

# Specification Incremental Publishing

- Status Quo
  - Problem
  - Numbers
  - Import
  - Replication
  - Miscellaneous
- Objectives and Requirements
  - Overview
  - Data Load Steps
- Intershop Product Capabilities
  - Automation
    - Process Chains
    - Alternatives
    - Comparison
  - Import Modes
    - Available Import Modes
      - Update
      - Replace
      - Ignore
      - Delete
      - Initial
      - Omit
    - Import Mode Selection
  - Product Sharing
    - Outbound Product Sharing
    - Inbound Product Sharing
    - Derived Products
    - Product Assignments
- Best Practices – Other Customers
- Solution Proposals
  - Overview
  - Product Data
    - Product Sharing Scenarios
    - Product Data Changes
    - Run-time Comparison
    - Price Sharing
    - Per Channel Replication
  - Process Automation
    - File Transfer
    - Update or Replace Mode
    - Import Review
    - Solr Configuration Update
    - Product Attribute Groups
    - Notification Emails
    - Catalog Category Import
    - Tips and Tricks
      - Enable or disable jobs
      - Temporarily disable tasks
      - Rename import files
  - Incremental Import
    - Product Import
    - Price Import
  - Performance and Reliability Improvements
- Effort Estimation
- Guides
  - Running Data Loads
    - Import Chain
    - Replication Chain
    - System Management Console
- Open Issues

## Status Quo

## Problem

The current Zamro Data Load – Import to Edit Cluster and Replication to Live Cluster – is based on a full import without support for deltas. In combination with a quite large number of products or product variants to be offered in the shop this leads to a range of problems in terms of performance and reliability.

Moreover the current process is based on a collection of manual steps, to be triggered in specific order and to be checked for a specific outcome or possible problems. So the current approach is rather complicated, time intense and error-prone.

## Numbers

The full import is run every 45 to 60 days. After export from PIM it takes 7 to 10 days to process the export files and generate products with ZDPT.

Resource		
Channels	BE, NL live; DE in progress; additional channels planned	
Locales	4	
Categories	Zamro	1800
Master Products	Zamro-NL	50,000
Variations	Zamro-NL	514,000
Single Products	Zamro-NL	4,400
Prices	Zamro-NL	518,000

## Import

The product import logic has already been touched. No implicit Solr index updates are triggered. The price list import is not touched and thus triggers cache updates and shows bad run-time performance. See Acceptance Edit run-times below:

Task	Mode	Hours	Remarks	Improvement Candidates
Product Import :: productNoLink_BE_1.xml	REPLACE	1,19		
Product Import :: productNoLink_BE_2.xml	REPLACE	0,69		
Product Import :: productNoLink_BE_3.xml	REPLACE	0,62		
Product Import :: productNoLink_BE_4.xml	REPLACE	0,90		
Product Import :: productNoLink_BE_5.xml	REPLACE	0,85		
Product Import :: product_BE_1.xml	REPLACE	0,27		
Product Import :: master_BE.xml	REPLACE	0,98		
Total, above steps		5,49		Delta, Automation, Parallelization
Price List Import :: productPricing_BE.xml	REPLACE	7,08	Includes Solr Update	Remove Solr Update, Automation, Parallelization
Rebuild Search Index :: product-search-idx-nl_BE.xml	REBUILD	3,05		
Replication		1,10		
Total, above steps		16,72		
Product Delete Import :: DeleteOnIntershop.csv	DELETE	3,21		
Total, above steps		19,93		

## Replication

All existing channels are for the time being replicated together in a single step. Performance wise sticking with current global replication approach is fine. For organizational purposes channel specific replication should be addressed.

The following replication groups are in use:

- *Branding*
- **Catalogs**
- *Channels/MasterRepository*
- *Image Definitions*
- **Image References**
- *Labels/Label Assignments*
- *Localization Data*
- *Organization*
- *Pages and Components*
- **Products**
- **Product Attribute Groups**
- **Product Prices**
- *Profanity Check Resources*
- **Search Indexes**

The replication groups listed in *gray italics* are only for reference and won't be part of the automated data load.

## Miscellaneous

Currently, ICM version 7.7.2.4 is run. An update to version 7.7.3 is planned.

## Objectives and Requirements

### Overview

Starting from the Status Quo described above drafting basic objectives is rather simple:

- Automation of the overall data load process to a reasonable degree
- Improve the currently seen performance of the data load
- Reduce the risk of running into technical errors during data load
- Memory management of Oracle data base is a known source of issues (to be addressed by [Carrenza](#))

### Data Load Steps

The following steps are basically describing the target data load process:

1. *PIM Request to update Product Data*
2. *Source file generation through [ZDPT](#)*
3. File transfer to ICM Edit cluster
4. Processing of XML import files
  - a. Catalogs
  - b. Products
  - c. Product Attribute Groups
  - d. Prices
  - e. Solr Filters
5. Rebuild [Solr](#) Index
6. Clear caches
7. Check of update results on ICM Edit cluster
8. Replication to ICM Live cluster
9. *Cache crawler*

Steps shown in *gray italics* are listed for the sake of completeness but are not in scope when discussing improvements in this document.

## Intershop Product Capabilities

### Automation

### Process Chains

A typical mechanism put into place when reducing the amount of manual steps to be executed during data load or similar activities is the use of [Pr](#)

[Process Chains](#). While these chains don't offer any import specific functionality they offer a good foundation with regards to automation in general. Thus development keeps focused on actual data load topics instead of designing an automation framework.

Common problems with the full-scale use of process chains arise from the fact that it is often difficult to reliably detect errors at specific tasks during execution. This is because a chain task can be any snippet of functionality represented by a job or a pipeline or a process chain reporting an error as a simple error log entry, an exception being thrown, a pipeline return value, a job configuration attribute changed and so on. In addition the steps are often executed in parallel. The execution of two steps can happen on the same or on different machines – independently from the question of serial or parallel execution mode. Therefore, a problem might be visible during a specific run while not occurring during another process chain run. However, it is important to understand that these difficulties basically arise from the requirements side. Dropping process chains in favor for a different solution offering similar features one would need to address exactly the same topics.

### Example Process Chain

See also: [Cookbook - Process Chains](#), [DS Distributed Process Chains](#)

## Alternatives

Usage of ICM offered process chains is only one thinkable scenario. It is also possible to offer only some kind of interface to the steps to be executed on ICM side in order to trigger actual execution from some external scheduling or maintenance system: ZDPT, Cron, Jenkins. **However, there should be some good reason for investing extra implementation effort here.**

See also: [Execute jobs remotely through ZDPT / REST](#).

## Comparison

Process Chain	Alternative Solution
▪ Available process control solution	▪ Planned process control solution
▪ Proven in many installations (large and small)	▪ Framework testing, bug-fixing, hidden show stoppers
▪ Graphical process representation in SMC back office	▪ Graphical process representation to be implemented
▪ Smooth integration into ICM	▪ Integration via new interface (REST, ...)
▪ Testing of process chain definition and execution on same end	• Testing: Communication overhead between process control at Zamro and execution and adjustment of actual logic at ISH
▪ Loose coupling	▪ Tight coupling

Loose coupling as Entity A (ZDPT) generates the import files and Entity B (ICM) processes the import files. Entity A doesn't require any knowledge regarding the internals of the data load performed on Entity B (ICM). Replacing ZDPT by a different software generating the files would keep the logic at ICM completely untouched and everything would continue to work. Replacing ICM by a different software doing the data load using some other process control mechanism would mean no changes regarding process control are required at ZDPT .

## Import Modes

### Available Import Modes

The following [import modes](#) are available in Intershop Commerce Management. While Product and Category imports are supporting all listed modes other imports typically do only support a reasonable subset.

### Update

Updates existing objects and creates records for new objects. Attributes and objects that do not exist in the import file are kept untouched.

### Replace

Replaces all data at existing objects and creates records for new objects. Attributes and objects that do not exist in the import file will be removed.

## Ignore

Ignores all objects that already exist in the database; creates records only for new objects and adds them to the database. For example, if a product is specified in the import source and the product is found in the database by the import/export service, it isn't modified.

## Delete

Deletes the specified objects from the database.

## Initial

Performs no database query to find existing objects. This provides for a quick import but causes an error whenever an object is imported that already exists. This mode is generally used during the DBInit process.

## Omit

Does not store objects defined in the import source. This can be useful for tests.

## Import Mode Selection

An import mode can be selected actively when using the back office import wizard or by defining the attribute `import-mode` at the respective business object's root tag within the XML import source file. The latter approach allows to run so called mixed mode imports by defining different import modes for the individual objects defined in the source. Moreover, it is possible to define separate modes at some of the business object's child tags. For example, a product can be imported in update mode while its category assignments are imported in replace mode at the same time in order to remove outdated assignments to categories as well. The values of attribute `import-mode` need to be specified in uppercase. Please check the respective schema definition for further details.

## Product Sharing

The basic idea of [product sharing](#) is to import products physically to a single master repository and then distribute these products down to an arbitrary number of channel repositories in term of visibility. Sharing products results in less import XML data, a smaller product table in the data base and as a result also improved replication performance. Moreover, from a shop manager perspective it can often be helpful to be able to update product data at a single place and have the result visible at all channels provided to the customers.

Of course there are situations where not every product or product attribute should be visible or look exactly the same in all available channels. Fortunately, the product sharing mechanism takes care of these real world requirements. It actually offers a lot of flexibility.

## Outbound Product Sharing

Outbound product sharing allows the selection of the target channels and the products to be shared into these channels performed at the source or master repository. The sharing direction is from the master repository to several channel repositories. This process shares all or all selected products of the source domain with the channels selected as target domains. The availability of Sharing Groups allows the organization of complex sharing scenarios using manual or rule-based product selection.

## Inbound Product Sharing

In addition the outbound sharing described above the Intershop Commerce Management allows to select certain products shared through a master repository to be activated for a specific channel (which is the target domain of the sharing relation). By default, the inbound product sharing selects all outbound shared products as active or visible in the respective channel. The sharing direction is from the channel repository to the master repository. This mechanism activates all or all selected products of the source domain for the channel at hand.

## Derived Products

While the previous sections described how to hide or activate selected products for specific channels the sharing mechanism is also offering granularity down to the product level itself. The product made visible in a channel can be adjusted on channel level to fit specific needs. The availability of so called derived products allows to define a delta to the product existing on master repository level. So for example, if the product on master repository level owns a custom attribute guarantee period containing a value 6 the attribute is also visible on channel level, but can be overridden to contain a value of 12 at a specific channel.

A product does usually not live in empty space. It is defined in many ways by its relation to additional objects. It can be part of a classification category, multiple navigation categories, comes with several images, prices and so on. On a technical level all these relations are defined through assignment tables or similar means. Therefore, all these relations exist independently from the product sharing approach. In other words, a product shared down into two channels can have totally or partly different prices, images, category assignments per channel if required. Nonetheless, it can also share relations defined on master repository level. The details are specific for different types of relations due to varying business needs or technical reasons.

The below tables shows several key numbers as available from other customers.

## Solution Proposals

In order to increase import performance and reliability there are four main areas that have to be considered:

- As product data import processes are subject of continuous improvement we recommend to do the implementation in an iterative approach and to address those topics first that promise the most benefit.

Iteration 0 is about current analysis, requirements gathering and providing a recommendation.

Iteration 1 is about defining the structure of Product Data that fits best Zamro's needs from organizational point of view, automation of the product import process, addressing critical items, e.g. Oracle PGA.

Further iterations are focusing on improving the import process, installing the Delta Import, and technical operations.

	Activity Description	Iteration	Effort	Responsible	Status	Due Date
Area 1: Product Data						

1.1	Provide Input regarding objectives, requirements and current status of the product import process.	0		Zamro	done	
1.2	Analyze Status Quo, document Requirements, provide information about best practices and provide recommendation.  Involve Intershop Engineering Product Owners.	0		ISH	done	
1.3	Provide access to the current System, Implementation and Full Set of Current Product Data.	0		Zamro	done	
1.4	Setup local Zamro environment and analyze Product Data to provide recommendation regarding the benefit regarding performance with Product Sharing.  Overall Sharing is recommended from an Organizational point of view. Regarding the import performance improvements local tests have to be made.	0		ISH	done	
1.5	Recommendation to be done by Intershop: <ul style="list-style-type: none"><li>What to consider regarding product Sharing. In which cases is Product Sharing recommended and in which cases not</li><li>How to deliver Product Data when Sharing is applied</li></ul>	0		ISH	done	
1.6	Zamro to decide for which country channels Product sharing will be applied.	1		Zamro	done	
1.7	Zamro to deliver restructured Product Data per country based on sharing decision <ul style="list-style-type: none"><li>Categories (currently shared)</li><li>Products</li><li>Product Attribute Groups</li><li>Prices (not shared)</li></ul>	1		Zamro	in progress	
1.8	Zamro to deliver full Product Data with Sharing automated from ZDPT	2		Zamro	open	
<b>Area 2: Process Automation</b>						
2.1.	Provide secure file share for Product Data Exchange ZDPT => ICM	1		Zamro	done	
2.2.	Decide on how to trigger the import process and required checks before publishing the updated product data to the Live system  Current assumption is that the Import job is manually triggered in SMC. The same applies for publishing after manual checks have been performed	1		Zamro	done	
2.3.	Clarify how Crawling the Site is currently done at Zamro currently done through 3rd party crawler started manually from Jenkins environment after replication is finished	0		Zamro	done	
2.4	Automate Import Data Provision <ul style="list-style-type: none"><li>PIM Request to update Product Data</li><li>Source file generation through ZDPT and placement on file share for ICM and IOM</li></ul>	1		Zamro	in progress	
2.5	Automate ICM Import and Publication Process with Intershop Process Chains <ul style="list-style-type: none"><li>File transfer to ICM Edit cluster</li><li>Processing of XML import files (REPLACE)</li><li>Rebuild Solr Index</li><li>Clear caches</li><li>Check of update results on ICM Edit cluster</li><li>Replication to ICM Live cluster</li></ul>	1		ISH	done	
2.6	Decide if additional test environment for process automation and process improvement is required.  Especially because of dependencies to country roll-outs / ongoing development and integration work, performance tests,.... <ul style="list-style-type: none"><li>Decision for new Integration Environment</li></ul>	1		Zamro	done	
2.7	Notifications with Email (SUCCESS, FAILURE) <ul style="list-style-type: none"><li>Minimal Solution: Basic notification (SUCCESS, FAILURE) of process chain (done)</li></ul>	2	1d-? (depending on requirements)	ISH	done	

2.8	Attribute Group Import	2	5d	ISH	done	
2.9	Implement Solr filter config update with or without server restart  See also: <a href="#">Solr Configuration Update</a>	2	3d	ISH	done	
2.10	Create DBMigrate steps to roll out config with deployment (not manually)  <ul style="list-style-type: none"> <li>to create chain execution jobs (1d)</li> <li>to create outbound sharing configuration (1d) <ul style="list-style-type: none"> <li>Assumption: complete sharing of org products to channels</li> </ul> </li> </ul> See also: <div> <b>IDP-18</b> - 2.10 create DBMigrate steps to roll out config with deployment (not manually) (~2SD) CW14 <b>DONE</b> </div>	2	2d	ISH	done	
2.11	Evaluate update to next product releases IOM 2.2 and ICM 7.9. ICM 7.9 with import process improvements and new Solr version.	3		Zamro / ISH	open	
2.12	...					
<b>Area 3: Process Improvement</b>						
3.0.	Provide ICM Delta Import Files (UPDATE, DELETE). Assumption is that the same Import automation can be used when Products are correctly marked with UPDATE, DELETE  <ul style="list-style-type: none"> <li>Zamro ZDPT currently status unclear. Was implemented but not Tested.</li> </ul> Timeline: end of third week	2		Zamro	open	
3.1	ICM Delta Import specifics. Are in overall process adjustments required when switching to delta and which restrictions apply (e.g. Solr update,...)  <ul style="list-style-type: none"> <li>delta product import</li> <li>delta price import</li> </ul> Remark: Catalogs, Solr Filters are always delivered in full set  Assumptions: <ul style="list-style-type: none"> <li>Analysis, implementation of minor adjustments, Testing</li> <li>Restriction: Deletion of existing product prices not possible in delta mode</li> </ul>	2	3d	ISH	open	
3.2	Decide on IOM Product Import and EOL Handling  <ul style="list-style-type: none"> <li>IOM can either run in full import mode or Delta. In Full Import Mode products that are no more delivered in the import are entering EOL Process.</li> <li>Currently IOM runs in Delta Mode as so far no full import could be delivered by ZDPT.</li> </ul>	2		Zamro	open	
3.3	Improve Price Import and disable price change events (Currently not optimized regarding Solr etc.).  Improvements: <ul style="list-style-type: none"> <li>switching off creation and processing of price change events</li> <li>setting up four parallel bulker threads</li> </ul>	2	2d	ISH	done	
3.4a	Input from Zamro which Exceptions in the occurred in the past in order to define the cases that will be implemented with 3.4.b Handling of Exceptions.	2		Zamro	done	
3.4b	Handling of Exceptions, implementation of frequently occurring exceptions	2	tbd (depending on Zamro input 3.4a)	ISH	done	
3.5	Automate Cache crawler execution (see 2.3)  <ul style="list-style-type: none"> <li>Decision that item is not in scope for Iteration 2</li> </ul>			Zamro or ISH	open	



3.6a	Testing and improvement of import automation - Full Data import <ul style="list-style-type: none"> <li>▪ Add process chain tasks to manually disable / re-enable jobs if required</li> <li>▪ Add process chain tasks to manually lock / unlock resources if required</li> </ul>	2	5-10d	ISH	open	
3.6b	Zamro Testing of Product data - Full Data Import	2	5-10d	Zamro	in progress	
3.7a	Testing and improvement of import automation - Delta Data import	2	5-10d	ISH	open	
3.7b	Zamro Testing of Product data - Delta Data Import	2	5-10d	Zamro	open	
3.8	Adjust Catalog import so that the categories of all catalogs are dropped before the import. <ul style="list-style-type: none"> <li>• drop the categories of all the catalogs</li> <li>• add new categories as provided</li> <li>• loading the masters is required as it recreates the links with the categories (which will be covered by the automation)</li> </ul>	2	tbd	ISH	done	
3.9	Improve logic 3.8 dropping the categories of all catalogs so that only changes are applied	later			open	
3.10	Add process chain tasks to analyze data base schema, synchronize context indexes if required	later			open	
3.11	Completeness Checks	later		ISH	open	
3.12	...					
<b>Area 4: Operations Improvement</b>						
4.1	Apply Oracle PGA Memory Patch <ul style="list-style-type: none"> <li>• Oracle Patch not applied yet</li> <li>• Oracle Memory Leak fix applied by Carrenza</li> </ul>	1		Zamro / Carrenza	open	
4.2	Monitoring of Product Import Process	later		Zamro	open	
4.3	Analyze Database differences Acceptance / Production	1		ISH	done	
4.4	Analyze Environment Differences Acceptance / Production	1		Zamro / Carrenza	done	
4.5	Define next Steps to improve Production performance for Import / Publish	1		Zamro / Carrenza / ISH	in progress	
4.6	Cut over procedure to switch from channel products to product sharing: <ul style="list-style-type: none"> <li>▪ Merge IDP-9 code into target branch <ul style="list-style-type: none"> <li>▪ Adjust migration path from <del>farpoint</del> to actual target version 1.1.0.9, as described in IDP-18</li> </ul> </li> <li>▪ Run data base export to have a backup available</li> <li>▪ Deployment to Acceptance</li> <li>▪ Run DB Migrate to roll-out configuration</li> <li>▪ Delete products, product images from all channels (delete scripts for products, product images and prices attached)</li> <li>▪ Place SSH private key for ZDPT access</li> <li>▪ Import Full Data Set to Acceptance by executing Import Process Chain <ul style="list-style-type: none"> <li>▪ Ensure required impex folder structure as a prerequisite</li> </ul> </li> <li>▪ Check imported data and execute Replication Process Chain</li> <li>▪ Test Full Product Data Functionally (Zamro) (in progress)</li> </ul>	2	10d	Zamro / ISH	in progress	
4.7	Run full data imports, identify bottlenecks and improve	2		ISH	in progress	

## Product Data

### Product Sharing Scenarios

This section tries to outline some sharing strategies that could fit for the actual data situation in the Zamro store. A general description of Intershop Product Sharing is available in a [dedicated chapter](#).

Scenario	Approach	Implication
<ul style="list-style-type: none"> <li>channels NL and BE do have exactly equal products</li> <li>channel DE has totally different products</li> </ul>	<ul style="list-style-type: none"> <li>products for NL and BE are maintained as a single set on organization level and shared into both channels</li> <li>product for DE are not shared from organization level but maintained on channel level (as currently true for NL and BE)</li> </ul>	<ul style="list-style-type: none"> <li>a change to an organization level product is instantly and implicitly visible to channels NL and BE and not visible to DE</li> <li>a change to a DE channel product has no impact to NL or BE</li> </ul>
<ul style="list-style-type: none"> <li>the NL products are a subset of the BE products or vice versa or both channels have some products not existing in the opponent channel</li> </ul>	<ul style="list-style-type: none"> <li>the union set of products is maintained as a single set on organization level and partially shared into both channels using outbound or inbound sharing settings reflecting the respective needs</li> </ul>	<ul style="list-style-type: none"> <li>a change to an organization level product is instantly and implicitly visible to channels NL and BE</li> <li>a new product added on organization level is implicitly shared or needs explicit white-listing to be shared to the desired channel – depending on the configured sharing settings</li> </ul>
<ul style="list-style-type: none"> <li>channels NL and BE do have the same set of products or contain subsets as described above and the products existing in both channels do sometimes have different attribute values</li> </ul>	<ul style="list-style-type: none"> <li>products for NL and BE are maintained as a single set on organization level and shared into both channels</li> <li>only the different attribute values are maintained on channel level overwriting the values from organization level</li> </ul>	<ul style="list-style-type: none"> <li>a change to an organization level product is instantly and implicitly visible to channels NL and BE whenever no different channel level attribute value has been defined</li> <li>changing channel level attributes will never have an impact on organization level nor other channels</li> </ul>
<ul style="list-style-type: none"> <li>channels NL and BE do have exactly equal products or contain subsets as described above</li> <li>channel DE shares some products with NL and BE and additionally contains a many products unique to the channel</li> </ul>	<ul style="list-style-type: none"> <li>the organization level products of channel NL and BE to be shown also for DE channel are partially shared into the DE channel using inbound sharing settings</li> <li>the additional products unique to the DE channel are maintained on channel level (as currently true for NL and BE)</li> </ul>	<ul style="list-style-type: none"> <li>a change to an organization level product shared into the DE channel is instantly and implicitly visible</li> <li>changing channel level products will never have an impact on organization level nor other channels</li> </ul>

Please also keep in mind that currently products are assigned to the categories on channel level. So while the actual category is already maintained on organization level and shared into the channels the products do exist physically on channel level. Therefore any assignment between product and category is technically defined on channel level. Assuming equal category assignments in both channels the number of category assignments in the data base could be drastically reduced. The degree of reduction becomes even higher with an increasing number of channels under product sharing. The same does also apply to all the image assignments or any kind of object assignment related to products. Specific questions not covered by the above table should be added to the Questions chapter of this document.

The provided test data offers an equal set of products for channels BE, NL and DE. The production environment doesn't know the DE channel yet. The channels BE and NL are subject to constant change. Nonetheless product sets seem to be quite comparable. Therefore simply importing all product data to the organization channel should be a valid approach.

We need to make sure to correctly combine the organization level product maintenance approach with the recently introduced requirement for channel level replication. With the products residing on organization level we no longer start channel replication from strictly separated channel product sets.

## Product Data Changes

Given the most simple scenario that all product data is maintained on organization level and from there shared down into the channels only minimal adjustments to the product data are required at all. All that actually changes is the location the data is imported into. Importing it into organization repository instead of a dedicated channel repository makes it organization level product data which will then be shared into the existing channels. A typical change is merging all localized information from the respective channels into the master import file:

Repository	nl-NL	nl-BE	nl-DE
Zamro-NL	X		
Zamro-BE		X	
Zamro-DE			X
Zamro	X	X	X

See localized values (`xml:lang` name-space) in product samples below:

### Current NL channel repository product sample

### Current DE channel repository product sample

### Future organization repository product sample

## Run-time Comparison

The below table compares data load times from local testing with and without product sharing applied. The testing setup uses a single remote data base machine for Edit and Live schemes. So when using a setup with separate data base machines the network bandwidth between these machines can considerable reduce replication performance. Nonetheless, the table shows the structural advantages of product sharing. The long running product import operations are executed only once. So the positive effect of the sharing approach becomes even more visible and important the more channels do actually become part of the sharing. Additional product imports to reflect potential channel specific product data are not considered below.

The rightmost column shows additional performance improvement from switching off creation and processing of price change events and setting up four parallel bulkier threads. See also [ENFINITY-12620](#).

Operation	Without Product Sharing		With Product Sharing		With Price Optimizations
Transfer category import files	organization	30 seconds*	organization	30 seconds*	30 seconds*
Transfer product import files	NL channel BE channel ...	120 seconds* 120 seconds* ...	organization	120 seconds*	60 seconds
Transfer price import files	NL channel BE channel ...	120 seconds* 120 seconds* ...	NL channel BE channel ...	120 seconds* 120 seconds* ...	120 seconds* 120 seconds* ...
Import Categories	organization	120 seconds*	organization	120 seconds*	120 seconds*
Import Products, no link	NL channel BE channel ...	11,423 seconds 11,423 seconds ...	organization	11,423 seconds	11,423 seconds
Import Products	NL channel BE channel ...	9,020 seconds 9,020 seconds ...	organization	9,020 seconds	9,020 seconds
Import Master Products	NL channel BE channel ...	9,469 seconds 9,469 seconds ...	organization	9,469 seconds	9,469 seconds
Import Prices	NL channel BE channel ...	12,723 seconds 12,723 seconds ...	NL channel BE channel ...	12,723 seconds 12,723 seconds ...	1,667 (5,028) seconds 1,667 (5,028) seconds ...
Import Solr Filters	NL channel BE channel ...	120 seconds*	NL channel BE channel ...	120 seconds*	120 seconds*

Rebuild Search Index	NL channel BE channel ...	5,253 seconds 5,253 seconds ...	NL channel BE channel ...	5,253 seconds 5,253 seconds ...	5,253 seconds 5,253 seconds ...
Total, Import		96,526 seconds		66,494 seconds	44,382 seconds
Replicate Categories	organization	10 seconds*	organization	10 seconds*	10 seconds*
Replicate Products	NL channel BE channel ...	992 seconds 992 seconds ...	organization	992 seconds	992 seconds
Replicate Product Attribute Groups	organization	10 seconds*	organization	10 seconds*	organization
Replicate Prices	NL channel BE channel ...	91 seconds 91 seconds ...	into NL channel into BE channel ...	91 seconds 91 seconds ...	91 seconds 91 seconds ...
Replicate Search Index	NL channel BE channel ...	? seconds ? seconds ...	NL channel BE channel ...	? seconds ? seconds ...	? seconds ? seconds ...
Total, Replication		2,186 seconds		1,194 seconds	1,194 seconds
Cache clear	-	60 seconds*	-	60 seconds*	60 seconds*
Total		98,772 seconds 27:26:12		67,748 seconds 18:49:08	45,636 seconds 12:40:36
*) estimated value, not based on actual testing					

## Price Sharing

Price list based product prices are imported on channel level only. Import of price lists on organization level is not possible without according customization effort. Only native product prices (cost price, list price) are part of the product import and can be maintained on organization level. The details and implications regarding price sharing – combined with product sharing – need further investigation.

## Per Channel Replication

Channel level replication is an out-of-the-box feature in ICM. No special implementation steps required here. So the focus should be to correctly consider per channel replication as part of the automated data load. As mentioned above at the sharing scenarios chapter, replicating only a single channel while updating the product data on organization level might create unwanted conditions. For example, a new product is imported into the organization while the corresponding price is only replicated to the NL channel. Therefore, the BE channel would contain the new product without a price available.

The below table shows the data load we want to apply, partitioned into time frames. A task being executed in the same time frame for multiple repositories (Zamro, Zamro-NL, Zamro-BE) is run in parallel. We can see that Price and Product channel replication is executed for channels in different runs of the data load. This could be achieved by enabling or disabling the respective job configuration during different runs while keeping all the channels in the data load chain definition.

Run , Time frame	Task(s)	Zamro	Zamro-NL	Zamro-BE
A 00	Import file transfer to ICM Edit cluster	X	X	X
A 01	Clear Product Catalog Categories	X		
A 02	Import Category XML	X		
A 03	Import Product XML (Products, Variation Masters)	X		
A 04	Import Product XML (Products, Variation Masters)		X	X
A 05	Import Product Attribute Group XML	X		
A 06	Import Price XML		X	X
A 07	Apply Solr Filter Updates		X	X

A 08	Rebuild Search Indexes		X	X
A 09	Cache Clear	X	X	X
A 10	Check import results on ICM Edit cluster	X	X	X
A 11	Replicate Categories	X		
A 12	Replicate Products	X		
A 13	Replicate Product Attribute Groups	X		
A 14	Replicate Prices, Replicate Products		X	
B 14				X
A 15	Replicate Search Index		X	X
<i>below tasks executed on ICM Live cluster</i>				
A 16	Cache Clear		X	X
A 17	Cache Crawling		X	X

## Process Automation

### File Transfer

The file transfer between [ZDPT](#) and ICM happens through public internet lines. Therefore we need to have a secured connection in place - a decision to use [SFTP](#) has been made. Regarding the transfer direction the pull approach has been chosen as the ICM environment doing the import naturally knows best when to do which file transfer. The ZDPT environment again simply provides the files for download at the moment when it is able to do so. Moreover, the pull approach keeps the file server away from the ICM machines which might be a little closer to public internet than the ZDPT machines.

Currently an SFTP user named *intershop* is used for pulling files from ZDPT machine running on [37.97.184.105](#). To be able to serve import files not only for the production environment, but also for acceptance or integration, separate user contexts should exist. For example the ZDPT could provide files to be imported below the following file system folders later being accessed through different SFTP logins:

```
/home/ish-production
/home/ish-acceptance
/home/ish-integration
```

The relative file system location of the import files describes their import domain context. Examples:

```
Zamro/catalog-1463123323001_update.xml    (catalogs to be created for Zamro organization)
Zamro-MasterRepository/product_update.xml (products to be created in Zamro / master repository)
Zamro-ZamroNL/product_update.xml         (products to be created in Zamro / Zamro NL channel repository)
Zamro-ZamroBE/product_update.xml         (products to be created in Zamro / Zamro BE channel repository)
```

All import files do have an 'xml' file name extension. Example:

```
product_update.xml
```

All import files provide an import mode identifier following the first underscore. Examples:

```
product_update.xml
product_delete.xml
product_replace.xml
```

All import files not starting with a valid import type identifier will be ignored. Example:

```
_foo
_product_update.xml
_LOADED_product_update.xml
Product_update.xml
```

Successfully downloaded import files get a '\_LOADED\_' prefix. Example:

```
product.xml => _LOADED_product.xml
```

Import files can have an optional suffix. The files of a specific import type are sorted lexicographically by their individual suffix to be imported in a defined order. Example:

order	full name	import type	import mode	suffix
-------	-----------	-------------	-------------	--------

1	product_update.xml	product	update	
2	product_updateNEW.xml	product	update	NEW
3	product_update_003.xml	product	update	_003
4	product_update_005.xml	product	update	_005
5	product_delete_006.xml	product	delete	_006
6	product_replace_01.xml	product	replace	_01
7	product_update_obsolete_005.xml	product	update	_obsolete_005

While the approach of lexicographic sorting is a very simple convention which still offers a lot of flexibility, it also requires some discipline to name files according to expected import order.

You should avoid to use index suffices twice. While formally correct the following example is rather confusing:

```
product_update_005.xml
product_update_obsolete_005.xml
```

Further you should avoid to use suffixes twice or omit suffixes when placing files for various import modes. Both of the following files are correctly named, import order is undefined however:

```
product_update_005.xml
product_delete_005.xml
```

Regular product import files are named 'product' followed by the import mode identifier followed by any optional string. Example:

```
product_update.xml
```

Master product import files are handled as part of regular product imports. File naming should make sure that masters will be imported after the mastered products they're pointing to. Examples:

```
product_update_master_001.xml
```

Product Attribute Group import files are named 'productattributegroup' followed by the import mode identifier followed by any optional string. Example:

```
productattributegroup_update.xml
```

Price import files are named 'price' followed by the import mode identifier followed by any optional string. Example:

```
pricelist_update.xml
```

Catalog import files are named 'catalog' followed by the catalog domain name followed by the import mode identifier followed by any optional string. Example:

```
catalog_1463123323001_update.xml
```

Solr Configuration files are named 'solrconfig' followed by either 'ISH-Config' or 'schema' followed by a minus followed by a search index identifier. Example:

```
solrconfig_ISH-Config-product-search-idx-nl_NL.xml
```

```
solrconfig_schema-product-search-idx-nl_NL.xml
```

Possible improvement: Track full import file life cycle (prefix '\_PROCESSED\_' after successful import instead of '\_LOADED\_' after successful load)

## Update or Replace Mode

While the file transfer chapter above shows only update files, the automation does implicitly support both modes. Mode selection is fully transparent to the remaining data load. If data load needs to update existing products an update file needs to be provided. In case existing products have to be replaced a replace file needs to be provided. Both files could contain equal data. It is only the naming that controls which import mode will be used in the end. For more details please refer to the chapters [Import Modes](#) and [File Transfer](#).

## Import Review

While large parts of the data load process from ZDPT up to ICM Live cluster should ideally happen without need for human interference there is one exception: A review of the latest changes applied to ICM Edit cluster should be done before a replication to the Live cluster is triggered. Omitting such a step would basically counteract the core idea of the Edit cluster being part of the data flow. So we basically define two processes: import and replication.

The import process is scheduled while the replication process is triggered manually. A mail can be sent to a defined list recipients whenever the scheduled import part finished successful or did run into any issues. In case of unforeseen problems this would also allow to skip a specific replication or run a partial replication. Of course it is also thinkable to invert the review logic so that a replication is started at a defined point in time after the import as long as no flag has been raised to omit the replication.

## Solr Configuration Update

Solr configuration updates do require the following three basic steps:

1. Transfer and placement of the updated Solr configuration files to according Solr directory within the ICM shared file system
2. Restart of the Solr server process or notification to the respective Solr core in order to re-read the updated configuration files
3. Rebuilding the search index based on the updated configuration known to the server Solr process

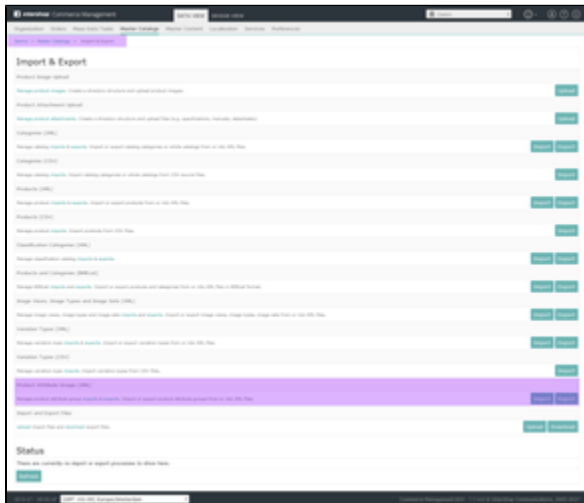
Without step two, the third step would execute a rebuild considering any product updates resulting from the product import. However, it wouldn't consider any potential Solr configuration updates. To address this shortcoming the automation chain - since build [1.0.0-integration-001.001.20170](#) [4071510](#) ensures the re-read of the updated Solr configuration before running the search index rebuild. The re-read is implemented using a notification approach which avoids actual Solr server process restarts.

**IDP-23 - 2.9 Implement Solr filter config update with or without server restart (Solr Configuration Update)**

See also: [DONE](#)

## Product Attribute Groups

Product attribute groups are used to control which custom attributes are shown in the storefront. Before build [1.0.0-integration-001.001.201705111202](#) attribute group updates were based on running manual DBInit steps. Meanwhile an XML based import mechanism has been made available. According import and export tasks can be triggered from the back office:



Imports and exports are available on organization and channel level. Moreover, a product attribute group import task has been added to the process chain based automated import.

**IDP-22 - 2.8 Attribute Group Import**

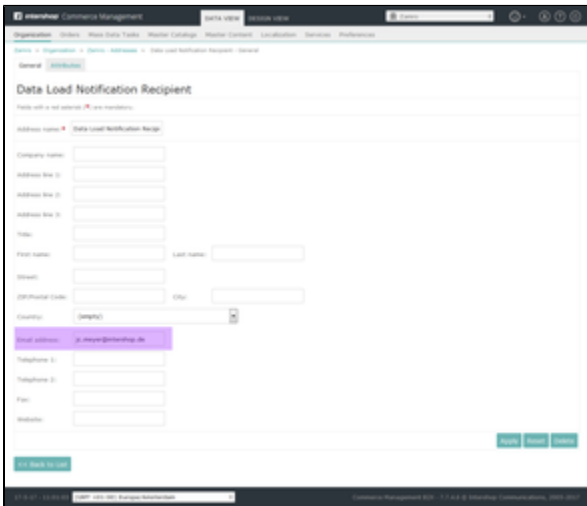
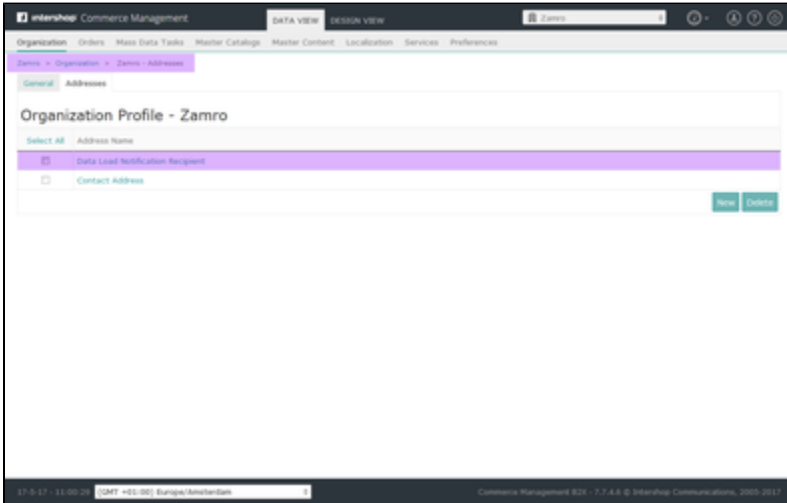
See also: [DONE](#)

## Notification Emails

Starting with build 1.0.0-integration-001.001.201705170916 notifications about important import or replication events will be send via email:

- started
- finished successfully
- finished with errors

To receive notification emails an organization address "Data Load Notification Recipient" holding a valid email address needs to be configured. Without such an address, a warning will be logged and the default "Contact Address" will be used as a fall back.



The solution described above matches a rather minimalist approach. For a more sophisticated solution, detailed requirements should be discussed first.

**IDP-24 - 2.7 Notifications with Email (SUCCESS, FAILURE) (minimal solution**

See also: **DONE**

## Catalog Category Import

As part of the automated data load the catalog categories to be shown will be delivered as a full set to be imported in initial mode. This means no categories should exist in the addressed catalogs when the actual import is starting. We have the following options to support this.

- name all categories to be deleted explicitly within the catalog import files – no custom logic, available out-of-the-box
- implement custom code to clear all categories from all existing catalogs in a separate step before doing the import files based catalog import
- run an actual catalog delta import providing delete entries for categories to be removed as also update entries for categories to be added or updated – would be the best option with regards to performance as only minimal changes are written to `CatalogCategory` and `ProductCategoryAssignment`



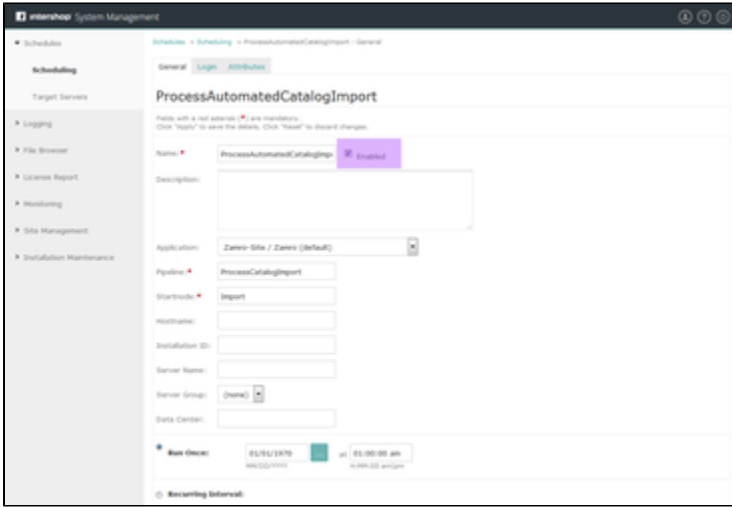
## Tips and Tricks

Sometimes, for testing purposes or error recovery, it might be required to have additional control over the executed process chain tasks. In case only a subset of tasks should actually be executed, three options are available:

- Enable or disable the jobs called through the process chain
- Temporarily disable according tasks in the process chain definition
- Rename import files to match or not match the look-up pattern

### Enable or disable jobs

As an example, if you want to avoid the import of catalog categories while the file share actually provides such import files, you can disable job *ProcessAutomatedCatalogImport* through the System Management Console. A disabled job will not be executed and a warning will be logged. Please don't forget to re-enable the job for regular imports to be executed later.



### Temporarily disable tasks

A kind of hack to avoid execution of a process chain task is to modify the chain definition on file system level. Open `$IS_SHARE/system/config/cluster/automated_import_chain.xml` in an editor and select the task you want to disable. Now either comment out the section you don't want to be executed or simply change the respective import type to an invalid value:

#### Example: Enabled catalog import

```
<p:pipeline
  name="ProcessAutomatedImportCatalog"
  pipeline="ProcessAutomatedImport"
  startnode="Start"
  domain="Zamro">
  <p:parameter name="ImportType">Catalog</p:parameter>
</p:pipeline>
```

### Example: Temporarily disabled catalog import

```
<p:pipeline
  name="ProcessAutomatedImportCatalog"
  pipeline="ProcessAutomatedImport"
  startnode="Start"
  domain="Zamro">
  <p:parameter name="ImportType">DISABLED_Catalog</p:parameter>
</p:pipeline>
```

## Rename import files

All import files not matching the valid file name patterns described in section [File Transfer](#) will be ignored. So if you don't want a specific import file to be imported you can simply rename it accordingly. The same is true for the directory level.

File will be imported:

product\_update.xml

Files will be ignored:

\_IGNORE-ME\_product\_update.xml

foo\_product\_update.xml

Files in this directory will potentially be imported:

Zamro-MasterRepository

Files in this directory will be ignored:

\_IGNORE-ME\_Zamro-MasterRepository

## Incremental Import

### Product Import

This section will describe incremental product import implementation details as needed. For a general introduction please refer to the chapters [Import Modes](#) and [File Transfer](#).



### Price Import

This section will describe incremental product import implementation details as needed. For a general introduction please refer to the chapters [Import Modes](#) and [File Transfer](#).

Updating existing price scale entries of a price list or adding new price scale entries is possible through update mode import. However, deletion of existing entries is [not possible before 7.7.3.0](#) or 7.8.0.1. In all previous versions the import mode attribute is allowed only on global price list level. Therefore an import file can't delete a specific price scale entry or specifically replace the price scale entries of an existing price list entry. For further details see [IS-17412](#). As a workaround it is possible to update any unwanted price to some out-of-range value and implement some special handling based on this.


## Performance and Reliability Improvements

The following items will be addressed as part of the automation development:

- Make price list import part of the data load chain: [ZMR-135](#) 
- Avoid price change event creation and use parallel import bulkers: [ENFINITY-12620](#) 

- Solr filter config update without server restart: [Solr Wiki - CoreAdmin](#) 

The following items list additional considerations for further performance improvement:

- Parallelization of data load steps, e.g. price import or search index rebuild 
- Check actual load on machines: data base vs. application servers
- Data base schema location and network bandwidth for replication

## Effort Estimation

The table below shows a high level effort overview regarding iteration one items taken from the development areas Product Data and Process Automation. Rows highlighted in blue show optional or potentially required effort. It describes net development effort to get the implementation running on a specific environment (acceptance, ..). No deployment to production or project management or sync meeting effort considered here.

Area	Implementation Task	Responsible	Effort in Days
1 Product Data	Restructure Product Data	Zamro	
2 Process Automation	Provide SFTP server for import files download	Zamro	
	Automate Import Data Provision	Zamro	
	Define Process Chain - Import	ISH	2
	Define Process Chain - Replication	ISH	2
	Test Process Chain - Import	ISH	5
	Test Process Chain - Replication	ISH	5
	Test Process Chain - Import	Zamro	
	Test Process Chain - Replication	Zamro	
	Implement Process Chain job configuration (Import, Replication) creation DBMigrate logic	ISH	2
	Implement import file transfer logic	ISH	5
	Implement multiple files import wrapper	ISH	3
	Implement missing connector logic pipelines	ISH	4
3 Process Improvement	Implement additional improvement steps based on insufficient testing results	ISH	5
Total			33

## Guides

### Running Data Loads

#### Import Chain

1. Login to the System Management of the Edit cluster of the environment you want to run the import for. E.g.: [SMC of Integration Edit](#)
2. Navigate to Schedules Scheduling, select the Zamro domain and click the *Apply* button
3. Check the *ProcessAutomatedImportChain* schedule and click the *Run* button
4. Click on the time stamp in the column named *Last Run* to watch run time information

#### Replication Chain

1. Login to the System Management of the Edit cluster of the environment you want to run the replication from. E.g.: [SMC of Integration Edit](#)
2. Navigate to Schedules Scheduling, select the Zamro domain and click the *Apply* button
3. Check the *ProcessAutomatedReplicationChain* schedule and click the *Run* button
4. Click on the time stamp in the column named *Last Run* to watch run time information

### System Management Console

It is essential that these jobs are running in correct domain and application contexts. The screen above was taken on a non-prod environment. For some unknown reason the Enterprise Backoffice application is named *enterprise* on prod environment, instead of *Zamro* as shown here. All downstream automated import jobs – like *ProcessAutomatedProductImport* for example – are generated on the fly if not available or reused when available.

## Open Issues

Issue	Decision	Comment	Responsible	Due Date
How can updated Attribute Groups be imported other than running DBMigrate?	An attribute group import is planned for a later ICM version. Write a custom import or wait for ICM update.		ISH	
Is a server restart mandatory after updating Solr Filters in file system?	According to Solr Wiki one can load a new core from the same configuration as an existing registered core at run-time. To be tested as part of automation development.		ISH	
Does import file transfer happen through intranet or public internet? Secure transfer on network or application layer?	Transfer through public internet – we need secure transfer on application layer.		Zamro	
Zamro Meeting 2016-11-16 in Jena: Data split into several import files. Import all in one is faster. Why are multiple files used then?	Workaround addressing Oracle PGA memory issue.	PGA memory issue patch to be applied.	Zamro	
Once automation of data load is in place, how often will the chain be run?	Full data load will be run as before while update runs will happen a lot more often.		Zamro	

Clarify how Crawling the Site is currently done at Zamro.	Crawling is done through 3rd party crawler started manually from Jenkins environment after replication is finished. We should stick with that approach for first iteration.		Zamro	
Will or should catalogs and catalog categories become part of the automated data load or are these considered static?	The 19 top level catalogs are considered static. Categories are delivered by ZDPT. So they need to be considered in data load.	Oliver to deliver example file for categories update.	Zamro	
Future of ZMR 1070? It proposes process control mastered through ZDPT which is contrary to what we describe here.	We would like to continue with the [ISH only process] on this project as discussed earlier. Please disregard the second PHP integration approach. We will use it internally as it helps us currently.		Zamro	
Decide for which country channels Product Sharing will be applied and which exceptions need to be implemented.	<ul style="list-style-type: none"> <li>Main product set that is common for all channels maintained on organization</li> <li>Specific supplier assortment maintained on channel level</li> </ul>		Zamro	
Deployment of additional job configurations: Created manually in the respective back offices or via custom DBMigrate logic?	<ul style="list-style-type: none"> <li>Test on a single environment first</li> <li>When tested successfully – implement DBMigrate logic</li> </ul>		Zamro	
If specific sharing configuration needs to be defined: Created manually in the respective back offices or via custom DBMigrate logic?			Zamro	
Will we have a dedicated test environment for Incremental Publishing adjustments?	Yes, a dedicated test environment will be set up.	New environment to be requested the same as Acceptance. Intershop to configure.	Zamro, ISH	