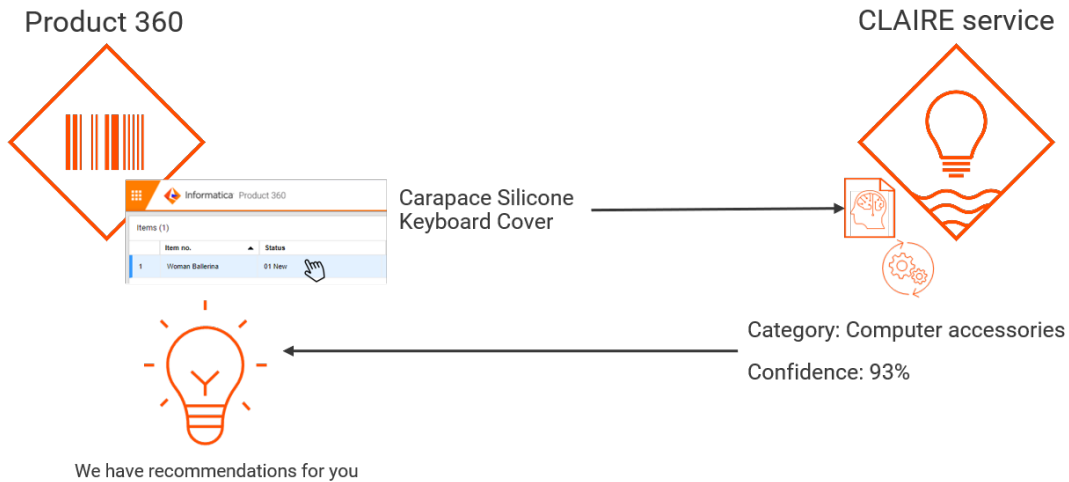
#### DQNavigationDefinition.xml

```
<!-- AutoClassification via CLAIRE -->
<ruleNavigation ruleName="Auto Classification Item" flexTemplateName=
"Classification with Claire" rootEntity="Article" />
```

Note that the `ruleName` property needs to be set to the actual rule name and the `flexTemplateName` to the name of the flex UI template you want to associate with the rule itself (e.g., a version of the Claire UI as seen below)

 For details on the expected performance of batch executions for auto classification please revise the chapter [Best Practices and Recommendations](#) (see page 506) under the section *Recommendations for best results*.

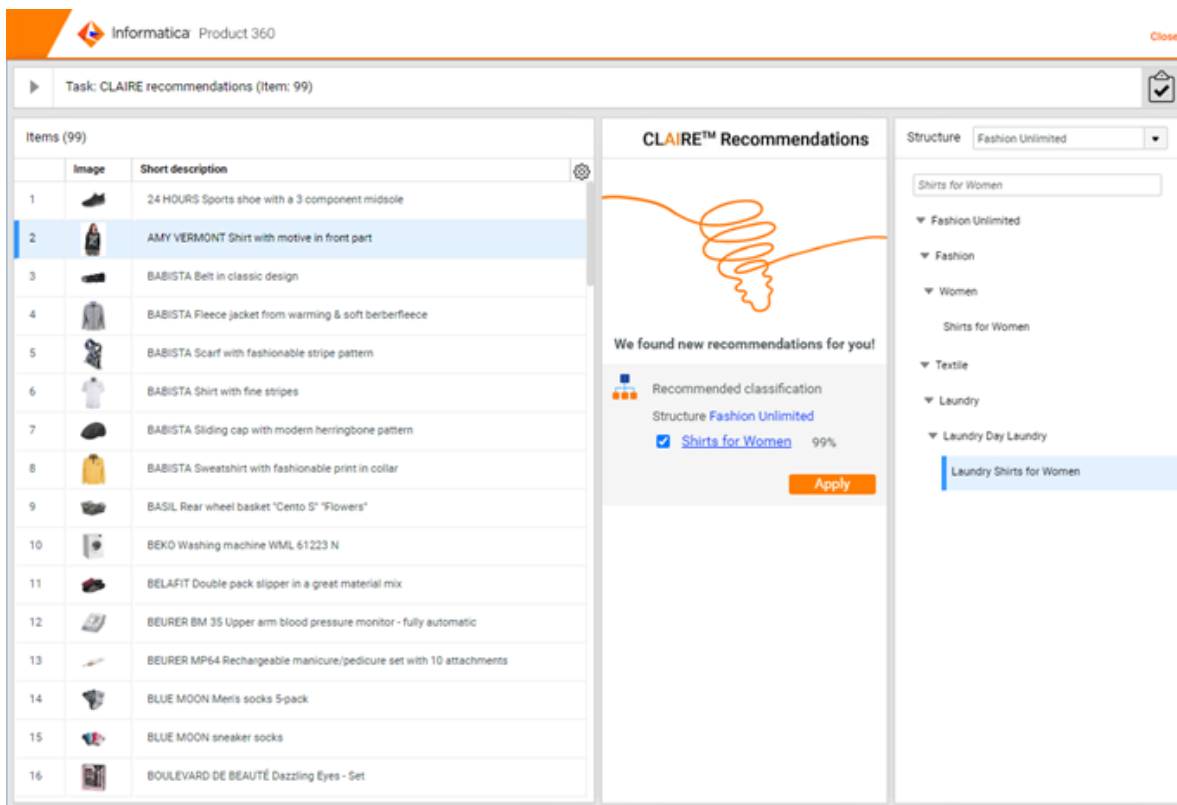
### 5.3.1.3 Configuration of CLAIRE Flex UI



The CLAIRE panel can be integrated into any flex UI with a component for classification. Below you see an example of how to configure a CLAIRE panel into your flex UI definition.

```
<group identifier="Claire info">
  <layoutData>
    <parameter key="colSpan" value="1" />
    <parameter key="rowSpan" value="7" />
  </layoutData>
  <component identifier="claire full" type="claire" i18NKey="Claire" >
    <layoutData>
      <parameter key="collapsible" value="true"/>
      <parameter key="collapsed" value="false"/>
    </layoutData>
    <parameter key="context" value="classification"/>
    <parameter key="sourceField" value="ArticleLang.DescriptionLong(en)"/>
    <parameter key="algorithm" value="deeplearning"/>
    <parameter key="threshold" value="80"/>
    <parameter key="selectionThreshold" value="80"/>
  </component>
</group>
```

Parameter name	Description	Valid values	Default
context	The context or use case the CLAIRE panel will be used for. Only recommendations for this use case will be shown. If empty, the server will attempt to load all available CLAIRE Panels, for example classification, translation and brand extraction.	classification	<empty>
sourceField	The source field which will be used for prediction. The field has to be fully qualified and formatted according to the REST API syntax.	ArticleLang. DescriptionLong(de)	ArticleLang. DescriptionLong(en)
algorithm	The algorithm used for classifications predictions which will be shown. If not configured all panels for classification will be shown. See section <i>Natural language processing and deep learning explained</i> for details.	deeplearning , nlp	<all>
threshold	All predictions of CLAIRE below or equal to this threshold will not be shown in the flex UI component	0-100	80
selectionThreshold	All predictions of CLAIRE above or equal to this threshold will be automatically selected in the CLAIRE flex UI component checkbox	0-100	80



## 5.3.2 Best Practices and Recommendations

### 5.3.2.1 Folder structure for model maintenance

All the models you train for natural language processing and deep learning are saved in the "model" folder (as defined in the configuration file). See chapter Configuration and Operation under the section *Configuration of CLAIRE recommendation service*.

- If you train a model using the NLP approach, the generated model is a file named after the structure system you trained for and with the extension ".sav". For example, if you train on the structure "FashionUnlimited", the generated model will be called "FashionUnlimited.sav".
- If you train a model using the DL approach, the generated model is a folder named after the structure system you trained for. For example, if you train on the structure "FashionUnlimited", the generated model is the folder "FashionUnlimited" within your model directory.

In addition to the generated models, a JSON file is also created for each initial training. This JSON file contains information about the trained model (structure, accuracy, ML approach) and is used to provide such insights to the user via the Desktop UI (see chapter Configuration and Operation under the section *Display available models*). The JSON file name has the following pattern:

`<Structure_System_name> + "_" + <ML_approach> + "_metrics.json" (i.e. FashionUnlimited_dl_metrics.json)`

In case a model has been trained on different environment it is possible to copy it into the "model" folder later on. In order to display it as available CLAIRE model on the Desktop UI, it is required to copy the corresponding generated JSON file as well. Say you train a model in a separate environment; you can copy the model file(s) as well as the JSON belonging to it into the "model" directory of your destination to make use of it properly.



If you want to delete a trained model from the "model" directory, delete its corresponding JSON file as well in order to avoid showing outdated information about a deleted model on the Desktop UI in the dialog "Available CLAIRE models".

### 5.3.2.2 Recommendations for best results

The accelerator offers two approaches to train models for product classification:

- **Natural language processing:** a pipeline based on classic algorithms from the NLP domain. Very fast to train, but in some cases, accuracy can be relatively low. These models can be quite big in size. The exact amount of space they need will depend on a particular dataset, but in our tests, some of them took as much as 30 GB. This might seem a bit excessive, but on the other hand, they should be more accurate compared to the models where the size is artificially inhibited.
- **Deep learning:** preferred approach, based on state-of-the-art research in deep learning. Such models are highly accurate, but are also time-consuming to train, especially, on big datasets.

See Chapter Configuration and Operation under the section *Natural language processing and deep learning explained* for details. Ordinarily, both require the user to be an expert in machine learning, but we kept all the details behind the scenes, to make the process as simple as possible. We would like to propose the following guidelines, to make sure you are getting the maximum out of this accelerator:

- **Keep the number of products in the training dataset to one million or below.** Our system and hardware requirements are tailored for datasets of this size. Also, the models we use do not need extremely big datasets to give high accuracy and going beyond one million is not likely to give a big boost.
- **For deep learning, set training time to around 10 hours.** In our tests, this timeframe resulted in models of good accuracy, and at the same time there was a probability of wasting too much time unnecessarily training much longer. Please keep in mind, that some additional time will be required for preprocessing the input data and for calculating performance metrics before the final model is available after training. For a dataset of a million records, it can take up to one hour, in addition to the time set for training. During training you can check the server window to see KPIs on the model and end the process earlier if you wish.
- **Avoid product categories that contain too few products.** The algorithms need a decent amount of data points in each category, in order to be successful. Before training even starts, we only consider categories that contain at least twenty products, the rest are discarded. It's being done because smaller categories will probably not be learned anyway, and at the same time they can harm performance on other categories.
- **Don't use inputs that are longer than 1,000 words.** All selected fields are merged into a single text input, and its size is truncated to 1,000 words. This is done to speed up training without losing accuracy.
- **Try to keep the number of product categories in 100s.** There is no hard limit on the number of product categories, but we find that a dataset of a million records can represent hundreds of categories quite well, but not thousands. Dealing with the number of classes in thousands and more usually requires much more advanced methods that are likely to require a lot of manual tuning by a data scientist, which is why we haven't further considered them in this accelerator.
- **For both natural language processing and deep learning, use small datasets when applying trained models in batch mode.** The actual speed would depend on configuration of the server, but internal tests on reasonable server hardware have shown an average performance of about 600 products/minute in our tests.

### 5.3.2.3 Model accuracy measurement

Before the training process starts, we split the input data into a training and a validation set (80/20 proportion). The model accuracy we report, is measured only on latter, to emulate how well the resulting model will perform on unseen data.

A good model performs well on two aspects:

1. If an item belongs to a certain category, the model assigns it to this category (also called *recall*).
2. If a model thinks an item belongs to a certain category, it does belong to it indeed (also called *precision*).

Both are very important. For instance, if a model just classifies anything as a watch, it will have high recall on watches, because all watches are being correctly identified. However, its precision will be low, as there will be many items from other categories incorrectly identified as watches.

This is why we use a measure called F1, which combines both recall and precision (this is a well-established and well-known metric used in many applications of machine learning). We calculate it for every product class in validation set separately, and then report the average of these scores (the technical term for that is *macro-F1*). It can take values between zero (bad) and one (good). Bigger values represent higher precision and recall across all product categories.

**A word of caution: it is not classification accuracy and can't be compared with it directly!**

Here are some guidelines on how to interpret F1 scores:

Value interval	Remarks
0.80-1.00	These values are quite rare, and usually can be achieved with a small number of classes only in the model (tens, not hundreds).
0.60-0.80	In our tests, these models gave good accuracy across product categories, but occasionally performed a bit less successfully on a small number of categories.
0.40-0.60	Perform well on the majority of product categories.
0.00-0.40	We wouldn't recommend using such models, as their performance can be unreliable. If you're seeing such scores, try reducing the number of classes and/or supplying more training data.



## 5.4 CLAIRE Brand Extraction

### 5.4.1 Configuration and Operation

#### 5.4.1.1 Brand Extraction Training

Brand name extraction is the task of identifying a brand of a product from a given text assuming the text contains the brand. For example, given a text "2009 Nissan rogue sl used gray silver suvs for sale lakeland fl", "Nissan" should be identified as the brand of the product (a SUV).

Brand name extraction is a supervised learning, and the first step is to train a model based on existing data. To generate a model the CLAIRE accelerator leverages the power of Product 360's export. To train a model, an export template has to be used in order to define the data for the training. The training data required to train the model basically consists of brand and text pairs. The text can be the product's title or description. The brand can be the manufacturer of the product. In your data there could be multiple text fields that contain brand information. If that is the case, we recommend choosing the field with the most number of product records and/or the field with shorter length of text values. For example, your training dataset contains 1000 product records, 900 of them have a long description and only 500 come with a short description. In this case, the best choice is to train over the long description field. But, if the difference between the number of records with long and short description is insignificant, we recommend choosing short description field as the text values are shorter (See the [Recommendation chapter](#) (see page 516) for more details).

In addition, the export template must configure the post processing step "*Brand extraction training*". This post processing step will send the created file to the CLAIRE brand extraction service which will eventually train a model based on the data.

Standard export templates for brand extraction training

With the CLAIRE accelerator package 3 pre-built export templates are provided under *Resources/Export Templates* and can be used as is or as template for your own brand extraction training exports.

- *AI Brand Extraction Training Items.ext*
- *AI Brand Extraction Training Products.ext*
- *AI Brand Extraction Training Variants.ext*

#### Training data

Please ensure that there is enough training data to create your machine learning model. A training based on a few thousand records only might not bring best results. Also, the labels should be meaningful and more than just a short description like "Blue T-Shirt". Generally, the prediction results are heavily dependent on the quantity and quality of the data you feed the training with.

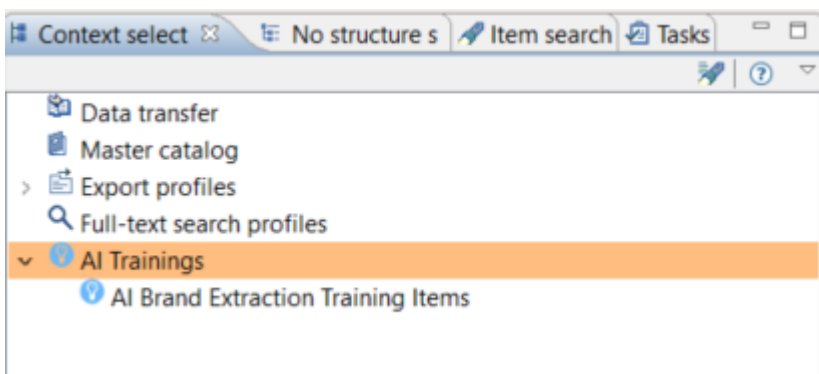
### Create your own export templates for brand extraction training

It is possible to create custom export templates for brand extraction training, e.g., if the data you want to train on is in custom fields. Some preconditions to keep in mind are:

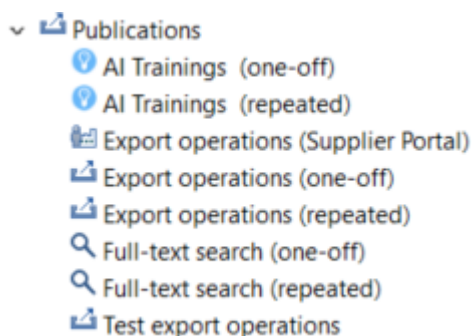
- It must be a csv file
- The purpose of the export template must be "AI Training"
- Column headers have to be "brand" and "title". The columns need to contain respectively the brand name and the textual data defined as label for the training. As mentioned above, there could be multiple text fields that contain brand information. If that is the case, we recommend choosing the field with the most data (most number of rows) and/or the field with shorter length of texts See our [Recommendation chapter](#) (see [page 516](#)) for more details).
- The export template must have the post processing step "Brand extraction training" attached

### Create AI brand extraction training export profiles

After creating an export template or importing the examples you can create AI brand extraction trainings in the context selection view. Although the "AI Brand Extraction Trainings" are basically exports we separated them from the "regular" export profiles for a better user experience.



Also, in the process overview the AI trainings are separated from the "regular" export job executions for better process traceability



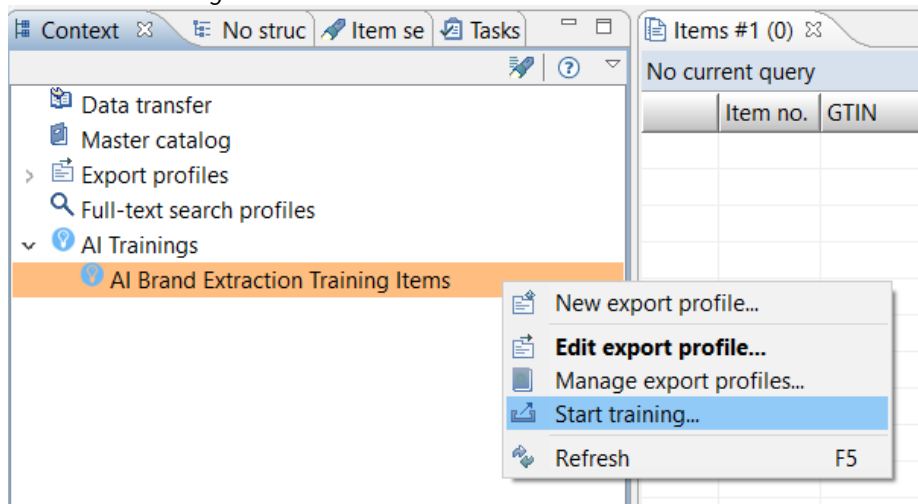
As we are using the export functionality of Product 360 you have the full power of configuring and scheduling export profiles to manage your AI trainings.

**i** Please note that we currently don't have any delta training. That means you have to retrain the model including exporting all of your data for every training.

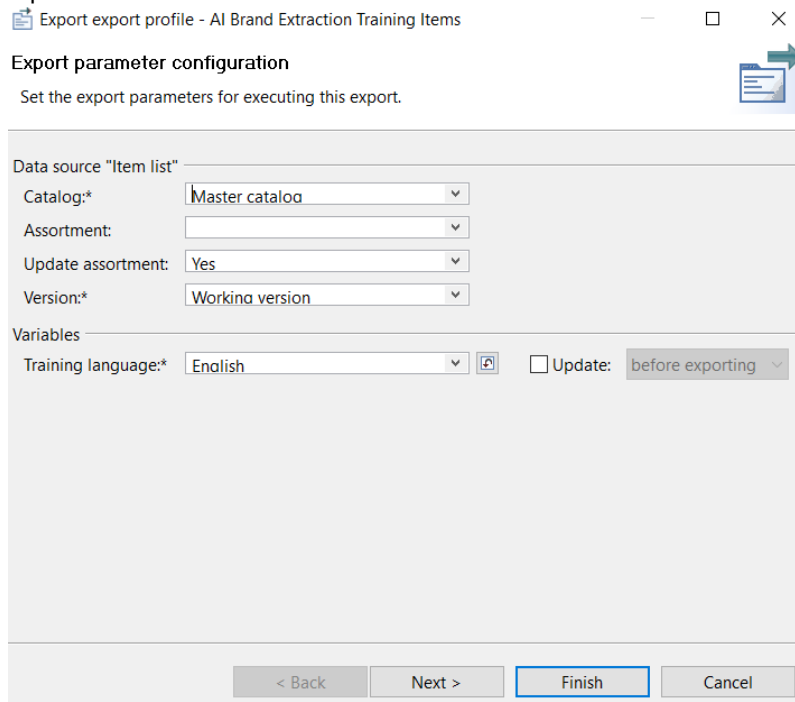
## Start AI brand extraction training

To start an AI Brand Extraction Training, follow the steps bellow:

1. Right click on the export profile
2. Select *Start training...*



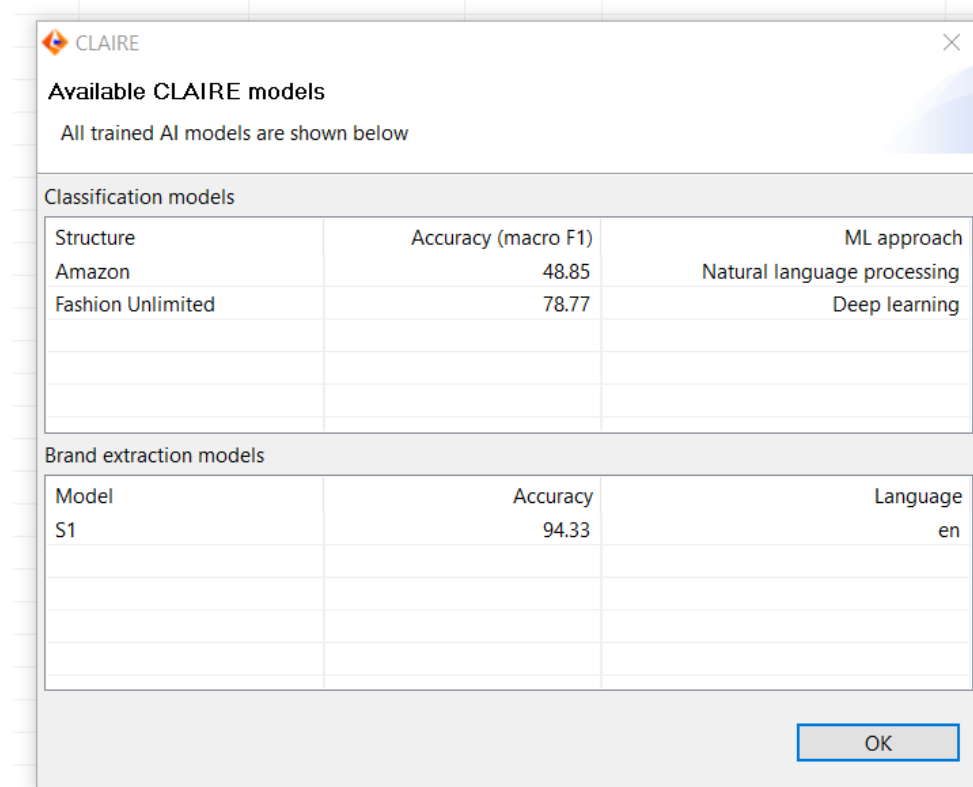
3. The export parameter configuration dialog will open up. At this stage you set the export parameters for executing the training including the training language variable which confirms the language of the text field exported to train the model.



4. Click on *Finish* and wait until the training is successfully completed.

#### Display available models

To check the accuracy of the brand extraction model, you can use the entry in the "Management" menu of the Desktop UI called "Show CLAIRE models". The table "Brand extraction models" displays all brand extraction trained models with their corresponding details.



#### 5.4.1.2 Configuration of Brand Extraction Flex UI

The CLAIRE panel can be integrated into any flex UI with a component for brand extraction. Below you see an example of how to configure a CLAIRE panel into your flex UI definition.

**Brand Extraction with Claire**

```

<group identifier="Claire info">
  <layoutData>
    <parameter key="colSpan" value="1"/>
    <parameter key="rowSpan" value="1"/>
  </layoutData>
  <component i18NKey="Claire" identifier="claire full" type="claire">
    <layoutData>
      <parameter key="collapsible" value="false"/>
    </layoutData>
  </component>
</group>

```

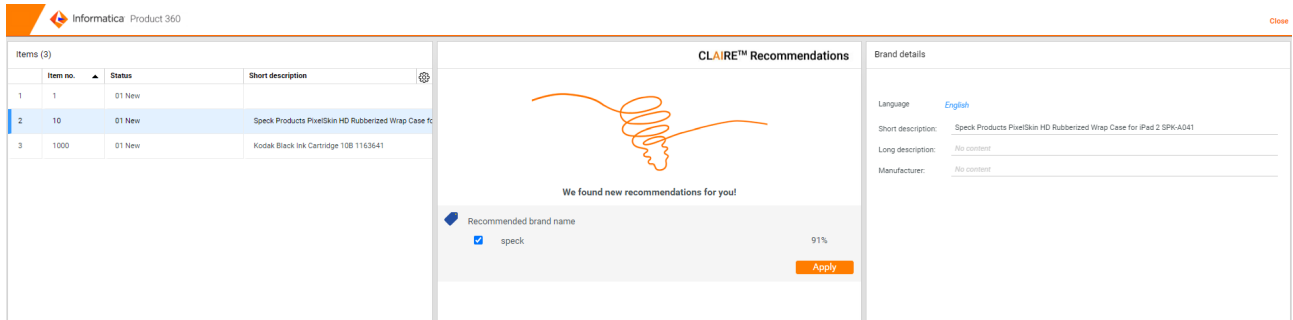
```

        <parameter key="collapsed" value="false"/>
    </layoutData>
    <parameter key="context" value="brandExtraction"/>
    <parameter key="brandExtractSourceField" value="ArticleLang.Descripti
onShort(en)"/>
    <parameter key="brandExtractTargetField" value="Article.ManufacturerN
ame"/>
    <parameter key="threshold" value="80"/>
    <parameter key="selectionThreshold" value="80"/>
    </component>
</group>

```

Parameter name	Description	Valid values examples	Default
context	The context or use case the CLAIRE panel will be used for. Only recommendations for this use case will be shown. If empty, the server will attempt to load all available CLAIRE Panels, for example classification, translation and brand extraction.	brandExtraction	<empty>
brandExtractSourceField	The source field which will be used for brand extraction. The field has to be fully qualified and formatted according to the REST API syntax.	ArticleLang.DescriptionShort(en)	<empty>
brandExtractTargetField	The target field to which the predicted brand name shall be written to.	Article.ManufacturerName	<empty>
threshold	All brand predictions with a confidence score below or equal to this threshold will not be shown in the flex UI component.	0-100	80
selectionThreshold	<p>All brand predictions with a confidence score above or equal to this threshold will be automatically selected in the CLAIRE flex UI component checkbox</p> <div>  In addition to the selectionThreshold, the check box for the predicted brand will only be automatically selected in the flex UI component if no value already exists for the brand name in the brandExtractTargetField to prevent unwanted overwriting. </div>	0-100	80

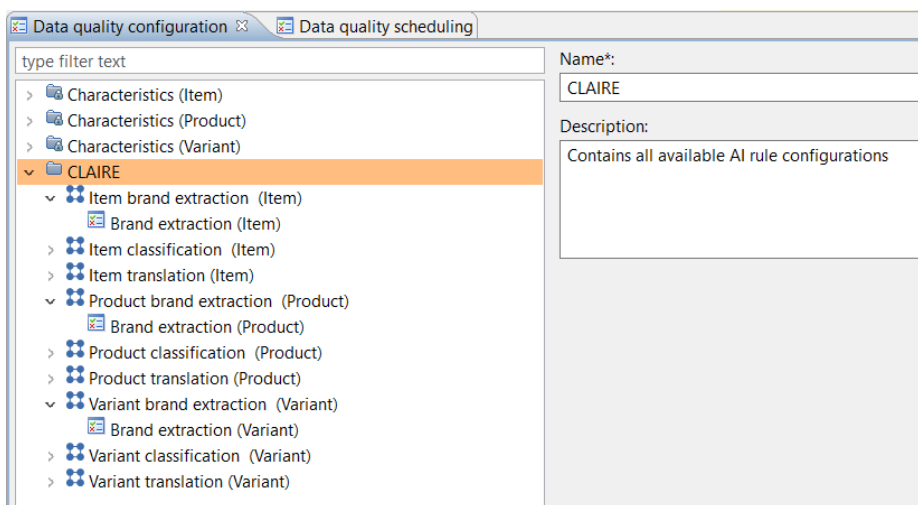
You can make the example flex UI template available in Product 360 Web by using the Desktop UI through *Management → Manage UI templates → Load UI templates from file*. The example file can be found under *CLAIRE\_Services → Resources → Flex UI Templates*. From this point on, you can open the brand extraction flex UI in the web UI by selecting one or multiple items and clicking on *Actions → Open Flex UI → Brand Extraction with Claire*.



### 5.4.1.3 Batch Execution of Brand Extraction

In order to provide the brand extraction in a batch process setup (for example after the import of data) the Claire accelerator comes with the following capabilities:

After installation and configuration of the accelerator there is a new data quality category named **CLAIRE** which also contains example rule configurations for brand extraction:



How to use the brand extraction rule configuration



It is highly recommended to **clone** one of the existing brand extraction rule configurations and then modify the input ports according to your needs. The "default" rule configuration is just a template and should not be used as is. Even if you delete this rule configuration, category or group, it will come up on next server start again.

After the data quality rule configuration has been cloned and the parameter adjusted, it can be used for triggers or direct execution to extract brands from language specific values of product records just as any other data quality rule configuration. The parameters are mandatory and described here:

Parameter	Description
Source Field	The brand will be extracted from a language specific field value. The selected field must be a string field qualified with a language.  <b>Note:</b> Only fully qualified fields are supported. Please don't select another data type for your rule configuration.
Target Field	The field, the extracted brand shall be written into. It must be a string field.
Threshold	Every prediction comes with a confidence score. In order to only add good predictions automatically to your data, you can adjust the minimum confidence score in percent.
Retain existing values	If true, already existing values in the target field will not be overwritten.

- ✓ In case no quality status entry is written for a product, variant or item after you triggered the execution of the rule, have a look into the process overview. There could be some reasons (like missing permissions) that might prevent the execution of a rule configuration for a product, variant or item.

The screenshot shows the 'Data quality configuration' window. On the left, a tree view shows the hierarchy: Characteristics (Item) > Characteristics (Product) > Characteristics (Variant) > CLAIRE > Item brand extraction (Item) > **Brand extraction (Item)**. The main panel displays the configuration for this rule.

**Name:** Brand extraction (Item) ☐ Hidden

**Description:**  
 This rule configuration is to automatically extract brand names from description fields.  
 Source field: The value which will be taken as input for the prediction.  
 Threshold: Only recommendations with a confidence equal or above this value (in %) will be considered.  
 Target field: The extracted brand name will be written to this field.  
 Retain existing values: If true, objects which already have a target field containing a value will be skipped.

**Rule name:** Brand Extraction

**Data type:** Item

Input port	Data type	Field
Source field		Short description (English)
Threshold		"80"
Target field		Manufacturer
Retain existing values		"true"

**Data type:** Item

Output port	Data type	Field



Our internal tests on batch executions of brand extraction based on one source and one target field using real customer data have shown the following performance results:

Number of product records	Duration
100	3 seconds
1.000	28 seconds
2.240	1 minute
10.000	4 minutes 6 seconds

## 5.4.2 Best Practices and Recommendations

### 5.4.2.1 Folder Structure for Model Maintenance

In each of the "data" and "model" folders which are configured in the *config.ini* file under the section "[ATTRIBUTE EXTRACTION]" in the *server.ai* folder, there is a folder named "attribute\_extraction" that is created to store related files.

When training is initiated, the following sub-folders are created:

- *data/attribute\_extraction/<model\_extraction>/training/<language>*
  - Example: *data/attribute\_extraction/S1/training/en*
- *model/attribute\_extraction/<model\_extraction>/training/<language>*
  - Example: *model/attribute\_extraction/S1/training/en*

where <model\_extraction> are either S1 or S2 (specified in the configuration), and <language> is the parameter specifying the language code sent to the extraction service. After the training is completed, a JSON file which contains information about the trained model (accuracy, algorithm, and language) is created inside the "model/attribute\_extraction" folder. The name of the JSON file has the following pattern:

**<model\_extraction>** + "\_" + **<language>** + "\_metrics.json" (e.g., S1\_en\_metrics.json)

After the training has been completed, the trained model is saved in the folder "model/attribute\_extraction/<model\_extraction>/<language>". For each extraction model (e.g., S1) and language (e.g., en) combination, there is only one JSON file. If you were to train a new model on a new dataset of the same combination, the newly trained model and JSON file are only saved if its accuracy is higher than that of the already existing trained model.



### 5.4.2.2 Recommendations for Best Results

For brand extraction, we have developed 2 solutions named S1 and S2. Model S1 (default model) is faster to train while model S2 is slower to train but often result in slightly higher accuracy. We have tested our solutions on various datasets and based on our experimental results, we recommend the followings:

- It is often better to use a larger training set to train models, especially when data is very heterogenous (e.g., brands can appear anywhere in the texts; the length of the texts vary widely). Therefore, we recommend using as much data as possible for training.
- If some products titles or descriptions do not carry brand names, include them in the training data as well. This is to teach the models to learn that the texts do not always have brands in them.
- The accuracy tends to be higher when the text from which the brand is extracted is shorter. Our solution is limited to handle texts of maximum 1,000,000 characters. Longer texts may cause memory allocation issues.
- We recommend using the S1 model for training as it is faster to train than S2. If the accuracy of the trained model is low (i.e., < 80), consider switching to S2 for better accuracy. Please note that the training time for S2 can be significantly longer than that of S1 (days vs hours). In one of our experiments, we trained an S1 model and an S2 model on a standard laptop CPU (MacBook Pro 2.3 GHz 8-Core Intel Core i9) using a dataset of 10K records and the average length of the texts was roughly 200 words. The training time for model S1 was about 3 hours while that for model S2 was almost 3 days. When we evaluated the trained models on a test dataset of 7K records, S2 gave about 4% improvement in accuracy compared to S1. The time it takes to train a model heavily depends on the size of the training data, the length of the texts from which the brands are extracted, the consistency of the data (e.g., brands tend to appear mostly at the beginning of the texts or vary), and the computing resources available.



We have extensively tested our solutions for English, German, Swedish and Finnish datasets, however, we also support other languages such as: Dutch, French, Italian, Norwegian, Portuguese, Spanish and Swedish.

## 5.5 CLAIRE Translation

### 5.5.1 Configuration and Operation

The CLAIRE translation service integration offers real-time translation flows to easily translate textual attributes and product descriptions into different target languages of your choice.

#### 5.5.1.1 Installation

Extract the Claire accelerator zip

Unpack the file *PIM\_<Version>\_Accelerators.zip* to a temporary folder on the Product 360 environment. The folder *PIM\_<Version>\_CLAIRE* contains the following artifacts:

*PIM\_<Version>\_Rev-xxxxx\_client\_ai.delta.zip*

contains the PIM Desktop UI features and plugins (not needed for the translation feature specifically)

*PIM\_<Version>\_Rev-xxxxx\_resources\_ai.delta.zip*

contains export templates, flex UI template examples and the ai server (not needed for the translation

feature specifically)

*PIM\_<Version>\_Rev-xxxxx\_server\_ai.delta.zip*

contains the feature and plugins to be put into the Product 360 server for translation API integration setups.

Installation of the Claire translation service

Unpack the *PIM\_<Version>\_Rev-xxxxx\_server\_ai.delta.zip*. The relevant plugins for the translation service are the following:

- *com.heiler.ppm.ai.server\_1.0.0.<Revision>.jar*
- *com.heiler.ppm.ai.translation.server.i18n\_1.0.0.<Revision>.jar*
- *com.heiler.ppm.ai.translation.server\_1.0.0.<Revision>.jar*
- *com.heiler.ppm.web.ai.translation\_1.0.0.<Revision>.jar*
- *com.heiler.ppm.web.ai\_1.0.0.<Revision>.jar*

We recommend to add all features into the folder *./server/features* and all plugins in the folder *./server/plugins* into your server environments. The translation service does not require any client plugins.

### 5.5.1.2 Configuration of Claire Translation Service

Option 1: Use the standard translation service configuration using Google Translate API

By default, and if no other translation service is contributed, the standard implementation uses Google Translate API.

When using the default translator, the setting used in the *claire.properties* file which is located in the *conf* folder of the Product 360 Server should be the following:

```
claire.translation.custom.used = false
```

Google Translate API setup

To use the Google Translate API, you must create an account and a project within *Google Cloud Platform*, enable billing and enable the Google Translate API for your Google Cloud project. Once this is done, you need to set up authentication and generate a Google Translate API key. The key will be used to configure Product 360 to run with Google Translate API.

The outlined setup steps are the following:

1. Create a *Google Cloud Platform (GCP)* account
2. Add your user account to your company's Google Cloud Project
3. Enable billing for your account
4. Enable the Cloud Translation API for your project
5. Generate an API Key to access the Google Translate API

More details about the setup process of Google Translate API can be found here <https://cloud.google.com/translate/docs/setup>.

## Google Translate API key usage

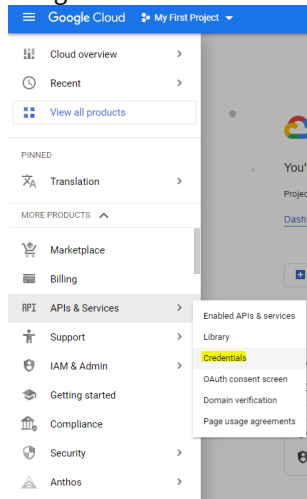
To access the Google Translate API from Product 360, a corresponding API key is needed. API keys are useful for accessing public data anonymously and are used to associate API requests with your project for quota and billing.

**i** Informatica MDM - Product 360 considers a "bring your own key" setup for the translation services providing full transparency and no extra charges from our end to make use of the capabilities.

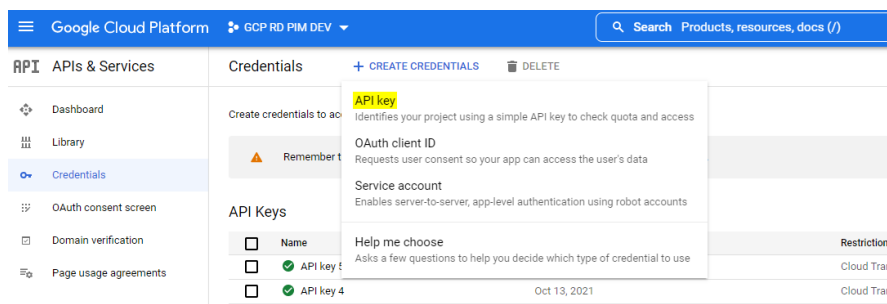
## Creating an API Key

You can use the Google Cloud Console to create and manage API keys. To create an API key:

1. Navigate to the *APIs & Services* → *Credentials* panel in the Cloud Console



2. At the top of the Credentials page, select *Create credentials*, then select *API key* from the dropdown menu



The *API key created* dialog box displays your newly created key.

**!** Always ensure that API keys are kept secure during both storage and transmission.

### Securing an API Key

API keys are unrestricted by default. Unrestricted keys can be used by anyone from anywhere. That's why it's highly recommended to set up application and API key restrictions. By adding restrictions, you can reduce the impact of a compromised API key.

Set application restrictions by restricting the API key to only work for calls from the IP address of the Product 360 server:

1. Navigate to the *APIs & Services* → *Credentials* panel in Cloud Console
2. Select the name of an existing API key
3. Under *Application restrictions*, select **IP addresses (web servers, cron jobs, etc.)** and enter the IP address of the Product 360 server

Set API restrictions and ensure that the API key can only be used for the Translation API.

1. In the *API restrictions* section, click **Restrict key**.
2. From the Dropdown menu, check *Cloud Translation API*
3. Click *Save*

For more details about Google Cloud API keys refer to <https://cloud.google.com/docs/authentication/api-key>

### Using an API Key

Within the Product 360 navigate to the *claire.properties* file and set the value of the generated API key like the following:

```
claire.translation.apiKey = [_to_encrypt_]<API key value>[_to_encrypt_]
```

We recommend to use the `[_to_encrypt_]` tag, to ensure that on the next restart of the Product 360 server, the API key will be encrypted and can no longer be read from the *claire.properties* file by others. [More information about the encryption process can be found under Installation and Operation → Installation → Server Installation](#)



When added to the *claire.properties* file, always keep the Claire Translation Service parameter name "*claire.translation.apiKey*". You may only want to remove or change the API key value.



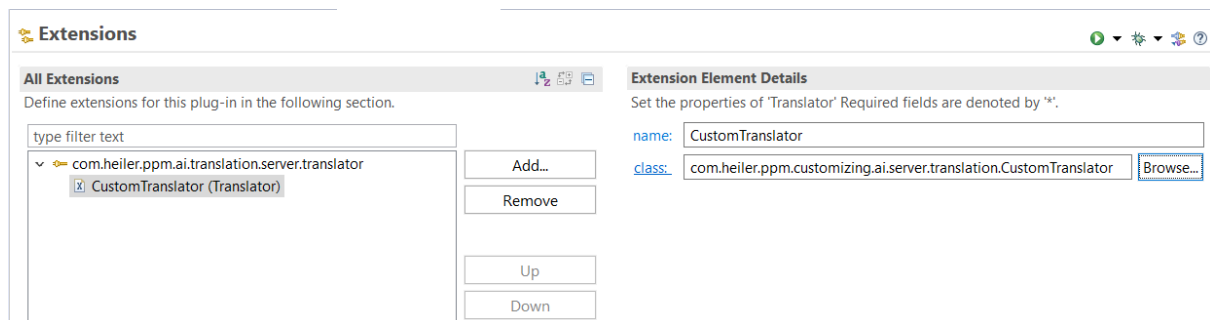
If *claire.translation.custom.used* is "false" and *claire.translation.apiKey* is empty, the translation feature is considered to be inactive.

Option 2: Implement and contribute your own custom translation service

In case you want to create and contribute your own custom translation service API integration, set the value of *claire.translation.custom.used* in the *claire.properties* file to "true" and follow the steps below. As an

addition to this documentation there is an example implementation in the SDK examples collection to get you started, the plugin name is *com.heiler.ppm.customizing.ai.server*.

1. Using the SDK of Product 360 create a new plugin, which has dependencies to *com.heiler.ppm.ai.server* and *com.heiler.ppm.ai.translation.server*.
2. Create a class that implements the interface *Translator.java* which is located in the package *com.heiler.ppm.ai.translation.server.service* according to your own translation service design.
3. If you need settings to configure your translation service, you can store them in the *claire.properties* file, key encryption is also supported as described above.  
Please have a look into the example implementation to learn more about the general approach and used objects.
4. Register your translator implementation using the extension point *com.heiler.ppm.ai.translation.server.translator*



After restarting the Product 360 Server, it will use the contributed translator. Confirmation that your translation service has been loaded successfully can be found in the console log.

```
17:39:30,041 INFO [ServerInitializer 0] [STARTUP] Initialize claire translation
server component
17:41:32,310 INFO [ServerInitializer 0] [TranslatorRegistry] Using contributed
translator for Claire Translation Panel: CustomTranslator
17:41:46,354 INFO [ServerInitializer 0] [STARTUP] Initialize claire translation
server component completed (Duration: 136313 ms)
```

### 5.5.1.3 Configuration of Translation Flex UI

The CLAIRE panel can be integrated into any Flex UI with a component for translation. Below you see an example of how to configure a CLAIRE panel into your flex UI definition.

#### Flex UI configuration for translation

```
<group identifier="Claire info">
  <layoutData>
    <parameter key="colSpan" value="1"/>
    <parameter key="rowSpan" value="1"/>
  </layoutData>
```

```

<component i18NKey="Claire" identifier="claire full" type="claire">
  <layoutData>
    <parameter key="collapsible" value="true"/>
    <parameter key="collapsed" value="false"/>
  </layoutData>
  <parameter key="context" value="translation"/>
  <parameter key="translationSourceFields" value="ArticleLang.DescriptionLong(en);ArticleLang.DescriptionShort(en);ArticleLang.Remarks(en);ArticleLang.Keyword(en)"/>
  <parameter key="translationTargetLanguages" value="es;fr;de"/>
</component>
</group>

```

Parameter name	Description	Valid values examples	Default
context	The context or use case the CLAIRE panel will be used for. Only recommendations for this use case will be shown. If empty, the server will attempt to load all available CLAIRE Panels, for example classification, translation and brand extraction.	translation	<empty>
translationSourceFields	The source field(s) which will be used for translation. The field(s) have to be fully qualified and language-specific fields of datatype String. It is possible to setup a single field or multiple for translation within the same flex UI. When using multiple fields, separate them with semicolons.	ArticleLang.DescriptionLong(en);ArticleLang.DescriptionShort(en);ArticleLang.Remarks(de);ArticleLang.Keyword(en)	<empty>

Parameter name	Description	Valid values examples	Default
translationTargetLanguages	<p>The language(s) each of the source fields shall be translated into. When using multiple target languages, separate them with semicolons. When adding a new language, make sure that the target language</p> <ol style="list-style-type: none"> <li>1. Is supported by google translate: <a href="https://cloud.google.com/translate/docs/languages">https://cloud.google.com/translate/docs/languages</a></li> <li>2. Corresponds to a key synonym of an entry in the Enum.Language enumeration in the repository</li> </ol>	es;fr;de	<empty>

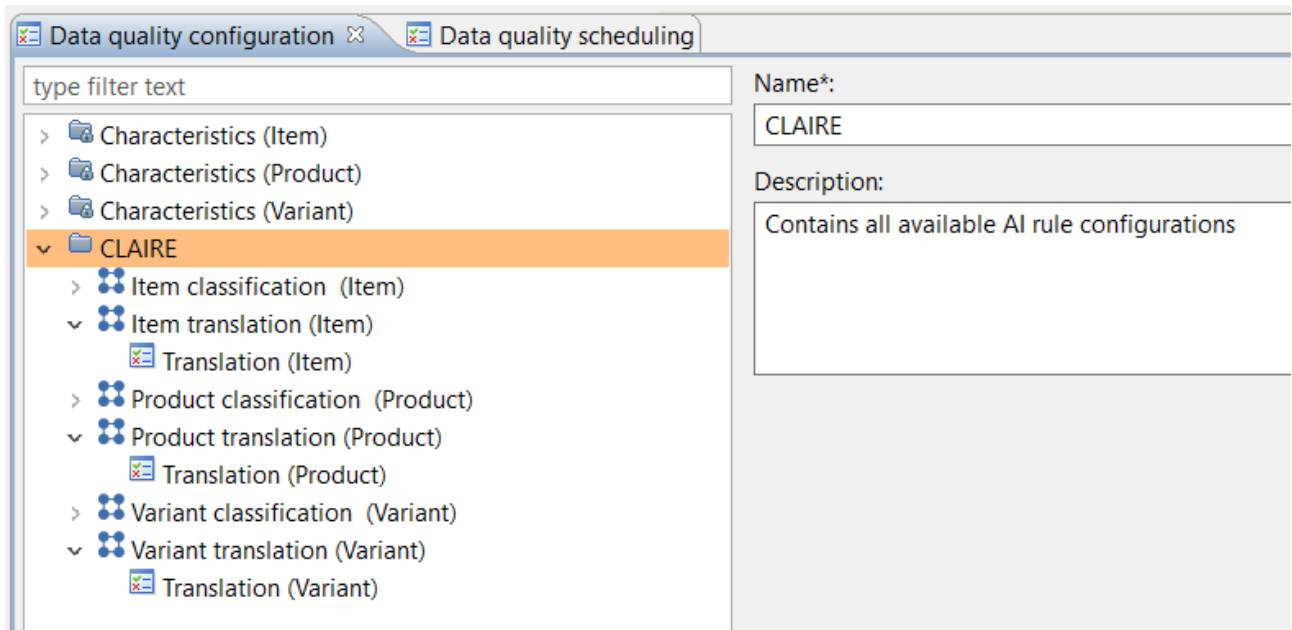
You can make the example flex UI template available in Product 360 Web UI by using the Desktop UI through *Management → Manage UI templates → Load UI templates from file*. The example file can be found under *CLAIRE\_Services → Resources → Flex UI Templates*. From this point onward, you can open the translation flex UI in the web UI by selecting one or multiple items and clicking on *Actions → Open Flex UI → Translation with Claire (en)*.

The screenshot displays the Informatica Product 360 interface. On the left, a table lists four items, with the fourth item (78410) selected. The central panel, titled 'CLAIRE™ Recommendations', shows a recommendation for the Koss EQ50 equalizer. It includes a 'Long description' in English and Spanish. The right panel, titled 'Marketing texts', shows the same recommendation in English and Spanish, with fields for 'Long description', 'Short description', 'Keywords', and 'Other remarks'.

### 5.5.1.4 Batch Translation

In order to provide the brand extraction in a batch process setup (for example after the import of data) the Claire accelerator comes with the following capabilities:

After installation and configuration of the accelerator there is a new data quality category named **CLAIRE**:



How to use the translation rule configuration



It is highly recommended to **clone** one of the existing Translation rule configurations and then modify the input ports according to your needs. The "default" rule configuration is just a template and should not be used as is. Even if you delete this rule configuration, category or group, it will come up on next server start again.

After the data quality rule configuration has been cloned and the parameter adjusted, it can be used for triggers or direct execution just as every other data quality rule configuration to translate language specific values of product records. The parameters are mandatory and described here:

Parameter	Description
Source fields	<p>Language specific values of these fields will be translated. At least one source field needs to be configured for a successful rule execution and the selected fields must be string fields qualified with a language.</p> <p>Note: Only fully qualified fields are supported. Please don't select another data type for your rule configuration.</p>
Target languages	<p>Language(s) in which the source fields will be translated. Semicolon separated list of language identifiers (e.g. de;fr;es). Valid values include synonyms of any entry in the Enum.Language enumeration in the Product360 repository.</p>



Parameter	Description
Retain existing values	If true, already existing values in the target language will not be overwritten and also won't be send to the translation service for translation to avoid unnecessary costs.



In case no quality status entry is written for a product, variant or item after you triggered the execution of the rule, have a look into the process overview. There could be some reasons (like missing permissions) that might prevent the execution of a rule configuration for a product, variant or item.

The screenshot shows the 'Data quality configuration' window. On the left, a tree view shows the hierarchy: Characteristics (Item) > Characteristics (Product) > Characteristics (Variant) > CLAIRE > Item translation (Item) > Translation (Item). The 'Translation (Item)' rule is selected. The main configuration area on the right includes:

- Name\*:** Translation (Item) [Hidden checkbox]
- Description:** This rule configuration is to translate language specific values of product records. Source field [1..5]: Language specific values of these fields will be translated. Target languages: Language(s) in which the source fields will be translated. Semicolon separated list of language identifiers (es;fr;de). Retain existing values: If true, already existing values in the target language will not be overwritten (true/false).
- Rule name\*:** Translation Service
- Data type:** Item
- Input port table:**

Input port	Data type	Field
Source field 1		Long description (English)
Source field 2		Short description (English)
Source field 3		Keywords (English)
Source field 4		
Source field 5		
Target languages		"de;fr"
Retain existing values		"true"
- Data type:** Item
- Output port table:**

Output port	Data type	Field

Allowed value examples for target languages

Language	Key synonyms
German	deu, ger, de, de_DE
English	eng, en, en_GB, en_EN, en_US
Spanish	esl, es, spa, es_ES

Language	Key synonyms
Finnish	fin, fi, fi_FI
French	fra, fre, fr, fr_FR
Italian	it, ita, it_IT
Dutch	dut, nl, nla, nl_NL
Norwegian	nor, no, no_NO, nb_NO, nn_NO
Russian	ru_RU, ru, rus
Swedish	sve, sv, sv_SE
Portuguese (Brazil)	por, pt_BR, por_BR, pt
Japanese	jpn, ja, ja_JP
Chinese	chi, zh, zh_CN
Korean	ko_KR, ko, kor



Our internal tests on batch execution (based on two source fields: Long description (English) and short description (English) and 3 target languages) using real customer data have shown the following performance results:

Number of product records	Duration
100	3 seconds

Number of product records	Duration
1,000	1 minute
10,000	10 minutes 20 seconds

The average number of characters per product record in our test data set (without counting white spaces) was 35 for the short description (English) and 926 characters for the long description (English).

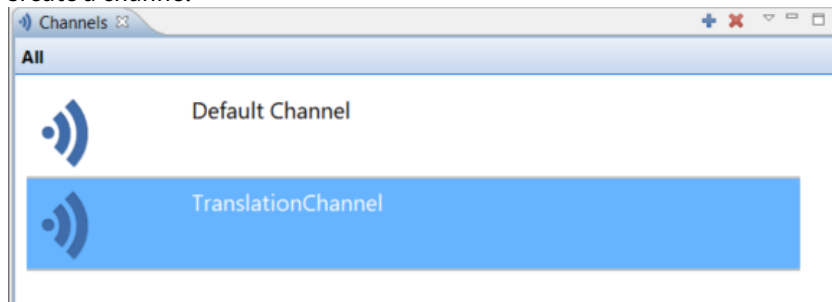
Important:

Always mind the charges of the translation service provider!

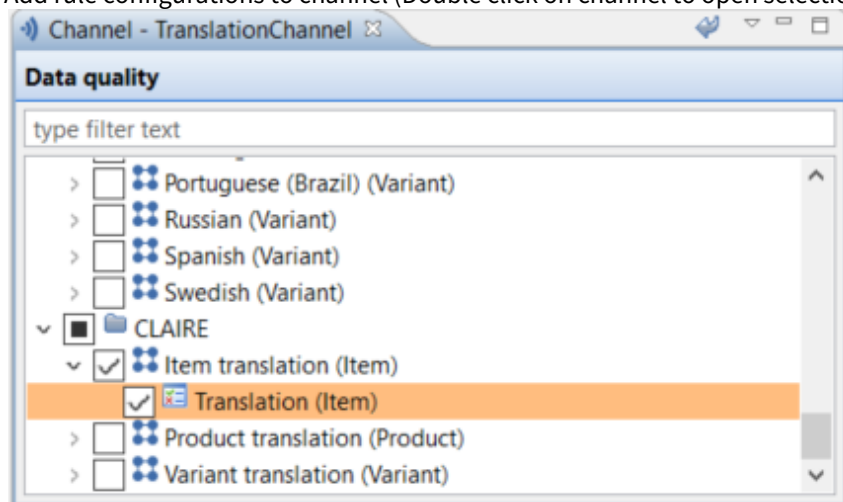
#### Restrict access to rule configurations

You might want to ensure that not every user can execute the batch execution. You can do this by assigning the rule configurations to a channel and securing that channel with object rights which will inherit to the rule configuration.

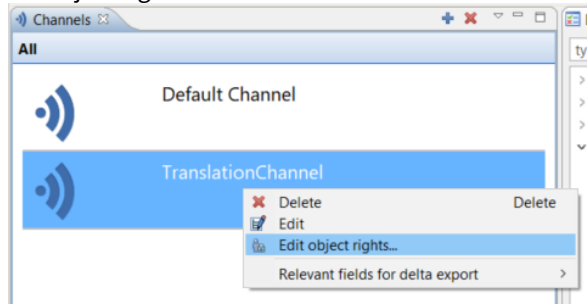
1. Create a channel



2. Add rule configurations to channel (Double click on channel to open selection)



### 3. Set object rights



A user from an excluded user group won't see the rule configuration anymore in the 'Execute data quality rules' dialog.

Restrict number of items for DQ rule execution

In order to control budgets and cross charges with the translation service provider, it is possible to restrict the number of items that can be processed within a single DQ rule execution.

In the file *claire.properties* you will find the property `claire.translation.dq.max.objects` with a default of '100' records per batch call. This number can be adjusted to your needs.

#### **claire.properties**

```
# The used translation service may cause some costs. In order to restrict
# unintentional translations, here you
# can limit the number of objects on which a translation dq rule is executed in one
# run.
# An empty value means there is no restriction.
claire.translation.dq.max.objects = 100
```

If the DQ rule configuration is executed on more than 100 items, then the job will be canceled, and no translations will be done. You will find an entry in the process overview with a corresponding message.

If the property is left blank, there will be no restriction at all.



Before the check is done, all items for which the user does not have write object rights are pre-filtered. In case a user selected 105 items but is missing the write object right for 6 of them, the DQ rule configuration will be executed and the translations will be executed.

#### 5.5.1.5 Attribute Support for Claire Translation

It is also possible to translate *language dependent* attribute values of datatype 'Character string' into defined target languages.

Configuration of attribute translations within a flex UI



Below is an example of a flex UI definition allowing for attribute values translation.

#### Flex UI configuration for attribute translation

```
<group identifier="Claire info">
  <layoutData>
    <parameter key="colSpan" value="1"/>
    <parameter key="rowSpan" value="1"/>
  </layoutData>
  <component i18NKey="Claire" identifier="claire full" type="claire">
    <layoutData>
      <parameter key="collapsible" value="true"/>
      <parameter key="collapsed" value="false"/>
    </layoutData>
    <parameter key="context" value="translation"/>
    <parameter key="translationSourceFields" value="ArticleLang.DescriptionShort(en)"
  />
    <parameter key="attributeTranslationSourceLanguage" value="en"/>
    <parameter key="attributeNamesInKeyLanguage" value="Size;Color"/>
    <parameter key="translationTargetLanguages" value="es;fr;de"/>
  </component>
</group>
```


The additional parameters specific to attribute values translation are the following:

Parameter name	Description	Valid values examples	Default
attributeTranslationSourceLanguage	The source language of the attributes that shall be translated. Valid values include synonyms of any entry in the Enum.Language enumeration within the Product360 repository. If no value is defined, no attribute will be displayed for translation on the UI.	en	<empty>

Parameter name	Description	Valid values examples	Default
attributeNamesInKeyLanguage	<p>The name(s) of the attribute(s) in the key language will be used for translation. If no value is defined, all existing attributes of data type 'Character string' will be displayed. When using multiple distinct attribute names, separate them with semicolons.</p> <div>  If you want to use an attribute name that contains a semicolon, you have to escape it by '\'.   Example: The attribute <i>Length;Width;Height</i> has to be configured as <code>&lt;parameter key="attributeNamesInKeyLanguage" value="Length\;Width\;Height"/&gt;</code> </div> <div>  The flex UI definition is using XML syntax which means that you have to care about some special characters.   Example: The attribute <i>Length&amp;Width</i> has to be configured as <code>&lt;parameter key="attributeNamesInKeyLanguage" value="Length&amp;Width"/&gt;</code> </div>	Size;Color	<empty>

The accelerator package provides a flex UI template specific for attribute translation which can be found under *CLAIRE\_Services* → *Resources* → *Flex UI Templates to get you started*.

Items (1)

	Image	Item no.	Status
1		AA-02	01 New

Detail (Item)

Short description:

T-Shirt


Color:

red

Size:

M

CLAIRE™ Recommendations



We found new recommendations for you!

Apply all

Short description

English (source): T-Shirt

Spanish

Camiseta de manga corta

French

T-shirt

German

T-Shirt

Apply

Attribute value(s) of Color

English (source): red

Spanish

rojo

French

rouge

German

rot

Apply

Attribute value(s) of Size

English (source): M

Attribute values

Language

English

Short description:

T-Shirt

Color:

red

Size:

M

Language

Spanish

Short description:

No content

[Color]:

No content

[Size]:

No content

Language

French

Short description:

No content

[Color]:

No content

[Size]:

No content

Language

German

Short description:

No content

Farbe:

No content

Größe:

No content

### Excluded Attributes

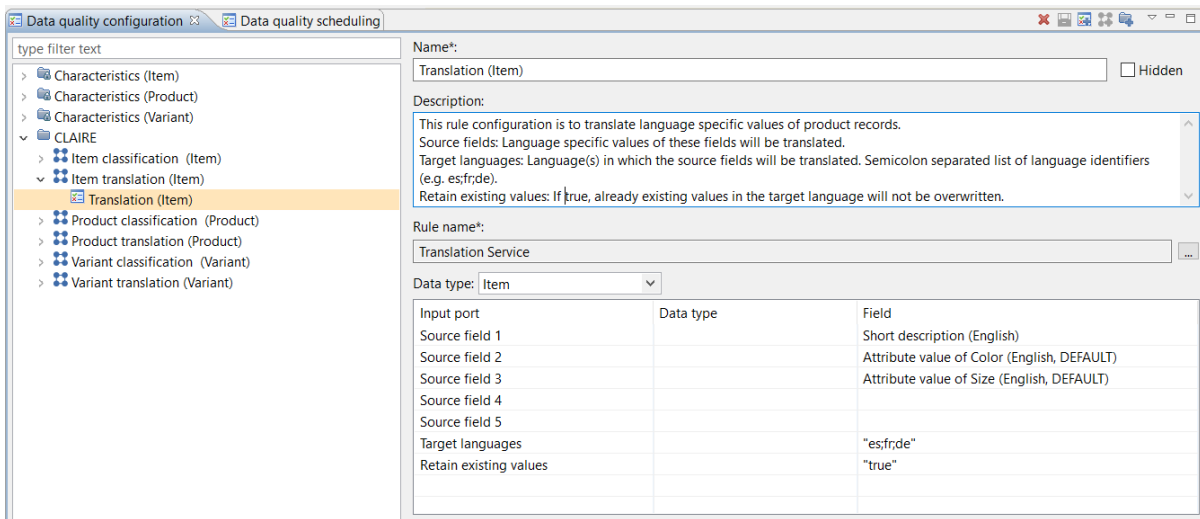
If you are missing an attribute, it might be because some attributes are excluded from the translation. Check for the following:

- Is the attribute of data type 'Character string'?
- Are preset values available for the attribute?

The preset values have to originate from a mapping to a structure group feature in a maintenance structure. Since preset values are based on structure values, it makes sense to translate those directly instead.

## Batch translation

You can use the existing translation data quality rule configuration to translate fully qualified attribute paths in batch mode. For details about the use of the translation rule configuration see the section *Batch Translation* from this chapter.



### 5.5.1.6 Characteristic Record Support for Claire Translation

It is also possible to translate language dependent characteristic record values of datatype 'Text' into defined target languages.

Configuration of characteristic record translation within a flex UI

Below is an example of a flex UI definition allowing for characteristic record values translation.



#### Flex UI configuration for attribute translation

```
<group identifier="Claire info">
  <layoutData>
    <parameter key="colSpan" value="1"/>
    <parameter key="rowSpan" value="1"/>
  </layoutData>
  <component i18NKey="Claire" identifier="claire full" type="claire">
    <layoutData>
      <parameter key="collapsible" value="true"/>
      <parameter key="collapsed" value="false"/>
    </layoutData>
    <parameter key="context" value="translation"/>
    <parameter key="translationSourceFields" value="ArticleLang.DescriptionShort(en)" />
    <parameter key="characteristicRecordTranslationSourceLanguage" value="en"/>
    <parameter key="characteristicCategories" value="Ingredients;Allergens"/>
    <parameter key="translationTargetLanguages" value="es;fr;de"/>
  </component>
</group>
```

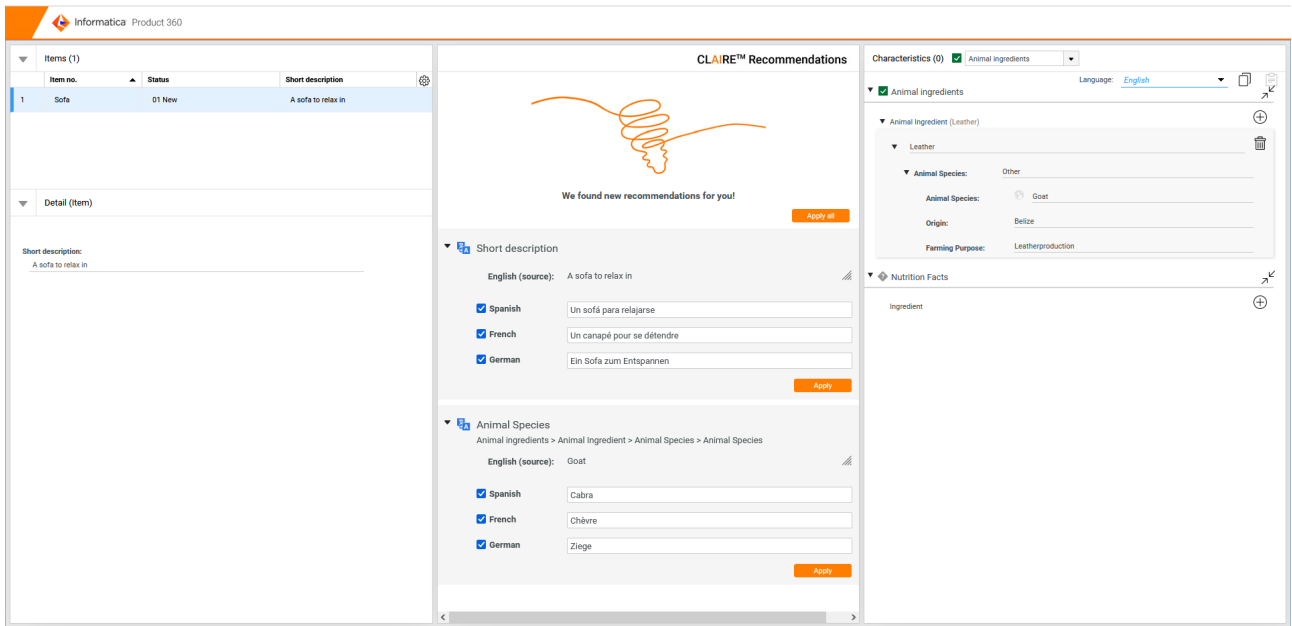
The additional parameters specific to characteristic record values translation are the following:



Parameter name	Description	Valid values examples	Default
characteristicRecordTranslationSourceLanguage	The source language of the characteristic records that shall be translated. Valid values include synonyms of any entry in the Enum.Language enumeration within the Product360 repository. If no value is defined, no characteristic records will be displayed for translation on the UI.	en	<empty>

Parameter name	Description	Valid values examples	Default
characteristicCategories	<p>The code(s) of the lookup values that represent the characteristic categories that will be used for translation. If no value is defined, records of all existing characteristics will be displayed. When using multiple category codes, separate them with semicolons.</p> <div>  If you want to use a category code that contains a semicolon, you have to escape it by '\'.             Example: The code <i>Length;Width;Height</i> has to be configured as <code>&lt;parameter key="characteristicCategories" value="Length\;Width\;Height"/&gt;</code> </div> <div>  The flex UI definition is using XML syntax which means that you have to care about some special characters.             Example: The category code <i>Length&amp;Width</i> has to be configured as <code>&lt;parameter key="characteristicCategories" value="Length&amp;Widt h"/&gt;</code> </div>	Ingredients;Allergens	<empty>

The accelerator package provides a flex UI template specific for characteristic record translation which can be found under *CLAIRE\_Services* → *Resources* → *Flex UI Templates*.



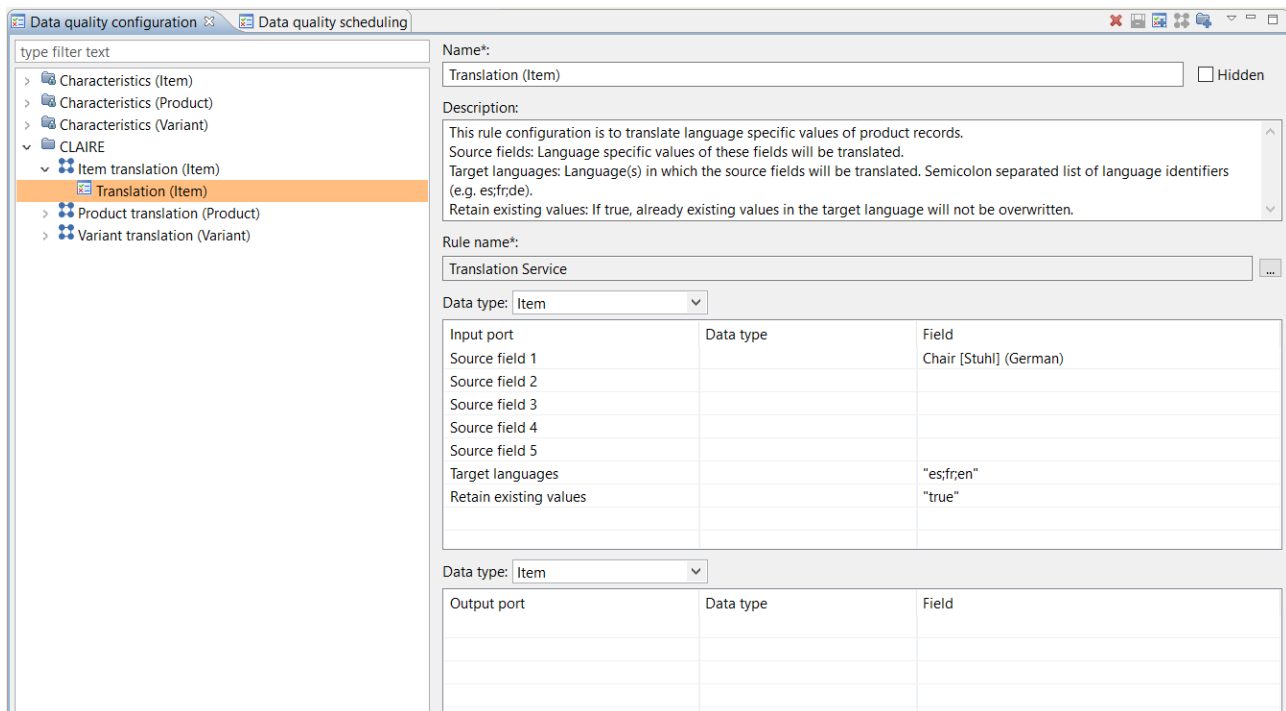
## Excluded characteristic records

If you are missing a characteristic record, it might be because some records are excluded from the translation. Check for the following:

- Is the corresponding characteristic of data type 'Text'?
- Is the corresponding characteristic language specific?
- Is the corresponding characteristic **not** read only?
- Is there a value in the source language to translate?
- Is the corresponding characteristic active?
- Is the category of the corresponding characteristic active?
- Is the product/variant/item assigned to a structure group that has an assignment to the category of the corresponding characteristic?
- Does the user have sufficient read and/or write access rights?

## Batch translation

You can use the existing translation data quality rule configuration to translate fully qualified paths of records of simple characteristics in a batch mode. For details about the use of the translation rule configuration see the section *Batch Translation* from this chapter.



## 5.6 Migration Guide from 10.1 to 10.5

### 5.6.1 Introduction

This section covers the changes that we have made to the CLAIRE accelerator from Product 360 version **10.1** to version **10.5**.

In addition to CLAIRE product classification, we have extended CLAIRE capabilities to also extract brand names from product titles or descriptions.

### 5.6.2 Changes Affecting Product Classification

To separate folders and/or files generated from brand extraction and product classification trainings, we did the following changes:

- separated paths to the data folders in which files used for trainings are saved for both use cases ( `upload_folder` and `data_path_extraction` parameters specified in the `config.ini` file)
- separated paths to the model folders in which models generated from both trainings are saved ( `model_path` and `model_path_extraction` parameters specified in the `config.ini` file)

To not loose CLAIRE product classification pre trained models, please copy them to the new location specified by `model_path` parameter and rerun the installation script. We recommend creating sub folders

in the data and model folders called `product_classification` and `attribute_extraction` as described [here](#) (see page 0).



- Old product classification data and model folders, if they were located in different locations than those defined in the current config.ini file, can be deleted.
- Old version of python (*Python version 3.8.5*) required for CLAIRE in Product 360 version 10.1 can be deinstalled.

## 6 Swagger UI

### 6.1 Purpose

The purpose of this document is to educate the user on the usage of the Product 360 Swagger UI accelerator.

The Swagger UI is a REST API client for the visual interaction with Informatica MDM - Product 360 Service APIs, specifically the List and Management API components.

This document covers all the aspects of the application setup, right from installation to adoption. This guide assumes basic understanding of Product 360 and corresponding REST functionality.

For more information on Product 360 and REST functionality, please refer to the Service API documentation packaged with the application.

### 6.2 Installation

#### 6.2.1 Prerequisites

The users using this application are expected to have at least a basic understanding of Informatica MDM - Product 360 and corresponding Service API functionality.

The setup/application prerequisites include:

- Oracle Java SE 8 or above or Azul OpenJDK 8 and above
- Browser (Chrome, Firefox or Microsoft Edge)
- Informatica MDM - Product 360 Server 10.x or above

#### 6.2.2 Terms

Below are the quick references to terms that the user may come across at multiple points:

- **Swagger** - Swagger is an open-source software set of tools to design, build, document, and use RESTful web services, developed by SmartBear Software.

- **Metadata Engine** - Java based dynamic REST API builder for Product 360 functionalities of List API and Meta API components using the repository data.
- **REST** - REST, or **RE**presentational **S**tate **T**ransfer, is an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other.

### 6.2.3 Download package

The Informatica MDM - Product 360 Swagger UI accelerator can be found within the **PIM\_10.1.0.00.00\_Accelerators.zip**.

Once downloaded, extract the zip to a folder. Preferably Informatica folder for ease of access. The extracted folder would look like below:

logs	File folder
service installation	File folder
static_out	File folder
application.properties	PROPERTIES File
log4j2.xml	XML File
spring-boot-swagger2-0.0.1-SNAP...	WAR File
start_swagger_ui.bat	Windows Batch File
start_swagger_ui.sh	SH File

### 6.2.4 Operation as a Windows Service

Navigate to the `SwaggerUI/service_installation` folder

logs	File folder
static_out	File folder
startup_swaggerui_service.bat	Windows Batch File
startup_swaggerui_service.sh	SH File
SwaggerUI.exe	Application
SwaggerUI.xml	XML File

To install Swagger UI as a Windows service, open command prompt in "Administrator Mode" and run the below command from extracted folder directory

```
SwaggerUI.exe install
```

Ensure Java and JDK/JRE are added to **PATH / JAVA\_PATH** variables in the Windows Environment

```
Administrator: Command Prompt
C:\Informatica\PIM_10.1.0.00.00_Accelerators\PIM_10.1.0.00.00_SwaggerUI\service installation> SwaggerUI.exe install
2020-11-25 04:26:27,488 INFO - Installing the service with id 'P360SwaggerUI'
```

Open "Services" app and start the **Informatica MDM – Product 360 Swagger UI** service

Name	Description	Status	Startup Type	Log On As
Informatica MDM - Product360 Swagger UI	The IKEEXT service hosts the Internet Key Exchange (IKE) and Authenticated Internet Protocol (AuthIP) key...	Running	Automatic (Trigger Start)	Local System
Informatica 10.2.0	Informatica Services	Running	Automatic (Delayed Start)	Local System
Informatica AVOS Apache Tomcat	Apache Tomcat 8.5.38 Server - https://tomcat.apache.org/	Running	Automatic (Delayed Start)	Local System
Informatica MDM - Product 360 - Control Center v1.0	Informatica MDM - Product 360 - Control Center v1.0	Running	Automatic (Delayed Start)	Local System
Informatica MDM - Product 360 v10.1	Informatica MDM - Product 360 v10.1	Running	Automatic (Delayed Start)	Local System
Informatica MDM - Product360 Swagger UI	Informatica MDM - Product360 Swagger UI	Running	Automatic	Local System
Informatica Media Manager Apache Tomcat	Apache Tomcat 8.5.39 Server - https://tomcat.apache.org/	Running	Manual	Local System
Informatica Media Manager Process Watcher - 2		Running	Manual	Local System

To uninstall the Swagger UI service, open command prompt in "Administrator Mode" and run the below command from extracted folder directory

```
SwaggerUI.exe uninstall
```

## 6.2.5 Operation in Standalone Mode

To run the Swagger UI as a standalone utility, navigate to the *SwaggerUI* folder and launch the application by executing the `start_swagger_ui.bat` / `start_swagger_ui.sh` scripts for Windows or Linux respectively. The application would open the login page of Product 360 Swagger UI in the default browser.

Ensure Java and JDK/JRE are added to **PATH / JAVA\_PATH** variables in your Windows/LINUX Environment

	spring-boot-swagger2-0.0.1-SNAPSHOT....	WAR File
	start_swagger_ui.bat	Windows Batch File
	start_swagger_ui.sh	SH File
	application.properties	PROPERTIES File
	log4j2.xml	XML File
	logs	File folder
	service installation	File folder
	static_out	File folder

## 6.3 Configuration and Operation

### 6.3.1 Configuration

The following application configurations are provided within `application.properties`, for adjusting the operation of Swagger UI

#### **application.properties**

```
# Title for displaying within the Swagger UI context
application.title=Informatica MDM - Product 360 Swagger UI

# Context settings for Swagger UI URL
server.contextPath=/swagger
server.port=9080

# Specify the maximum number of threads used for building the OpenAPI specification
YAML file.
# Allowed value is an integer greater than 0.
server.maxThreads = 10

# The maximum time for the user session in minutes
server.maxSession= 300

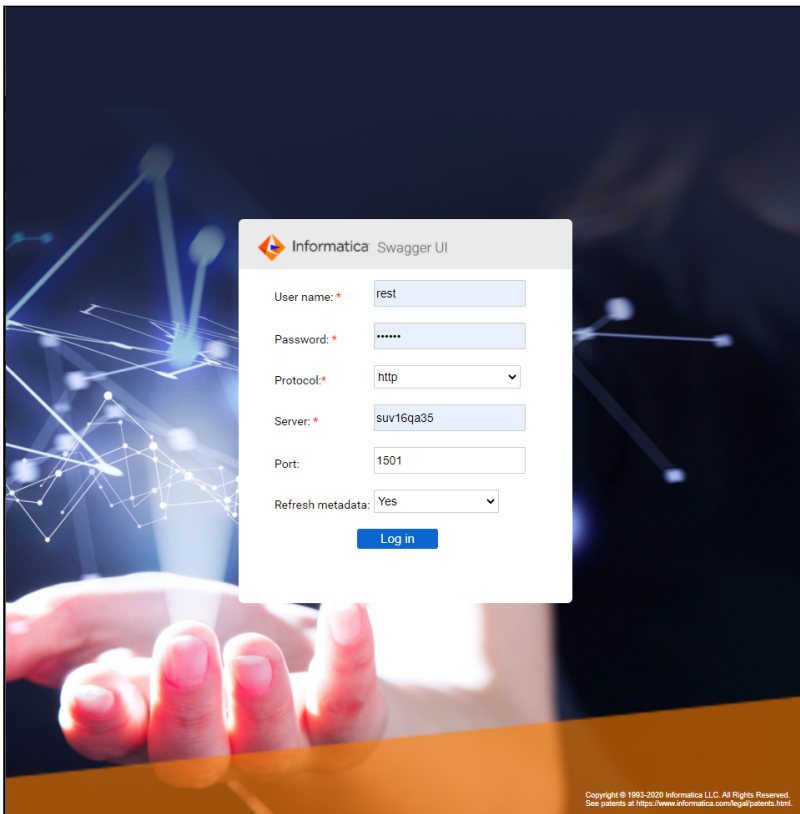
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
spring.output.ansi.enabled=ALWAYS
application.formatted-version=1.0
```

### 6.3.2 Operation

The Swagger UI landing page, the login window, should be presented in the default browser with URL “http://localhost:9080/swagger/swagger-ui.html”. This URL reflects the default properties set in the configurations. This might be different based on your setup.

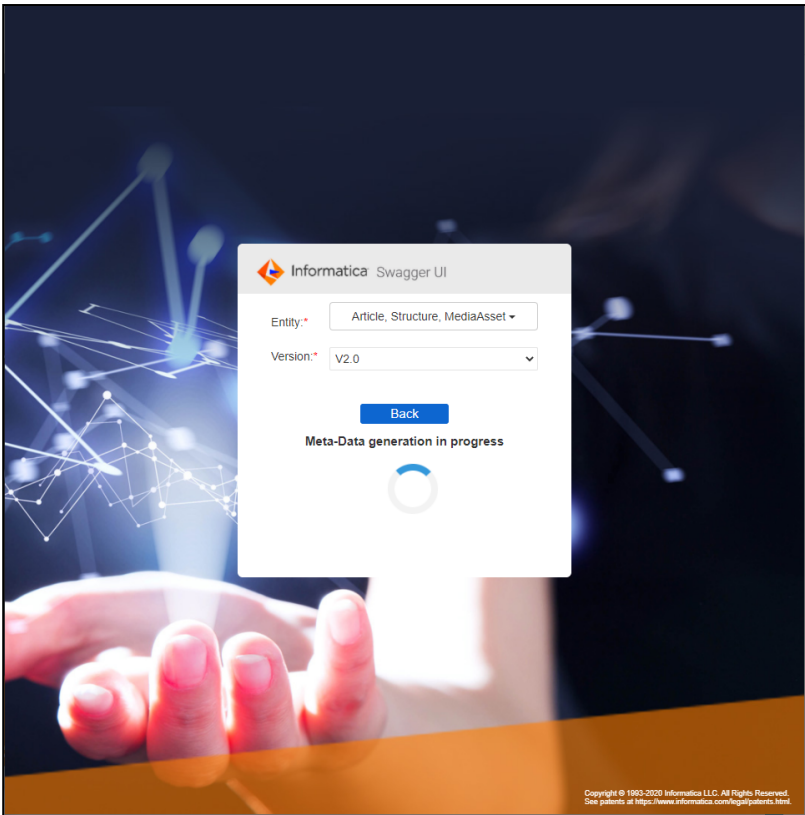
Enter the REST API User and Product 360 Server credentials received from the admin, into the login window. For the first time, select "Yes" from the "Refresh Metadata" drop-down and click login. At this point the application fetches the latest repository information and builds a list of entities available within the specific Product 360 installation. In case of no interim changes in the repository, for future logins this step can be skipped by choosing "No" under "Refresh metadata". However, with any changes to the entity information in the repository this metadata refresh action should be performed again.



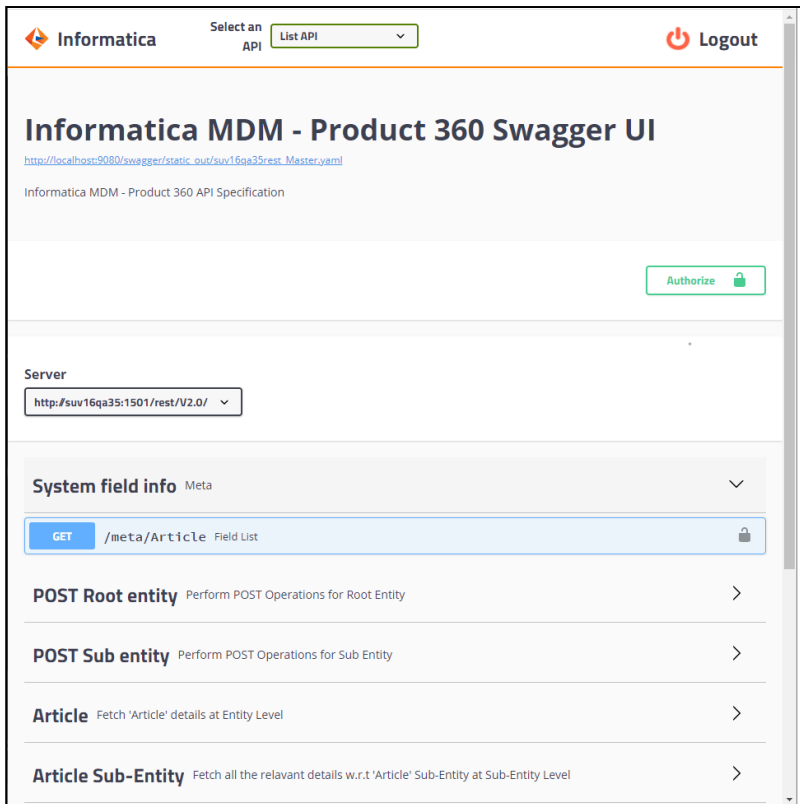


Once the metadata is refreshed, a page with the supported entities will be presented. Select the entity types, that you would like to operate upon using the List API, and click "Generate". This action initiates the generation of the custom repository specific OpenAPI specification for the supported Service API.

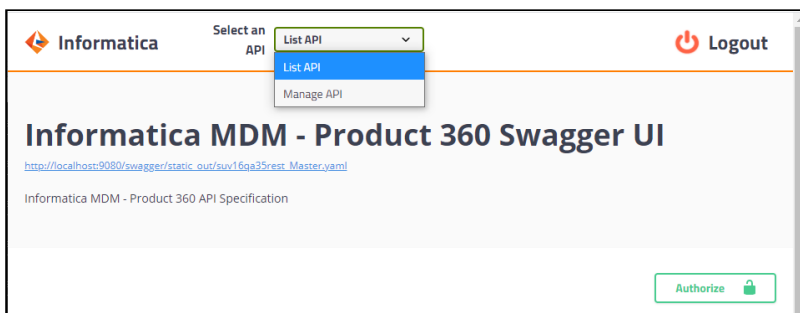
Depending upon the number of entities / sub-entities in the repository and the network overhead, the metadata generation and the corresponding OpenAPI specification generation might take a few minutes. As mentioned earlier, this step would have to be performed only whenever there is any change in the metadata information.



Once the OpenAPI specification is generated, the home page of Product 360 Swagger UI, as shown below, should appear.



Select one of the API specifications from the drop-down list. Once a selection is made the corresponding OpenAPI YAML specification file can be downloaded, if needed for client REST API implementations, from the link appearing under the Informatica logo.



Before executing any query, you must be authorized. Click on the Authorize button on the home page and enter your user credentials. Once successfully authorized you can execute the required operations. Later pages show an example for the execution of a `GET` operation for an entity.

Available authorizations

httpBasic (http, Basic)

Username:

rest

Password:

.....

Close

Authorize

### 6.3.2.1 Example : Fetch 'Article' details at Entity Level, by Items in a catalog

On the Swagger UI page is a list of entity and sub-entity panels corresponding to the selection made earlier, while generating the OpenAPI specification. Upon clicking any of the entity / sub-entity panels, the server stubs to the respective API endpoints and permitted REST operations would be displayed.

For the current example we choose Article > GET /list/Article/byCatalog

Server

http://suv16qa35:1501/rest/V2.0/

System field info

Meta

>

POST Root entity

Perform POST Operations for Root Entity

>

POST Sub entity

Perform POST Operations for Sub Entity

>

Article

Fetch 'Article' details at Entity Level

>

Article Sub-Entity

Fetch all the relevant details w.r.t 'Article' Sub-Entity at Sub-Entity Level

>

Structure

Fetch 'Structure' details at Entity Level

>

Structure Sub-Entity

Fetch all the relevant details w.r.t 'Structure' Sub-Entity at Sub-Entity Level

>

MediaAsset

Fetch 'MediaAsset' details at Entity Level

>

MediaAsset Sub-Entity

Fetch all the relevant details w.r.t 'MediaAsset' Sub-Entity at Sub-Entity Level

>

Article

Fetch 'Article' details at Entity Level

GET

/list/Article/byAssortment

Items in assortment

🔒

GET

/list/Article/byCatalog

Items in a catalog

🔒

Returns all items in the specified catalog. The master catalog (default) or any supplier catalog may be specified here.

Parameters

Try it out

On selecting the API endpoint, supported parameters and their descriptions are displayed. To set the parameters and execute, click the “Try it out” button. A set of text boxes, drop-downs and multi-select fields against each parameter would now be enabled

The top screenshot shows the 'Parameters' section of the API interface. It contains a table with the following parameters:

Name	Description
catalog string (query)	The catalog for which the items are to be determined
revision string (query)	The version in which the items must exist
compareRevision string (query)	The version against which all items that have changed have to be displayed.
dataQualityStatus string (query)	The data quality status to be used as the filter.
characteristicValueFilter string (query)	The characteristic value filter query
channel string (query)	The channel to be used as the filter.
ruleConfiguration string (query)	The validation rule to be used as the filter.
fields array [string] (query)	Fields to be displayed in the result

The bottom screenshot shows the same interface with the 'Try it out' button highlighted.

From the listed options, API parameters can be set by selecting fields from dynamically loaded drop-down lists (use Ctrl+Select / Shift+Select, for selecting multiple options) or by entering text based on the parameter type. In the example below, Catalog and display fields parameters are chosen from a drop-down list and parameters such as pageSize are set explicitly using the text box against it. Obligatory required parameters are indicated with a \*required against the parameter.

Name	Description
catalog string (query)	The catalog for which the items are to be determined
revision string (query)	Must exist
compareRevision	Items that have changed have to be displayed.

fields	Fields to be displayed in the result
array [string] (query)	<div> Article.AclProxy  Article.CatalogProxy  Article.ContentUnit  Article.CurrentStatus </div>

Once the necessary parameters are set, clicking "Execute" would generate the Curl and Request URLs, that can be used directly in downstream REST API client applications and workflows. The selected query is also executed and the response for the query is shown in the "Response body" section in XML/JSON format based on the selection in the "Accepted Header" parameter.

Execute

Clear

Responses

Curl

```
curl -X GET "http://suv16qa35:1501/rest/V2.0/List/Article/byCatalog?catalog=Master%20catalog&fields=Article.AclFlag,Article.AclProxy,Article.CatalogProxy&pageSize=10" -H "accept: application/xml" -H "Authorization: Basic cmVzdDpoZWlsZXI="
```

Request URL

```
http://suv16qa35:1501/rest/V2.0/List/Article/byCatalog?catalog=Master%20catalog&fields=Article.AclFlag,Article.AclProxy,Article.CatalogProxy&pageSize=10
```

Server response

Code

Details

200

Response body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<entityItemTable>
  <cacheId>no-cache</cacheId>
  <entityIdentifier>Article</entityIdentifier>
  <totalSize>1246</totalSize>
  <startIndex>0</startIndex>
  <pageSize>10</pageSize>
  <rowCount>10</rowCount>
  <columnCount>0</columnCount>
  <columns>
    <rows>
      <object>
        <id>1301</id>
        <entityId>1000</entityId>
      </object>
      <value>
        <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">
          </value>
        <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">
          </value>
        <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:type="entityItemReference">
          <id>1</id>
          <entityId>2908</entityId>
        </value>
      </values>
    </rows>
  </columns>
</entityItemTable>
```

Other REST API operations can be executed in a similar fashion as mentioned above.

## 6.4 Best Practices, Recommendations and Known Issues

For the optimal usage of Swagger UI, the following recommendations are made:

- The installation of the Swagger UI as a service on the same node as that of the Product 360 Server, prevents any latency on account on network and thereby would lead to quicker generation of metadata and OpenAPI specification.
- In case of repositories with complex metadata and large number of entities/sub-entities the following optimizations, through the Swagger UI configuration settings, could ensure timely generation of the specification:
  - Increasing the JVM heap size, recommended default value is 512mb.
  - Increasing the number of threads, recommended default value is 10 threads.

Additionally, the following limitations are known for the current version of Swagger UI:

- It currently supports only the List API and Management API components within Product 360
- Channel Entity as part of Product 360 10.1, is not yet supported

# Copyright

© Copyright Informatica LLC 1993, 2024

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica and the Informatica logo are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at <https://www.informatica.com/trademarks.html>. Other company and product names may be trade names or trademarks of their respective owners.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at [infa\\_documentation@informatica.com](mailto:infa_documentation@informatica.com).

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.