Administration Guide

Document Version: 1.7 – 2017-09-29

PUBLIC

Administration Guide - SAP Hybris Commerce, integration package for SAP for Retail 2.2

Content

Change History

Table 1:

Document Version	Publication Date	Change
1.0	2016-11-25	Initial version
1.1	2016-12-02	Article Replication Performance [page 40]
1.1	2016-12-02	Changes in subsection *Quick Check on In-Store Product Stock Level*, in topic yinstorecsfrontendaddon AddOn [page 125]
1.2	2017-01-26	Subsection *Variant-Creating Characteristics* in section *Boundaries* modified, in topic Dependencies, Prerequisites, Boundaries [page 14]
1.3	2017-03-01	Section *Compliance with Global Data Privacy Regulation (GDPR)* added, in topic Security Information [page 129]
1.3	2017-03-01	Note (*Order Management for SAP Hybris Commerce (formerly called Order Management Services or OMS) is not part of the OAA solution*) added, in topic Omnichannel Article Availability and Sourcing [page 57]
1.4	2017-04-25	Section *Support for Different CAR Versions* added, in topic sapoaacommerceservices Extension [page 74]
1.5	2017-06-27	Added note in topic sapoaacommerceservices Extension [page 74], in section *Rough Stock Indicators (RSI)*
1.5	2017-06-27	Removed GDPR section in topic Security Information [page 129]
1.6	2017-08-29	New topic Major Functional Enhancements [page 86] added Section *System Setup* in topic Omnichannel Promotion Pricing [page 84] modified Direct links to PDF deliverables for *SAP Hybris Commerce, integration package for SAP for Retail* inserted
1.6	2017-08-29	Removed section *Rapid Deployment Solution for SAP Retail Integration*, in topic Article Data Management [page 10]
1.7	2017-09-29	Added topic Hybris Data Hub Database Cleanup [page 41]

1 Getting Started with the Integration Package

SAP Hybris Commerce, integration package for SAP for Retail 2.2 is an add-on to SAP Hybris Commerce 6.2. It provides retail-specific enhancements of SAP Hybris Data Hub and SAP Hybris Commerce. These enhancements are deployed and run within SAP Hybris Data Hub or on the SAP Hybris Commerce server. They include extensions for article data management, omnichannel article availability and sourcing, omnichannel promotion pricing, and in-store customer engagement.

SAP Hybris Commerce, integration package for SAP for Retail enhances the generic SAP ERP back-end integration (SAP Back-End Integration: SAP ERP and SAP CRM) with specific functionality for retail articles and thus enables support of the generic integration scenarios for the retail and fashion industry. Specific support for the fashion industry, such as support for seasons or segmentation, is not provided and no fashion industry-specific content is included in the integration package. However, the integration package is released for an SAP Retail back end as well as an SAP Fashion Management back end within the boundaries of the delivered scope.

⚠ Caution

SAP S/4HANA is not supported.

Prerequisites

You have installed either of the following:

- **Full edition** of SAP Hybris Commerce
- **B2C Edge edition** of SAP Hybris Commerce

⚠ Caution

Any other edition of parts of SAP Hybris Commerce is not sufficient for installation of the integration package.

Scope

SAP Hybris Commerce, integration package for SAP for Retail 2.2 provides extensions for the following business capabilities:

- **Article data management (ADM)**
  Article data management provides the replication of article master data from SAP Retail via Data Hub to SAP Hybris Commerce.
  The following data is replicated:
  - Single articles

    - Generic articles including variants
    - Structured articles
    - Classification data based on a merchandise category hierarchy
- **Omnichannel article availability and sourcing (OAA)**
  Omnichannel article availability and sourcing provides retail-specific availability information and sourcing information to your customers. This information is identical and consumable across all sales and communication channels (points of sale, online, mobile, call centers) and provides a consistent customer experience.
  It supports the following e-commerce scenarios:
    - Click-and-ship scenario
    - Click-and-collect scenario

  > Note
  >
  > OAA is based on the integration of the following software components: SAP Retail, SAP Customer Activity Repository, SAP Hybris Commerce, and SAP Hybris Commerce, integration package for SAP for Retail, with SAP Customer Activity Repository being the system of record for OAA functionality. The extensions provided with the integration package serve to enable OAA functionality within SAP Hybris Commerce and SAP Hybris Data Hub. This is why the documentation in this guide covers technical guides of the OAA extensions only. Most of the available OAA documentation is located in other places:
  >
  > - For release notes, see the release notes in the application help of SAP Customer Activity Repository.
  > - For installation information, see the Installation Guide of the respective software application.
  > - For integration information, see section *Configure Omnichannel Article Availability for Use with SAP Customer Activity Repository* in the *Common Installation Guide* of SAP Customer Activity Repository retail applications bundle.
  > - For application help including business configuration and business setup, see the application help of SAP Customer Activity Repository, under *Inventory Visibility* > *Omnichannel Article Availability and Sourcing* > *Business Configuration and Business Setup*.
  > - For a link list of all OAA-relevant documentation, see the application help of SAP Customer Activity Repository, under *Inventory Visibility* > *Omnichannel Article Availability and Sourcing* > *Overview of OAA-Related Documentation*.

- **Omnichannel promotion pricing (OPP)**
  With OPP, SAP provides a solution to ensure correct and consistent effective sales prices across all sales channels along with the ability to introduce new pricing rule types with low implementation effort. It includes the following features:
    - A central price and promotion repository that stores all the relevant information for the calculation of effective sales prices
    - An IDoc outbound to send regular prices and OPP promotions from the central repository to the different sales channel applications
    - A price calculation logic to calculate the effective sales price for a set of products (promotion pricing service)

      > Note
      >
      > The promotion pricing service is subject to a separate license. If you want to use it, you must obtain a separate license. For more information, please contact your account executive.

    - A central and a local deployment option for the promotion pricing service

Note

OPP is based on the integration of the following software components: SAP Customer Activity Repository, SAP Hybris Commerce, and SAP Hybris Commerce, integration package for SAP for Retail, with SAP Customer Activity Repository being the home of the central price and promotion repository. The extension provided with the integration package serves to enable OPP functionality within SAP Hybris Commerce.

Most of the available OPP documentation is located in other places:

- For release notes, see the release notes in the application help of SAP Customer Activity Repository.
- For installation information, see the Installation Guide of the respective software application.
- For integration information, see section *Configure Omnichannel Promotion Pricing with SAP Customer Activity Repository* in the *Common Installation Guide* of SAP Customer Activity Repository retail application bundle.
- For application help including business configuration and business setup, see the application help of SAP Customer Activity Repository, under *Omnichannel Promotion Pricing*.
- For technical information about the promotion pricing service, see the following documents that are delivered as part of the software artifact for SAP Hybris Commerce, integration package for SAP for Retail, in addition to this administrator guide:
  - Development and Extension Guide for Omnichannel Promotion Pricing
  - Client API for Omnichannel Promotion Pricing

- In-store customer engagement (ISCE)
  In-store customer engagement helps store associates perform their tasks when interacting with customers. This includes customer engagement such as customer identification and individual consulting based on personalized offer and product recommendations. This also includes order fulfillment such as handing over goods and invoices to the customer.

Business Cases

ADM and OAA / OPP do not depend on each other. Common business cases are as follows:

- Customers with a focus on using ADM might or might not use OAA or OPP as well.
- Customers with a focus on using OAA or OPP will most probably also use ADM for replicating articles from SAP Retail to SAP Hybris Commerce.

Installation of the Integration Package

1. Download SAP Hybris Commerce, integration package for SAP for Retail from SAP ONE Support Launchpad under *Software Downloads*.
   Under *Installations and Upgrades*, search the alphabetical index for SAP INDUSTRY PACK RETAIL.
2. Unzip the file.
   The integration package installs on top of the SAP Hybris Commerce installation.

Note

The integration package contains several AddOns that need a special setup. Especially if you want an automatic core data import and a self-configuration, an initialization of SAP Hybris Commerce could be necessary. To install and configure the AddOns, follow the instructions in the individual sections of this document.

3. Activate the required commerce extensions. For more information, see Configuring Available Extensions.
   - For article data management:
     - `saparticlemodel`
     - `saparticleb2caddon`
     - `saparticlesearch`
     - `saparticlebackoffice`
   - For omnichannel article availability and sourcing:
     - `sapoaaaddon`
     - `sapoaacommercefacades`
     - `sapoaacommerceservices`
     - `sapoaamodel`
     - `sapoaaorderexchange`
     - `sapoaabackoffice`

     ⚠ Caution

     Make sure that the `sapproductavailability` and `warehousing` extensions are deactivated. These extensions also provide availability information but follow different approaches. If active, they interfere with OAA functionality.
   - For omnichannel promotion pricing:
     - `sapppspricing`
   - For in-store customer engagement:
     - `instorecsbackoffice`
     - `instorecsfacade`
     - `instorecsservice`
     - `sapinstorecsservice`
     - `yinstorecsfrontendaddon`
4. Install the Data Hub extensions:
   Copy the relevant *.jar files from folder `<hybris>/bin/ext-integration/datahub/extensions/sapretail` to folder `${TOMCAT_HOME}/webapps/datahub-webapp/WEB-INF/lib`. These files include:
   - For article data management: `saparticle-<version number>.jar`
   - For omnichannel article availability and sourcing:
     - `sapoaaorder-*-<version number>.jar`
     - `sapoaarsi-*-<version number>.jar`
     - `sapoaasite-*-<version number>.jar`
5. Restart the Tomcat server.

Security-Relevant Information

Since the enhancements provided by the integration package are deployed and run within SAP Hybris Data Hub or on the SAP Hybris Commerce server, you have to follow the security guidelines of the specific module and platform to set up a secure system landscape and to configure and run your applications in a secure way.

For more information, see the following documentation links:

- For Omnichannel Promotion Pricing: Security Information [page 116]
- For In-Store Customer Engagement: Security Information [page 129]
- For SAP Integrations: SAP Integration Security
- For SAP Hybris Data Hub: Data Hub Security
- For SAP Hybris Commerce: Hybris Security and Security and User Management

Please also check the Security Guides of the SAP Solutions that are involved in your scenario setup, which you can find on SAP Service Marketplace at http://service.sap.com/securityguides.

Reporting Customer Incidents

When you report an incident (that is, create an SAP support message) for your installation, you must specify an application component. The components to be used for SAP Hybris Commerce, integration package for SAP for Retail are as follows:

Table 2:

Application Component	Functionality
CEC-COM-BSC-MDX-ART	Article data management
CEC-COM-BSC-OAA	Omnichannel article availability and sourcing
CEC-COM-BSC-PPS	Omnichannel promotion pricing
CEC-COM-BSC-ICE	In-store customer engagement

2 Article Data Management

Article Data Management provides the replication of single articles, generic articles, structured articles, and classification data from SAP Retail via Data Hub to SAP Hybris Commerce.

Article Data Management as part of SAP Hybris Commerce, integration package for SAP for Retail is based on SAP Hybris Commerce 6.2. It provides the replication of article master data from SAP Retail via Data Hub to SAP Hybris Commerce.

The following data is replicated:

- Single articles
- Generic articles including variants
- Structured articles
- Classification data based on a merchandise category hierarchy

Glossary

Table 3: Terms and Definitions

English Term	Definition
article	An article is the smallest unit or customer pack that can be ordered independently, and that cannot be split further into any smaller units. In SAP ERP, *products* are commonly referred to as *materials*, whereas in SAP Retail they are *articles*.
single article	Standard article as sold to the consumer, for example, a packet of chocolate chip cookies.
generic article	A special type of configurable article for structuring the variants of an article (such as a fashion item). A generic article can have both *variant-creating characteristics* and *non-variant-creating characteristics*.
variant	If a number of articles differ only in certain characteristics - such as in color, size, or style - the individual articles are referred to as *variants* and grouped together as a "generic article".
structured article	Group of articles (such as set, prepack, or display) that is handled in certain logistics processes as separate article. You assign the components to the structured article when maintaining the bill of material (BOM). The structured article itself is the BOM header, its components the BOM items. Structured articles can be resolved into their components, for example, upon goods receipt.

English Term	Definition
merchandise category	Every article is assigned to a single merchandise category across a whole company. This allows you to classify and structure the entire range of goods offered by your company (breadth and depth).
merchandise category hierarchy	Merchandise categories can be grouped into hierarchies. Every merchandise category can be assigned to one merchandise category hierarchy level.
classification system	System that is used to describe objects on the basis of *characteristics* by grouping them together into *classes* of objects (that is, by classifying them) according to freely definable criteria. *Classes* and *characteristics* in SAP Retail correspond to *categories* and *features* in SAP Hybris Commerce.
tax category	Classification that the system uses to determine country-specific taxes during pricing.

Related Information

SAP Library for SAP Business Suite on SAP Help Portal:
Article Master Data
Documentation on SAP Hybris Commerce Help
Products and Stock
Replication of Product Data
Synchronizing Data between SAP ERP and SAP Hybris Commerce
Getting Started with SAP ERP Integration

2.1 What's New in Version 2.0

New features in Article Data Management in version 2.0

Version 2.0 of the SAP Hybris Commerce, integration package for SAP for Retail is based on SAP Hybris Commerce 6.0. In the following, you can find detailed information about the new features in Article Data Management:

Table 4:

Feature	Description	More Information
Article discountable	Indicates if an article is discountable in SAP Hybris Commerce	Dependencies, Prerequisites, Boundaries [page 14] saparticlemodel Extension [page 43] Contents of saparticle DH Extension: Single Articles [page 29]
Structured articles	Replication of structured articles from SAP Retail to SAP Hybris Commerce	Replicating Structured Articles from SAP Retail [page 24] Contents of saparticle DH Extension: Structured Articles [page 36]
`saparticlemodel` extension	Enables retail-specific master data and processes	saparticlemodel Extension [page 43]
`saparticleb2caddon` extension	Displays structured article component data	saparticleb2caddon AddOn [page 46]
`saparticlesearch` extension	Enhances the standard SAP Hybris Commerce search functionality	saparticlesearch Extension [page 51]
`saparticlebackoffice` extension	Enables the maintenance of structured articles	saparticlebackoffice Extension [page 52]

2.2 Configuration

You need to download SAP Hybris Commerce, integration package for SAP for Retail from SAP Software Download Center, install the extensions, and configure the data replication.

This document focuses mainly on a setup of the SAP Hybris Commerce, integration package for SAP for Retail 2.2 with your SAP Retail back-end system, based on the asynchronous order management (AOM) from SAP Integrations for SAP Hybris Commerce. The SAP Hybris Commerce, integration package for SAP for Retail consists of several extensions. The `saparticle` Data Hub extension replicates retail article data from the SAP Retail back-end system to SAP Hybris Commerce via Data Hub. These replicated retail articles can then be used to create orders in SAP Hybris Commerce, and are replicated back to the SAP Retail system via the AOM scenario.

The installation of the AOM scenario and SAP Hybris Commerce is described in SAP ERP Integration and System Requirements for SAP Integrations.

The SAP Hybris Commerce, integration package for SAP for Retail is built on **SAP Hybris Commerce 6.2**. It is distributed as a separate package, which is installed after SAP Hybris Commerce and the asynchronous order management (AOM).

Technically, the `saparticle` Data Hub extension is based on the `saperpproduct` Data Hub extensions, which are part of the SAP Hybris solution integration. The `saperpproduct` Data Hub extensions include the following: `saperpproduct-raw`, `saperpproduct-canonical`, and `saperpproduct-target`.

Configuring Data Replication

Check the *Customizing/Configuration of Master Data Replication* section in Getting Started with SAP ERP Integration. As the back-end system, you need SAP ERP 6.0 EHP7 and higher. For the article master data replication, you need the `ARTMAS` IDoc with type `ARTMAS06`. If you have set up the distribution of article master data changes with the Shared Master Data Tool, changes to the master data objects are selected for distribution by change pointers. To manage article master data change pointers, follow the instructions in SAP Note 1966001.

Configuring Structured Articles

With SAP Hybris Commerce, integration package for SAP for Retail 2.2 structured articles such as sales set, prepacks, or displays are supported in SAP Hybris Commerce. This requires specific settings in the SAP Retail back-end system.

Bill of Material in SAP Retail Sales Documents

The explosion of structured articles is not supported in SAP Hybris Commerce. To avoid any collision or mismatch of item numbers between SAP Retail sales documents and the order entries of an order in SAP Hybris Commerce, it is recommended to switch off the bill of material (BOM) explosion for SAP Retail sales documents and to use the processing on the main item level instead. For more information about how to specify the processing level for bill of materials in sales documents, see Bills of Materials in Sales Documents.

To disable the bill of material explosion in SAP Retail sales documents, proceed as follows:

1. In Customizing for SAP Retail, choose *Logistics - General* > *Material Master* > *Retail-Specific Settings* > *Settings for Structured Materials* > *Check Settings Related to Structured Materials*.
2. Choose the activity *Define Item Categories*.
3. Choose the item category that you want to change.
4. In the *Bill of Material/Configuration* section, do the following:
   - Leave the *Structure scope* field empty (*Do not explode material structure*).
   - Enter **`SD01`** (*Sales and distribution*) in the *Application* field.

Configuring Replication of Article Master Data

In SAP Retail, you can create characteristics that should be used for the classification of articles. The characteristic values can be defined separately in the new characteristic value storage.

If you want to use the new characteristic value storage, you need to activate the replication in the Customizing at *Logistics - General* > *Material Master* > *Retail-Specific Settings* > *Settings for New Characteristic Value Storage* > *Activate Distribution Using ALE*.

Related Information

saperpproduct Data Hub Extensions
Data Hub
Change Pointer (Master Data Distribution)
Storing Characteristic Values

2.3 Dependencies, Prerequisites, Boundaries

The `saparticle` Data Hub extension depends on several Data Hub extensions of Hybris. As a prerequisite to article data replication, you need to replicate SAP Retail Customizing data and to configure basic settings for the replication. Not all data can always be replicated.

The `saparticle` Data Hub extension provides functionality for the transfer of retail articles to SAP Hybris Commerce.

The article master data is sent from the SAP Retail system using ALE technology via IDocs. The `HttpInboundService` receives the IDocs, and sends Spring-Integration XML messages for them. The incoming IDoc XML file is routed to the `saparticleARTMASMappingService`, which transforms the IDoc into the raw model. For more information, see sapidocintegration Data Hub Extension.

The composition step creates the canonical items from the raw items. Therefore, the `saparticle` Data Hub extension provides canonical grouping handlers, which filter the relevant raw items for each canonical item type. Additionally, composition handlers are used for the processing of the canonical items.

With the publication step, the canonical items are transformed into the target items, which are used for the generation of the Impex file. The Impex file is sent to the target system SAP Hybris Commerce for the creation of the Hybris product in the corresponding catalogs. To determine the relevant catalog of each sales area assigned to an article, the mapping of the sales area to product catalog is used, which is defined in the configuration data of the `saperpproduct` Data Hub extensions (`CanonicalSalesAreaToCatalogMapping`). For more information, see Making Settings in Backoffice for Product Master Data Replication.

Dependencies

saparticle Data Hub Extension

The `saparticle` Data Hub extension depends on the following Data Hub extensions of Hybris. In the following hierarchy, each extension is dependent on the extension above it:

- `saperpproduct`
  - `sapidocintegration`
    - `sapcoreconfiguration`

For the `saperpproduct` Data Hub extensions, the following applies:

- Material replication is not used since it is not applicable for retail scenarios.
- Replication of the merchandise category hierarchy is required for article replication.
- Replication of stock information is optional.

saparticlemodel Extension

To sell structured articles, you need appropriate stock for the structured article in SAP Hybris Commerce.

In SAP Retail, a bill of material (BOM) explosion for sales documents can be set up, which means that sub-items for the components are added to the sales document when a structured article is added to the document. A BOM explosion is not possible in the Hybris order management. If a BOM explosion for sales documents is customized in SAP Retail, collisions and mismatches of the item numbering can appear in SAP Hybris Commerce, when the Asynchronous Order Management for B2C and B2B Scenarios is used. If the asynchronous order management (AOM) is used, it is recommended to switch off the BOM explosion for sales documents for structured articles that are replicated to SAP Hybris Commerce. For more information, see the *Configuration of Structured Articles* section at Configuration [page 12].

Prerequisites

Replication of SAP Retail Customizing Data to Hybris

Before you can start replicating article data from SAP Retail to Hybris, you need to replicate the necessary customizing data.

The general process is based on Synchronizing Data between SAP ERP and SAP Hybris Commerce, which refers to SAP Note 1983231.

For article data replication, you need to enhance the mentioned report `ZDOWNLOAD_CUSTOMIZING_AS_IMPEX` so that the merchandise categories (classification class type `026`) are replicated as well.

Enhance the select statement with `OR klart = '026'` as follows:

```
...
*Classification Units
    DATA lt_klart TYPE TABLE OF tcla-klart.
    FIELD-SYMBOLS <fs_klart> LIKE LINE OF lt_klart.
    SELECT klart FROM tcla INTO TABLE lt_klart WHERE obtab = 'MARA' AND ( klart =
'001' OR klart = '200' OR klart = '300' OR klart = '026' ).
...
```

Configuring Basic Settings for Data Replication in SAP Retail

The general processing is described in section Configuring Basic Settings for Data Replication.

For the replication of article data, you need to configure specific settings in SAP Retail. This includes outbound and inbound parameters for partner profiles, and specific message types for the distribution model.

To create partner profiles, the following outbound parameters are necessary:

Table 5: Partner Profiles: Outbound Parameters

Message Type	Basic Type	Description
`ARTMAS`	`ARTMAS06`	Distribute articles
`BOMMAT`	`BOMMAT04`	Distribute structured articles (BOM)
`CHRMAS`	`CHRMAS05`	Distribute characteristics
`CLFMAS`	`CLFMAS02`	Distribute master object classification
`CLSMAS`	`CLSMAS04`	Distribute classes with document links

To set up the distribution model, the following message types are necessary:

Table 6: Set Up Distribution Model

Message Type/BAPI	Type	Description
`BOMMAT`	IDoc	Distribute structured articles (BOM)
`CHRMAS`	IDoc	Distribute characteristics
`CLFMAS`	IDoc	Distribute master object classification
`CLSMAS`	IDoc	Distribute classes with document links
`LOISTD`	IDoc	Distribute stocks
`RetailMaterial.Clone`	BAPI	Distribute articles

Note

IDoc `LOISTD` only needs to be maintained if you want to replicate stock information. For the replication of prices and conditions, IDoc `COND_A` needs to be maintained.

If you also want to replicate pricing data, refer to section Prices, Discounts, Bonus Buys (extension `sappricing`).

Order of Replication

Ensure that the merchandise category hierarchy (transaction `WGSE`) is replicated before replicating article data.

Variant Categories and Variant Value Categories in SAP Hybris Commerce Accelerators

The SAP Retail generic articles are represented in SAP Hybris Commerce by multi-dimensional products. The multi-dimensional products use `VariantCategories` and `VariantValueCategories` to store generic article variant information. During the replication process, the `VariantCategories` and `VariantValueCategories` are replicated into catalog `ERP_CLASSIFICATION_026`.

Thus, the catalog `ERP_CLASSIFICATION_026` needs to be assigned to the base store in the Backoffice Administration Cockpit. The catalog version `ERP_IMPORT` needs to be set active. For more information, see saparticleb2caddon AddOn [page 46].

Boundaries

Due to the different data models and technical differences between SAP Retail and SAP Hybris Commerce, not all data can always be replicated. Potential functionality that falls into such a category is listed in the following paragraphs. Note that this does not represent a complete list of all use cases that are not supported.

Data Types of Classification Data

The SAP ERP attribute types `CHAR` (text) and `NUM` (numeric) are supported with single- and multi-value attributes for the SAP ERP-Hybris integration of classification data. For the attribute type `NUM`, intervals as a value are not supported.

Variant-Creating Characteristics

Variant-creating characteristics defined in characteristic profiles are supported. Until version 2.2, patch level 2.2.0.2, other alternatives relating to the definition of variant-creating characteristics, especially direct assignment of variant-creating characteristics to merchandise category hierarchy levels or merchandise categories, were not supported. Thus, any inheritance of variant-creating characteristics was not supported either. With patch level 2.2.0.3, these boundaries have been resolved. For more information, see SAP Note 2407029.

Until version 2.2, patch level 2.2.0.2, the assignment of variant-creating characteristics via a characteristic profile to a single article was not supported. With patch level 2.2.0.3, this boundary has been resolved. For more information, see SAP Note 2362922.

Only variant-creating characteristics of attribute type `CHAR` and `NUM` are supported. If SAP Note 2269467 is implemented, intervals as a value are also supported for the attribute type NUM.

Discount Allowed

In the point of sales (POS) data of an SAP Retail article it can be specified if a discount for an article is allowed. This can be done for an appropriate distribution chain (sales organization and distribution channel) or a plant in a distribution chain. For SAP Hybris Commerce, only POS data entries on the distribution chain level are evaluated. All entries on plant level are ignored. The discount allowed value is overtaken to the discountable attribute of a Hybris product that is part of a product catalog, which is mapped to the SAP Retail distribution chain.

2.4 Replication of Article Master Data from SAP Retail

Replication of article master data requires uploading SAP configuration data to the Data Hub.

The following documents explain the procedures for replicating article master data, the corresponding classification structure, and its values, from SAP Retail to SAP Hybris Commerce.

Note

In SAP ERP, products are commonly referred to as *materials*, whereas in SAP Retail they are *articles*. Furthermore, *classes* and *characteristics* in SAP Retail correspond to *categories* and *features* in SAP Hybris Commerce.

Prerequisites

You have imported SAP configuration data to the Data Hub. Proceed as follows in the Backoffice Administration Cockpit:

1. In the navigation area, choose *SAP Integration* > *SAP Administration Configuration*.
2. Click *Start Upload*.

2.4.1 Replicating the Classification Structure from SAP Retail

Use transaction `WGSE` to replicate classification structures.

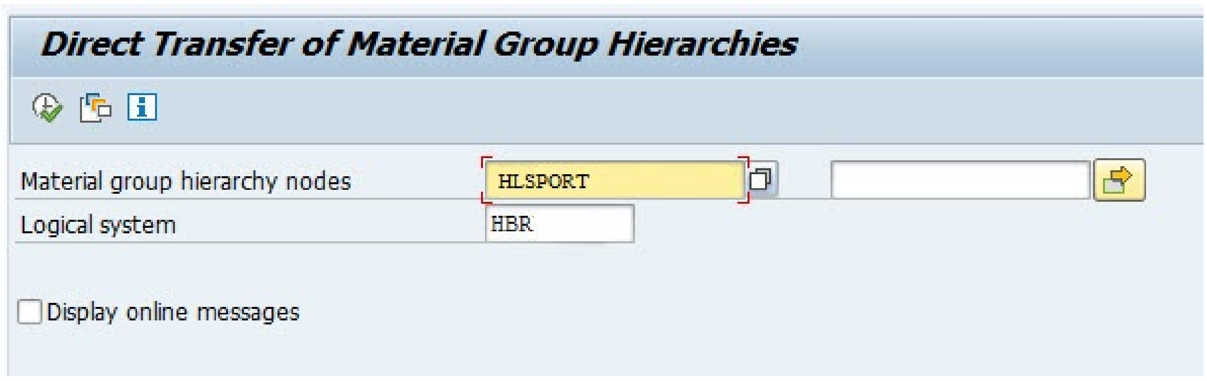
Context

In SAP Retail, the classification is implemented by using material groups (also known as merchandise categories), which are grouped in a material group hierarchy tree (also known as a merchandise category hierarchy), by assigning them to particular material group hierarchy nodes. This tree represents a classification structure that needs to be replicated to SAP Hybris Commerce as a classification catalog.

To replicate this classification structure, the node on the highest level of the material group hierarchy tree needs to be used in transaction `WGSE`:

Procedure

1. Launch *Direct Transfer of Material Group Hierarchies* (transaction `WGSE`).
2. Enter the highest node as the material group hierarchy, which will represent the Hybris classification catalog root.
3. Enter the logical system.
4. Execute the transaction.


Direct Transfer of Material Group Hierarchies

Material group hierarchy nodes	HLSPORT	
Logical system	HBR	

☐ Display online messages

5. To check the replication results in the target Hybris system, launch the Backoffice Administration Cockpit and choose *Catalog* > *Classification Systems* > *Classifying Categories*.

2.4.1.1 Understanding the Material Group Used

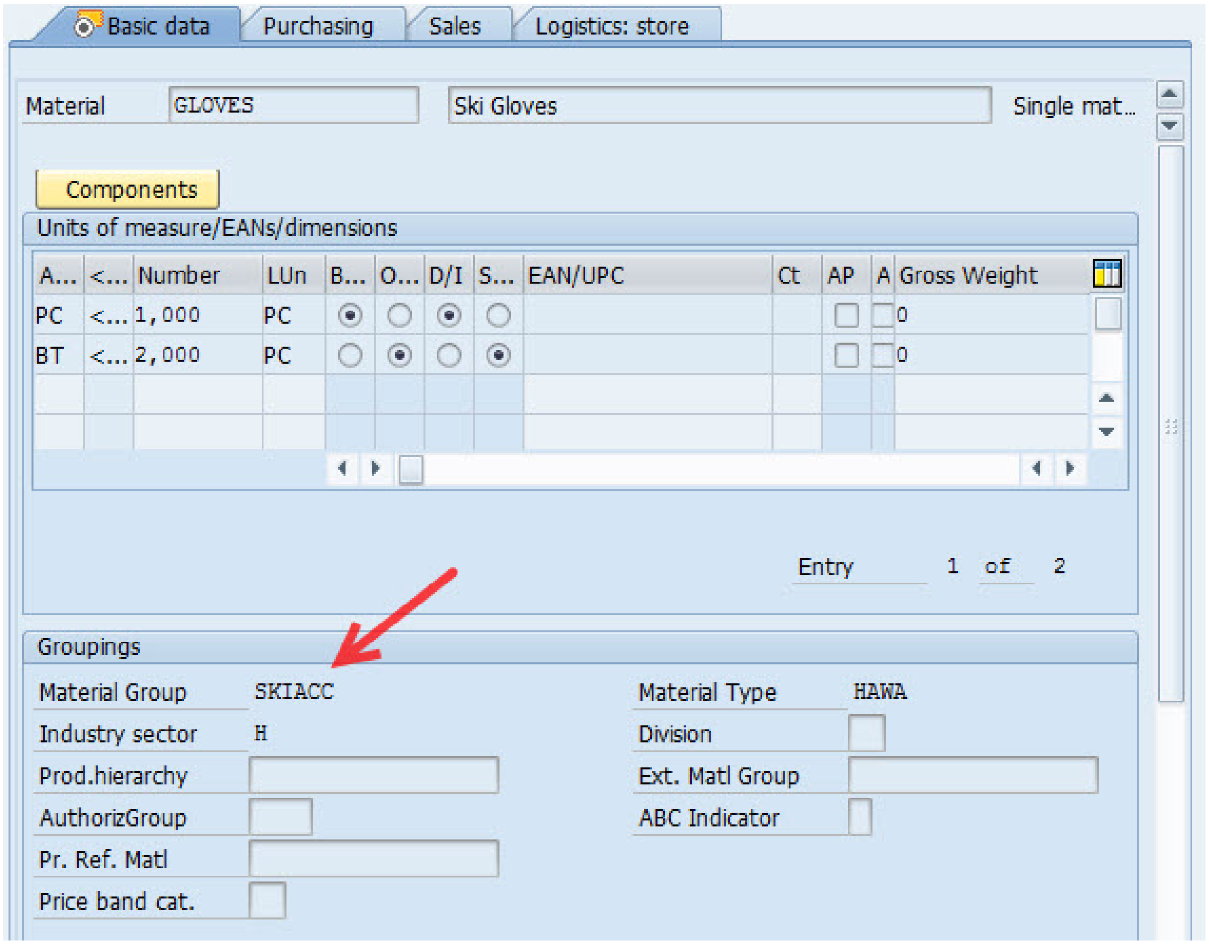
Use transaction `MM43` to find the material group assigned to a material, and transaction `CL6C` to display the corresponding class hierarchy.

Context

To get a better understanding of the material group used, and its assignment to the merchandise category hierarchy, do the following in SAP Retail:

Procedure

1. Launch *Display Material* (transaction `MM43`).
2. Choose a retail article for which you want to know the assigned material group.



Material	GLOVES	Ski Gloves

Single mat...

Units of measure/EANs/dimensions

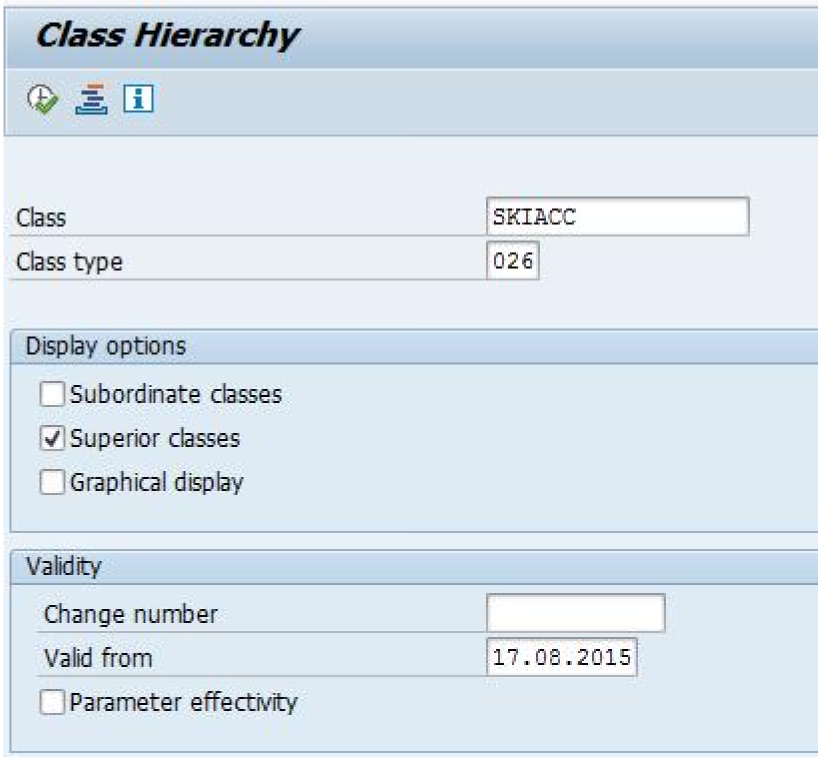
A...	<...	Number	LUn	B...	O...	D/I	S...	EAN/UPC	Ct	AP	A	Gross Weight
PC	<...	1,000	PC	●	○	●	○			☐	☐	0
BT	<...	2,000	PC	○	●	○	●			☐	☐	0

Entry 1 of 2

Groupings

Material Group	SKIACC	Material Type	HAWA
Industry sector	H	Division	
Prod.hierarchy		Ext. Matl Group	
AuthorizGroup		ABC Indicator	
Pr. Ref. Matl			
Price band cat.			

3. Launch *Class Hierarchy* (transaction `CL6C`).
4. Enter the material group value as the class, value **`026`** as the class type, and select *Superior classes*.



Class	SKIACC
Class type	026

Display options

☐ Subordinate classes

☑ Superior classes

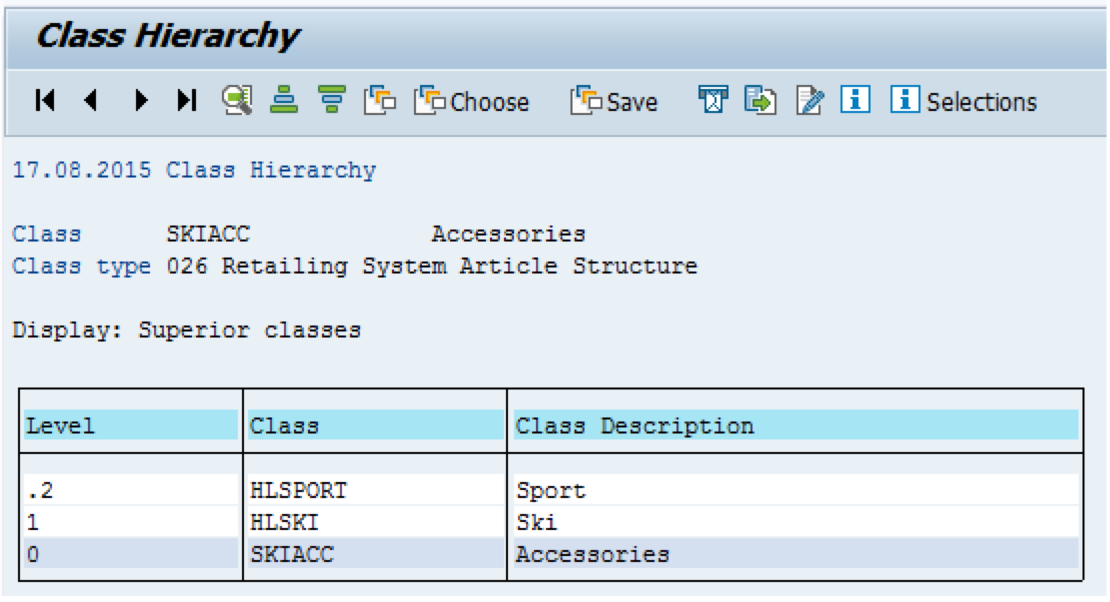
☐ Graphical display

Validity

Change number	
Valid from	17.08.2015

☐ Parameter effectivity

5. After executing the transaction, you get the corresponding tree structure. The highest level number represents the highest node in the tree.



```
17.08.2015 Class Hierarchy

Class      SKIACC              Accessories
Class type 026 Retailing System Article Structure

Display: Superior classes
```

Level	Class	Class Description
.2	HLSPORT	Sport
1	HLSKI	Ski
0	SKIACC	Accessories

2.4.2 Replicating Single and Generic Articles from SAP Retail

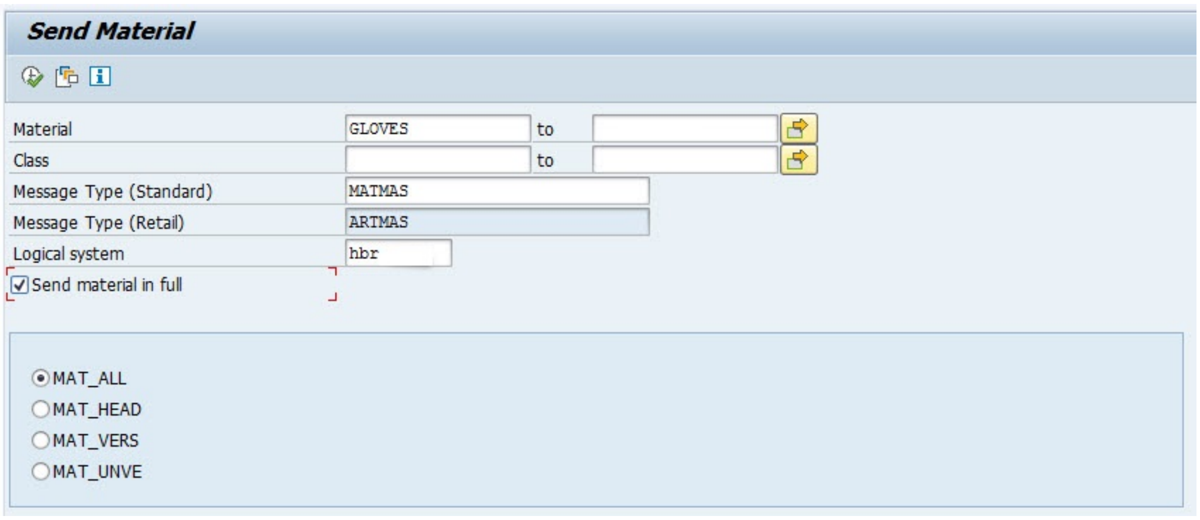
Use transaction `BD10` to replicate single and generic articles.

Context

To replicate single and generic articles, including maintained characteristic values, do the following in SAP Retail:

Procedure

1. Launch *Send Material* (transaction `BD10`).
2. Enter the material and the logical system.
3. Select *Send material in full*.
4. Execute the transaction.



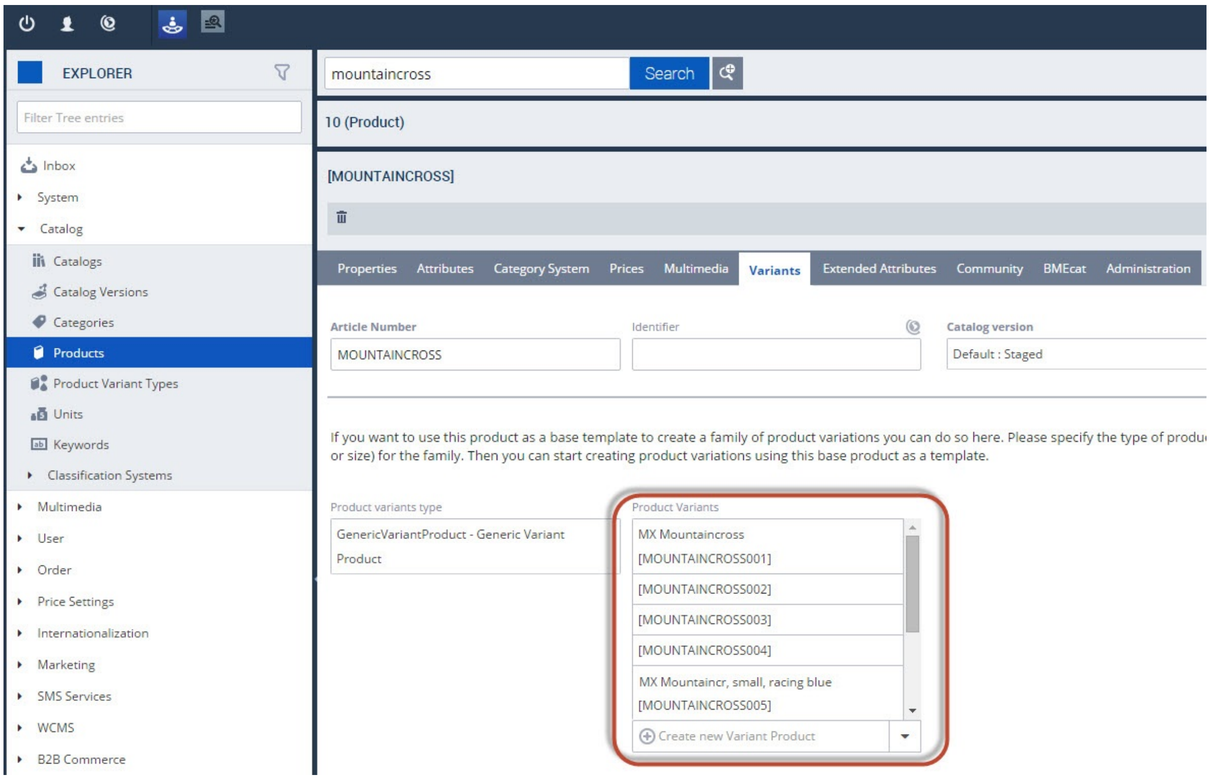
Material	GLOVES	to	
Class		to	
Message Type (Standard)	MATMAS		
Message Type (Retail)	ARTMAS		
Logical system	hbr		

[✓] Send material in full

● MAT_ALL
○ MAT_HEAD
○ MAT_VERS
○ MAT_UNVE

5. Verify the replication results in the target Hybris system in the Backoffice Administration Cockpit, by navigating to *Catalog* > *Products*.

   The replicated characteristic values can be found on the *Attributes* tab. The generic article variants are replicated using the concept of B2B Multi-Dimensional Products in Hybris. These variants can be found on the *Variants* tab.



mountaincross

10 (Product)

[MOUNTAINCROSS]

Article Number	Identifier	Catalog version
MOUNTAINCROSS		Default : Staged

If you want to use this product as a base template to create a family of product variations you can do so here. Please specify the type of produ
or size) for the family. Then you can start creating product variations using this base product as a template.

Product variants type: GenericVariantProduct - Generic Variant Product

Product Variants:
- MX Mountaincross [MOUNTAINCROSS001]
- [MOUNTAINCROSS002]
- [MOUNTAINCROSS003]
- [MOUNTAINCROSS004]
- MX Mountaincr, small, racing blue [MOUNTAINCROSS005]

If you want to use the standard B2B or B2C accelerators logic for the display of multi-dimensional products, you could encounter problems regarding missing variant information on the product detail page. For a possible solution, see Dependencies, Prerequisites, Boundaries [page 14].

2.4.2.1 Understanding the Maintained Variants

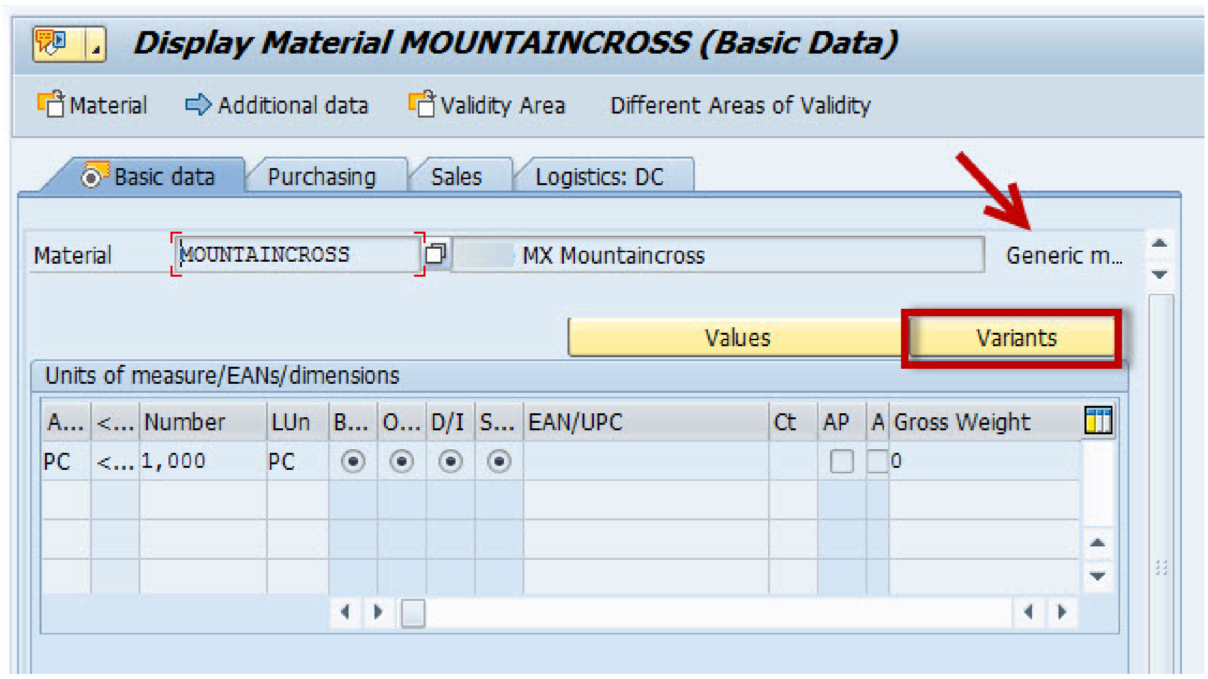
Use transaction `MM43` to display the variants of a generic article.

Context

To get a better understanding of the maintained variants in SAP Retail, do the following:

Procedure

1. Launch *Display Material* (transaction `MM43`).
2. Click *Variants*.


Display Material MOUNTAINCROSS (Basic Data)
Material
Additional data
Validity Area
Different Areas of Validity
Basic data
Purchasing
Sales
Logistics: DC
Values
Variants
Units of measure/EANs/dimensions

Material MOUNTAINCROSS MX Mountaincross Generic m...

A...	<...	Number	LUn	B...	O...	D/I	S...	EAN/UPC	Ct	AP	A	Gross Weight
PC	<...	1,000	PC	●	●	●	●			☐	☐	0

2.4.3 Replicating Structured Articles from SAP Retail

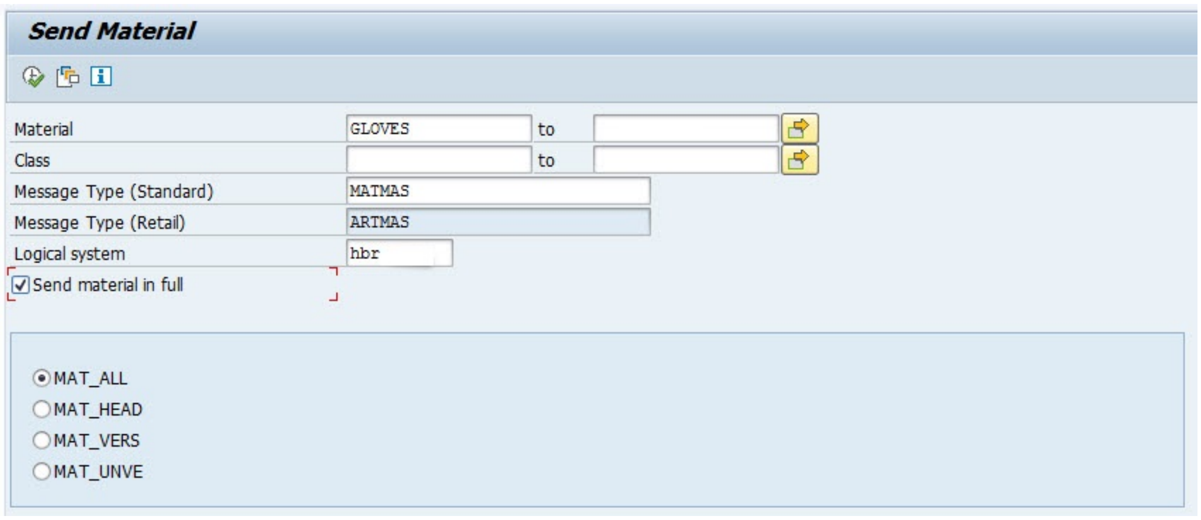
Use transaction `BD10` to replicate structured articles.

Context

To replicate structured articles, including their components, do the following in SAP Retail:

Procedure

1. Launch *Send Material* (transaction `BD10`).
2. Enter the material and the logical system.
3. Select *Send material in full* to send the component list data as well.
4. Execute the transaction.



Send Material

Material	GLOVES	to	
Class		to	
Message Type (Standard)	MATMAS		
Message Type (Retail)	ARTMAS		
Logical system	hbr		

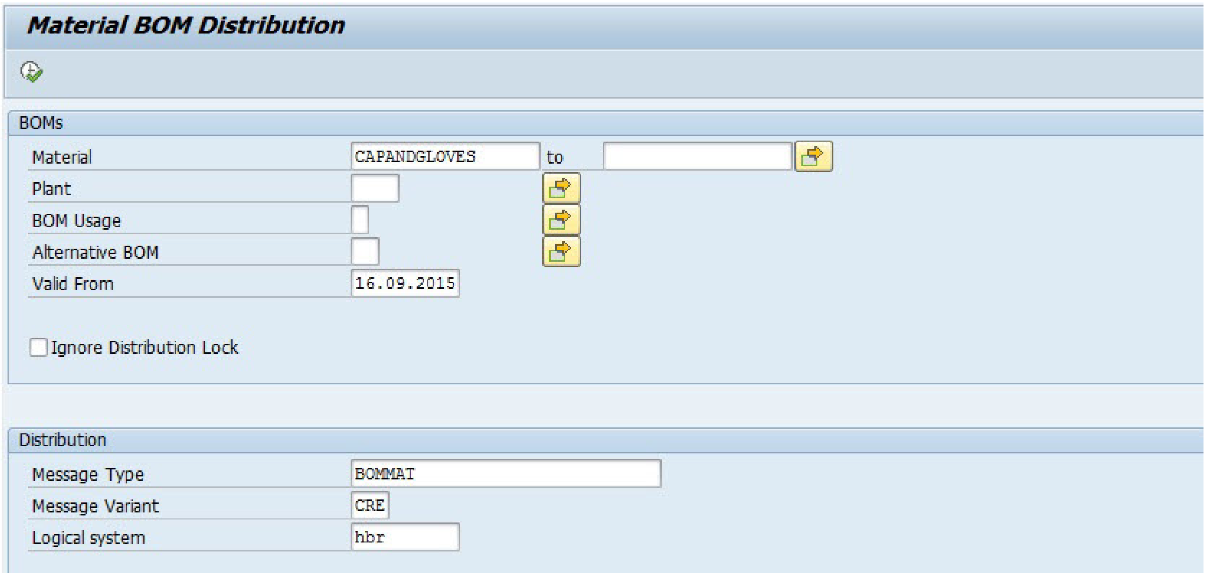
☑ Send material in full

● MAT_ALL
○ MAT_HEAD
○ MAT_VERS
○ MAT_UNVE

> **Note**
>
> It is recommended to replicate the relevant component articles in advance or to enter them in step 2 as well. If you do not replicate article components (of type `CanonicalArticleComponent`), the status `FAILURE(INCOMPLETE)` remains on the Data Hub. To resolve the error, you must send the corresponding article component `ARTMAS` IDoc (transaction `BD10`). For more information about article components, see Contents of saparticle DH Extension: Structured Articles [page 36].

To replicate the component list only, do the following in SAP Retail:

5. Launch *Material BOM Distribution* (transaction `BD30`).
6. Enter the material and the logical system.
7. Execute the transaction.



Material BOM Distribution

BOMs

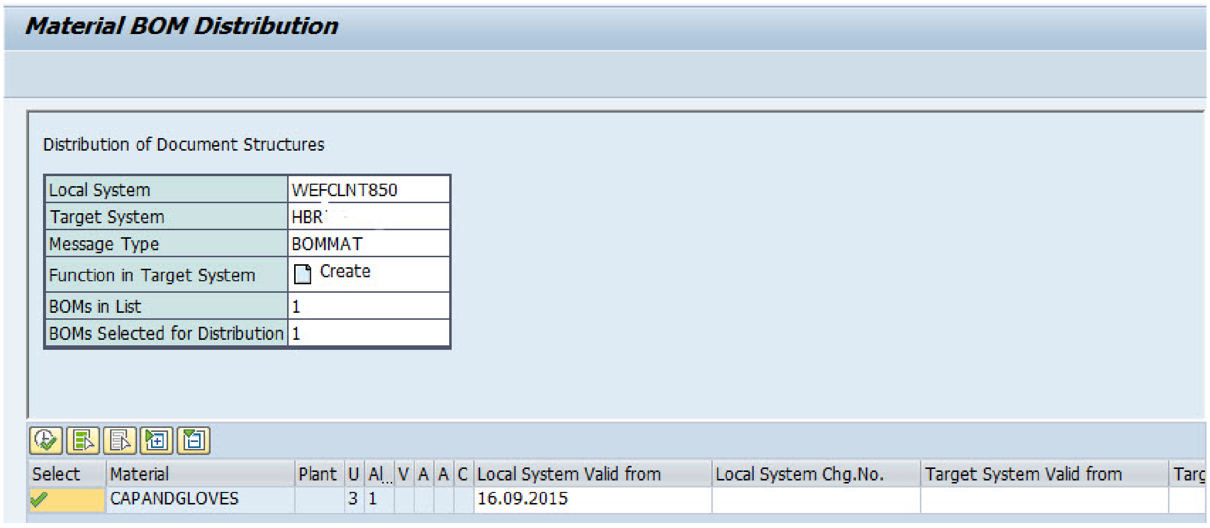
Material	CAPANDGLOVES	to	
Plant			
BOM Usage			
Alternative BOM			
Valid From	16.09.2015		

☐ Ignore Distribution Lock

Distribution

Message Type	BOMMAT
Message Variant	CRE
Logical system	hbr

8. On the next screen, check the entries that you want to replicate.
9. Distribute your selection.



Material BOM Distribution

Distribution of Document Structures

Local System	WEFCLNT850
Target System	HBR
Message Type	BOMMAT
Function in Target System	Create
BOMs in List	1
BOMs Selected for Distribution	1

Select	Material	Plant	U	Al..	V	A	A	C	Local System Valid from	Local System Chg.No.	Target System Valid from	Targ
✓	CAPANDGLOVES		3	1					16.09.2015			

Note

The structured article and its components must be replicated before.

To enable selling of structured articles in Hybris storefront applications, you must replicate stock data to SAP Hybris Commerce by using report `RMCPAMRP` in SAP Retail. For more information about the replication of stock, see Products and Stock.

The replication results can be found in the Backoffice Administration Cockpit, by navigating to *Catalog* > *Products*.

2.4.3.1 Understanding the Maintained Components

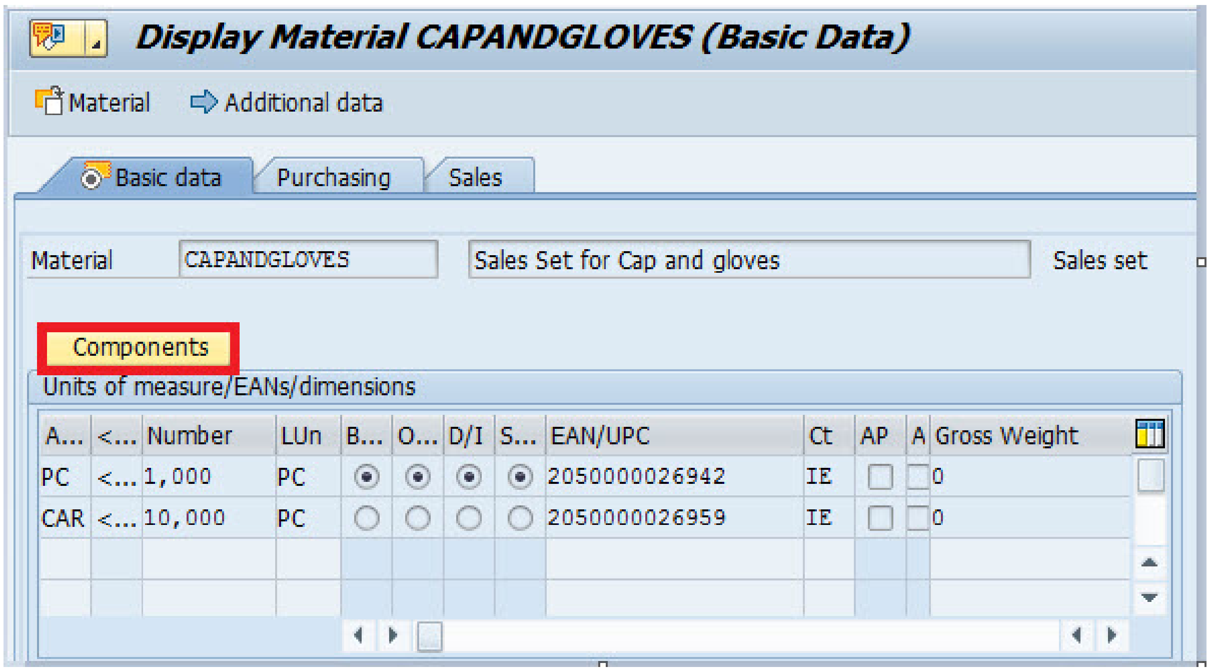
Use transaction `MM43` to display the components of a structured article.

Context

To get a better understanding of the maintained components in SAP Retail, do the following:

Procedure

1. Launch *Display Material* (transaction `MM43`).
2. Enter the structured article and click *Components* in the *Basic Data* screen.


Display Material CAPANDGLOVES (Basic Data)
Material
Additional data
Basic data
Purchasing
Sales
Components
Units of measure/EANs/dimensions

Material	CAPANDGLOVES	Sales Set for Cap and gloves	Sales set								

A...	<...	Number	LUn	B...	O...	D/I	S...	EAN/UPC	Ct	AP	A	Gross Weight
PC	<...	1,000	PC	●	●	●	●	2050000026942	IE	☐	☐	0
CAR	<...	10,000	PC	○	○	○	○	2050000026959	IE	☐	☐	0

2.5 saparticle Data Hub Extension

The `saparticle.properties` file contains default values for the configuration of the `saparticle` Data Hub extension. The article data is transferred via the `ARTMAS` IDoc. The IDoc mapping service for the article is defined in the `saparticle-datahub-extension-spring.xml` file.

The `saparticle` Data Hub extension enables the processing of SAP Retail IDocs that are used for replicating SAP Retail article master data to SAP Hybris Commerce via the Data Hub.

Configuration of the saparticle Extension

The following configuration settings can be made:

- Definition of the default catalog (`saparticle.default.catalog` / `saparticle.default.catalog.version`) for articles that are not sales-area-specific
- Definition of the classification catalog for the merchandise categories (`saparticle.merchandise.classification.catalog` / `saparticle.merchandise.classification.catalog.version`) containing the characteristic profiles
- Definition of the datapool for the saparticle data (`saparticle.pool`)
- Definition of the pagesize for reading canonical items without integration key (`saparticle.canonicalitems.read.pagesize`)

The following default values are defined in the `saparticle.properties` file in the `WEB-INF/classes` directory of the `saparticle` extension.

```
# Default catalog for non sales area specific articles
saparticle.default.catalog=Default
saparticle.default.catalog.version=Staged
# Datahub pools
saparticle.pool=GLOBAL

# Classification catalog for merchandise categories
saparticle.merchandise.classification.catalog=ERP_CLASSIFICATION_026
saparticle.merchandise.classification.catalog.version=ERP_IMPORT
# Pagesize for reading canonical items without integration key
saparticle.canonicalitems.read.pagesize=10000
# Default approval status. Possible values are check, approved, unapproved, <ignore>
# <ignore> is set as default so that an initial product gets the approval status
check for SAP hybris Commerce.
# The approval status can be changed for SAP hybris Commerce and will not be
changed by an ERP import.
# Any other constant value will change the approval status in every ERP import.
saparticle.default.approvalstatus=<ignore>
```

Contents of the saparticle Extension

The following article types from SAP Retail can be transferred to SAP Hybris Commerce:

- Single article: standard article
- Generic article and variants: Variants are articles that differ only in certain characteristics such as color, size, or flavor. They are grouped together as a generic article.
- Structured article

For the replication of generic articles and variants, it is necessary to transfer the corresponding characteristic profiles and characteristics as well.

Note

Please be aware that the following described classes may be changed in future releases. Especially enhancements of handlers and method resolvers need to be adjusted after applying new releases.

IDoc Mapping Service

The article data is transferred via the `ARTMAS` IDoc. The IDoc mapping service for the article is defined in the `saparticle-datahub-extension-spring.xml` file.

Source Code

```
<!-- IDOC Mapping service definition -->
<bean id="saparticleARTMASMappingService"
    class="com.hybris.datahub.sapidocintegration.IDOCMappingService">
   <property name="rawFragmentDataExtensionSource" value="saparticle" />
   <property name="rawFragmentDataType" value="RawARTMAS" />
</bean>
```

The component list of a structured article is transferred via the `BOMMAT` IDoc. The IDoc mapping service for the BOMMAT is defined in the `saparticle-datahub-extension-spring.xml` file.

Source Code

```
<bean id="saparticleBOMMATMappingService"
class="com.hybris.datahub.sapidocintegration.IDOCMappingService">
    <property name="rawFragmentDataExtensionSource"
value="sapstructuredarticle" />
    <property name="rawFragmentDataType" value="RawBOMMAT" />
</bean>
```

The `saparticle-datahub-extension-spring.xml` file contains the definition of the needed spring integration input channel.

Source Code

```
<int:header-value-router input-channel="idocXmlInboundChannel"
header-name="IDOCTYP">
    <int:mapping value="BOMMAT04" channel="BOMMAT" />
</int:header-value-router>
<int:service-activator input-channel="BOMMAT"
    output-channel="rawFragmentDataInputChannel"
ref="saparticleBOMMATMappingService"
    method="map" />
```

2.5.1 Contents of saparticle DH Extension: Single Articles

Model, composition, and publication of single articles

Model

The `saparticle` model is defined in the `saparticle-datahub-exension.xml` file.

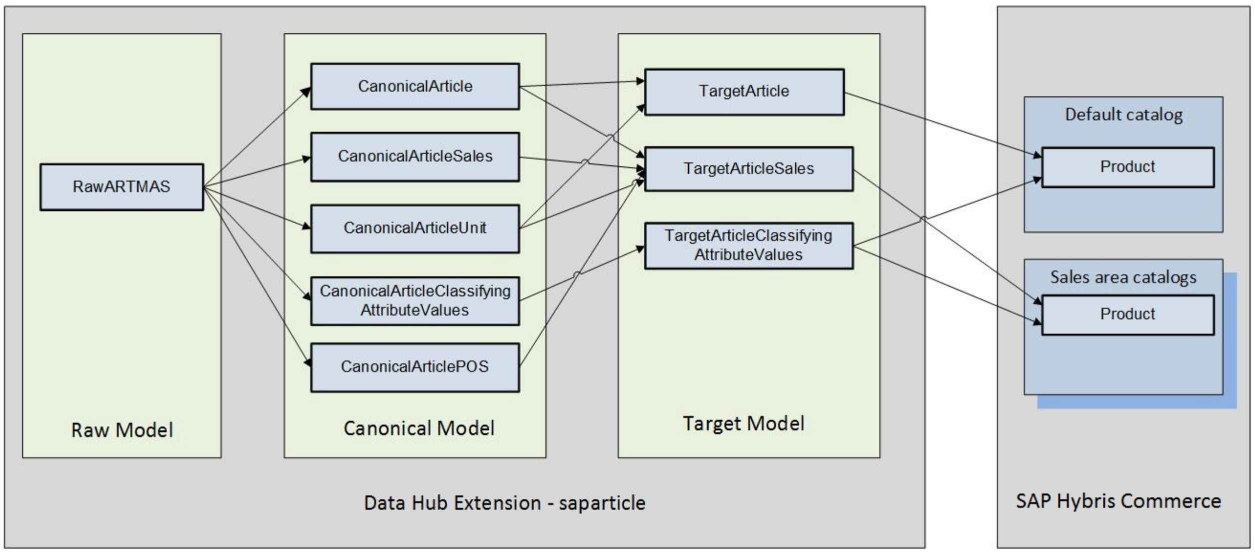


Table 7: Model of Single Articles

Item Type	Item	Description
Raw Items	`RawARTMAS`	Raw article representation of an article for the import from SAP Retail into SAP Hybris Commerce.
Canonical Items	`CanonicalArticle`	This item represents an article in the Data Hub.
	`CanonicalArticleSales`	This item represents the sales-area-specific data of an article in the Data Hub.
	`CanonicalArticleUnit`	This item represents the available units of measurement and assigned `GTIN/UPC` of an article in the Data Hub.
	`CanonicalArticleClassifyingAttributeValues`	This item represents the values of the classifying attributes of an article in the Data Hub.
	`CanonicalArticleTaxClassification`	This item represents the tax classification of an article in the Data Hub.
	`CanonicalArticlePOS`	This item represents point of sales data of an article in the Data Hub. Especially the discount allowed attribute is handled by this item.
Target System	`HybrisCore`	The target system is used to publish the product data to SAP Hybris Commerce.
Target Items	`TargetArticle`	This item represents the article data for the replication to the default catalog.
	`TargetArticleSales`	This item represents the article data for the replication to the sales catalog.
	`TargetArticleClassifyingAttributeValues`	This item represents the values of the classifying attributes.

Composition

Table 8: Composition of Single Articles

Type of Handler	Handler	Description
Grouping Handlers	`FilterKeysGroupingHandler`	This handler of the `saperpproduct` extensions is used to filter out all raw items unless it has all defined raw item attributes set.
	`TraceAllGroupingHandler`	This handler is used to trace the state of the composition group for the requested canonical item.

Type of Handler	Handler	Description
Composition Handlers	`DeleteCompositionHandler`	The handler of the `saperpproduct` extensions is used to delete the values of a classifying attribute for the canonical item `CanonicalArticleClassifyingAttributeValues`. This is required, because otherwise the new attribute values will be merged with existing values.
	`LocalizedCompositionHandler`	The handler of the `saperpproduct` extensions is used to determine the language-dependent descriptions for the canonical item `CanonicalArticle`.

Publication

Table 9: Publication of Single Articles

Type of Handler	Handler	Description
Grouping Handlers	`ArticleAttributeValueSalesPublicationGroupingHandler`	This handler selects all relevant sales areas from the configuration of the sales area to catalog mapping. For each article of the canonical item `CanonicalArticleClassifyingAttributeValues`, the grouping handler checks whether the article is relevant for the configured sales area, and clones this entry for the relevant sales area and catalog version. The grouping handler returns the modified group as input for the composition of the `CanonicalArticleClassifyingAttributeValues` item.
Determination Handlers and Method Resolvers	`DetermineArticleSalesEAN`	The handler determines the `EAN/GTIN` for the sales unit of the sales area article. The available units of measurement of an article and the assigned `EANs/GTINs` are stored in the canonical item `CanonicalArticleUnit`.
	`DetermineVariantCategoryAndCategoryName`	This handler determines the name of the supercategory for the classifying attributes, which is located in the ERP classification catalog and will be assigned to the product. The supercategory name will be composed of the merchandise group of the Retail article and the corresponding classification catalog/version.
	`DetermineAttributeValues`	This handler determines the values of the classifying attributes. The classifying attribute values, which are assigned as fixed values, will be enhanced by the attribute name. For other classifying attributes that have no fixed values assigned to them, the values will be transferred without a prefix. The determination of the fixed values of a classifying attribute is processed in the canonical item `CanonicalAttribute` of the `saperpproduct` extensions.

Type of Handler	Handler	Description
	`DetermineCatalogExportCode`	This handler determines the export code for the classifying attribute values in the target item `TargetArticleClassifyingAttributeValues`. The export code for the attribute of the hybris product depends on the identifier for the target catalog that is transferred (sales catalog: PK / default catalog: catalog ID and version).
	`DetermineUnitOfMeasure`	This handler checks if the unit of measure of a given canonical item exists within the SAP Hybris Commerce. It determines if a corresponding entry in `CanonicalHybrisUnit` is available. If this entry does not exist, the unit of measure of the target item is set empty and an appropriate log message is created.
	`DetermineTaxClassification`	This handler determines the tax category and tax classification of an article. For each country a tax category can be defined. The tax country is defined in the canonical item `CanonicalProductSalesAreaToCatalogMapping`. The tax classification for each article and country is retrieved from the canonical item `CanonicalArticleTaxClassification` and will be assigned to the product as a product tax class. The name of the product tax class is returned as a concatenated string of country, category, and classification. Example: `de_MWST_1`.

Conversion of Numeric Attribute Values

The attribute values for datatype NUM (numeric) are transferred in the `ARTMAS` IDoc as a localized string concatenated with the unit of measurement, for example, `1.25 mm`. To extract the number value from this string, the locale (decimal point format) used for the creation of the IDoc is required. The locale for IDoc input is defined in the file `saparticle-datahub-extension-spring.xmlsaparticle` extension as follows:

```
<bean id="sapArticleNumericAttributeFormatConverter"
class="com.sap.retail.datahub.saparticle.tools.NumericAttributeFormatConverter">
        <property name="IDocInputLocale" value="en" />
</bean>
```

For the output of numbers, the decimal point (U.S. format) is used by default. If another output format is required, the class <bean id="sapArticleNumericAttributeFormatConverter" class="com.sap.retail.datahub.saparticle.tools.NumericAttributeFormatConverter"> <property name="IDocInputLocale" value="en" /> </bean> in the WEB-INF/classes directory of the `NumericAttributeFormatConverter` of the `saparticle` in the WEB-INF/classes directory of the extension can be replaced by an appropriate implementation for the bean `sapArticleNumericAttributeFormatConverter` in the `sap-article-datahub-extension-spring.xml`.

2.5.2 Contents of saparticle DH Extension: Generic Articles

Model, composition, and publication of generic articles

Model

Generic articles and their variants data are transferred by `ARTMAS` IDocs. One IDoc contains one generic article and its variants.

The concept of B2B Multi-Dimensional Products will be used for target item types in SAP Hybris Commerce.

The model is defined in the `saparticle-datahub-exension.xml` file.

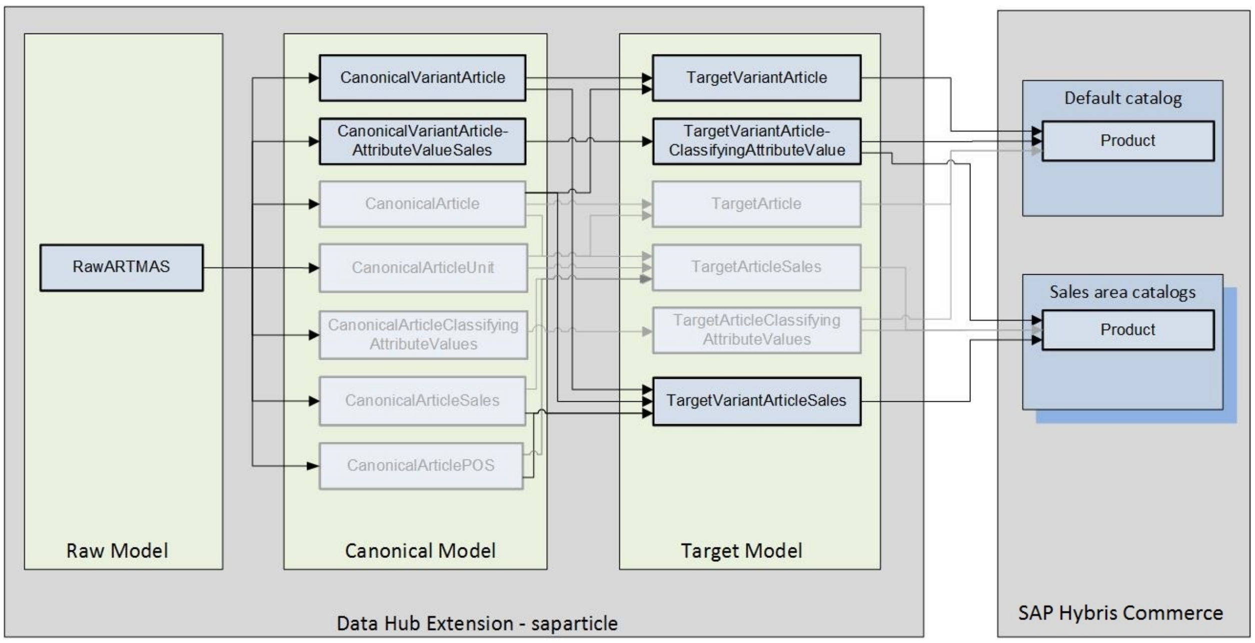


Table 10: Model of Generic Articles

Item Type	Item	Description
Raw Items	`RawARTMAS`	Raw article representation of an article for the import from SAP Retail into SAP Hybris Commerce.
Canonical Items	`CanonicalVariantArticle`	This item represents a variant and its relation to the generic article in the Data Hub.
	`CanonicalArticleVariantAttributes`	This item represents all characteristics that are transferred for an article or variant in the Data Hub.
	`CanonicalVariantArticleAttributeValueSales`	This item represents the values for each characteristic that is transferred for an article or variant in the Data Hub.

Item Type	Item	Description
Target System	`HybrisCore`	The target system is used to publish the product data to SAP Hybris Commerce.
Target Items	`TargetVariantArticle`	This item represents the variant data for the replication to the default catalog.
	`TargetVariantArticleSales`	This item represents the variant data for the replication to the sales catalog.
	`TargetVariantArticleClassifyingAttributeValues`	This item represents the values of the classifying attributes of a variant.

Composition

Table 11: Composition of Generic Articles

Type of Handler	Handler	Description
Grouping Handlers	`VariantArticleAttributeValuesGroupingHandler`	This handler removes all entries for the generic article from the table of `CanonicalVariantArticleAttributeValueSales`. Thus only characteristics and values for variants remain.
	`ArticleClassifyingAttributeValuesGroupingHandler`	This handler removes all entries for variants from the table of `CanonicalArticleClassifyingAttributeValueSales`. Thus only characteristics and values for single or generic articles remain.
	`GenericArticleFilterGroupingHandler`	This grouping handler filters out the invalid entries of the `rawARTMAS` items for generic articles. Therefore, the variant-relevant attributes of each generic article are evaluated. If the assigned characteristic profile contains variant-relevant characteristics, which allows to assign numeric ranges as values, it is required that the version 004 of segment `E1BPE1AUSPRT` of the `ARTMAS` IDoc is available. The enhancements are described in SAP Note 2269467.

Publication

Table 12: Publication of Generic Articles

Type of Handler	Handler	Description
Grouping Handlers	`VariantArticleAttributeValueSalesPublicationGroupingHandler`	This handler replicates classification data from the default catalog to all relevant sales catalogs.
	`VariantArticlePublicationGroupingHandler`	The canonical items `CanonicalArticle` and `CanonicalArticleSales` contain data for the generic articles, and their variants. This handler determines the correct target `TargetArticle`, `TargetArticleSales` or `TargetVariantArticleSales` for these canonical items.
Determination Handlers and Method Resolvers	`DetermineVariantArticleEAN`	This handler determines the `EAN/GTIN` for the variant articles by reading it from the corresponding `CanonicalArticle`. The available units of measurement of an article and the assigned `EANs/GTINs` are stored in the canonical item `CanonicalArticleUnit`.
	`DetermineVariantCategoryValues`	This handler determines the attributes that are relevant for variants, and sets the values for the variants.
	`DetermineVariantCategoryValuesForVariantArticle`	This handler determines the attributes that are relevant for variants, and sets the values for the variants. This is done for items `TargetVariantArticle` and `TargetVariantArticleSales`.
	`DetermineVariantCategoryandCategoryName`	This handler is used in the targets `TargetArticle` and `TargetArticleSales`. It determines the variant categories of the given generic article for the assignment, as supercategories. The supercategory name will be composed of the merchandise group of the Retail article, the parent variant category, and the corresponding classification catalog.
	`DetermineUnitOfMeasure`	This handler checks if the unit of measure of a given canonical item exists within SAP Hybris Commerce determining if a corresponding entry in `CanonicalHybrisUnit` is available. If it does not exist, the unit of measure of the target item is set empty and an appropriate log message is created.

2.5.3 Contents of saparticle DH Extension: Structured Articles

Model, composition, and publication of structured articles

The structured article and its component articles are replicated via the `ARTMAS` IDoc such as single articles. Additionally, the component list (bill of material) is replicated via the `BOMMAT` IDoc. For this replication, the following Data Hub model applies:

Model

The structured article component list model is defined in the `saparticle-datahub-extension.xml` file.

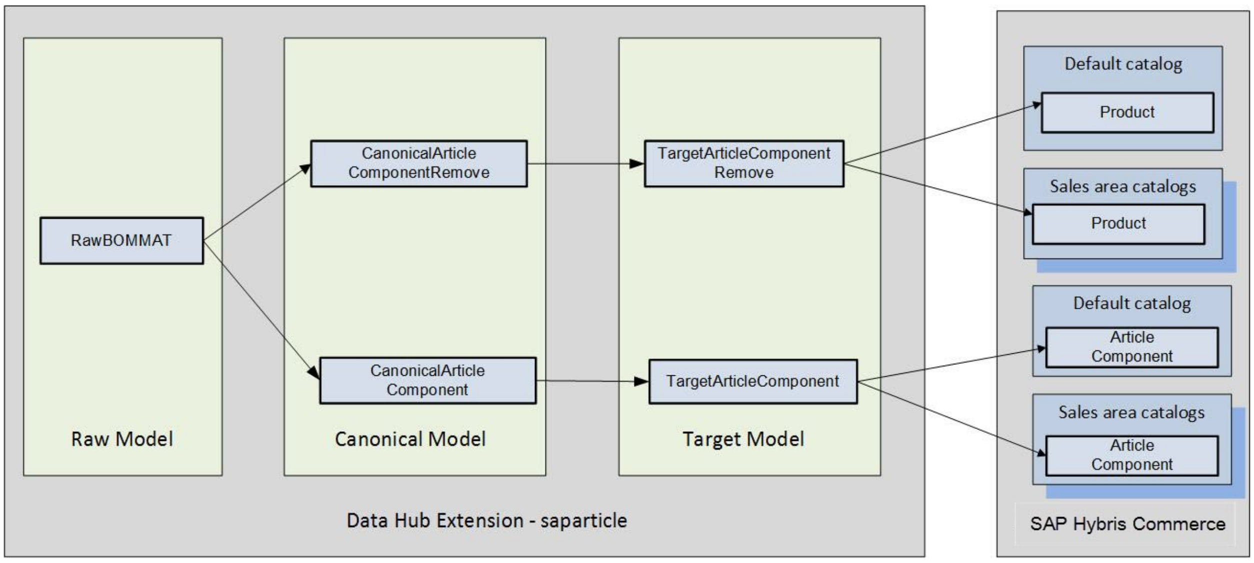


Table 13: Model of Structured Articles

Item Type	Item	Description
Raw Items	`RawBOMMAT`	Raw representation of the component list of a structured article for the import from SAP Retail into SAP Hybris Commerce.
Canonical Items	`CanonicalArticleComponent`	This item represents a component list entry in the Data Hub.
	`CanonicalArticleComponentRemove`	This item represents a structured article for which a component list is replicated and any existing components must be removed, before the new component list can be overtaken, if the article already exists in SAP Hybris Commerce.
Target System	`HybrisCore`	The target system is used to publish the product data to SAP Hybris Commerce.
Target Items	`TargetArticleComponent`	This item represents the structured article component list data for the replication to the relevant catalogs.

Item Type	Item	Description
	`TargetArticleComponentRemove`	This item represents the structured articles for which new or changed component data replication to the sales catalog exists. For these articles the article component information must be cleared before the new component list can be overtaken.

Publication

Table 14: Publication of Structured Articles

Type of Handler	Handler	Description
Grouping Handlers	`ArticleComponentSalesPublicationGroupingHandler`	This handler checks if already the corresponding structured article and component articles that are part of a `CanonicalArticleComponent` item are transferred to SAP Hybris Commerce. If not, a warning message is added to the Data Hub log. If the articles are available, the handler selects all relevant sales areas from the configuration of the sales area to catalog mapping. For each entry of the canonical item `CanonicalArticleComponent`, the grouping handler checks whether the structured article and the component article are relevant for the configured sales area and clones this entry for the relevant sales area and catalog version. The grouping handler returns the modified group as input for the composition of the `CanonicalArticleComponent` item to create `TargetArticleComponent` items.
	`ArticleComponentRemoveSalesPublicationGroupingHandler`	This handler checks if already the corresponding structured article that is part of a `CanonicalArticleComponentRemove` item is transferred to SAP Hybris Commerce. If not, a warning message is added to the Data Hub log. If the structured article is available, the handler selects all relevant sales areas from the configuration of the sales area to catalog mapping. For each entry of the canonical item `CanonicalArticleComponentRemove`, the grouping handler checks whether the structured article is relevant for the configured sales area and clones this entry for the relevant sales area and catalog version. The grouping handler returns the modified group as input for the composition of the `CanonicalArticleComponentRemove` item to create `TargetArticleComponentRemove` items. A `TargetArticleComponentRemove` item removes existing component entries from a structured article in SAP Hybris Commerce to take over the current structured article component list.

2.5.4 Contents of saparticle DH Extension: Characteristic Profiles and Variant-Creating Characteristics

Model, composition, and publication of characteristic profiles and variant-creating characteristics

Model

The characteristic profile contains variant-creating characteristics, amongst others. These are transferred as variant categories to SAP Hybris Commerce. Their corresponding values are replicated as variant value categories.

The profile itself is replicated as root classifying category (not part of the merchandise category hierarchy) and contains only classifying characteristics (features) and no variant-creating characteristics.

Classification data is transferred via `CHRMAS`, `CLSMAS`, and `CLFMAS` IDocs.

Note

The article classification data is replicated via the `ARTMAS` IDoc. The `CLFMAS` IDoc is only used for the replication of assignments of characteristic profiles, merchandise category hierarchies, and merchandise categories.

The replication of merchandise category hierarchies is part of the `saperpproduct` extensions. In the following sections, only the enhancements of the `saparticle` extension that are needed to create the required `VariantCategories` with its `VariantValueCategories` are described. These are the basis for replicating the generic articles as B2B Multi-Dimensional Products.

The model is defined in the `saparticle-datahub-exension.xml` file.

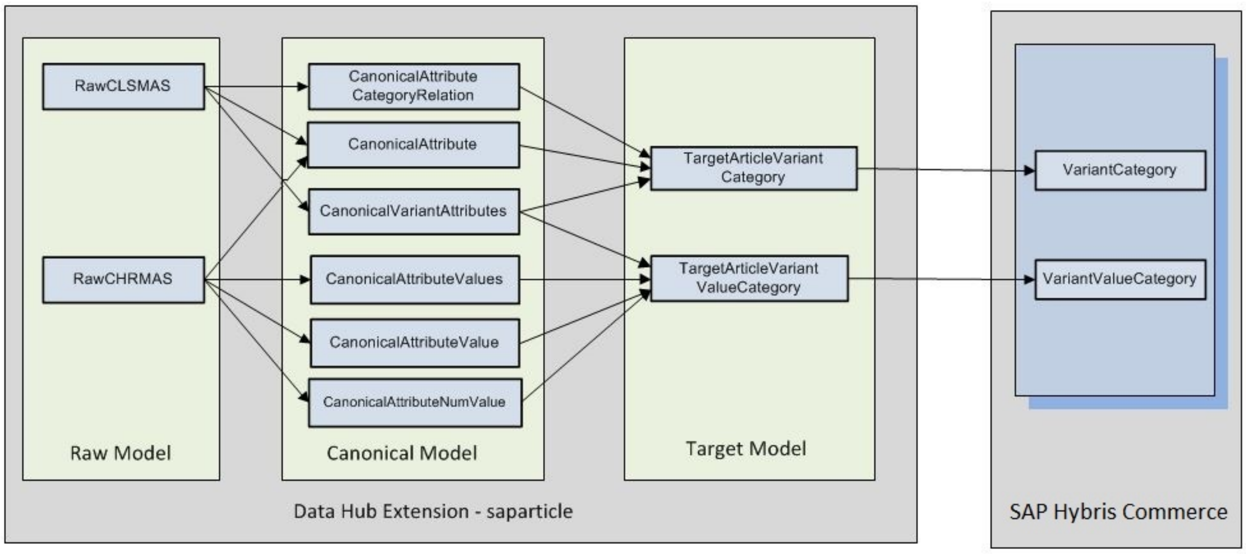

RawCLSMAS
RawCHRMAS
Raw Model
CanonicalAttribute CategoryRelation
CanonicalAttribute
CanonicalVariantAttributes
CanonicalAttributeValues
CanonicalAttributeValue
CanonicalAttributeNumValue
Canonical Model
TargetArticleVariant Category
TargetArticleVariant ValueCategory
Target Model
Data Hub Extension - saparticle
VariantCategory
VariantValueCategory
SAP Hybris Commerce

Table 15: Model of Variant Categories and Variant Value Categories

Item Type	Item	Description
Raw Items	`RawCLSMAS`	Raw classification representation of a classification for the import from SAP Retail into SAP Hybris Commerce (part of `saperpproduct` extensions).
	`RawCHRMAS`	Raw characteristic representation of a characteristic for the import from SAP Retail into SAP Hybris Commerce (part of `saperpproduct` extensions).
Canonical Items	`CanonicalVariantAttributes`	This item represents the variant-relevant attributes per category.
	`CanonicalAttributeCategoryRelation`	This item represents classifying attributes including the variant-relevant information.
	`CanonicalAttributeValue`	This item represents the values of the classifying attributes.
	`CanonicalAttributeValues`	This item represents all values per classifying attribute.
	`CanonicalAttributeNumValue`	This item represents the values of the attributes with numeric data type.
Target Items	`TargetArticleVariantCategory`	This item represents the variant-relevant classifying attributes.
	`TargetArticleVariantValueCategory`	This item represents the values of the variant-relevant classifying attributes.

Composition

Table 16: Composition of Variant Categories and Variant Value Categories

Type of Handler	Handler	Description
Grouping Handlers	`AttributeGroupingHandler`	This composition handler determines the values of attributes for the canonical item `CanonicalAttribute`.
	`AttributeValueGroupingHandler`	This composition handler determines the values of attributes for the canonical item `CanonicalAttributeValue`.
	`VariantAttributeGroupingHandler`	This composition handler ensures that only variant-relevant attributes are passed for the canonical item `CanonicalVariantAttributes`.

Publication

Table 17: Publication of Variant Categories and Variant Value Categories

Type of Handler	Handler	Description
Grouping Handlers	`VariantAttributePublicationGroupingHandler`	This handler creates a `Variant Category` for every attribute ID of `CanonicalVariantAttributes`. During this process, it also creates the parent-child relationships between `VariantCategories` and the child relationship between `VariantCategory` and `VariantValueCategory` values.
	`VariantAttributeValuePublicationGroupingHandler`	This handler processes all variant attribute IDs passed as `CanonicalVariantAttributes` and creates a `VariantValueCategory` for every value of each attribute using `CanonicalAttributeValues`.
	`CanonicalCategoryPublicationGroupingHandler`	This publication grouping handler removes the parent information for characteristic profiles.
	`ClassAttributeAssignmentPublicationGroupingHandler`	This publication grouping handler prevents variant-relevant classifying attributes to be added to characteristic profiles.

2.5.5 Article Replication Performance

Use transactions `WE30` and `BD56` to optimize the data volume of an IDoc.

Required Segments

As described in the Best Practices Guide for Performance, it is possible to optimize the data volume of an IDoc by sending appropriate data from SAP Retail to the Data Hub. In the default delivery, only the following segments are required:

- `E1BPE1AUSPRT`
- `E1BPE1MAKTRT`
- `E1BPE1MARART`
- `E1BPE1MARMRT`
- `E1BPE1MATHEAD`
- `E1BPE1MVKERT`
- `E1BPE1VARKEY`

You can check the segment structure of IDoc `ARTMAS06` with *Develop IDoc Types* (transaction `WE30`).

Note

For performance reasons, it is strongly recommended to transfer only processed segments to the Data Hub. This can be done by using the transaction `BD56` in *Change View "Segment Filters"*.

Using Filters

In *Display/Change Distribution Model* (transaction `BD64`) you can create filters. In these filters, you can define the data that should be replicated with the IDoc. The other data is filtered.

Example

Sales data and sales-dependent data (MVKE) of each article are only created for the distribution chain *R100/R1*. The other distribution chains are filtered. Proceed as follows:

1. Start *Display/Change Distribution Model* (transaction `BD64`).
2. Create a filter group for the sales data of object `RetailMaterial` and method `Clone`.
3. Create filter values for distribution channel *R1* and sales organization *R100*.

For more information about filtering IDocs, see 595027.

2.5.6 Hybris Data Hub Database Cleanup

You need to exclude the canonical items mandatory for article replication, from the data hub database cleanup.

If the data hub database cleanup (see Activating Data Hub Database Cleanup) is activated in SAP Hybris Commerce, it will delete all your successfully published canonical items. However, during article replication, the canonical items of classification and configuration are required for validation checks and customization. Thus, they need to be excluded from the data hub database cleanup.

To exclude the canonical items mandatory for article replication from the data hub database cleanup, configure the `datahub.cleanup.publisheditems.excluded.types` parameter as follows:

Source Code

```
datahub.cleanup.publisheditems.excluded.types=
SAPBaseStoreConfiguration,SAPBaseStoreConfigurationMapping,SAPGlobalConfiguration,
SAPHTTPDestination,SAPIDocTargetSystem,SAPLogicalSystem,CanonicalAttribute,Canonic
alAttributeCategoryRelation,CanonicalAttributeNumRangeValue,CanonicalAttributeNumV
alue,CanonicalAttributeValue,CanonicalAttributeValues,CanonicalCategory,CanonicalV
ariantAttributes
```

2.5.7 Migration from saperpproduct to saparticle DH Extension

You must migrate the article replication field enhancements from the `saperpproduct` Data Hub extensions to the `saparticle` Data Hub extension.

In SAP Hybris Commerce 5.6 and former releases, the `sapproduct` Data Hub extension provides basic support for article replication as well. As of SAP Hybris Commerce 5.7, this support is not available anymore in the `saperpproduct` Data Hub extensions.

Starting with SAP Hybris Commerce, integration package for SAP for Retail, this functionality will be continued in new extensions that will be added to this package. This document describes how to migrate the article replication field enhancements from the `saperpproduct` Data Hub extensions to the `saparticle` Data Hub extension.

Migration from Version 1.0 to 1.1

Field Enhancements: Raw Items

The raw item `RawARTMAS` stays as it is, but it contains more fields.

Field Enhancements: Canonical Items

You need to move the field enhancements from:

- `CanonicalBaseProduct` to `CanonicalArticle`
- `CanonicalSalesProduct` to `CanonicalSalesArticle`

Field Enhancements: Target Items

You need to move the field enhancements from:

- `BaseProduct` to `TargetArticle/TargetVariantArticle`
- `SalesProduct` to `TargetSalesArticle/TargetVariantArticleSales`

Migration from Version 1.1 to 2.0

Validation Check for Generic Articles

The validation checks for generic articles regarding the supported data types (NUM and CHAR) of variant-creating characteristics have been moved from the publication phase to the composition phase. Therefore, it must be ensured that there are no open canonical items in the Data Hub database before the upgrade from the SAP Hybris Commerce, integration package for SAP for Retail 1.1 to 2.0.

2.6 saparticlemodel Extension

The `saparticlemodel` extension contains data models and components to enable retail-specific master data and processes.

The `saparticlemodel` extension contains the data model and service layer components for the item type `ArticleComponents` and enhancements of the product in SAP Hybris Commerce to enable the modeling of structured articles and to enable the specification if an article is discountable or not.

Discountable Attribute

The discountable article attribute indicates if discounts for an article are possible. The following values are available:

Table 18: Discountable Attribute Values

Value	Description
true	Article discount is possible
false	Article discount is not possible
n/a	Article discountable information is not maintained

Article Components

Article components enable you to create articles such as displays, prepacks, or sales sets (also known as structured articles). These articles consist of a list of sub-articles or components. You can use article components to specify how many units of a specific component are availabe in the component list of a structured article.

Structured Articles

The individual type of a structured article can be specified by the structured article type. The following article type values are available:

Table 19: Structured Article Types

Article Type	Description
Display	Group of single articles or variant articles of one or more base articles, which is purchased as a separate article and as such has an article number and price. The components are grouped together by the manufacturer or vendor. In the wholesale or B2B trade, displays are purchased and sold, whereas in the re-tail trade (B2C), only the components are sold.

Article Type	Description
Sales Set	Group of single articles or variant articles of one or more base articles,which is sold as a separate article and as such has an article number and price. It is normally assembled at the re-tailer.
Prepack	Group of variant articles of one or more base articles, which is purchased as a separate article and as such has an article number and a price. These articles can be grouped together by the retailer as well as by the manufacturer or vendor. In the wholesale trade, prepacks are purchased and sold, whereas in the retail trade, only the components are sold.

The article components of a structured article are listed in the attribute **Components**. It can be used to create and maintain the component list of the structured article.

Article Component Attributes

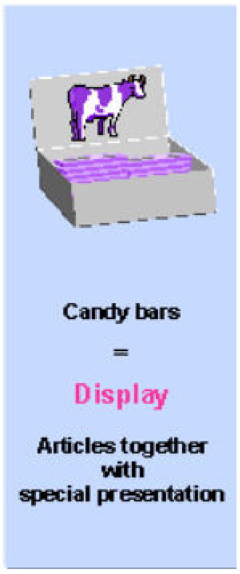
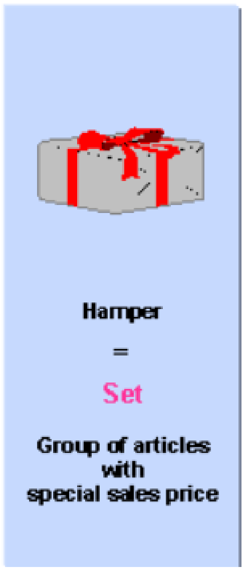
An article component contains the following attributes:

Table 20: Article Component Attributes

Attribute Name	Type	Description
structuredarticle	Product	The structured article itself, for example, a display of chocolate bars
component	Product	The component article, for example, milk chocolate
unit	Unit	The unit of measure in which a compo-nent exists in the component list, for ex-ample, a bar or piece
quantity	Integer	The quantity of the component units ex-isting in the component list, for example, 3 bars

Structured Article Examples

Table 21: Structured Article Examples

Structured Article Type	Description
 Candy bars = Display Articles together with special presentation	A display of chocolate bars containing: • 1 bar of milk chocolate • 2 bars of dark chocolate • 3 bars of white chocolate
 Hamper = Set Group of articles with special sales price	A hamper (set) of Italian food containing: • 1 bottle of wine • 2 packs spagetthi • 1 bottle olive oil • 1 bottle balsamico • 1 piece of parmesan • 1 piece of parma ham

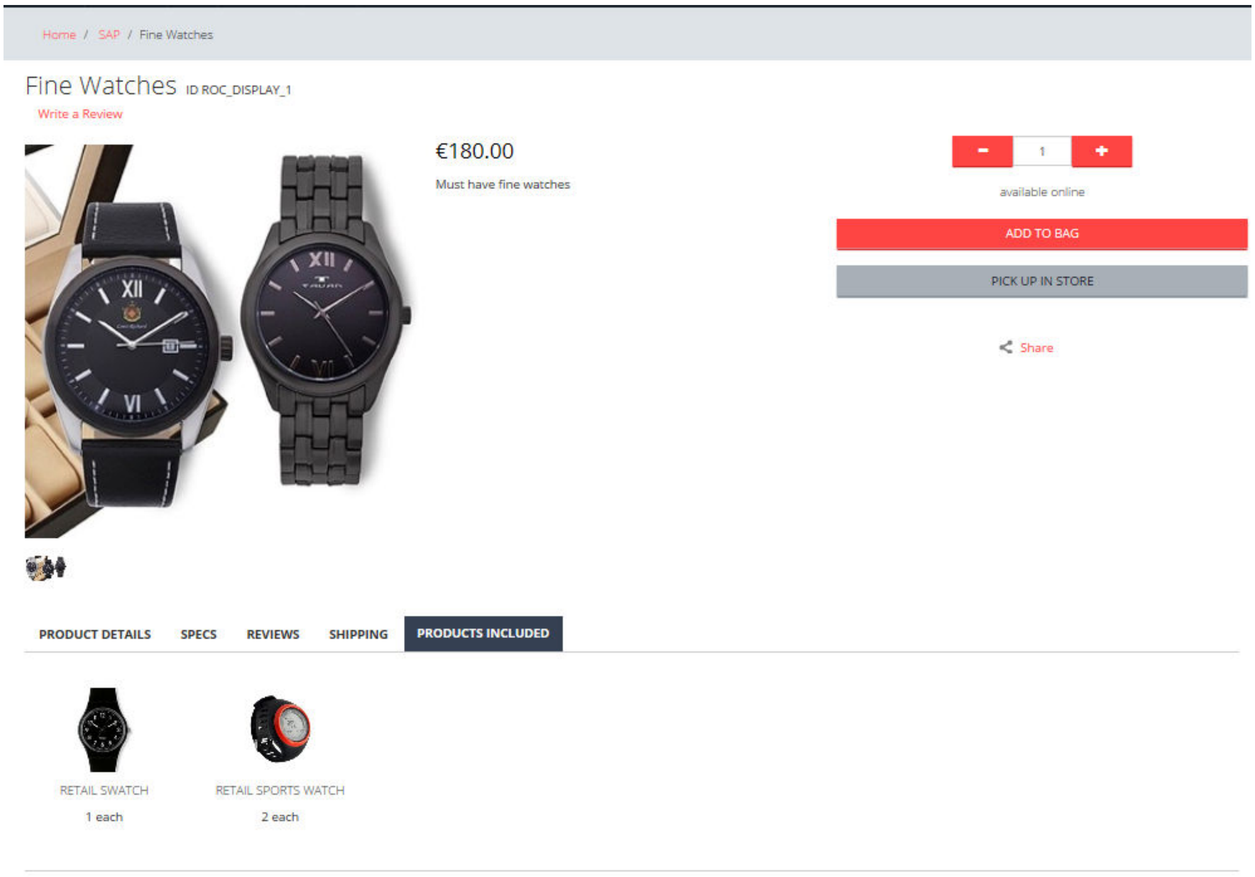
Structured Article Type	Description
Shoe prepacks = Prepack Prepacked variants of a generic article	A running shoe prepack containing: • 1 pair size 7 • 1 pair size 8 • 2 pairs size 9 • 3 pairs size 10 • 2 pairs size 11 • 1 pair size 12

2.7 saparticleb2caddon AddOn

The `saparticleb2caddon` AddOn extends target B2C storefront applications to display structured article component data. Structured article component data is part of the `saparticlemodel` extension.

Content

The component data of a structured article is displayed on the additional tab *Products Included* on the product detail page of a structured article. The article component data tab is provided by the new CMS component `ArticleComponentTabParagraphComponent` that extends the `CMSTabParagraphComponent` component.



Note

The *Products Included* tab is only displayed if a structured article has components.

Definition

Table 22: Definition

Feature	`saparticleb2caddon` AddOn
Description	The `saparticleb2caddon` AddOn extends a target B2C storefront to enable the display of structured article components in the product detail view.

Feature	`saparticleb2caddon` AddOn
Requires	addonsupport Extension acceleratorservices Extension hmc Extension cms2 Extension commerceservices Extension commercefacades Extension saparticlemodel Extension [page 43]
Author	SAP

Installation

The `saparticleb2caddon` AddOn needs to be added to the `localextensions.xml` file. The extensions listed in the table above are also required and must be added to this file.

The `saparticleb2caddon` AddOn must be added to your storefront extensions based on the `yacceleratorstorefront` template by using the following `ant` command:

```
ant addoninstall -Daddonnames="saparticleb2caddon"-
DaddonStorefront.yacceleratorstorefront="<mystorefrontextensionname>"
```

For more information about how to install AddOns using `ant addoninstall`, see Installing an AddOn for a Specific Storefront.

Supported Markets and Channels

The `saparticleb2caddon` AddOn supports the B2C market. However, it can only be used for the desktop channel by using the responsive UI.

Table 23: Supported Markets and Channels

Supported	B2C Commerce	B2B Commerce	Telco Commerce
Market	✓	⛔	⛔

Supported	B2C Commerce		B2B Commerce		Telco Commerce	
	Table 24:		Table 25:		Table 26:	
	Desktop	Mobile	Desktop	Mobile	Desktop	Mobile
Channel	✓	−	−	−	−	−

CMS Component Registration

To add the article component data CMS component to the product tabs slot of the product details page in a given storefront content catalog, the `saparticleb2caddon` AddOn uses the `/saparticleb2caddon/resources/saparticleb2caddon/import/contentCatalogs/{storefront}ContentCatalog/cms-content.impex` ImpEx files to hook into the core data import. For more information, see addonsupport Extension and impex Extension

The `saparticleb2caddon` AddOn provides ImpEx files for the following storefronts and their content catalogs:

- apparel-deContentCatalog
- apparel-ukContentCatalog
- electronicsContentCatalog

For other storefronts, simply create a corresponding folder `/saparticleb2caddon/resources/saparticleb2caddon/import/contentCatalogs/{storefront}ContentCatalog/`. For your storefront in the `saparticleb2caddon` AddOn, copy one of the `cms-content.impex` files and adapt it to support your storefront.

To import the ImpEx files, perform the following:

- Option A:
  For an automatic import, you can use the command:

  ```
  ant clean all initialize
  ```

- Option B:
  To avoid a complete initialization of SAP Hybris Commerce, you can import the appropriate ImpEx file manually by using the import function of the Hybris Administration Console (hac) or the import wizard of the Hybris Management Console (hmc). For more information, see the corresponding sections of the ImpEx User Guide.

  Note

  The ImpEx file references the `Staged` catalog version. For manual imports to the `Online` content catalog version, you need to change the catalog version to `Online` in the ImpEx file before you start the import.

  Example: `cms-content.impex` ImpEx file for `Online` version of `apparel-ukContentCatalog`

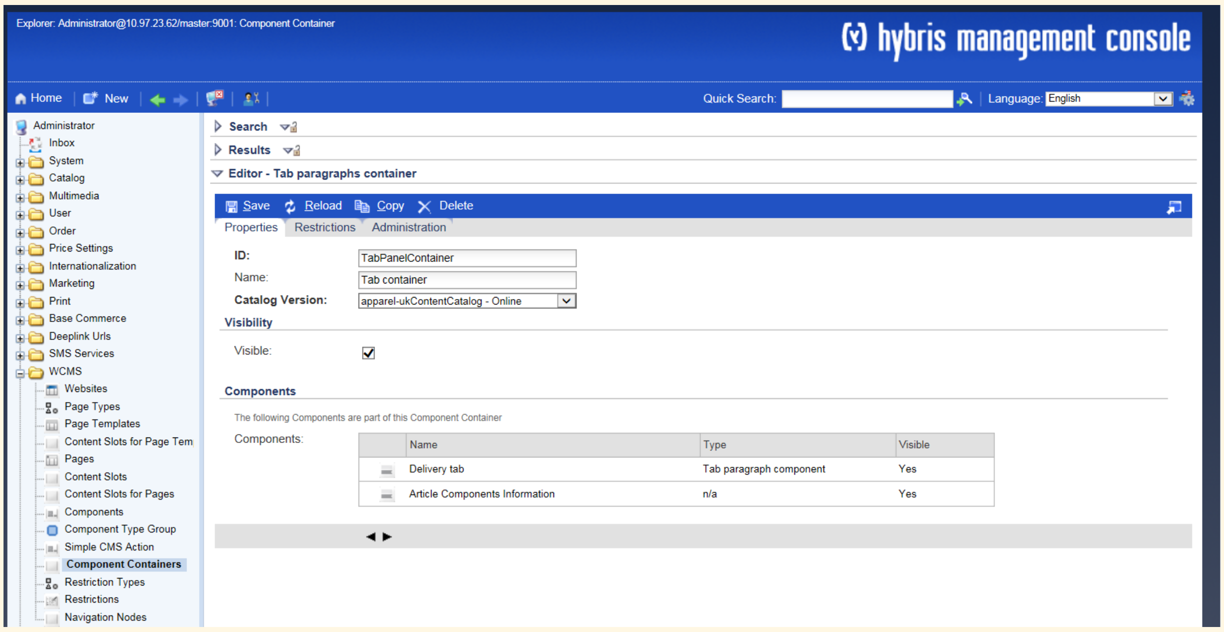
  ```
  $contentCatalog=apparel-ukContentCatalog
  $contentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=
  $contentCatalog]),CatalogVersion.version[default=Online])[default=
  $contentCatalog:Online]
  ```

```
$jarResourceCms=jar:com.sap.retail.commercesuite.saparticleb2caddon.constants.s
aparticleb2caddonConstants&/saparticleb2caddon/import/cockpit/cmscockpit
INSERT_UPDATE ArticleComponentTabParagraphComponent;
$contentCV[unique=true];uid[unique=true];name
;;ArticleComponentsInformation;Article Components Information
INSERT_UPDATE CMSTabParagraphContainer;
$contentCV[unique=true];uid[unique=true];name;visible;simpleCMSComponents(uid,
$contentCV);&componentRef
;;TabPanelContainer;Tab
container;true;deliveryTab,ArticleComponentsInformation;TabPanelContainer
```

As a result, the *Article Components Information* CMS component is added to the tab container in the WCMS for a storefront content catalog.

Example

Tab container for *apparel-ukContentCatalog - Online*



ID: TabPanelContainer
Name: Tab container
Catalog Version: apparel-ukContentCatalog - Online

Visibility

Visible: ✓

Components

The following Components are part of this Component Container

Components:

	Name	Type	Visible
▬	Delivery tab	Tab paragraph component	Yes
▬	Article Components Information	n/a	Yes

Variant Selector

The variant selector on the product detail page of the default B2C accelerators only shows `VariantValueCategories` that are in the same catalog as the product itself. ERP variants and their values however are replicated into the `ERP_CLASSIFICATION_026` catalog by default in order to avoid redundancies when providing multiple sales area catalogs. To consider the `ERP_CLASSIFICATION_026` catalog also as valid catalog, the `commerceProductService` was enhanced with bean `sapadmCommerceProductService`, which enhances the scope of allowed super category catalogs by the `ERP_CLASSIFICATION_026` catalog.

The bean definition can be found in the `saparticleb2caddon-spring.xml` file.

Note

If you want to replicate the merchandise categories into another catalog than `ERP_CLASSIFICATION_026`, enter the new classification catalog ID for properties saparticleb2caddon.merchandise.classification.catalog for this extension and saparticle.merchandise.classification.catalog of the saparticle Data Hub extension. Both values need to be in sync.

Responsive UI

The saparticleb2caddon extension supports the responsive UI. To force the responsive behavior, change the {HYBRIS_CONFIG_DIR}/local.properties file and add the following property:

```
commerceservices.default.desktop.ui.experience=responsive
```

Related Information

Hybris AddOn Concept

2.8 saparticlesearch Extension

The saparticlesearch extension enhances the standard SAP Hybris Commerce search functionality.

Contents of the saparticlesearch Extension

This extension enhances the standard SAP Hybris Commerce search functionality in a way that structured articles can also be found by the names and keywords of their component articles. Therefore, the search index has been enhanced by the following properties that are multi-value and localized:

- componentNames
- componentKeywords

Technical Details

The following value providers/resolvers deliver the values of the new search properties when the according search index is built:

- sapStructuredArticleComponentNamesValueResolver
- sapStructuredArticleComponentKeywordsValueResolver

Additionally, the `saparticlesearchCommerceSearchTextPopulatorEnhancer` bean enhances the standard SAP Hybris Commerce search populator for free text search (`commerceSearchTextPopulator`) by the new search properties:

The boost factor of both properties is lower than the existing search properties in order to get a high scoring for direct hits.

The beans are defined in the`resources/saparticlesearch-spring-solrfacetsearch.xml` file.

Default Content for Enhancing the B2C Catalog Search Indices

The above-mentioned search properties are added to the existing product catalog search indices of the B2C accelerators (`apparel-de/ukProductType`, `electronicsProductType`) as sample data.

Furthermore, the according indexer query for *Update* is enhanced so that changes on component articles lead to an update of the `solr` index of the according structured articles. See `resources/saparticlesearch/import/stores/[apparel-de|apparel-uk|electronics]/solr.impex`.

The definition of the according sample data event listener `saparticlesearchSampleDataEventListener` can be found in `resources/saparticlesearch-spring.xml`.

2.9 saparticlebackoffice Extension

The `saparticlebackoffice` extension enhances the standard Backoffice Administration Cockpit to enable maintenance of structured articles.

Contents of the saparticlebackoffice Extension

This extension enhances the standard Backoffice application for products in a way that structured articles can also be displayed and maintained. Therefore, an additional tab *SAP Retail* is available that displays specific attributes for structured articles. For more information about how to work with structured article components, see Working with Structured Article Components [page 53].

Technical Details

The additional tab *SAP Retail* is defined in the `/resources/saparticlebackoffice-backoffice-config.xml` file. This extends the standard Backoffice Administration Cockpit for products.

Additionally, the file contains field definitions for structured article type (`structuredArticleType`) and structured article components (`component`) to work with structured articles.

To create also new components for structured articles, a new creation wizard `ComponentWizard` has been added.

The standard creation wizard for products has also been enhanced to maintain structured article type when creating a new product:

2.9.1 Working with Structured Article Components

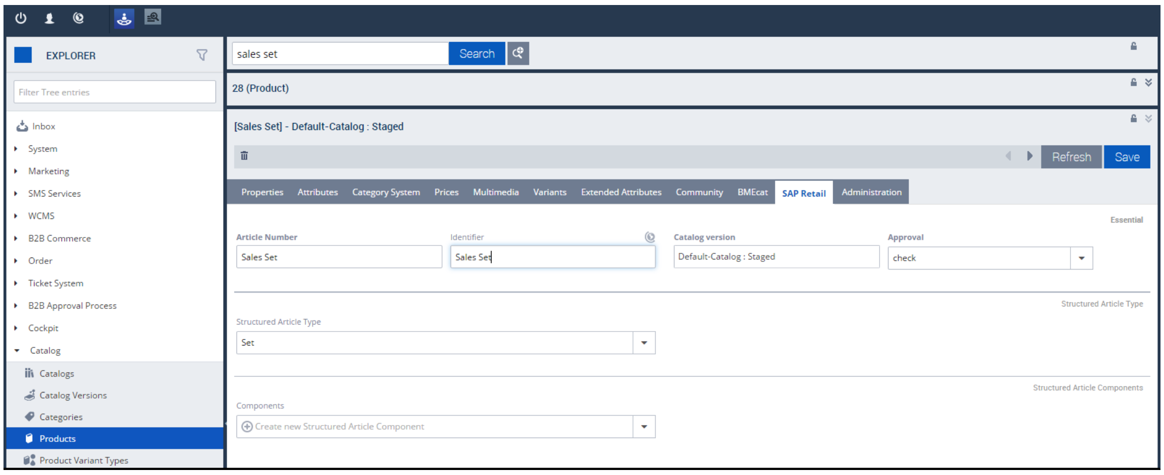
Use the Backoffice Administration Cockpit to work with structured article components.

Context

You can create new structured article components, search for structured article components, and create new standard products in the Backoffice Administration Cockpit. This is only possible, if the `saparticlemodel` extension is installed.

Procedure

1. Open the *Structured Article Components* section at the *SAP Retail* tab
   a. Choose *Catalog* > *Products* and then choose a structured article in the product search result.
   b. Choose the *SAP Retail* tab.



sales set

28 (Product)

[Sales Set] - Default-Catalog : Staged

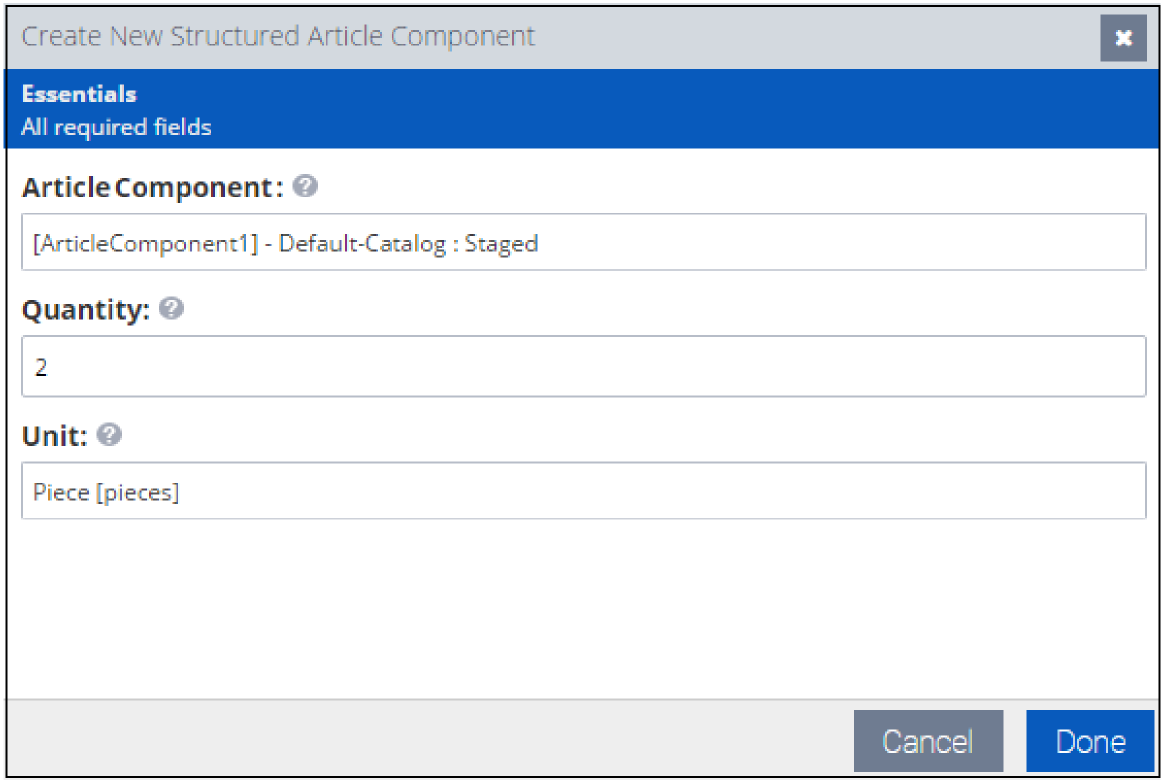
Article Number: Sales Set
Identifier: Sales Set
Catalog version: Default-Catalog : Staged
Approval: check

Structured Article Type: Set

Components: Create new Structured Article Component

   As soon as the structured article type is set to *Display*, *Set*, or *Prepack*, the *Structured Article Components* section appears under the *Structured Article Type* section.

2. Create a new structured article component
   a. Click *Create new Structured Article Component* in the *Components* field in the *Structured Article Components* section.
   b. Enter the structured article component, the quality, and the unit in the creation wizard.


Create New Structured Article Component

Essentials
All required fields

Article Component:

[ArticleComponent1] - Default-Catalog : Staged

Quantity:

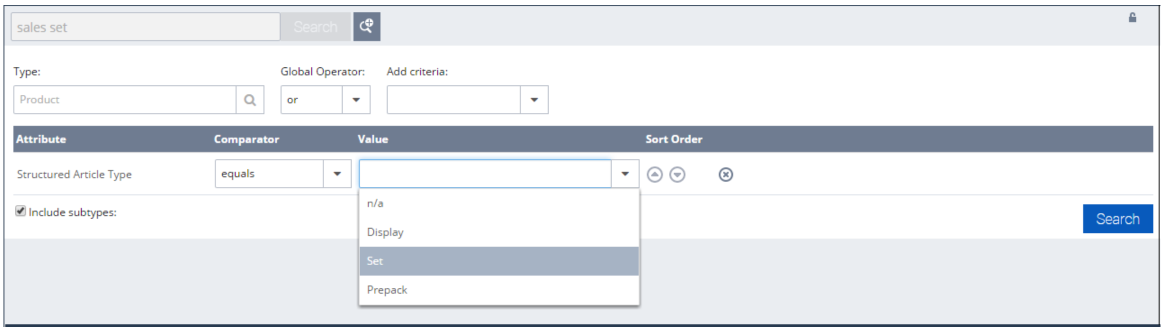
2

Unit:

Piece [pieces]

3. Search for structured article types in the standard product search.

   a. Choose *Catalog* > *Products* and click the *Advanced Search* button in the product search.
   b. Choose *Structured Article Type* in the dropdown box of the *Add criteria:* field.
   c. Choose *Display*, *Set*, or *Prepack* in the dropdown box of the *Value* field.
   d. Click the *Search* button to search for the selected structured article type.


sales set

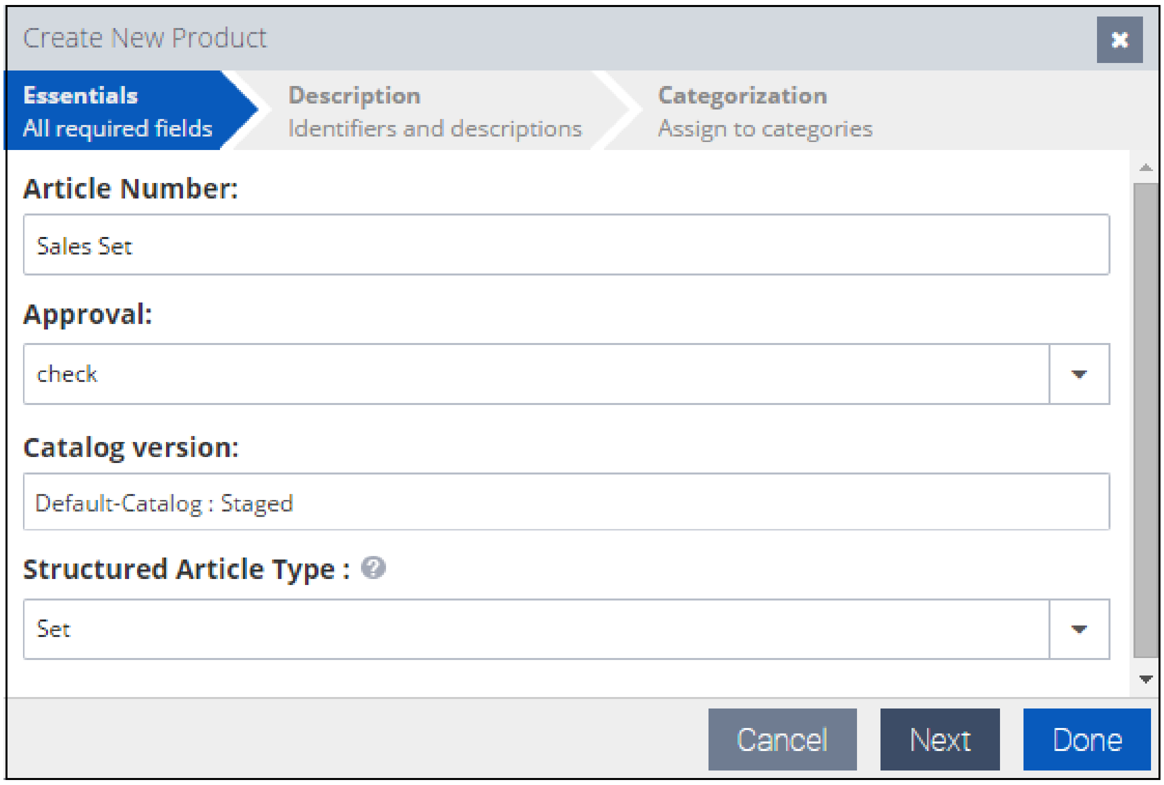
Type: Product
Global Operator: or
Add criteria:

Attribute	Comparator	Value	Sort Order
Structured Article Type	equals		

Include subtypes: ✓

4. Create a new structured article and choose a structured article type.

   a. Choose *Catalog* > *Products* and click the *+ Product* button.
   b. Enter the article name, choose an approval status, enter the catalog version, and choose a structured article type in the creation wizard.



Article Number: Sales Set

Approval: check

Catalog version: Default-Catalog : Staged

Structured Article Type : Set

2.10 Troubleshooting

Troubleshooting in SAP Retail, Data Hub, and SAP Hybris Commerce

In SAP Retail

Check the IDoc sending status in *IDoc List* (transaction `WE05`).

If `VariantValueCategories` are missing in SAP Hybris Commerce, check the generated `CHRMAS` IDoc with *IDoc List* (transaction `WE05`). Values should be available in segment `E1CAWNM`. Check the basic type of the IDoc, which is `CHRMAS05`.

If characteristic values are defined in *Edit Characteristic Values* (transaction `WRFCHVAL`), ensure that the checkbox *Activate ALE Distribution* is selected in Customizing of SAP Retail at *Logistics - General* > *Material Master* > *Retail-Specific Settings* > *Settings for New Characteristic Value Storage* > *Activate Distribution Using ALE*. For more information, see SAP Note 1785737.

In the Data Hub

To get more information, increase the log level for specific packages. Proceed as follows:

- Set the log level of `com.sap.retail.datahub.saparticle` to *WARN*, to get log entries in the case of data inconsistencies and replication problems. The log level can be set in production. Log level *DEBUG* gives more detailed information, but should not be set in production.
- Set the log level of `com.hybris.datahub.sapidocintegration.IDOCMappingService` to *DEBUG*, to get information about parsing time and fragments of an IDoc. This can be set in production, since it activates only one trace entry per received IDoc. Log level *TRACE* prints a list transformed maps. Do not use log level *TRACE* in production, since the logged data amount could be up to several megabytes.
- Set the log level of `com.hybris.datahub.sapidocintegration.spring.HttpInboundService` to *INFO*, to log received IDocs. Do not set in production because the IDoc size could be up to several megabytes.

Check Data Hub errors in the Data HubBackoffice. For more information, see Overview of UI and Installing and Initializing the Data HubBackoffice UI.

In SAP Hybris Commerce

Check the Impex log in the Backoffice Administration Cockpit. Alternatively, monitor Impex folder `${HYBRIS_DATA_DIR}/media/sys_master/impex` for CSV files, which indicates errors during import.

3 Omnichannel Article Availability and Sourcing

Omnichannel article availability and sourcing (OAA) provides retail-specific availability information and sourcing information to your customers across all sales and communication channels. It is based on the integration of the following software components: SAP Retail, SAP Customer Activity Repository, SAP Hybris Commerce, and SAP Hybris Commerce, integration package for SAP for Retail.

OAA provides **retail-specific availability information and sourcing information** to your customers. This information is identical and consumable **across all sales and communication channels** (points of sale, online, mobile, call centers) and provides a consistent customer experience. OAA information can be provided for articles delivered from distribution centers (DCs), vendors, or brick-and-mortar stores.

OAA supports the following e-commerce scenarios:

- Click-and-ship scenario
- Click-and-collect scenario

⚠ Caution

In the standard delivery, availability information and sourcing information is available in the facade layer of SAP Hybris Commerce only; it is not available on the UI yet. To be able to present this information to your online store customers, you must make it available in the UI layer of the programming model first in a custom project.

To facilitate this process, SAP provides a set of modification examples along with explanatory documentation delivered via SAP Note 2360000 or via SAP Help Portal.

System Setup

The standard delivery of the OAA functionality is based on the implementation and integration of the following software applications:

- **SAP Retail** 6.0, minimum release EHP7 SP13 / EHP8 SP04, as back-end system for order fulfillment and availability calculation for DC articles or vendor articles

  i Note

  SAP Hybris Commerce, integration package for SAP for Retail enhances the generic SAP ERP back-end integration (SAP Back-End Integration: SAP ERP and SAP CRM) with specific functionality for retail articles and thus enables support of the generic integration scenarios for the retail and fashion industry. Specific support for the fashion industry, such as support for seasons or segmentation, is not provided and no fashion industry-specific content is included in the integration package. However, the integration package is released for an SAP Retail back end as well as an SAP Fashion Management back end within the boundaries of the delivered scope.

Caution

SAP S/4HANA is not supported.

- **SAP Customer Activity Repository** 3.0 or higher as the central repository hub for all customer activities from all channels
  If you want to deliver not only from DCs or vendors but also from brick-and-mortar stores, you must have set up SAP Customer Activity Repository to collect all POS data from your stores.
- **SAP Hybris Commerce** 6.2 or higher, full edition or B2C Edge edition
  - B2C Accelerator (part of SAP Hybris Commerce) for deployment of online stores and for storing the rough stock indicator information
  - Data Hub (part of SAP Hybris Commerce) as middleware used for replication of IDocs between SAP Customer Activity Repository and SAP Hybris Commerce
  - Hybris-SAP Solution Integration: SAP asynchronous order management functionality

  Note

  Order Management for SAP Hybris Commerce (formerly called Order Management Services or OMS) is not part of the OAA solution.
- **SAP Hybris Commerce, integration package for SAP for Retail** 2.2 or higher for replication of Retail articles and for OAA functionality

Note

In your system landscape, system relations may be set up as follows:

- SAP Retail : SAP Customer Activity Repository 1:1
- SAP Retail : SAP Hybris Commerce 1:n
- SAP Customer Activity Repository : SAP Hybris Commerce 1:n

Note

For installation information, see the Installation Guide of the respective software application.

For integration information, see the section *Configure Omnichannel Article Availability and Sourcing for Use with SAP Customer Activity Repository* in the Common Installation Guide of SAP Customer Activity Repository retail applications bundle.

Extensibility

Note

- In a custom project, you may **replace SAP Hybris Commerce** with online store software from a different vendor, using the REST APIs provided.
- There can be consumers of the OAA functionality other than online stores, such as product recommendations and offer recommendations from SAP Hybris Marketing.

- You may replace SAP Customer Activity Repository with a different repository hub, by replicating the availability information to this hub as follows:
    - For DC articles, create your own implementation of business add-in (BAdI) `BADI_OAA_ATP_RESULT_HANDLER` in SAP Retail.
    - For store articles, replicate the availability information (TLOG data) in a custom implementation, from the system that holds this information (SAP Customer Activity Repository or a different system).
    - For vendor articles, call RFC `OAA_RFC_GET_VENDOR_STOCK` in SAP Retail.

Business Configuration and Business Setup

To enable the OAA functionality, you have to carry out a set of busines configuration and business setup steps first in all connected systems (SAP Retail, SAP Customer Activity Repository, SAP Hybris Commerce (see application help of SAP Customer Activity Repository, under *Inventory Visibility* > *Omnichannel Article Availability and Sourcing* > *Business Configuration and Business Setup*).

Features

- **Availability calculation for DC articles** uses the **standard ATP logic of SAP Retail** but improves its performance considerably: The ATP run is executed on a number of **parallelized work processes**, and the result is stored as an ATP snapshot in SAP Customer Activity Repository. Since you as a retailer typically need to handle large volumes of articles and of transactions and need to update availability information frequently, this process helps meet your requirements with regard to performance.
- **Availability calculation for store articles** considers physical stock in the stores only (no projected stock) and is carried out via the **Inventory Visibility capabilities of SAP Customer Activity Repository**. Inventory Visibility reads POS data and data (sales orders) from SAP Retail SD.
- **Availability calculation for vendor articles** checks the uploaded vendor stock information. If the vendor did not provide any stock information, availability is assumed to be **infinite**. This assumption covers scenarios where vendors do not provide exact stock figures but promise to ship all articles within a specified time period.
- Articles in checkout and orders that have not reached SAP Retail yet are included in the availability calculation via **temporary reservations**. This minimizes the risk that you as a retailer cannot fulfill orders as promised to your end customers in the different channels.
- In the standard delivery, **availability information is aggregated on different levels**, depending on the scenario and on how far the online store customer has advanced in the ordering process. The more advanced he or she is, the more accurate the availability and sourcing information gets.
    - In a click-and-ship scenario, availability information can be fully aggregated into **rough stock indicators**. Rough stock indicators can be presented, for example, as three different traffic lights representing in stock, low stock, or out of stock. Alternatively, availability information can be partly aggregated or not aggregated at all; in both cases, it is provided through **schedule lines** that show which quantities are available on which date.
    - In a click-and-collect scenario, availability information always indicates the exact quantities available on the current day.
- **Sourcing** tries to find the best sources for all articles that an online store customer wants to order. Sources can be all the **DCs, vendors, or brick-and-mortar stores** from which the articles can be delivered.

Caution

With OAA, structured articles that consist of several components, such as displays, sets, prepacks, are only supported on header level. If you made a setting in Customizing to carry out availability calculation on component level, this function is not supported with OAA.

Note

For correct availability and sourcing results, identical items in the cart must be merged. In the standard delivery, merging in SAP Hybris Commerce is automatic.

3.1 sapoaaorder Data Hub Extensions

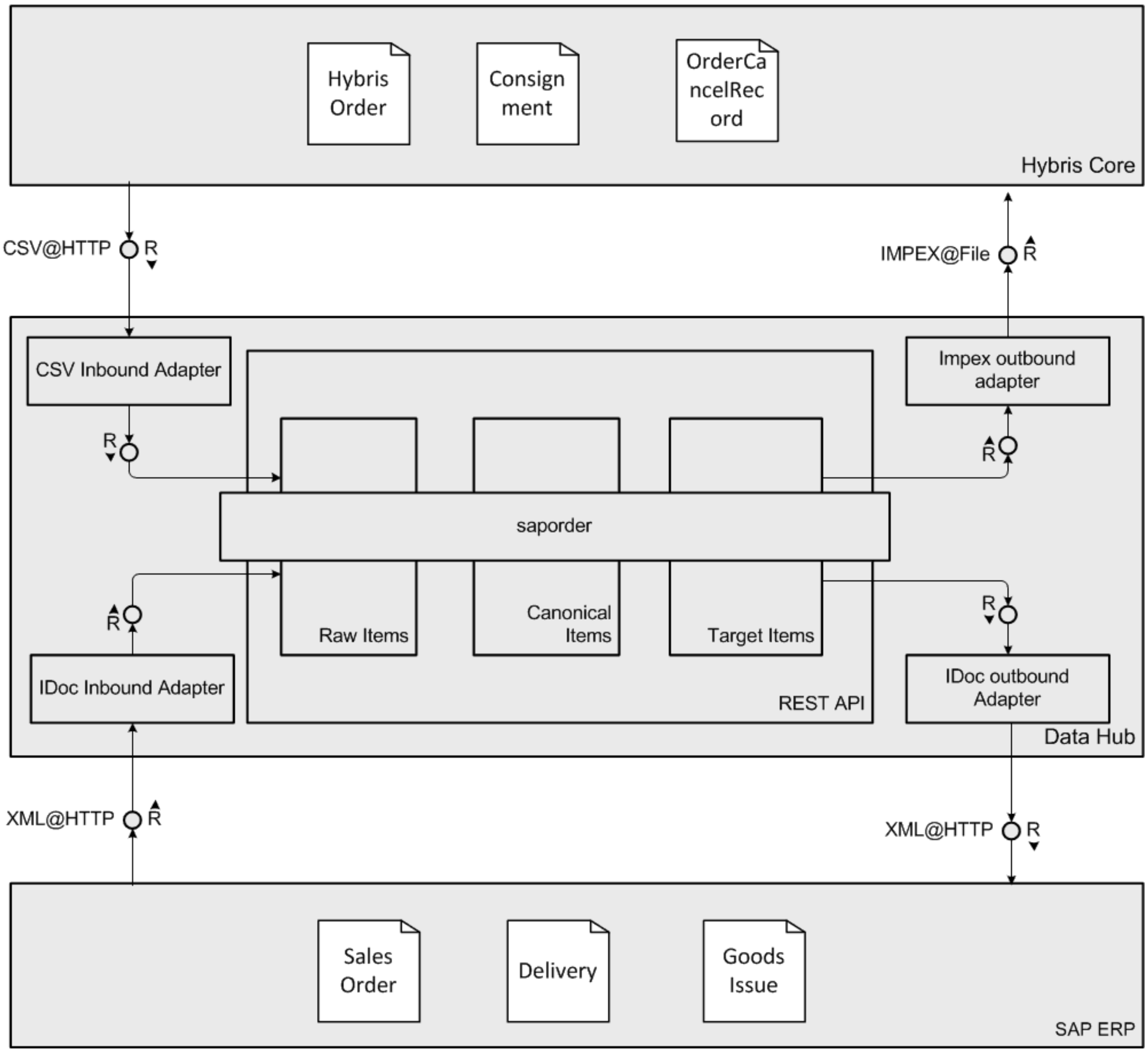
The `sapoaaorder` Data Hub extensions extend the `saporder` Data Hub extensions with OAA specific fields. The `saporder` Data Hub extensions provide the integration content necessary to connect an SAP Hybris Commerce core installation to an SAP Retail system for order fulfillment.

The extensions are as follows:

- `sapoaaorder-raw`
- `sapoaaorder-canonical`
- `sapoaaorder-target`

Overview

The following graphic gives an overview of the order-related integration between the SAP Hybris Commerce core installation and the SAP Retail system via the Data Hub.



The order including the OAA-specific fields is sent to the Data Hub via CSV.

Dependencies and Prerequisites

This section lists the steps required to run the `sapoaaorder` Data Hub extensions.

1. You have installed and configured the SAP Hybris Data Hub.
2. You have deployed the following extensions in `<your-tomcat-folder>\webapps\datahub-webapp\WEB-INF\lib`:
    - `sapoaaorder`
    - `saporder`
    - `sapoutboundadapter`
    - `sapidocintegration`

- `sapcoreconfiguration`

3. The `local.properties` file contains the necessary configurations for pool system and target system.

Configuration

For the `sapoaaorder` Data Hub extensions, no specific configuration is needed.

However, you must have configured the `saporder` Data Hub extensions.

OAA Modifications

Table 27: Modifications to saporder Data Hub Extension

Item	Modification
`RawHybrisOrder`	The following fields were added: • `(reservationId)` • `siteId` • `scheduleLines` - all schedule lines of an item
`CanonicalOrder`	The following field was added: `reservationId`
`CanonicalOrderItem`	The following fields were added: • `siteId` • `scheduleLines` • `vendorItemCategory` The following fields are filled from the `ScheduleLinePublicationHandler` only on `ScheduleLine` target segment: • `scheduleLineNumber` • `requiredDate` • `requiredQuantity` The following fields were added to the `CanonicalPartnerRole` item: • `entryNumber` This number is used to transfer a vendor item partner in the drop shipment sce-nario.
`$E1BPSDHD1`	The following fields were added: `COMPL_DLV` -> Set to false

Item	Modification
`$E1BPSDITM`	The following field was added: `PLANT`
`$E1BPSCHDL`	The following fields were added or modified: • `SCHED_LINE` • `REQ_QTY` • `REQ_DATE`
`$E1BPPARNR`	The following fields were added or modified: • `ITM_NUMBER`

Publication Handler

The `ScheduleLinePublicationHandler` is only applicable if the target item type is the schedule line segment and the target system is the SAP Retail system.

It creates a copy of the `CanonicalOrderItem` for each schedule line in the `scheduleLines` field of the `CanonicalOrderItem`. The schedule lines have to be separated by "|" and the quantity and date have to be separated by ";".

Troubleshooting

You can access the information contained in the SAP Hybris Data Hub directly using the following standard browser GET requests:

- `http://localhost:8080/datahub-webapp/v1/pools/SAPORDER_OUTBOUND_POOL/items/RawHybrisOrder.xml`
- `http://localhost:8080/datahub-webapp/v1/pools/SAPORDER_OUTBOUND_POOL/items/CanonicalOrder.xml`
- `http://localhost:8080/datahub-webapp/v1/pools/SAPORDER_OUTBOUND_POOL/items/CanonicalOrderItem.xml`

Related Information

Installing Data Hub
saporder Data Hub Extensions

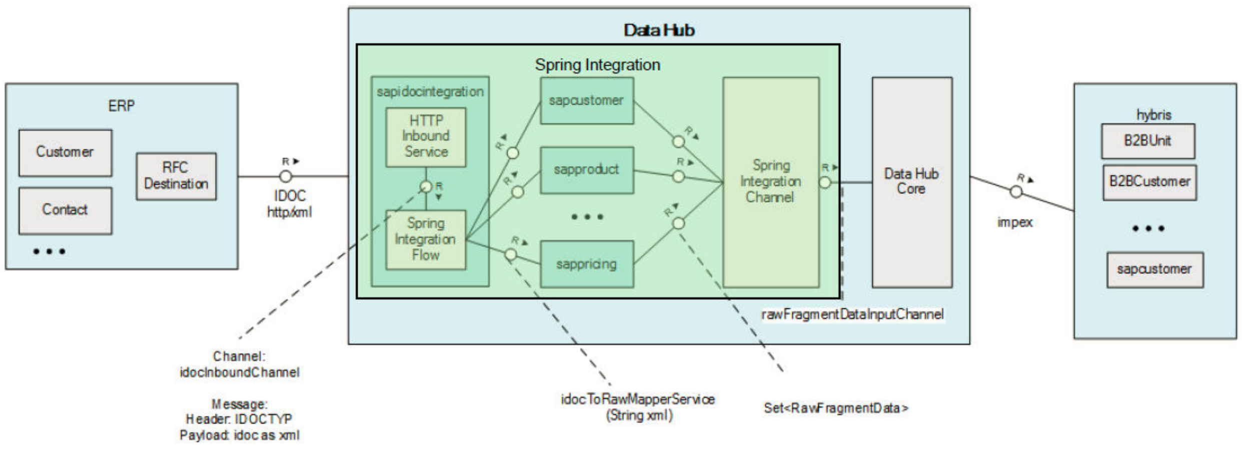
3.2 sapoaasite Data Hub Extensions

The `sapoaasite` Data Hub extensions enable processing of the SAP Retail IDoc `W_OAA_SITES` to the SAP Hybris Commerce core installation.

The extensions are as follows:

- `sapoaasite-raw`
- `sapoaasite-canonical`
- `sapoaasite-target`

Overview



The `sapoaasite` extensions are OAA-specific IDoc integration extensions and are used for the replication of plants (sites) sent from SAP Retail.

An HTTP request is used to send IDocs from SAP Retail to the SAP Hybris Data Hub, and once there, a Spring channel transfers the payload to the `idocInboundChannel`. The payload is an XML file that is transformed into raw fragments by the `IDOCMappingService`. The transformation is generic and optimized.

The raw fragment flow into the SAP Hybris Data Hub progresses from `rawItem`, to `canonicalItem`, and then to `targetItem`, after which it is sent to the SAP Hybris server using the ImpEx technology.

IDocs are modeled using `rawItems`. The `rawItem` attributes take their names from IDoc segments and subsegments, separated by dashes. For example, if the sales organization attribute is named `E1MARAM-E1MVKEM-VKORG`, the naming convention works as follows:

- `VKORG` is the field name.
- The field is part of the `E1MVKEM` segment.
- This segment is part of the `E1MARAM` segment.

This convention is followed in all `rawItem` attributes that model IDocs to work with the `IDOCMappingService`.

Dependencies and Prerequisites

This section lists the steps required to run the `sapoaasite` Data Hub extensions.

1. You have installed and configured the SAP Hybris Data Hub.
2. You have deployed the following extensions in `<your-tomcat-folder>\webapps\datahub-webapp\WEB-INF\lib`:
   - `sapoaaasite`
   - `sapidocintegration`
   - `sapcoreconfiguration`
3. The `local.properties` file contains the necessary configurations for pool system and target system.

Configuration: Data Hub Properties

The `local.properties` file should look as follows:

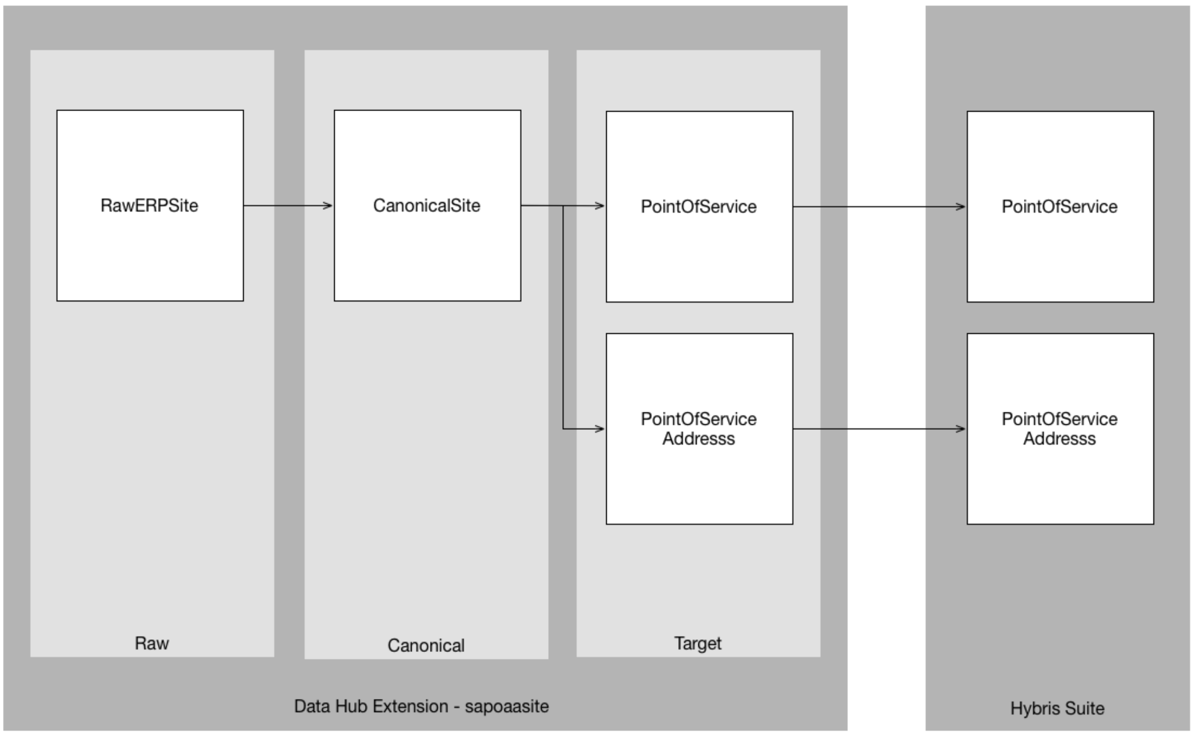
Source Code

```
sapcoreconfiguration.pool=SAPCONFIGURATION_POOL
sapcoreconfiguration.autocompose.pools=GLOBAL,SAPCONFIGURATION_POOL,SAPOAASITE_INB
OUND_POOL
sapcoreconfiguration.autopublish.targetsystemsbypools=GLOBAL.HybrisCore,SAPOAASITE
_INBOUND_POOL.HybrisCore
targetsystem.hybriscore.url=http://localhost:9001/datahubadapter
targetsystem.hybriscore.username=
targetsystem.hybriscore.password=
```

The values should be set based on your own configuration. The `local.properties` file should be located in `<your-tomcat-folder> \webapps\datahub-webapp\WEB-INF\classes\`.

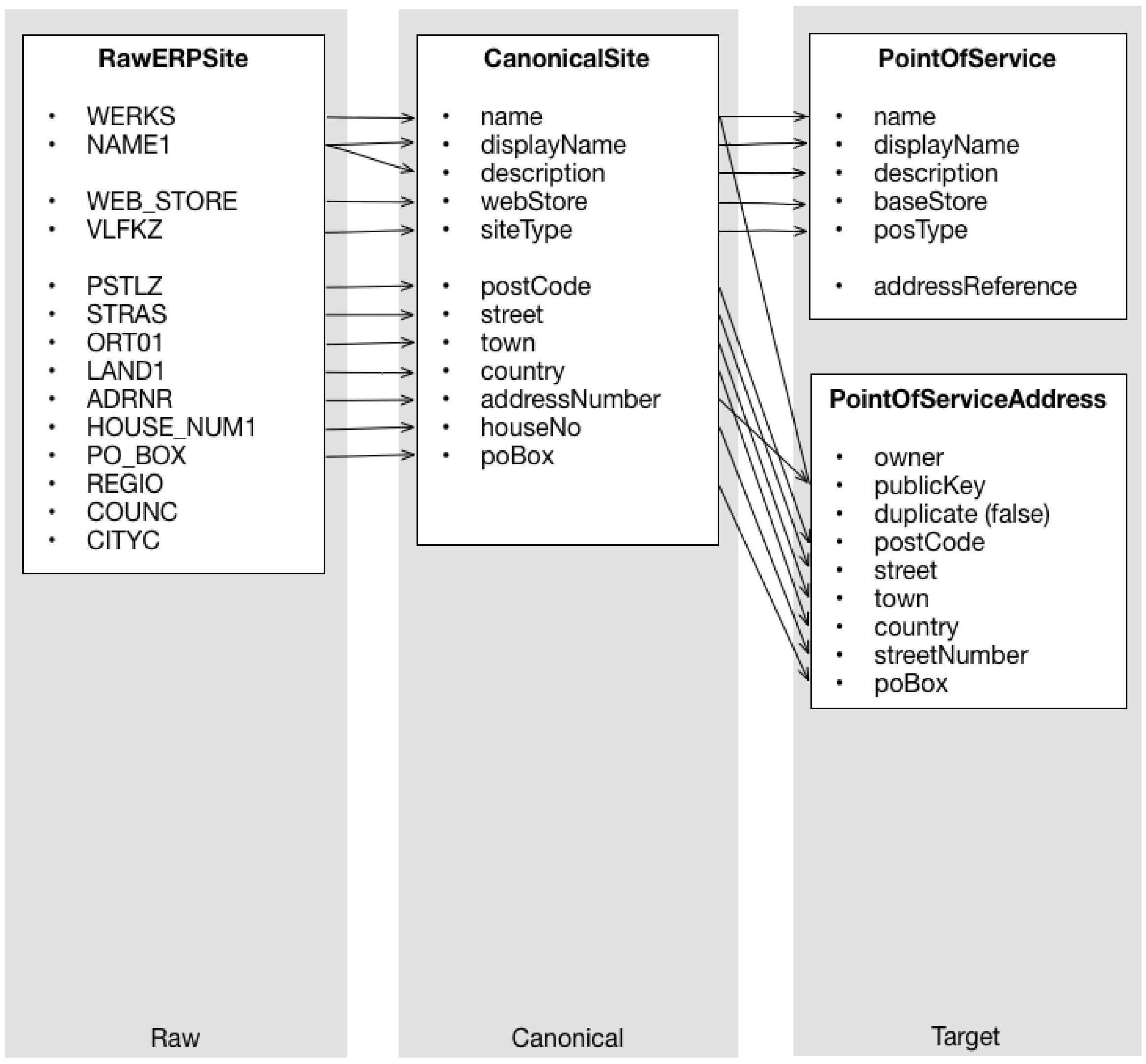
IDOC RawItem CanonicalItem TargetItem Flow

The diagram below explains how information contained in the `W_OAA_SITES` IDocs is processed by the SAP Hybris Data Hub before being sent to SAP Hybris Commerce.



RawItem to CanonicalItem to TargetItem Field Mapping

The diagram below provides an overview of the field mapping. For the most up-to-date values, see the `META-INF\sapoaasite-datahub-extension.xml` file.



sapoaasite - site inbound

posType Mapping

The `posType` is mapped to the `VLFKZ` IDoc field of the SAP Retail back end. When `VLFKZ` has an "A" as input, the `posType` is the `PointOfServiceTypeEnum.code` *Store*, otherwise it is *WAREHOUSE*.

cacShippingPoint Mapping

In SAP Retail, shipping point determination is used to find a shipping point that is used for the click-and-collect scenario. This shipping point is replicated via site replication and is stored in an SAP Hybris point of service. When a click-and-collect order item is created, the shipping point stored for click-and-collect is set at the corresponding SAP Retail SD order item.

Troubleshooting

You can access the information contained in the SAP Hybris Data Hub directly using the following standard browser GET requests:

- `http://localhost:8080/datahub-webapp/v1/pools/SAPOAASITE_INBOUND_POOL/items/RawERPSite.xml`
- `http://localhost:8080/datahub-webapp/v1/pools/SAPOAASITE_INBOUND_POOL/items/CanonicalSite.xml`

Related Information

Installing Data Hub

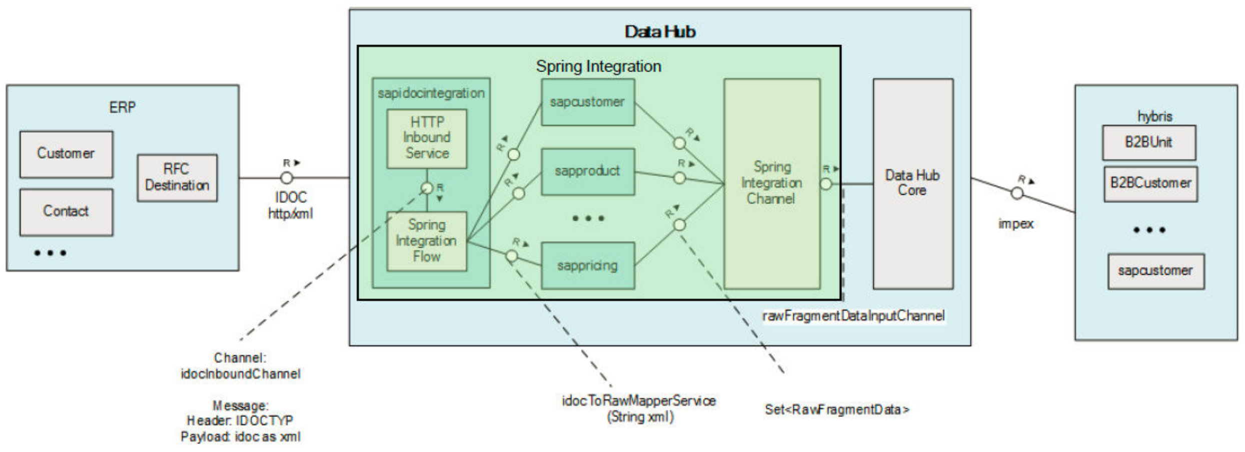
3.3 sapoaarsi Data Hub Extensions

The `sapoaarsi` Data Hub extensions enable processing of the SAP Retail IDoc `OAA_RSI_IDOCTYPE` to the SAP Hybris Commerce core installation.

The extensions are as follows:

- `sapoaarsi-raw`
- `sapoaarsi-canonical`
- `sapoaarsi-target`

Overview



The sapoaarsi extensions are OAA-specific IDoc integration extensions and are used for the replication of rough stock indicators from SAP Customer Activity Repository to SAP Hybris Commerce.

An HTTP request is used to send IDocs from SAP Customer Activity Repository to the SAP Hybris Data Hub, and once there, a Spring channel transfers the payload to the `idocInboundChannel`. The payload is an XML file that is transformed into raw fragments by the `IDOCMappingService`. The transformation is generic and optimized.

The raw fragment flow into the SAP Hybris Data Hub progresses from `rawItem`, to `canonicalItem`, and then to `targetItem`, after which it is sent to the SAP Hybris server using the ImpEx technology.

IDocs are modeled using `rawItems`. The `rawItem` attributes take their names from IDoc segments and subsegments, separated by dashes.

This convention is followed in all `rawItem` attributes that model IDocs to work with the `IDOCMappingService`.

Dependencies and Prerequisites

This section lists the steps required to run the `sapoaarsi` Data Hub extensions.

1. You have installed and configured the SAP Hybris Data Hub.
2. You have deployed the following extensions in `<your-tomcat-folder>\webapps\datahub-webapp\WEB-INF\lib`:
   - `sapoaaarsi`
   - `sapidocintegration`
   - `sapcoreconfiguration`
3. The `local.properties` file contains the necessary configurations for pool system and target system.
4. You have activated the `sapoaamodel` extension.

Configuration: Data Hub Properties

The `local.properties` file should look as follows:

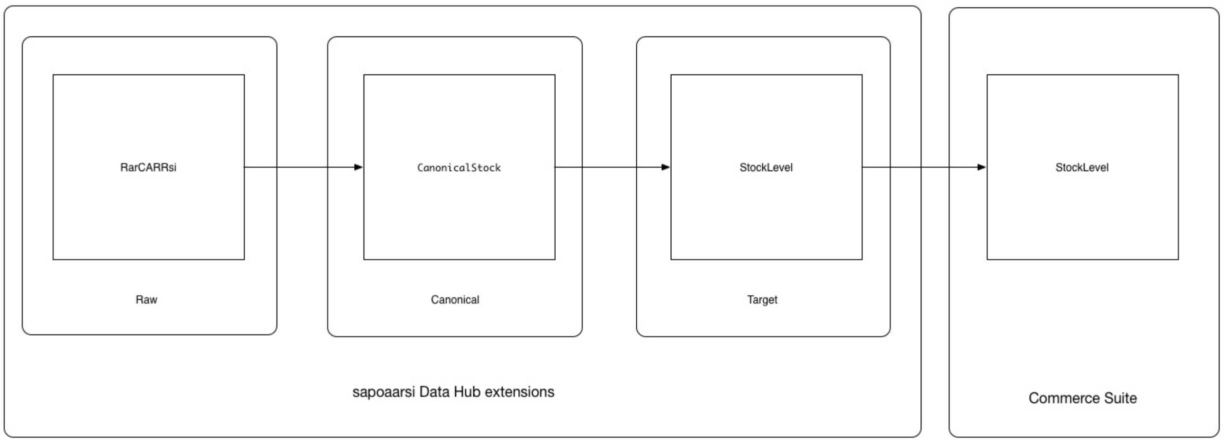
Source Code

```
sapcoreconfiguration.pool=SAPCONFIGURATION_POOL
sapcoreconfiguration.autocompose.pools=GLOBAL,SAPCONFIGURATION_POOL,SAPOAARSI_INBO
UND_POOL
sapcoreconfiguration.autopublish.targetsystemsbypools=GLOBAL.HybrisCore,SAPOAARSI_
INBOUND_POOL.HybrisCore
targetsystem.hybriscore.url=http://localhost:9001/datahubadapter
targetsystem.hybriscore.username=
targetsystem.hybriscore.password=
```

The values should be set based on your own configuration. The `local.properties` file should be located in `<your-tomcat-folder> \webapps\datahub-webapp\WEB-INF\classes\`.

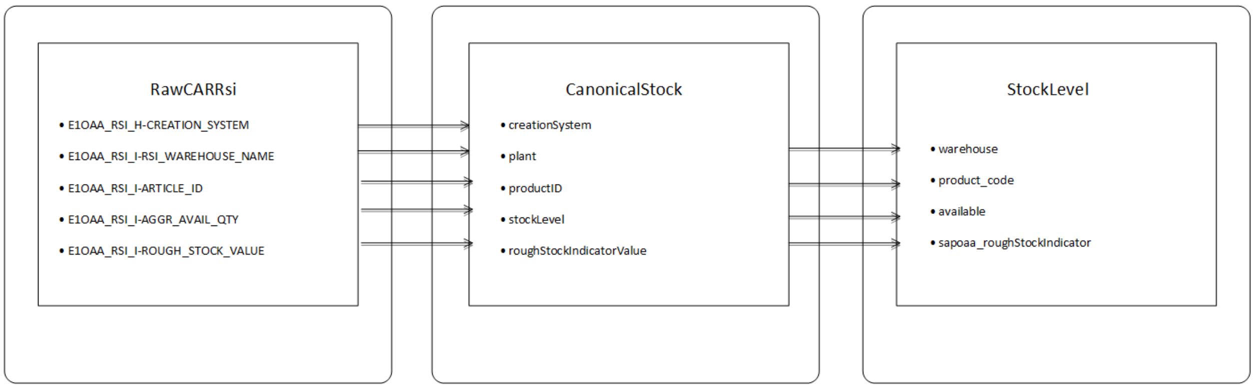
IDOC RawItem CanonicalItem TargetItem Flow

The diagram below explains how information contained in the `OAA_RSI_IDOCTYPE` IDocs is processed by the SAP Hybris Data Hub before being sent to SAP Hybris Commerce.



RawItem to CanonicalItem to TargetItem Field Mapping

The diagram below provides an overview of the field mapping. For the most up-to-date values, see the `META-INF\sapoaarsi-canonical-datahub-extension.xml` and `META-INF\sapoaarsi-target-datahub-extension.xml` files.


RawCARRsi
• E1OAA_RSI_H-CREATION_SYSTEM
• E1OAA_RSI_I-RSI_WAREHOUSE_NAME
• E1OAA_RSI_I-ARTICLE_ID
• E1OAA_RSI_I-AGGR_AVAIL_QTY
• E1OAA_RSI_I-ROUGH_STOCK_VALUE
CanonicalStock
• creationSystem
• plant
• productID
• stockLevel
• roughStockIndicatorValue
StockLevel
• warehouse
• product_code
• available
• sapoaa_roughStockIndicator

Troubleshooting

You can access the information contained in the SAP Hybris Data Hub directly using the following standard browser GET requests:

- `http://localhost:8080/datahub-webapp/v1/pools/SAPOAARSI_INBOUND_POOL/items/RawCARRsi.xml`

- `http://localhost:8080/datahub-webapp/v1/pools/SAPOAARSI_INBOUND_POOL/items/CanonicalStockLevel.xml`

Related Information

Installing Data Hub

3.4 sapoaaddon Extension

The `sapoaaddon` extension provides omnichannel article availability and sourcing (OAA) functionality during different checkout steps.

Dependencies

The `sapoaaddon` extension requires the following extensions:

- `hmc`
- `addonsupport`
- `acceleratorstorefrontcommons`
- `sapoaacommercefacades`

Supported Markets and Channels

The `sapoaaddon` extension is designed for B2C commerce. It can be used for both the desktop and mobile channels.

Table 28: Markets and Channels Supported by the sapoaaddon Extension

Supported	B2C Commerce	B2B Commerce	Telco Commerce
Market	✓	–	–
Channel	Desktop ✓ Mobile ✓	Desktop – Mobile –	Desktop – Mobile –

Main Purpose

The `sapoaaaddon` AddOn provides Omnichannel Article Availability (OAA) functionality including sourcing validation during different checkout steps. To achieve this, the `checkoutStepValidators` for the delivery method and the delivery address are used to perform OAA sourcing when entering and - in case of delivery method - when exiting these checkout steps. If the cart has only items to be collected and the user navigates to checkout, the default checkout steps for choosing the delivery address and delivery method are skipped and the user is redirected to the payment section.

OAA functionality is also supported during express checkout.

How to Install

Add the `sapoaaaddon` extension to your `localextensions.xml` and make sure that the required extensions listed above are also included.

To install, enter the **`addoninstall`** command for `sapoaaaddon` in the command line dialog box. For example, to install the extension in an online store called `yacceleratorstorefront`, enter the following:

Source Code

```
ant addoninstall -Daddonnames="sapoaaaddon" -
DaddonStorefront.yacceleratorstorefront="yacceleratorstorefront"
```

Modifications Checklist

The modifications that this extension makes to the Accelerator are listed below:

Impex Configuration Scripts	⊖	Service Layer	⊖	Page Controllers	⊖
Core Data Listeners	⊖	Facade DTO	⊖	Tags	⊖
Model Layer	⊖	Facade Layer	✓ The extension overwrites the `DefaultAcceleratorCheckoutFacade` with facade `DefaultSapOaaAcceleratorCheckoutFacade`.	TLD	⊖

Model Interceptors	⛔	CMS Components	⛔	Filters	⛔
Cockpit Configuration	⛔	Page Templates	⛔	MVC Interceptors	⛔
Cockpit Beans	⛔	JavaScript	⛔	Spring Security	⛔
Validation Rules	✓ The extension adds the following checkout step validators: • `ResponsiveDeliveryMethodCheckoutStepValidator` • `ResponsiveDeliveryAddressCheckoutStepValidator`	CSS	⛔	Message Resources	✓

3.5 sapoaacommercefacades Extension

The `sapoaacommercefacades` extension provides the data definitions that match the models created in `sapoaamodel`, as well as populators that convert the model classes into data classes. It also enhances the SAP Hybris Commerce standard checkout facade and adds a method to execute sourcing with the current session cart.

Overview

The `sapoaacommercefacades` extension provides a way to execute sourcing in checkout, with the current session cart of the user. It offers populators to convert the model structures into the data structures defined.

Configuration

This extension does not need any configuration. In this extension, the `defaultOrderEntryProductConverter` is overwritten and the `productDeliveryModeAvailabilityPopulator` is removed, to prevent calling SAP Customer Activity Repository on each update call or add to cart call.

Sourcing

The `DefaultCheckoutFacade` has been enhanced and a new interface called `OaaCheckoutFacade` has been implemented to enable a sourcing call from the facade layer. This sourcing call is executed by the `sapoaaaddon` extension.

Dependencies

The `sapoaacommerceservices` extension depends on the following extensions:

- `sapoaacommerceservices`: OAA service extension
- `commerceFacades`: SAP Hybris Commerce standard facades

Related Information

Converters and Populators

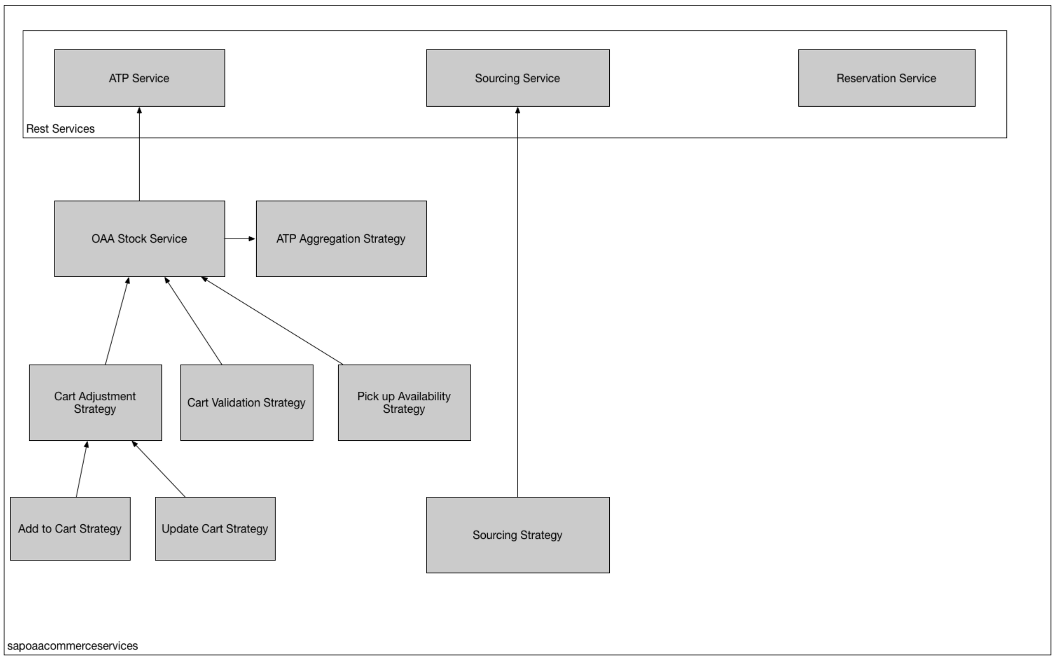
3.6 sapoaacommerceservices Extension

The `sapoaacommerceservices` extension provides services needed to integrate OAA into SAP Hybris Commerce.

Overview

This extension contains services that are used to integrate OAA-specific functionality into SAP Hybris Commerce. It provides **rough stock indicators** for the article catalog and **availability information** from SAP Customer Activity Repository for low stock articles in the shopping cart, calls **sourcing** including creation of **temporary reservations** in checkout, and offers a stock service to call the OAA REST services.

The diagram below shows the overall architecture:



Configuration

Support for Different CAR Versions

This extension can be connected against different versions of SAP Customer Activity Repository. As there were incompatible changes in the OAA REST service APIs, you have to set a Spring profile property in order to control the mapping to the different API versions. To do so, maintain the `spring.profiles.active` property in the `local.properties` file of your SAP Hybris Commerce installation.

Example: `spring.profiles.active=sapoaa_carApiVersionLatest`

The following profiles are supported:

Table 29:

CAR Version	Profile Value
CAR 3.0 (CARAB 2.0)	`sapoaa_carApiVersionLatest`
CAR 2.0 FP3 (CARAB 1.0 FP3)	`sapoaa_carApiVersion1`

Backoffice Configuration

For this extension, you need to make the following customizing settings in the Backoffice application of SAP Hybris Commerce.

- Enter a **consumer ID** and an **OAA profile** in the **SAP base store configuration**. These values are added to the REST service call; if the fields are initial, the REST service call fails.
- Create an **HTTP destination** and **client** for the SAP Customer Activity Repository system and add both to the **SAP global configuration**.

Offline Mode

REST service `connect` and `read timeout` parameters can be configured in the `<local.properties>` file with the `sapoaacommerceservices.rest.connectTimeout` (default 5000) and `sapoaacommerceservices.rest.readTimeout` (default 10000) properties. For more information, see the Java documentation for the `URLConnection` class.

SSL Support

For the REST service call, you also need to import the certificates from your SAP Customer Activity Repository system to your Java keystore.

Rough Stock Indicators (RSIs)

A rough stock indicator is stored using stock levels. StockLevel is extended by attribute `sapoaa_roughstockindicator`, which can contain any kind of string. Typical values could be R = Red, Y = Yellow, or G = Green. The standard SAP Hybris Commerce installation only provides rough stock values *in stock*, *no stock*, and *low stock*.

i Note

For correct calculation of rough stock indicators, it is important that only one SAP OAA virtual warehouse is assigned to a base store and that this SAP OAA virtual warehouse is set as the default warehouse. Additional warehouses may be assigned to the base store, as required for example by Asynchronous Order Management, but for the RSI functionality to work, only one may be set as the default warehouse, namely the SAP OAA virtual warehouse.

The mapping of the OAA values to the SAP Hybris Commerce values is made in class `DefaultSapOaaStockLevelStatusStrategy`. To create your own mapping, implement your own version of interface `StockLevelStatusStrategy`.

Availability in Cart

To ensure that availability information is provided in the cart, the default Hybris strategies `CommerceAddToCartStrategy` and `CommerceUpdateCartEntryStrategy` were extended to get the allowed cart adjustment for an article. An OAA-specific extension of the default Hybris commerce stock service is used to get the aggregated article availability via the REST service for ATP calculation (ATP service). The commerce stock service uses the `ATPAggregationStrategy` to calculate the aggregated quantity of available articles. For each article that is added to the cart or where the quantity is updated, the ATP service performs a synchronous REST call to get the current stock availability information from SAP Customer Activity Repository. If the requested quantity in the cart is higher than the determined available quantity, the quantity in the cart is reduced and the user is notified according to Hybris standard logic.

When navigating to **checkout** all cart items are validated to check current stock availability information for all articles. For this purpose, the `CartValidationStrategy` is also extended to perform an ATP call to check the current stock availability information. If the requested quantity in the cart is higher than the determined available quantity, the quantity in the cart is reduced and the user is notified according to Hybris standard logic. If the requested quantity of at least one item is not available, navigating to checkout is not possible.

Tip

The `CartValidationStrategy.validateCart` method is called from `CartPageController` on navigation to checkout and from `CheckoutController` when the checkout is called - this leads to two identical calls in a row. You can remove one of these calls, to reduce load on the SAP Customer Activity Repository system and to save time for the user.

The **ATP service** offers different methods to retrieve ATP information for articles and different sources via REST service calls from SAP Customer Activity Repository. The result handler checks the REST service result and - if successful - returns the result to the initiator. When the REST service returns an error or the ATP service is unavailable, an exception is raised and the current article availability cannot be determined. In this case, the requested quantity of an article is used without checking availability information.

In the default OAA implementation of the `ATPAggregationStrategy`, the quantity information is aggregated over all time series. See also modification example *Adapting ATP Aggregation Strategy*.

To **create your own strategies**, you can implement the `CommerceAddToCartStrategy`, `CommerceUpdateCartEntryStrategy`, `CartValidationStrategy` and `ATPAggregationStrategy` interfaces and update the corresponding bean references.

Availability in Store Locator

To ensure that the article is available in the preferred store, the commerce stock service returns the article stock via the ATP service. For this, the functionality used for availability in the cart is reused because the REST calls are nearly identical; only the `PointOfService` name is added in the REST URI for the GET call. This call is made for every store in the store locator result list.

Since Hybris calls are sequential calls, trying to find too many stores may lead to a performance bottleneck. As a solution, a batch REST service is available. See also modification example *Using Batch Call in Store Finder*.

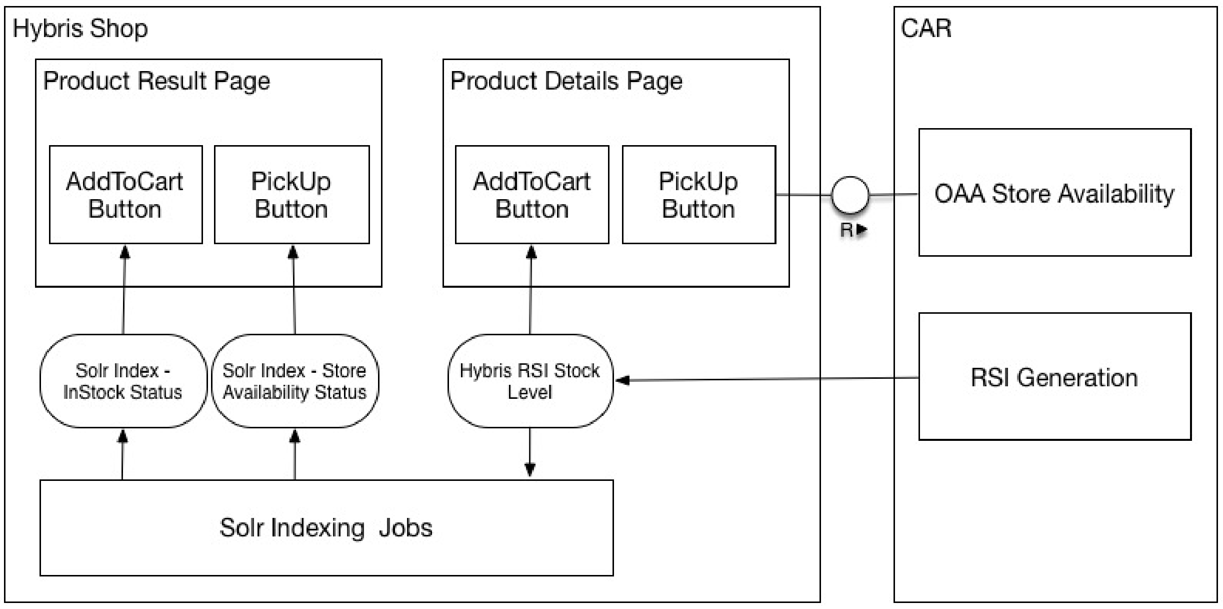
With **low stock articles**, the aggregated quantities of the rough stock indicators are used. See also modification example *Checking Current Availability in Case of Low Stock*.

Availability in Catalog and Article Details

Availability information in the catalog is retrieved from Solr as shown in the diagram below on the left-hand side (Product Result Page). The Solr information controls the *Add to cart* and *Pick up in store* buttons in the catalog.

The Solr index is filled with rough stock indicator (RSI) information via an indexing job that calls the class `RoughStockIndicatorValueResolver`. This value provider is assigned to the property `inStockFlag` of the indexed article type.

In the article details page, the *Add to cart* button is controlled by the RSI functionality directly via the SAP OAA virtual warehouse. For the *Pick up* button, a real time call to SAP Customer Activity Repository is made.



Sourcing

In the REST service for sourcing, the sourcing service is called with the current cart of the user. The result handler checks the REST service result and when successful, persists the result to the cart. If the REST service returns an error or the result is unknown (no point of service found for the base store), an exception is raised and the checkout step fails.

In the sourcing strategy, the decision is made whether or not the sourcing service is called. To **create your own sourcing strategy**, you can implement the `SourcingStrategy` interface and update the bean reference.

Reservation

In **checkout**, the sourcing service is called, and it creates a temporary reservation of type 'C' in SAP Customer Activity Repository. To update the temporary reservation after submission of an order there is an update reservation REST service that can be called with the `DefaultReservationService`.

You can also use the `DefaultReservationService` to **delete the entire reservation or a single reservation item**. You can also use the REST service to create temporary reservations (without running sourcing), but for creating and reading the reservation, no default service layer implementation is provided. The reservation is deleted when the session is closed (see below) - currently only when the customer logs out manually. The reservation item is deleted when the item is deleted in the cart. This happens in the `CommerceUpdateCartEntryStrategy`.

In the **reservation strategy**, the decision is made whether or not the reservation service is called. To create your own reservation strategy, you can implement the `ReservationStrategy` interface and update the bean reference.

OAA Stock Service

The `DefaultStockService` of SAP Hybris Commerce has been enhanced with an OAA specific service class. There is a new interface called `SapOaaCommerceStockService` - this interface offers corresponding methods to call the ATP service.

Session Close Strategy

The `sessionCloseStrategy` bean is overwritten by the `DefaultOaaSessionCloseStrategy`. When the Jalo or the HTTP session is closed, the system checks whether there is a reservation for the session cart that was not submitted, and deletes the reservation in SAP Customer Activity Repository via `DefaultReservationStrategy`. Currently, this only works when the customer logs out of the online store.

Offline Scenario

When SAP Customer Activity Repository is not responding due to a planned or unplanned downtime, the system goes into a restricted mode (offline mode) where the customer can order as much (ship to) as he wants, but cannot pick up items. Pick-up items already added to the cart are removed. When the order is submitted, the order is **not replicated** to SAP Retail. The system keeps trying to call sourcing via cron job (see backorder process in the documentation of the `sapoaaorderexchange` extension) and replicates the order only when sourcing was successful.

Dependencies

The `sapoaacommerceservices` extension depends on the following extensions:

- `sapoaamodel`: provides OAA-specific fields in the model classes that are replicated
- `sapcoreconfiguration`: provides the configuration
- `commerceservices`: standard services in SAP Hybris Commerce

3.7 sapoaamodel Extension

The `sapoaamodel` extension provides additional attributes that are required for OAA-specific processes.

Overview

This extension adds new item types, or adds attributes to existing item types, for OAA processes.

Modification of Item Types

Table 30: Modification of Item Types Made by the sapoaamodel Extension

Item Type	Modification
`AbstractOrder`	The following fields were added: *Reservation in SAP Customer Activity Repository*.
`AbstractOrderEntry`	The following fields were added: *Schedule Lines*, *SAP Source*, *SAP Vendor*, and *Reservation in SAP Customer Activity Repository*. *Schedule Lines* is a new type with the following fields: *Confirmed Date* and *Confirmed Quantity*.
`StockLevel`	For rough stock indicators, a new property was added: *Rough Stock Indicator*.
`PointOfService`	For articles to be picked up in a store, a new property was added: *Shipping Point for Click-and-Collect Scenario*.
`SAPConfiguration`	The following fields were added: *OAA Profile ID*, *Consumer ID*, and *Item Category for Drop Shipment*.
`SAPGlobalConfiguration`	For the REST service calls, a new SAPHTTPDestination was added: *SAP CAR HTTP Destination* with property *SAP CAR HTTP Client*.

Dependencies

The `sapoaamodel` extension has dependencies to the following SAP Hybris Commerce extensions:

- `sapmodel`
- `sapcoreconfiguration`
- `basecommerce`

3.8 sapoaaorderexchange Extension

The `sapoaaorderexchange` extension enhances the `saporderexchange` extension of SAP Asynchronous Order Management. It enables the replication of orders that include OAA-specific fields.

Overview

The following objects were enhanced:

Table 31: Objects Enhanced by sapoaaorderexchange Extension

Activity	Objects in SAP Hybris Commerce	Objects in SAP Retail	Direction
Order creation	Order and subsequent entries	Sales order	Outbound

Configuration

The `saporderexchange` extension must be configured.

If you want to change the date pattern that is used on the `ScheduleLines`, you have to change the `datahubadapter.datahuboutbound.date.pattern` property. Note that you have to change the property in the `saporder` extension as well, otherwise the date cannot be parsed in the SAP Hybris Data Hub.

The number of retries of the backorder process can be defined via the `saporderexchange.orderoutbound.maxRetries` property.

Enhancements

To add OAA-specific contributors to the default list of contributors in file `sapoaaorderexchange-spring.xml`, the default enhancement method of SAP Asynchronous Order Management is used. For this purpose, the `defaultSapRawHybrisOrderBuilder` bean is overwritten to add the following contributors:

- `DefaultOaaOrderContributor`: adds the Hybris GUID to the `RawHybrisOrder`, to update the temporary reservations on SAP Customer Activity Repository on SLT replication.
- `DefaultOaaOrderEntryContributor`: adds the `SAPSource` and the `ScheduleLines` to the `RawHybrisOrder`, which defines the plant (site) in the SAP Retail back end.

Schedule Lines

The schedule lines for each entry are added to a single field in the Hybris order entry. Example:

Table 32: Example of Schedule Lines

Quantity	Date
1,0	18.09.2016
4,0	20.09.2016

These schedule lines are mapped to "1,0;2016-09-18 00:00:00.0|4,0;2016-09-20 00:00:00.0".

Order Process

SAP order process `ysaporderfulfillment\process\sap-order-process.xml` is enhanced to update the temporary reservation in SAP Customer Activity Repository with the corresponding order ID from SAP Hybris Commerce. For this purpose, action class `com.sap.retail.oaa.orderexchange.actions.CheckSAPOaaOrderAction` enhances the `CheckSAPOrderAction` to call the sourcing REST service and creates a reservation of type 'O' (order) with an order ID in SAP Customer Activity Repository.The spring bean configuration can be found in file `sapoaaorderexchange/resources/sapoaaorderexchange-spring.xml`:

Source Code

```
<alias name="sapOrderexchangeDefaultCheckOrderAction"
alias="sapOrderexchangeCheckOrderAction" />
<bean id="sapOrderexchangeDefaultCheckOrderAction"
class="com.sap.retail.oaa.orderexchange.actions.CheckSAPOaaOrderAction"
parent="abstractAction">
     <property name="baseSiteService" ref="baseSiteService" />
     <property name="sourcingService" ref="oaaCommerceSourcingService"/>
</bean>
```

Backorder Processing

If SAP Customer Activity Repository is not responding and the `CheckSAPOrderAction` fails, orders remain in error state and are not replicated to SAP Retail. For the postprocessing of these orders, there is a cron job called `sapOaaOrderexchangeRepairCronJob` that handles backorder processing and that you can schedule.

The cron job calls sourcing for the order again, and when sourcing was successful, the order is replicated to SAP Retail. When sourcing was not successful, the order remains in error state and takes part in the next run of the cron job. The number of times an order takes part in backorder processing can be defined via the `saporderexchange.orderoutbound.maxRetries` property in the `sap-order-process`.

Dependencies

The `sapoaaorderexchange` extension depends on the following extensions:

- `sapoaamodel`: provides OAA-specific fields in the model classes that are replicated
- `saporderexchange`: provides the default order replication that is enhanced
- `ysaporderfulfillment`: required due to SAP order process enhancement

3.9 sapoaabackoffice Extension

The `sapoaabackoffice` extension provides the user interface configurations for OAA in the Backoffice application of SAP Hybris Commerce.

Overview

This extension defines the structure of the *Omnichannel Article Availability and Sourcing* tab page and section in the Backoffice application of SAP Hybris Commerce. Furthermore, it adds configuration attribute texts and translations for the following attributes, which are part of the `sapoaamodel` or `sapoaabackoffice` extensions:

- *Schedule Lines* (`sapoaamodel`)
- *SAP Source* (`sapoaamodel`)
- *SAP Vendor* (`sapoaamodel`)
- *Reservation in SAP Customer Activity Repository* (`sapoaamodel`)
- *Omnichannel Article Availability and Sourcing* (`sapoaabackoffice`)

Dependencies

The `sapoaabackoffice` extension has dependencies to the following SAP Hybris Commerce extensions:

- `sapoaamodel`
- `platformbackoffice`

4 Omnichannel Promotion Pricing

Omnichannel promotion pricing provides a central price and promotion repository and a promotion pricing service for the calculation of effective sales prices in all sales and communication channels.

Today, the use of multiple price calculation functionalities in different sales applications and different repositories can result in inconsistent prices across the sales channels. This also leads to massive development and test effort when introducing new pricing rules. For your customers, this means that no seamless omnichannel buying experience is provided, which has a profound effect on customers' satisfaction and loyalty.

SAP's vision is to ensure correct and consistent effective sales prices across all sales channels along with the ability to introduce new pricing rule types with low implementation effort.

The first step to achieve this is to provide a central price and promotion repository as part of omnichannel promotion pricing (OPP). This repository is located within SAP Customer Activity Repository, an application that is based on SAP HANA. It contains all regular prices and OPP promotions that are used to calculate the effective sales prices in the sales application:

- **Regular prices** can originate from an SAP Retail system, an SAP ERP system, or any other non-SAP system.
- **Offers** can be maintained in SAP Promotion Management for Retail (SAP PMR) as well as in the new SAP Fiori transactional app *Manage Promotional Offers*, or they can be imported from other systems. After the offer has been created, it is transformed into a format that complies with the format of the Association for Retail Technology Standards (ARTS). An offer with this format is called **OPP promotion**.

For the calculation of the effective sales price, OPP provides the promotion pricing service.

This service calculates the price by applying pricing rules to a regular price. This price is available in realtime for the online store.

The promotion pricing service can be deployed locally or centrally for SAP Hybris Commerce:

- **Local Deployment**
  With this deployment option, regular prices and OPP promotions are replicated from the central price and promotion repository that is located in SAP Customer Activity Repository to SAP Hybris Commerce. The promotion pricing service is embedded in SAP Hybris Commerce. In this way, effective sales prices can be calculated without an additional remote system being continuously available.
- **Central Deployment**
  With this deployment option, the promotion pricing service runs on the central price and promotion repository located in SAP Customer Activity Repository powered by SAP HANA XS advanced (XSA).

Note

The following promotional features are not supported with the integration of the promotion pricing service into SAP Hybris Commerce:

- Coupons
- Loyalty points
- External action
- Customer card
- Promotions on article hierarchy nodes

For more information about OPP functionality, see the application help of SAP Customer Activity Repository on SAP Help Portal under https://help.sap.com/viewer/p/CARAB> *<Version>* › *Application Help* › *SAP Customer Activity Repository* › *Omnichannel Promotion Pricing* .

System Setup

In addition to the standard installation process of the SAP Hybris Commerce, integration package for SAP for Retail, you need the following software applications for an OPP scenario:

- **SAP Customer Activity Repository** 2.0 FP3 or higher as the home of the price and promotion repository, and the outbound functionality for the local scenario of the promotion pricing service
- **SAP Customer Activity Repository** 3.0 or higher, and SAP HANA XS advanced (XSA) v1.0.34 or higher, for the central scenario of the promotion pricing service
- SAP Fiori transactional app *Manage Promotional Offers* or SAP Promotion Management for Retail (SAP PMR) to create and maintain offers in SAP Customer Activity Repository
- **SAP S/4HANA** or **SAP Retail** as the back-end system for the regular prices that have to be uploaded to the price and promotion repository

> Note
>
> You can use a different SAP system or a non-SAP system to upload regular prices to the price and promotion repository.

For information about installation and configuration of OPP in SAP Customer Activity Repository, see the Common Installation Guide of SAP Customer Activity Repository applications bundle on SAP Help Portal under https://help.sap.com/viewer/p/CARAB> *<Version>* › *Installation and Upgrade* › *Common Installation Guide* › *Post-Installation* › *SAP Customer Activity Repository* › *Configure Omnichannel Promotion Pricing for Use with SAP Customer Activity Repository* .

Related Information

Configuration [page 86]
sapppspricing Extension [page 90]
Security Information [page 116]

4.1 Major Functional Enhancements

SAP Hybris Commerce, integration package for SAP for Retail 2.1

Version 2.1 of SAP Hybris Commerce, integration package for SAP for Retail comprises the following major new features:

Table 33:

Feature	Description	More Information
Local scenario of the promotion pricing service	With this scenario, regular prices and OPP promotions are replicated from the central price and promotion repository, located in SAP Customer Activity Repository, to SAP Hybris Commerce. The promotion pricing service is embedded in SAP Hybris Commerce. In this way, effective sales prices can be calculated without an additional remote system being continuously available.	The required product version of CARAB for this scenario is CARAB 1.0 FP3 or higher. For more information, see the corresponding OPP release note for SAP Customer Activity Repository.
Central scenario of the promotion pricing service	With this scenario, the promotion pricing service runs on the central price and promotion repository located in SAP Customer Activity Repository. The service is powered by SAP HANA XS advanced (XSA).	The required product version of CARAB for this scenario is CARAB 2.0 or higher. For more information, see the corresponding OPP release for SAP Customer Activity Repository.

4.2 Configuration

Before you can use the local promotion pricing service, you have to configure the data replication to send IDocs from SAP Customer Activity Repository to SAP Hybris Commerce. To delete OPP promotions and regular prices stored in the local database in SAP Hybris Commerce, you can create and configure two cron jobs. Additionally, you need to configure the promotion pricing service in the Backoffice application of SAP Hybris Commerce.

Configuration of Data Replication

For the local deployment of the promotion pricing service, you have to replicate regular prices and OPP promotions from the central price and promotion repository to SAP Hybris Commerce. For more information about the configuration of the outbound processing of regular prices and OPP promotions via IDocs, see SAP Note 2363312, or the Common Installation Guide of SAP Customer Activity Repository applications bundle under https://help.sap.com/viewer/e38aeef2739c45e89a87cb999f54972c/2.0.1/en-US/7b006dfe1f6947d6bc271b9faf7a1313.html .

Configuring IDoc Inbound Processing

To protect your SAP Hybris Commerce installation from external malicious data, all received IDocs need to be authorized. The authentication of the IDoc inbound processing uses basic authentication with user and password. To be able to send regular prices and OPP promotions with the related IDoc types from SAP Customer Activity Repository to SAP Hybris Commerce and to process these IDocs successfully in SAP Hybris Commerce, a user must be maintained that fulfills the following requirements:

- It must be of type *Employee*.
- It must be assigned to user group `sap_pps_idocinbound`.
- It must not be locked.
- A password must be maintained for the user.

The user group `sap_pps_idocinbound` is created automatically during initialization. Additionally, an example user `sap_pps_idocuser` is created that you can use as a template to create your own users. This example user is locked by default and there is no password maintained.

If this data is not created automatically during initialization, perform the following steps to run a system update in Hybris administration console (HAC):

1. Start Hybris administration console (HAC) and choose *Platform* > *Update*.
2. Select the checkbox *Create essential data* and choose *Update*.
   After the update, the user group `sap_pps_idocinbound` and a user with ID `sap_pps_idocuser` assigned to it are created.

Note

For security reasons, this new user is disabled by default. To enable the user, deselect the checkbox *Login disabled* in the *Password* tab and assign a valid password to the user in the *Password* tab.

Configuring Deletion Functionality

Reorganisation of data is important to keep performance stable. If you use the local deployment of the promotion pricing service, OPP promotions and regular prices are stored in a local database in SAP Hybris Commerce. To remove obsolete OPP promotions and regular prices, the following Hybris job definitions are offered in the `sapppspricing` extension:

Table 34: Deletion Job Definitions

Job Definition	Description
`sapDefaultDeletePromotionsJob`	With this job definition, you can delete OPP promotions with the following parameters: • *Number of Days Since Expiry Date* If this parameter is set, only those OPP promotions are deleted that have expired since at least the specified number of days. This parameter may also be 0. • *Also Delete Active OPP Promotions* If this parameter is set, OPP promotions are deleted regardless of their status (active, inactive, logically deleted). If this parameter is not set, active OPP promotions are not deleted.
`sapDefaultDeletePricesJob`	With this job definition, you can delete regular prices with the following parameters: • *Number of Days Since Expiry Date* If this parameter is set, only those regular prices are deleted that have expired for more than the specified number of days. This parameter may also be 0. For example, if you enter 10 days, all regular prices are deleted with a valid-to date that lies more than 10 days in the past.

To schedule the deletion of obsolete data, you have to create and configure the corresponding cron jobs. In the Backoffice application of SAP Hybris Commerce, you can choose the following cron jobs:

Table 35: Deletion Cron Jobs

Cron Job	Description
`DeletePromotionsCronJob`	With this cron job, you can delete OPP promotions. To configure this cron job, you have to enter both parameters for OPP promotions and determine when the job is to be exectued.
`DeletePricesCronJob`	With this cron job, you can delete regular prices. To configure this cron job, you have to enter the parameter for regular prices and determine when the job is to be executed.

Note

If you define a cron job based on the Hybris model, you can define or assign a user for which this job is to be executed. Each user must be assigned to the user group `sap_pps_cronjob`. If the user is not assigned to this group, the cron job cannot be executed.

Configuration of Promotion Pricing Service

In the Backoffice application of SAP Hybris Commerce, choose *SAP Integration* > *SAP Base Store Configuration* > *<name of the SAP base store configuration>*, and specify the following parameters in the *Omnichannel Promotion Pricing* tab to configure your promotion pricing service:

Table 36: PPS Configuration in Backoffice

Fields	Description
Name	Enter the name of the SAP base store configuration
Use PPS	Select *True* to use the promotion pricing service. With this parameter, prices and promotions are retrieved from a central price and promotion repository and the effective prices are calculated by calling the promotion pricing service. Note If you do not activate this parameter, the following parameters have no effect.
Business Unit ID	With this parameter, you can identify the location of the online store in the system. For example, you can define a location for stores and online stores by entering a location ID in SAP Customer Activitty Repository. If you use the promotion pricing service with SAP Hybris Commerce, the business unit ID specifies for which specific locations regular prices and OPP promotions are requested. Note The business unit ID corresponds to the location ID that you defined in SAP Customer Activity Repository.
Cache Catalog Prices	Select *True* to store catalog prices for quick access and increase the performance of your online store. For example, you can display catalog prices by retrieving the prices from a cache instead of calling the promotion pricing service.
Deployment Scenario	Select a scenario to deploy your promotion pricing service.

Fields	Description
HTTP Destination for Central PPS	Enter an HTTP destination. With this parameter, you specify the source from which the promotion pricing service can be called. If you want to deploy your service centrally, choose *SAP Integration* > *HTTP Destination*, and define the HTTP destination for your promoiton pricing service. Note It is only necessary to define an HTTP destination if you choose the central deployment scenario to configure your promotion pricing service.

Related Information

Configuration of the Promotion Pricing Service [page 108]
sapppspricing Extension [page 90]

4.3 sapppspricing Extension

The `sapppspricing` extension provides omnichannel promotion pricing (OPP) functionality in SAP Hybris Commerce.

With this extension, you can use the promotion pricing service to calculate effective sales prices for catalog items and for shopping carts in SAP Hybris Commerce.With `sapppspricing` the following features are supported:

- OPP promotions on product level
- OPP promotions on merchandise category hierarchy nodes

  Note

  The merchandise category hierarchy must have been imported from SAP ERP.
- OPP promotions based on cart total
- Automatic distribution of header discounts to items. For example, discounts on transaction level are distributed to item level.
- Automatic conversion of percentage discounts to the absolute values to facilitate further processing
- Access to the local promotion pricing service by replacing the standard Hybris functionality for the price calculation
- Access to a central promotion pricing service
- Caching of catalog prices for improved performance, in particular for a central promotion pricing service
- Calculation of catalog prices and prices in a shopping cart

Note

For the price and promotion calculation of shopping carts, the promotion pricing service only calculates the effective sales prices including discounts on item level. The structure of the shopping cart does not change. Therefore, only discounts per shopping cart entry are available with SAP Hybris Commerce, discounts per single piece are not supported.

Note

- The PPS based price calculation and the SAP ERP based price calculation are not supported in the same SAP Hybris installation.
- Information other than discount information about applied OPP promotions is not used in SAP Hybris Commerce.

Related Information

Integration into SAP Hybris Commerce [page 91]
Running the PPS-Based Price Calculation [page 103]
Extending the Price Calculation [page 106]
Configuration of the Promotion Pricing Service [page 108]
Provided Spring Beans [page 112]
Dependencies [page 115]

4.3.1 Integration into SAP Hybris Commerce

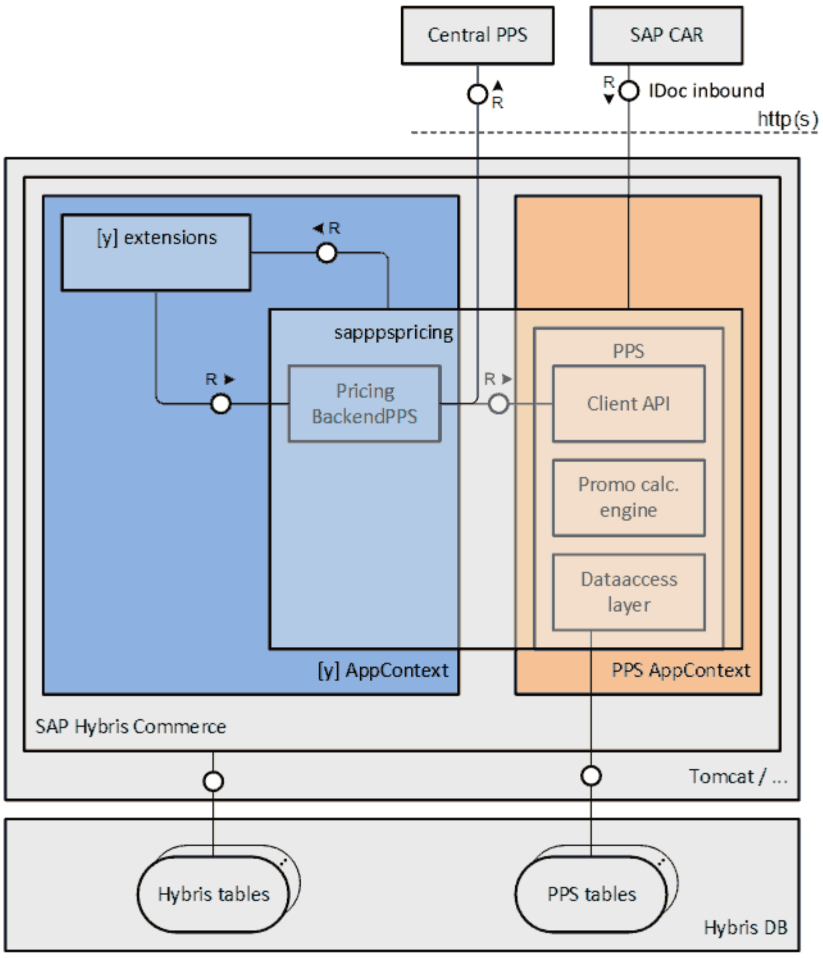
Promotion Pricing Service as a Black Box

The promotion pricing service (PPS) allows a flexible deployment on many platforms and does not rely on platform-specific features. However, the architecture of SAP Hybris Commerce is based on the so-called Hybris extension concept that is described in the official Hybris documentation.

The Hybris extension `sapppspricing` is introduced for the following purposes:

- It acts as wrapper for the PPS and allows the deployment of the PPS in SAP Hybris Commerce.
- It redirects calls for the calculation of catalog prices and cart prices in SAP Hybris Commerce to the promotion pricing service.

The following figure shows a simplified view of the inner structure of the `sapppspricing` extension:



From a Hybris core extension perspective, `sapppspricing` looks like an ordinary extension. However, it contains a separate application (the PPS) that is located in its own Spring application context. This context is not connected to the application context hierarchy of SAP Hybris Commerce. This enables good decoupling of SAP Hybris Commerce and PPS artifacts.

Nevertheless, the local PPS shares its classpath with Hybris extensions, which makes it possible to internally access classes that are offered by the local PPS. From a file perspective, the local PPS consists of a set of JARs within the `lib/` and `WEB-INF/lib` folders of the `sapppspricing` extension.

`sapppspricing` contains both, the local PPS and the classes that redirect the price calculation to the PPS. Therefore, this extension contains a service provider and a service consumer. The required price calculation artifact and the concepts to be applied are mapped as follows:

- Artifacts that are needed to make the PPS work logically belong to the service provider side. Hence the PPS concepts are applied to allow flexible deployment.
  For example, the inbound processing of regular prices and OPP promotions does not use a Data Hub extension and therefore makes no use of impex. In particular, the Spring beans relevant for the inbound processing are located within the PPS application context. Also, the database tables used by PPS do not use the Hybris `items.xml` concept.
- Artifacts that are needed to call the PPS logically belong to the service consumer side. Hence the SAP Hybris Commerce concepts are applied.

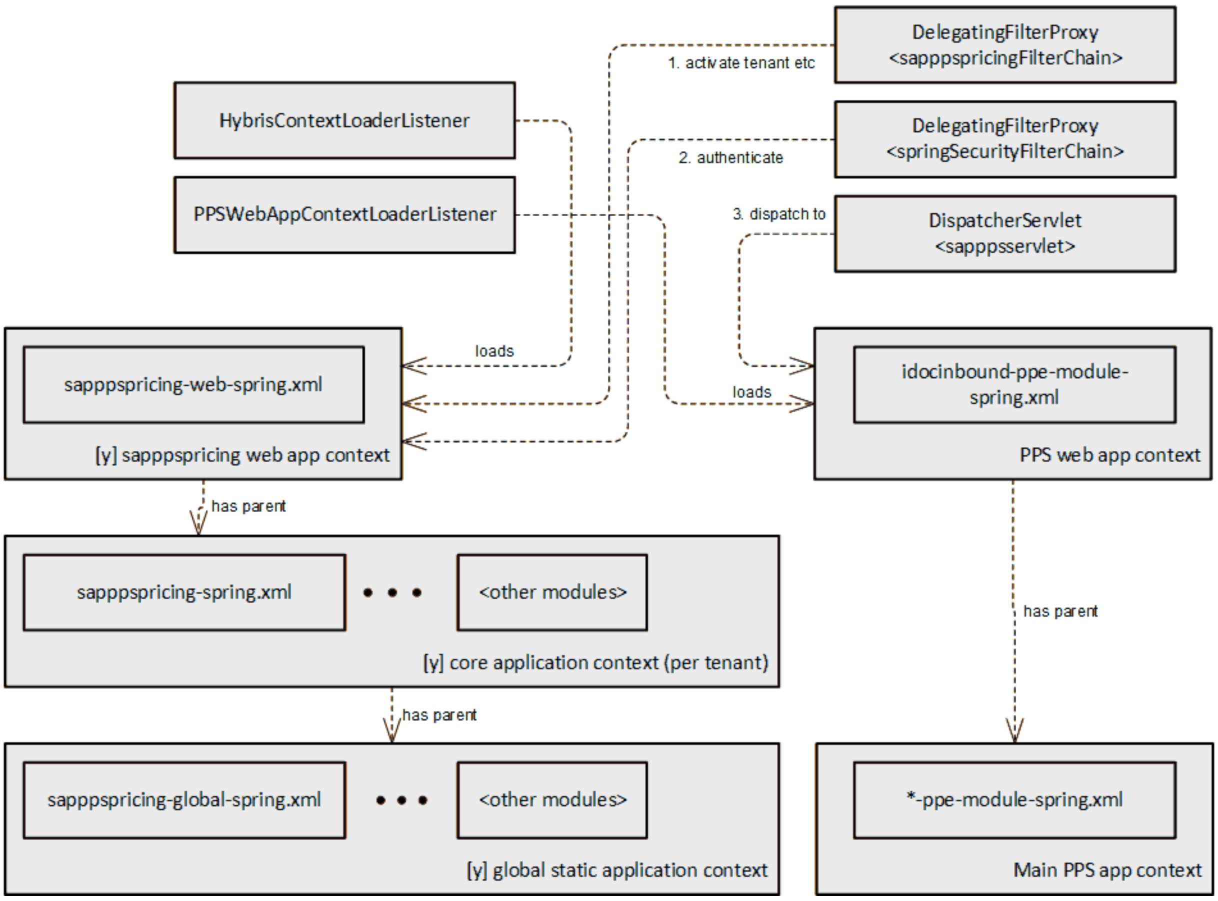
This means thatthe Spring beans for redirecting the price calls of SAP Hybris Commerce are located in the SAP Hybris Commerce application context hierarchy. SAP Hybris Commerce and PPS use the same database to store regular prices and OPP promotions. `sapppspricing` is defined as a web module, which means that it has its own Web application context. This is required to process incoming IDocs.

Used Application Contexts and Bootstrapping

The `sapppspricing` extension uses the following types of application contexts:

- Hybris context
- PPS application context

The following figure shows these application contexts and how they are used during HTTP request processing:


DelegatingFilterProxy <sapppspricingFilterChain>
1. activate tenant etc
DelegatingFilterProxy <springSecurityFilterChain>
2. authenticate
DispatcherServlet <sapppsservlet>
3. dispatch to
HybrisContextLoaderListener
PPSWebAppContextLoaderListener
loads
loads
sapppspricing-web-spring.xml
[y] sapppspricing web app context
idocinbound-ppe-module-spring.xml
PPS web app context
has parent
sapppspricing-spring.xml
<other modules>
[y] core application context (per tenant)
has parent
has parent
sapppspricing-global-spring.xml
<other modules>
[y] global static application context
*-ppe-module-spring.xml
Main PPS app context

The boxes on the left represent the Hybris application context hierarchy. The `sapppspricing` extension has beans that are defined on all 3 levels of this hierarchy:

- In the cross-tenant global static application context, a cache region is introduced for caching catalog prices calculated with the PPS. Here, also the creation of the main PPS application context is triggered.
- In the tenant-specific core application context, the actual business logic of `sapppspricing` is located, for example, the service layer that offers the price calculation in SAP Hybris Commerce.
- In the web application context `sapppspricing` contains beans that are required for authenticating the incoming HTTP requests.

Note

The Hybris application context hierarchy is launched via the class `HybrisContextLoaderListener`.

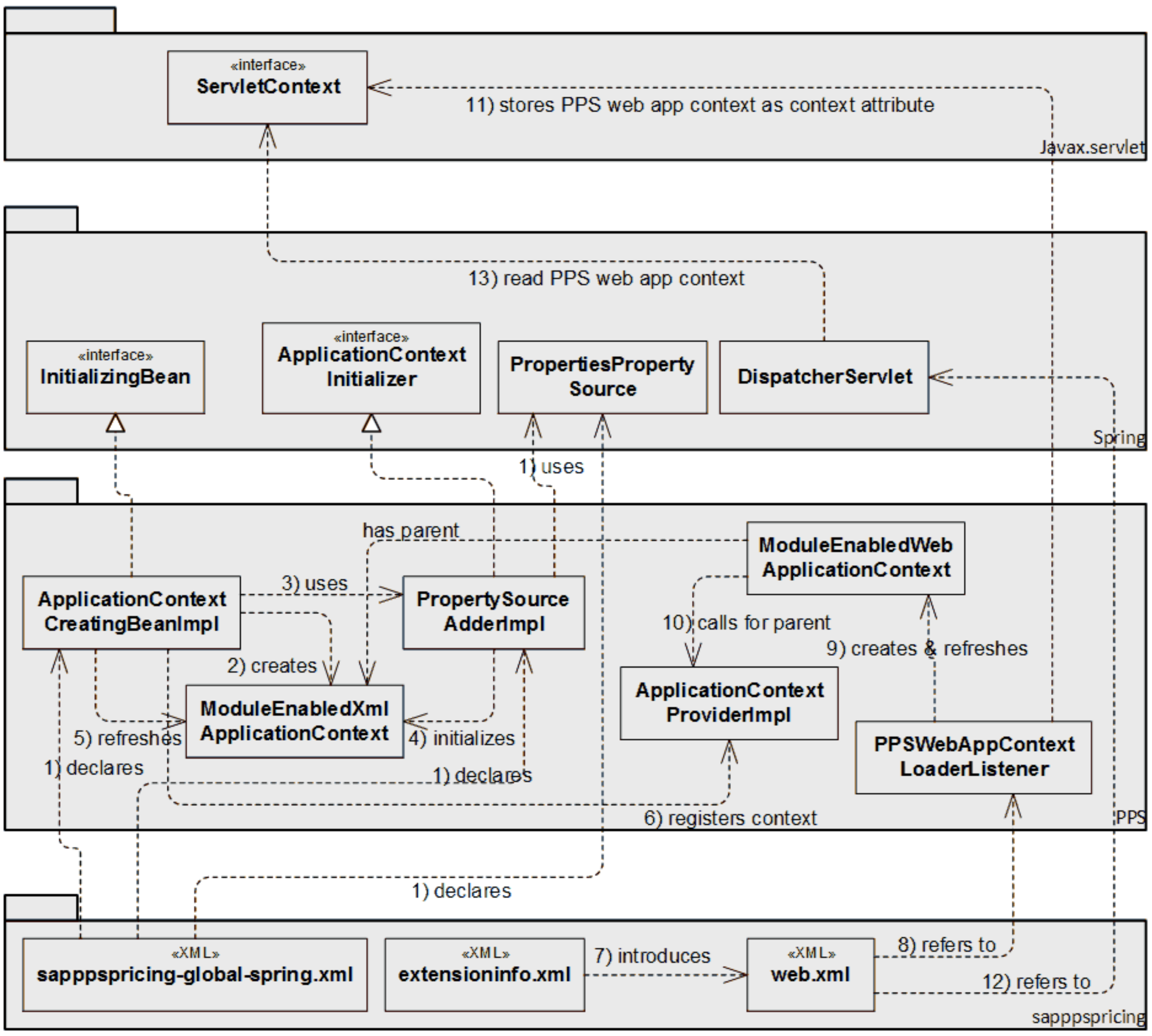
The boxes on the right represent the PPS application context, which is a 2-level hierarchy:

- The majority of the business logic is located in the PPS app context, in the beans specified in the `*-ppe-module-spring.xml` files.
- This is the parent of a web application context that is required as a target for the web requests dispatched by the `DispatcherServlet` specified in the `sapppspricing web.xml` file. Additionally, this web application context contains the beans required for IDoc inbound processing.

Note

The PPS application context hierarchy is launched via the class `PPSWebAppContextLoaderListener`.

The following figure gives a detailed overview of how the PPS application context hierarchy is created during startup:



«interface» ServletContext
11) stores PPS web app context as context attribute
Javax.servlet
13) read PPS web app context
«interface» InitializingBean
«interface» ApplicationContext Initializer
PropertiesProperty Source
DispatcherServlet
Spring
1) uses
has parent
ModuleEnabledWeb ApplicationContext
ApplicationContext CreatingBeanImpl
3) uses
PropertySource AdderImpl
10) calls for parent
9) creates & refreshes
2) creates
ModuleEnabledXml ApplicationContext
ApplicationContext ProviderImpl
5) refreshes
4) initializes
PPSWebAppContext LoaderListener
1) declares
1) declares
6) registers context
PPS
1) declares
«XML» sapppspricing-global-spring.xml
«XML» extensioninfo.xml
7) introduces
«XML» web.xml
8) refers to
12) refers to
sapppspricing

For the creation of the PPS application context hierarchy, the following steps are performed with the startup of the application. These steps refer to the steps in the figure above:

1. The spring context resource file `sapppspricing-global-spring.xml` contains the following beans that are relevant for the startup of the PPS application context:
    - `ApplicationContextCreatingBeanImpl`
      This bean implements the Spring interface `InitializingBean`. This interface ensures that some logic implemented in this bean is executed automatically during startup of the Hybris application context.
    - Beans that declare an additional Spring `PropertiesPropertySource`
      These beans allow the injection of properties that are available during the creation of the Hybris application context into the PPS application context.
      The following database connection parameters are available:

      Source Code

      ```
      <alias name="sapDefaultPPSHybrisPropertySource"
      alias="sapPPSHybrisPropertySource" />
      <bean id="sapDefaultPPSHybrisPropertySource"
      class="org.springframework.core.env.PropertiesPropertySource">
          <constructor-arg value="hybrisSource" />
          <constructor-arg>
              <map>
                  <!-- Take over DB connection parameters of hybris as long as
      no sapppspricing specific properties are set -->
                  <!-- Note: sapppspricing.db.* properties are NOT FOR
      PRODUCTIVE USE! -->
                  <entry key="sap.dataaccess-common.db.url" value="$
      {sapppspricing.db.url:${db.url}}" />
                  <entry key="sap.dataaccess-common.db.userName" value="$
      {sapppspricing.db.username:${db.username}}" />
                  <entry key="sap.dataaccess-common.db.passWord" value="$
      {sapppspricing.db.password:${db.password}}" />
                  <entry key="sap.dataaccess-common.db.driverClassName" value="$
      {sapppspricing.db.driver:${db.driver}}" />
                  <entry key="sap.dataaccess-common.db.client" value="$
      {sapppspricing.dbclient}" />
                  <entry key="sap.core.ppsconfiglocation" value="$
      {sapppspricing.pps.configfile}" />
                  <entry key="sap.sapppspricing.db.dstype" value="#{ '$
      {db.pool.fromJNDI}'.isEmpty() or '${db.pool.fromJNDI}'.startsWith('$') ?
      'Direct' : 'Jndi'}" />
                  <entry key="sap.sapppspricing.db.jndiname" value="#{ '$
      {db.pool.fromJNDI}'.isEmpty() or '${db.pool.fromJNDI}'.startsWith('$') ?
      'dummy' : '${db.pool.fromJNDI}'}" />
              </map>
          </constructor-arg>
      </bean>
      ```

2. After being instantiated, the `ApplicationContextProvidingBeanImpl` creates the PPS application context.
3. After creation of the application context, the `ProperySourceAdderImpl` is used for this context.
4. The additional configuration properties are injected into the PPS application context.
5. The main PPS application context is refreshed and ready to use.The additional configuration properties are injected into the PPS application context.
6. This context is registered in `ApplicationContextProviderImpl`. Therefore, it can be accessed from anywhere by the application logic.
7. The extension `sapppspricing` is declared as a web module. Therefore, the extension has its own `web.xml` file.

8. In `web.xml` the listener `PPSWebAppContextLoaderListener` is registered. This listener is invoked automatically during startup of the `sapppspricing` web application. The startup is triggered after the Hybris global static application context has been initialized. Therefore, the main PPS application context has been initialized, since this action is triggered during the startup of the Hybris global static application context.
9. When invoked, this listener creates a `ModuleEnabledWebApplicationContext` that is required in a web application context. This context supports the declaration of additional modules that are not yet present in the main application context. The module `idocinbound` is located in this context.
10. When the web application context is refreshed, it calls `ApplicationContextProviderImpl` to get access to the main PPS application context that is automatically considered as parent content.
11. This web application context is stored as an attribute in the servlet context of the web application.
12. The `web.xml` declares a servlet implemented by the Spring `DispatcherServlet`. This dispatches the HTTP requests into the PPS web application context.
13. Access to the PPS web application context is provided by specifying the name of the servlet context attribute that holds a reference to this context.

The distribution of the PPS modules to separate application contexts (the main PPS application context and the web application context) is performed because of the dependencies of the `idocinbound`The additional configuration properties are injected into the PPS application module to additional open source libraries. To avoid side effects, only the classpath of the corresponding web application for the IDoc inbound is enhanced. The classpath of the hybris platform below is not enhanced.

Deployment Options

`sapppspricing` is prepared to use a central PPS and a local PPS. These deployment options have advantages and disadvantages in terms of the following aspects:

Table 37: Aspects of Local and Central Deployment

Aspect	Local PPS	Central PPS
Processing time	Each request only involves a direct Java method call. There is no remote communi-cation needed.	Each request involves HTTP(S) commu-nication with payload conversion, net-work delay, and authentication.
Data redundancy	Regular prices and OPP promotions are copied to the local repository in SAP Hybris Commerce.	Regular prices and OPP promotions are calculated with the original data of the central price and promotion repository.
Changing price information	Changed price and promotion information is only available after the data replication to the local repository. For the latest price in-formation, it is necessary to schedule up-dates regulary.	Changed price information is available immediately. There is no data replication needed. **Note** Delayed visibility can only occur due to caching.

Aspect	Local PPS	Central PPS
Availability of the online store	The online store does not need an additional external system to work properly.	The online store needs an additional external service to work properly.
Hybris tenant support	There is only one PPS per tenant. Therefore, the Hybris tenant concept is not supported.	A base store can have its own configuration per tenant. This also applies for the OPP configuration including the HTTP destination of the central PPS. Therefore, tenant-specific configuration of the PPS is supported.

New Price Calculation Services

For the integration into SAP Hybris Commerce, `sapppspricing` overrides 3 central services:

- `PriceService` to calculate prices for products in the catalog
- `CalculationService` to calculate a shopping cart
- `PromotionsService` to apply promotions in the shopping cart

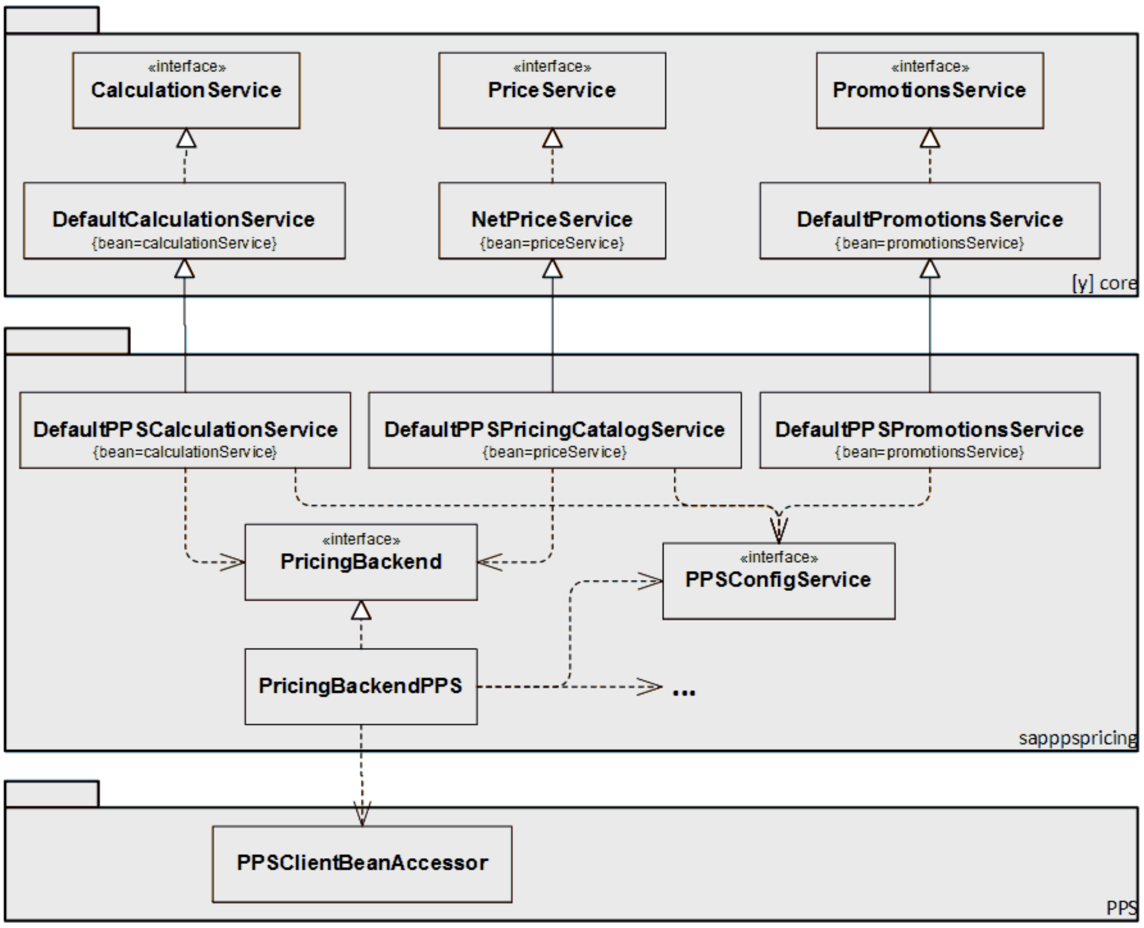
The default implementation of the `PromotionsService` is replaced by a dummy implementation, effectively deactivating the Hybris-based promotions. The PPS always considers OPP promotions maintained in the central repository. The Hybris-based promotions may not be used.

The default implementations of the`PriceService` and the `CalculationService` are adjusted so that price and discount calculation are not based on the Hybris `PriceFactory` anymore, but make use of the PPS. The price calculation via PPS is offered via the interface `PricingBackend`, having `PricingBackendPPS` as the default implementation. This realizes the transition between the Hybris and the PPS domain. Calling PPS-specific functionality (in particular the local PPS) is realized by calling the Spring beans of the PPS application context, which are offered via the `PPSClientBeanAccessor` interface.

Note

PPS-based price calculation is only effective if activated in the Backoffice application of SAP Hybris Commerce. Read access to the configuration is offered by the `PPSConfigService`.

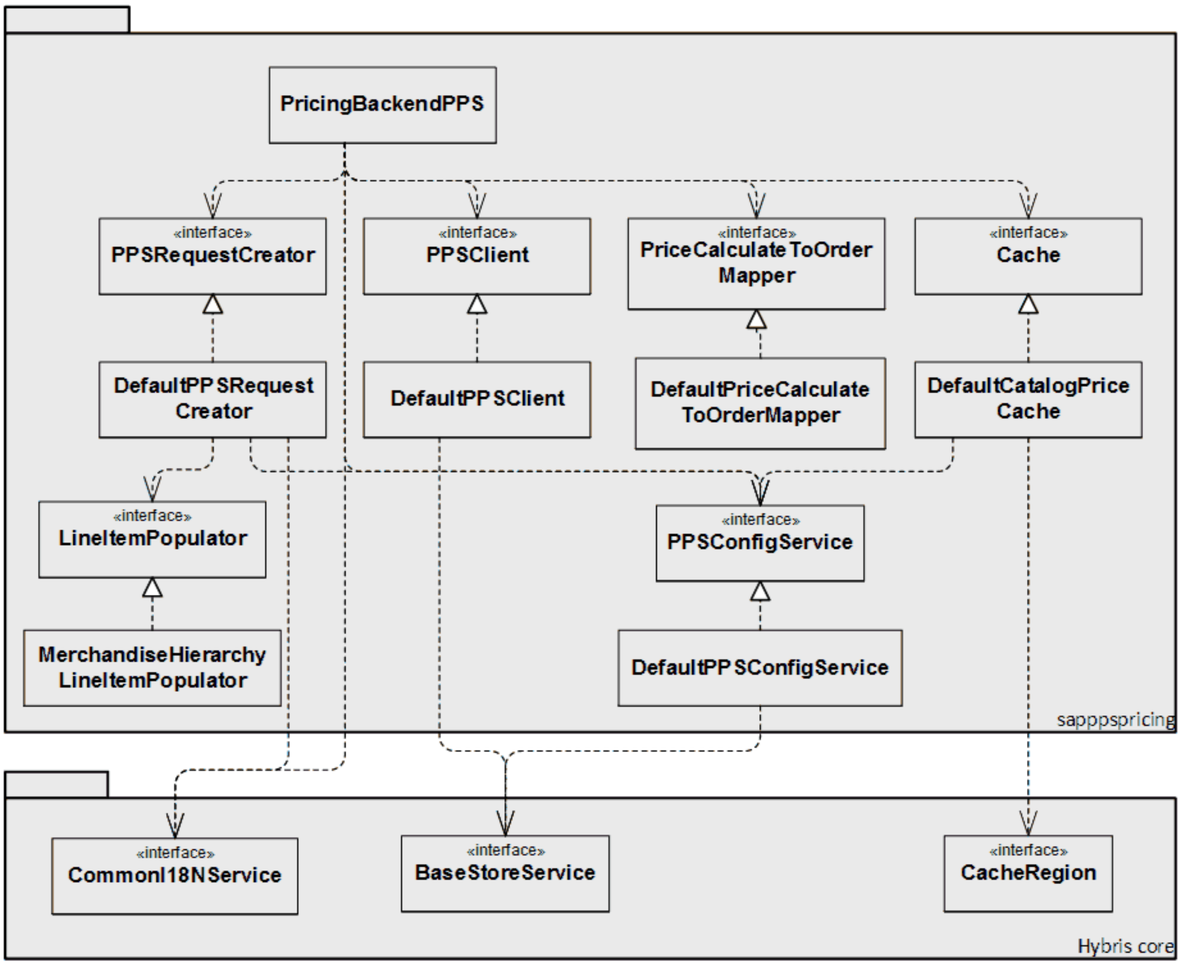
The following figure shows the replacement of the Hybris default implementations with the new implementaions of `sapppspricing`:



«interface» CalculationService
«interface» PriceService
«interface» PromotionsService
DefaultCalculationService {bean=calculationService}
NetPriceService {bean=priceService}
DefaultPromotionsService {bean=promotionsService}
[y] core
DefaultPPSCalculationService {bean=calculationService}
DefaultPPSPricingCatalogService {bean=priceService}
DefaultPPSPromotionsService {bean=promotionsService}
«interface» PricingBackend
«interface» PPSConfigService
PricingBackendPPS
...
sapppspricing
PPSClientBeanAccessor
PPS

Inner Structure of the Price Calculation

The following figure shows the inner structure of `sapppspricing` and its dependencies to Hybris core functionality:



PricingBackendPPS
«interface» PPSRequestCreator
«interface» PPSClient
«interface» PriceCalculateToOrder Mapper
«interface» Cache
DefaultPPSRequest Creator
DefaultPPSClient
DefaultPriceCalculate ToOrderMapper
DefaultCatalogPrice Cache
«interface» LineItemPopulator
«interface» PPSConfigService
MerchandiseHierarchy LineItemPopulator
DefaultPPSConfigService
sapppspricing
«interface» CommonI18NService
«interface» BaseStoreService
«interface» CacheRegion
Hybris core

The main class `PricingBackendPPS` delegates the work to the following more specialized classes:

Table 38: Subordinate Classes of PricingBackendPPS

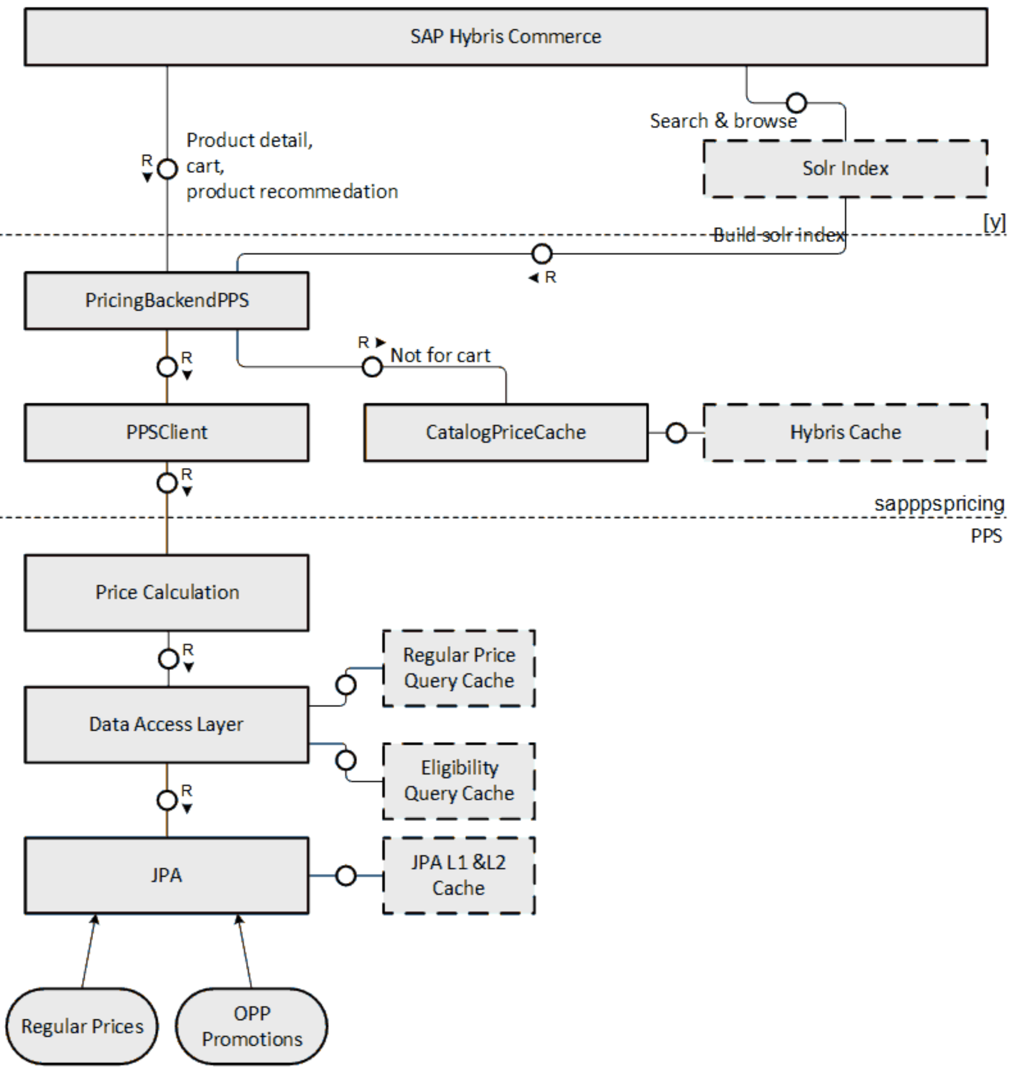
Class	Description
`DefaultPPSRequestCreator`	This class creates the request to be sent to the PPS. It adds the basic information to the request (item ID, quantity, unit of measure, and so on), and additionally supports several `LineItemPopulator`. In the standard shipment this class includes the `MerchandiseHierarchyLineItemPopulator`, which adds merchandise category hierarchy information to the request per item.

Class	Description
`DefaultCatalogPriceCache`	This class caches catalog prices calculated via PPS. Not every call to the `PriceService` interface of Hybris leads to a PPS call. This class mainly creates the cache keys and executes the access to a Hybris `CacheRegion` reserved for catalog prices calculated via PPS. For more information about caching, see section *Caching*.
`DefaultPPSClient`	This class sends the created request to the central or local PPS and receives the answer from PPS.
`DefaultPriceCalculateToOrderMapper`	This class maps the response of a price calculation request to PPS to the hybris order or cart. Catalog prices are mapped within `PricingBackendPPS` directly.
`DefaultPPSConfigService`	This class accesses the `sapppspricing`-specific configuration for the PPS that is part of `SAPConfigurationModel`. The `SAPConfigurationModel` can be assigned on base store level.

The current language and currency code are provided via the `CommonI18NService`. This is needed to receive promotion information in the correct language and to verify that the returned amounts have the expected currency. The `BaseStoreService` provides access to the `SAPConfigurationModel` that contains the `sapppspricing` configuration. In addition, it provides the ID of the current base store that is used as a key for cookies returned by a central PPS.

Caching

The following figure gives an overview of how prices and promotions are cached when using the PPS for price calculation:



The Solr index is used during catalog browsing and product search. It alsocontains price information. In the Solr index, no OPP-related changes were made.

The `sapppspricing` functionality to determine catalog prices is used in the following contexts:

- On the product details page
- For product recommendations
- For building the content of the Solr index

This offers the possibility to cache price calculation results using existing Hybris infrastructure. Therefore, a new cache region is used. These prices are only used in a catalog context. This cache is in particular useful in a centrally deployed PPS. A cache key is used, which is the internal primary key of the product. This key is specific to the Hybris catalog version. For calculating prices in a shopping cart, no caching is offered. In case of shopping cart calculation, order price calculation, or catalog prices that are not contained in the cache, a price calculation via PPS is triggered. In PPS, caches are used for the following purposes:

- The regular prices of products are cached using Spring cache abstraction
- The results of the search for promotional eligibilities based on request content that are cached by using Spring cache abstraction
- The OPP promotions are cached using level 1 and 2 object caching provided by the JPA provider (`EclipseLink`)

For more information, see the Development and Extension Guide that is part of the software artifact of SAP Hybris Commerce, integration package for SAP for Retail .

Hybris Tenants

PPS does not support the concept of Hybris tenants.Therefore, only one instance of the local PPS is launched during the startup of SAP Hybris Commerce.

However, if you choose the central PPS, the configuration data is stored per tenant and you can specifiy a different central PPS per tenant. As described in section *Caching* of chapter Integration into SAP Hybris Commerce [page 91], the result of the catalog price calculation can be stored in a cache region offered by Hybris. In this cache the data is stored per tenant.

Database Connections

In SAP Hybris Commerce, the implementation of a `javax.sql.DataSource`, which represents the database connection, can be tenant-specific. Therefore, the local PPS uses its own logic to access a `java.sql.DataSource` implementation that typically includes the pooling of database connections.

This is done by introducing another PPS module called `sapppspricing`, which is located in the `sapppspricing` extension (files `sapppspricing-ppe-module-metadata.xml` and `sappspricing-ppe-module-spring.xml`). This only adds the `java.sql.DataSource` implementation to be used for the database connection. The following two options are available to specify the database connection:

- **A direct specification of the database connection parameters**
  With this option, the Hybris database connection parameters are also used by default for the local PPS. If you need to, for example, specify a different database schema, these Hybris properties can be overruled by the following `sapppspricing`-specific properties:

  Table 39: sapppspricing-Specific Properties

Hybris Global DB Properties	sapppspricing DB Properties
`db.url`	`sapppspricing.db.url`
`db.username`	`sapppspricing.db.username`
`db.password`	`sapppspricing.db.password`

Hybris Global DB Properties	sapppspricing DB Properties
`db.driver`	`sapppspricing.db.driver`

- **A specification of the DataSource by its JNDI name as specified in property** `db.pool.fromJNDI`
  If this property is set, it takes precedence over the direct specification of database connection parameters. The logic for the corresponding `javax.sql.DataSource` is implemented using Spring Expression Language in `sapppspricing-global-spring.xml` to determine `DataSource` type, and `sapppspricing-ppe-module-spring.xml` to set bean alias `sapDataSource` accordingly.

Authentication for the Central Promotion Pricig Service

For the central PPS, authentication is strongly recommended. The corresponding HTTP connection properties are maintained via an SAP HTTP Destination in the Backoffice application of SAP Hybris Commerce. This HTTP destination is assigned to the `sapppspricing`-specific configuration.

Currently, only basic authentication is supported. However, the PPS client makes use of JESSIONIDs that are generated by the server. IDs returned by the server are sent back with the next request. This may help to skip authentication on server side.

Tip

The HTTP sessions are shared across SAP Hybris Commerce sessions, for example, the price requests from different end users will reuse the same session. However, the HTTP sessions for different base stores are kept separate.

Related Information

Extension Concept in SAP Hybris Commerce

4.3.2 Running the PPS-Based Price Calculation

IDoc Inbound Processing

In order to run a local PPS, SAP Hybris Commerce must be supplied with regular prices and OPP promotions. For this purpose, OPP offers an IDoc inbound interface. With this, IDocs can be processed if they are sent as HTTP(S) POST requests with an XML payload. Therefore, OPP supports HTTP basic authentication. IDoc inbound processing is offered by the PPS module `idocinbound`. For more information about this PPS module, see the

Development and Extension Guide that is part of the software artifact of SAP Hybris Commerce, integration package for SAP for Retail.

Note

The following is not supported with OPP:

- The import of regular prices and OPP promotions via Data Hub
- SOAP protocol

In the standard delivery, the web service for receiving OPP IDocs can be reached as follows:

Table 40: URLs for OPP IDoc Web Service

Protocol	URL
HTTP	http://<server>:9001/sapppspricing/idocinbound
HTTPS	https://<server>:9002/sapppspricing/idocinbound

The processing of incoming HTTP requests is split as follows:

- The authentication of the request for the corresponding URL
- The inbound processing of the IDocs

Dispatching the request to the PPS application context for inbound processing is done via the Spring The authentication of the HTTP request uses the Hybris user management in the Hybris application context. The authentication of a Hybris user is described in the Configuration documentation of the IDoc inbound processing`DispatcherServlet`. This class uses the web application context stored in Servlet context attribute `SAP_PPS_WEBAPPCONTEXT`. This attribute is populated by the listener `PPSWebAppContextLoaderListener` that creates the PPS web application context and wires this to the main PPS application context containing the actual business logic:

Source Code

```
<!-- Create & initialize PPS web app context on startup -->
<listener>
    <listener-
class>com.sap.ppengine.core.spring.impl.PPSWebAppContextLoaderListener</listener-
class>
</listener>
<!-- Servlet directing all requests to PPS app logic -->
<servlet>
    <servlet-name>sapppsservlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
    <init-param>
        <!-- Name of the servlet context attribute holding the PPS web app
context -->
        <param-name>contextAttribute</param-name>
        <param-value>SAP_PPS_WEBAPPCONTEXT</param-value>
    </init-param>
    <load-on-startup>10</load-on-startup>
</servlet>
```

Using of the Hybris Administration Console

Logging Configuration

The local PPS shares the logging implementation of SAP Hybris Commerce. Therefore, there is no difference between the logging configuration.

SQL Query

If you have not made any changes to the configuration of the PPS database connection, you can easily access the PPS tables. The following code snippet shows how to access the promotion header table:

Source Code

```
SELECT * FROM SAPPSPROMOTION
```

Scripting Languages

If you want to access the Spring beans of the main PPS application context, you have to choose a slightly different syntax. This is because you have to access the PPS application context from the Hybris application context, as shown in the following code snippet. The bean `sapPPSClientBeanAccessor` is located in the Hybris context. The bean `sapDataSource` is located in the PPS context.

Source Code

```
spring.getBean("sapPPSClientBeanAccessor").getContext().getBean("sapDataSource")
```

Catalog Price Cache

The statistics for the catalog price cache, which is introduced by the `sapppspricing` extension, are available under the region `sapPPSProductPriceCacheRegion`.

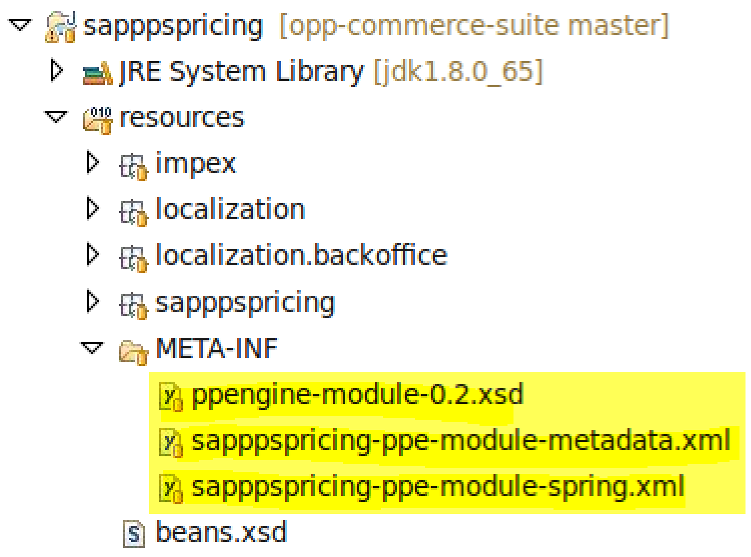
PPS and SAP Asynchronous Order Management

If you use the SAP Asynchronous Order Management for the integration between SAP Hybris Commerce and SAP ERP, you have to specify condition types for offers on promotion price derivation rule level. In Customizing for SAP Customer Activity Repository under *Omnichannel Promotion Pricing (OPP)* > *Business Add-Ins (BAdIs)* > *Offer Transformation*, you can do this via *BAdI: Promotion Builder* in table field `/ROP/PROMO_RULE_TYPE_CODE`. If you do not populate this field, value `EDIS` is used as default condition type in sales order processes.

4.3.3 Extending the Price Calculation

There are the following complementary scenarios for the extension of the price calculation logic:

- **The extension only refers to the client side of the price calculation.**
  If, for example, a coupon number is added to the request sent to the PPS, the standard Hybris extensibility concepts apply. This means that you can create a new Hybris extension declaring a dependency to `sapppspricing` in `extensioninfo.xml`, redefine the relevant Spring beans, and so on.
- **The extension refers to the server side of the price calculation located on the platform class path.**
  If the logic to be extended is part of the main PPS application context, the standard PPS extension concept applies. This means that you can create a new PPS module on the classpath with the corresponding PPS modules dependencies, redefine the relevant Spring beans of the PPS application context, and so on.
  For the definition of a new PPS module in SAP Hybris Commerce, we recommend to wrap it into an Hybris extension and add the metadata and Spring resource file in the resources folder, for example, to the `sapppspricing` extension that has an additional PPS module that provides the `javax.sql.DataSource` to use:



```
sapppspricing [opp-commerce-suite master]
  JRE System Library [jdk1.8.0_65]
  resources
    impex
    localization
    localization.backoffice
    sapppspricing
    META-INF
      ppengine-module-0.2.xsd
      sapppspricing-ppe-module-metadata.xml
      sapppspricing-ppe-module-spring.xml
    beans.xsd
```

- **The extension refers to the server side of the price calculation that is not located on the platform class path, but is only part of the** `sapppspricing` **web application**.

Example

Extending the IDoc Inbound

If you want to extend the IDoc inbound, you have to apply the following concepts to extend PPS extensions and Hybris web applications:

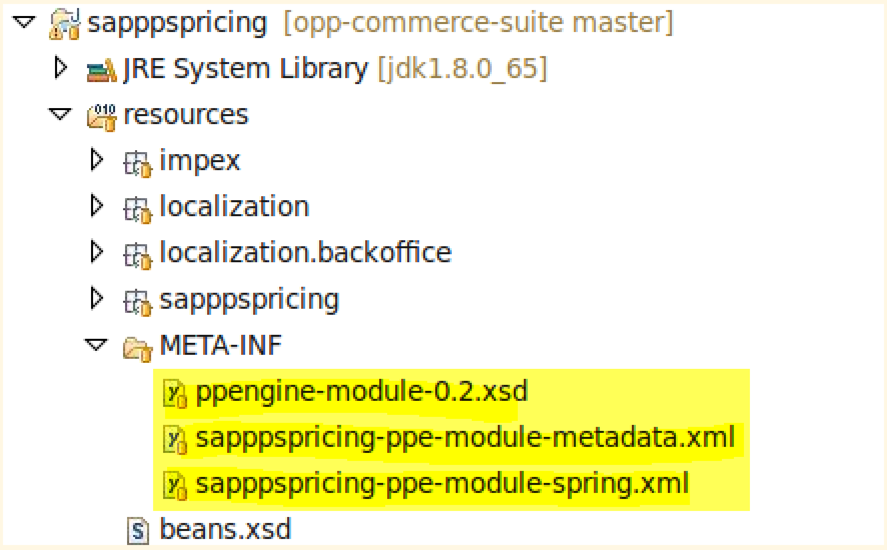
1. Create a Hybris extension as a wrapper for the additional PPS module.
2. In `extensioninfo.xml` of your extension, declare this extension as a web module by extending the web appplication `sapppspricingwebapp`. In the example below this is done taking the extension name `sappsext`:

Source Code

```
<extensioninfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="extensioninfo.xsd">
```

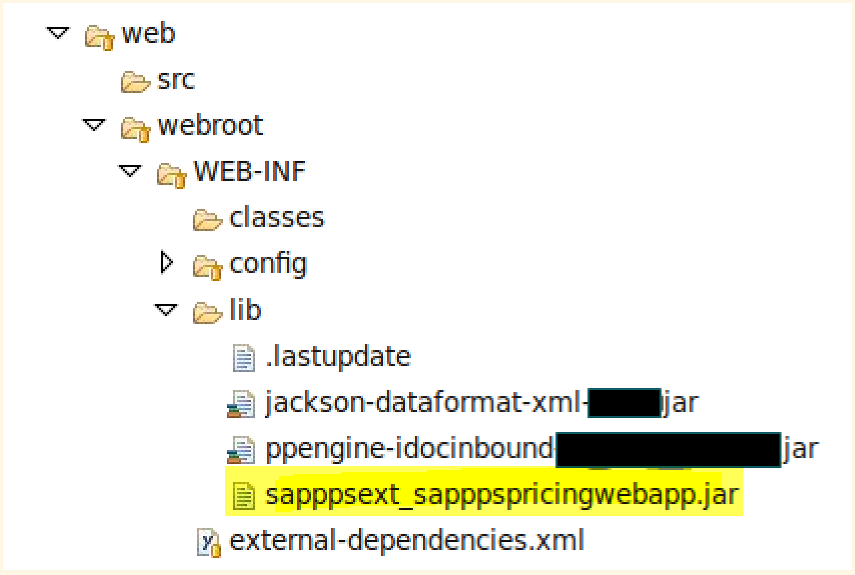
```
    <extension abstractclassprefix="Generated" classprefix="Sapppsext"
name="sapppsext">
        <requires-extension name="sapppspricing" />
        <webmodule jspcompile="false" webroot="/sapppsext"/>
        <meta key="extends.webmodules" value="sapppspricingwebapp" />
    </extension>
</extensioninfo>
```

3. In this extension, note the following when creating the folder structure:
   - Place the Java classes in folder `sapppspricingwebapp/src`.
   - Place other resources on the classpath in folder `sapppspricingwebapp/resources`, for example the PPS module.



```
sapppspricing [opp-commerce-suite master]
  JRE System Library [jdk1.8.0_65]
  resources
    impex
    localization
    localization.backoffice
    sapppspricing
    META-INF
      ppengine-module-0.2.xsd
      sapppspricing-ppe-module-metadata.xml
      sapppspricing-ppe-module-spring.xml
    beans.xsd
```

4. During the ant build, these folders are bundled into a JAR and copied into the `WEB-INF/lib` folder of the `sapppspricing` extension:



```
web
  src
  webroot
    WEB-INF
      classes
      config
      lib
        .lastupdate
        jackson-dataformat-xml-[redacted]jar
        ppengine-idocinbound[redacted]jar
        sapppsext_sapppspricingwebapp.jar
      external-dependencies.xml
```

Related Information

Extension Concept in SAP Hybris Commerce

4.3.4 Configuration of the Promotion Pricing Service

PPS is configured via the properties file `ppe-local.properties`. The configuration options are described in the Development and Extension Guide, under https://help.sap.com/viewer/p/IPR ▶ *<Version>* ▶ *Development* ▶ *Development and Extension Guide for Omnichannel Promotion Pricing* ◀.

For more information about configuring the IDoc inbound processing and the promotion pricing service in the Backoffice application of SAP Hybris Commerce, see Configuration [page 86].

For the deployment of the PPS in `sapppspricing`, the following differences have to be considered:

- The location of the file `ppe-local.properties` is not on the classpath. You can find the location in the hybris config folder. This folder should contain both, the `local.properties` file (Hybris settings) and the `ppe-local.properties` file (PPS settings).
- The following PPS configuration properties are set automatically:
  - `sap.dataaccess-common.db.url`
  - `sap.dataaccess-common.db.userName`
  - `sap.dataaccess-common.db.passWord`
  - `sap.dataaccess-common.db.driverClassName`
  - `sap.dataaccess-common.db.client`
  - `sap.sapppspricing.db.jndiname`
  - `sap.sapppspricing.db.dstype`

  Note

  Do not set these properties again via `ppe-local.properties`, since the settings for the properties do not have defined precedence.
- In the standard delivery (Hybris on HSQLDB), the PPS-specific database tables are located in the same database schema (`PUBLIC`) as the Hybris tables. This means that the file `ppe-schema-orm.xml` is not used.

  Note

  In a productive installation, it is recommended to separate the Hybris tables and the PPS tables into separate schemas. For more information, check the documentation of your database platform.

The logical system to read regular prices and OPP promotions is the only mandatory configuration in the `ppe-local.properties` file. This is unique per PPS installation and does therefore not depend on the base store. Depending on the underlying database, it may be necessary to specify a validation query that checks if the pooled connection to the database is still alive.

Source Code

```
# Replace with your logical system
sap.dataaccess-common.logSys=ABCCLNT800
# Validation query - valid for many DB platforms
sap.dataaccess-localdb.connectionpool.validationQuery=select 1
# Other queries below
# hsqldb - select 1 from INFORMATION_SCHEMA.SYSTEM_USERS
# Oracle - select 1 from dual
# DB2 - select 1 from sysibm.sysdummy1
# mysql - select 1
# Microsoft SQL Server - select 1
# Postgresql - select 1
# Derby - values 1
# H2 - select 1
```

Spring Configuration Properties in Hybris

The following properties in the Hybris application context are relevant for the `sapppspricing` extension:

Table 41: Spring Configuration Properties Relevant for sapppspricing Extension

Name	Description	Default Value	Comment
`sapppspricing.pps.configfile`	Name of the file that contains the configuration properties of the PPS	file:${HYBRIS_CONFIG_DIR}/ppe-local.properties	Default value means that the PPS properties are expected in the file `ppe-local.properties` in the `config` folder of SAP Hybris Commerce
`sapppspricing.dbclient`	SAP client for which the price and promotion information is to be stored in case of a local PPS	000	The client is part of the key of the PPS tables and is copied to the PPS property `sap.dataaccess-common.db.client`. This information is mandatory, but currently not used.
`sappspricing.merchcatclassificationcatalogid`	ID of the classification catalog that contains the replicated SAP Retail merchandise hierarchy	ERP_CLASSIFICATION_026	This ID is used to enhance the price calculation requests with merchandise hierarchy information to support promotions on merchandise category hierarchy level .
`sapppspricing.catalogpricecacheenabled`	Switch to enable or disable the caching of catalog prices	true	
`sapppspricing.catalogpricecache.maxentries`	Maximum number of entries in the catalog price cache	10000	

Name	Description	Default Value	Comment
`sapppspricing.catalogpricecache.evictionpolicy`	Eviction policy if the maximum number of entries is reached for the catalog cache	LRU	The following values exist:: • LRU (least recently used) • LFU (least frequently used) • FIFO (first in first out)
`sapppspricing.catalogpricecache.statsenabled`	Enablement of the recording of access statistics	true	
`sapppspricing.catalogpricecache.exclusivecomputation`	Exclusive computation	false	If set to false, the same value can be loaded multiple times. However, the cache lock contention is reduced and cache performs better. If the same value is read by multiple threads, only the first thread loads it from a database. Others are waiting and get value loaded.
`sapppspricing.catalogpricecache.timetolive`	Maximum time to live in seconds of a cache entry before it is considered as outdated and removed from the catalog price cache	600	Note Do not mix up with PPS side caches.
`sapppspricing.db.url`	URL of the database connection that is used for the PPS tables only	no default	This URL has precedence over property `db.url`.
`sapppspricing.db.username`	Database user used for the PPS tables only	no default	This user has precedence over property `db.username`. It is not intended for productive use.
`sapppspricing.db.password`	Password of the database user that is used for the PPS tables only	no default	This password has precedence over property `db.password`. It is not intended for productive use.
`sapppspricing.db.driver`	Database driver that is used for the PPS tables only	no default	This driver has precedence over property `db.driver`.
`sapppspricing.catalogerrorprice`	Price to be displayed in the catalog if the corresponding PPS call failed	99999.99	Note If increasing this value, you have to set the facet search price ranges.

Name	Description	Default Value	Comment
db.url	URL of the database connection that is used by SAP Hybris Commerce	n/a	The value of this property of the Hybris core application context is copied to the PPS property sap.dataaccess-common.db.url.
db.username	Database user that is used by SAP Hybris Commerce to access the database	n/a	The value of this property of the Hybris core application context is copied to the PPS property sap.dataaccess-common.db.userName.
db.password	Password of the database user	n/a	The value of this property of the Hybris core application context is copied to the PPS property sap.dataaccess-common.db.passWord.
db.driver	Database driver that is used by SAP Hybris Commerce	n/a	The value of this property of the Hybris core application context is copied to the PPS property sap.dataaccess-common.db.driverClassName.
db.pool.fromJNDI	Name of the JNDI datasource	n/a	The value of this property of the Hybris core application context is copied to the PPS property sap.sapppspricing.db.jndiname.

Spring Configuration Properties in PPS

The following properties in the PPS application context are added by the `sapppspricing` PPS extension:

Table 42: Spring Configuration Properties Added by sapppspricing Extension

Name	Description	Default Value	Comment
sap.sapppspricing.db.dstype	Type of the `DataSource` that is to be used	(calculated)	Either `Direct`, or `Jndi`
sap.sapppspricing.db.jndiname	Name of the `JNDI DataSource` that is to be used	(calculated)	Either the value of the Hybris Spring configuration property `db.pool.fromJNDI` or `dummy`

Related Information

Configuration [page 86]

4.3.5 Provided Spring Beans

SAP Hybris Commerce contains a hierarchy of application contexts. `sapppspricing` adds beans on all three levels of the Hybris application hierarchy and on one level of the PPS application context hierarchy.

Beans in the Hybris Web Application Context

Table 43: Beans in the Hybris Web Application Context

Name	Alias	Description
`sapppspricingFilterChain`		Filter chain to activate Hybris tenant and so on
./.		Spring HTTP security configuration. Supports HTTP basic authentication.
`accessDecisionManager`		Access decision manager authorizing the HTTP request
./.		Spring Security authentication manager. Refers to the `coreAuthenticationProvider` as authentication source.
`coreAuthenticationProvider`		Hybris implementation of the Spring authentication provider that delegates to the Hybris user management. Refers to bean `corePreAuthenticationChecks` and `coreUserDetailsService`.
`corePreAuthenticationChecks`		Rejects users that are not assigned to user group `sap_pps_idocinbound`, or a user of type *Employee*.
`coreUserDetailsService`		Hybris user details service

Beans in the Hybris Core Application Context

On this level, the regular beans are defined that contain application logic.

Table 44: Beans in the Hybris Core Application Context

Name	Alias	Description
`sapDefaultPPSConfigService`	`sapPPSConfigService`	Provides access to the configuration for calculating prices via PPS.
`sapDefaultPPSCalculationService`	`calculationService`	Delegates shopping cart or order calculation to PPS-based logic.
`sapPPSPromotionsService`	`promotionsService`	Delegates promotion handling to PP- based logic.
`sapDefaultPPSPricingCatalogService`	`priceService`	Delegates catalog price calculation to PPS-based logic.
`sapDefaultPPSPricingBackend`	`sapPPSPricingBackend`	Main bean of `sapppspricing` extension. Delegates most of the work to the beans that are listed below in this table.
`sapDefaultPPSRequestCreator`	`sapPPSRequestCreator`	Creates a PPS pricing request. Used by PPS pricing back end.
`sapDefaultPriceCalculateToOrderMapper`	`sapPriceCalculateToOrderMapper`	Maps a PPS response to a Hybris order or shopping cart. Used by PPS pricing back end.
`sapDefaultPPSClient`	`sapPPSClient`	Client for the PPS. Sends the prepared request and receives the response. Supports local and central calls. Used by PPS pricing back end
`sapDefaultPPSCatalogPriceCache`	`sapPPSCatalogPriceCache`	Helper to access the Hybris cache for catalog prices. Creates the cache key. Used by PPS pricing back end.
`sapDefaultMerchHierLineItemPopulator`	`sapMerchHierLineItemPopulator`	Populator of a PPS calculation request item. Adds the base merchandise category of the corresponding article including its parent hierarchy nodes. Used by the PPS request creator.
`sapDefaultPPSClientBeanAccessor`	`sapPPSClientBeanAccessor`	Helper class to access certain Spring beans living in the PPS application context without having to know their names
`sapDefaultJacksonJsonConverter`	`sapJacksonJsonConverter`	`HttpMessageConverter` bean using Jackson (from `FasterXML`) to convert between Java objects and JSON messages sent to and received from a central PPS. Used by PPS client, created by `sapJacksonJsonConverterBuilder`.
`sapDefaultJacksonJsonConverterBuilder`	`sapJacksonJsonConverterBuilder`	Builder for the sapJacksonJsonConverter bean

Name	Alias	Description
sapDefaultDbServiceAccessor	sapDbServiceAccessor	Accessor for the PPS database access layer from outside. This bean is used to delete obsolete OPP promotions and regular prices.
sapDefaultDeletePromotionsJob	sapDeletePromotionsJob	Deletes obsolete OPP promotions, referenced by Cron Job.
sapDefaultDeletePricesJob	sapDeletePricesJob	Deletes obsolete regular prices, referenced by cron job.

Beans in the Hybris Global Application Context

This level contains beans that are visible across Hybris tenants. In particular, caches are defined on this level.

Table 45: Beans in the Hybris Global Application Context

Name	Alias	Description
`sapDefaultPPSProductPriceCacheRegion`	`sapPPSProductPriceCacheRegion`	Defines and configures the cache region that contains the catalog prices of products. For example, for product recommendations or the product details page
`sapPPSProductPriceCacheRegionAdder`	./.	Adds the catalog price cache region to the total list of cache region for the cache management of SAP Hybris Commerc.e
`sapPPSAppContextCreator`	./.	Creates the Spring application context of the local PPS upon startup of SAP Hybris Commerce. Refers to `sapPPSContextInitializers` that allows further configuration.
`sapDefaultPPSContextInitializers`	`sapPPSContextInitializers`	List of application context initializers for the PPS application context. In the standard delivery, only `sapPPSHybrisPropertySourceAdder` is used.
`sapDefaultPPSHybrisPropertySourceAdder`	`sapPPSHybrisPropertySourceAdder`	Context initializer that adds a further property source to the default set of property sources to be used when resolving Spring properties during the startup of the PPS application context. This allows the injection of information from outside. The used property source is `sapPPSHybrisPropertySource`.

Name	Alias	Description
`sapDefaultPPSHybrisPropertySource`	`sapPPSHybrisPropertySource`	Spring property source that provides values for PPS Spring properties filled by properties defined within SAP Hybris Commerce. In particular, the database connection parameters are injected into the local PPS via this property source.

Beans in the Main PPS Application Context

The following beans are introduced by the `sapppspricing` PPS module. The bean assignment to the alias `sapDataSource` depends on the value of the Spring configuration property `sap.sapppspricing.db.dstype`.

Table 46: Beans in the Main PPS Application Context

Name	Alias	Description
`sapDefaultDirectDataSource`	`sapDataSource`	`DataSource` using connection pool configured explicitly via URL, driver, and so on
`sapDefaultJndiDataSource`		`JNDI DataSource`

Related Information

Spring Framework in SAP Hybris Commerce

4.3.6 Dependencies

The `sapppspricing` extension depends on the following extensions:

- `springintegrationlibs`
  This extension provides additional open source libraries for IDoc processing.
- `sapmodel`
  This extension provides the basis for the SAP Hybris integrations.
- `sapcoreconfiguration`
  This extension provides the configuration access.
- `commerceservices`
  This extension redefines central beans for calculating prices and discounts.
- `webservicescommons`
  This extension provides libraries of `EclipseLink`.

In addition, not as a strict dependency, `sapppspricing` makes use of replicated merchandise category hierarchy nodes replicated into SAP Hybris Commerce.

Related Information

Contents of saparticle DH Extension: Characteristic Profiles and Variant-Creating Characteristics [page 38]
Spring Framework in SAP Hybris Commerce

4.4 Security Information

To send IDocs from SAP Customer Activity Repository to SAP Hybris Commerce, you have to make the corresponding security settings, and configure basic authentication to protect your installation from malicious data.

This document provides an overview of the security-relevant information that applies to omnichannel promotion pricing for the IDoc inbound processing and the inbound to SAP Hybris Commerce.

IDoc Inbound Processing

To protect your SAP Hybris Commerce installation from external malicious data, all received IDocs need to be authorized. For more information about the configuration of basic authentification with user and password, see Configuration [page 86]> *Configuration of Data Replication* > *Configuring IDoc Inbound Processing* .

HTTPS Connection for Replication of IDocs

We strongly recommend to use SSL to secure any HTTP connection from the NetWeaver Application Server ABAP (NW AS ABAP) of SAP Customer Activity Repository to the SAP Hybris Commerce server. The following documents describe how to configure this security mechanism:

- For NW AS ABAP: Configuring the AS ABAP for Supporting SSL
- For Hybris server and Data Hub: Configuring the Apache Tomcat 7 for Supporting SSL

Output Encoding

If you want to display data from the database tables that are delivered with omnichannel promotion pricing, make sure that the correct output encoding is implemented.

5 In-Store Customer Engagement

In-store customer engagement supports store associates to perform their tasks when interacting with customers.

In-store customer engagement is based on the Hybris Assisted Service Module (ASM) while extending it with the following solutions:

- Customer profile based on data from SAP Hybris Marketing and SAP Customer Activity Repository (CAR)
- Quick check on in-store stock level for a product
- Click and collect for logistical processes based on SAP ERP
    - Handover of prepared deliveries
    - Order adjustments for cross- and up-sell
    - Invoice document review

For more information about the Hybris Assisted Servce Module, see About Assisted Service Module.

System Setup

The delivery of the in-store functionality is based on the following software applications:

- SAP Retail 6.0, minimum release EHP7 SP13/EHP8 SP4
- SAP Hybris Marketing, minimum release 1608
- SAP Customer Activity Repository (CAR), minimum release 2.0 FP2

Customer Profile

When logged into the ASM, the store associate can call up the profile of the customer that he or she has selected in the customer list before. This profile is available via the *360 Degree Customer View* icon in ASM or in the customer list. By choosing *Customer Profile* in the *360 Degree View*, the store associate is provided with an overview of the customer activities. This view contains statistical data such as sales volume, purchase ratio, sales order history, and many more key performance indicators. To enhance the store associate's support possibilities, you can also add the Hybris AddOn *SAP Product Recommendations*, which can simply be added by using the WCMS Cockpit.

Embedded in the ASM-based Additional Information Framework, the CMS component `Customer 360 Component` in the WCMS Cockpit allows you to adjust the value interpretation of the tiles. The CMS component can be found on the CMS page *Customer 360 Page*. The statistical data tiles are structured into four representing segments. This is reflected in the CMS component's maintainable attributes. To support the interpretation of the tiles for the in-store associate they are clickable. When clicking them, they turn around showing the maintained boundary values.

Boundary Definition in Statistical Data Tiles

The following attributes define the boundaries used by the tiles showing statistical data. They can be devided in the following groups:

1. Ascending boundaries
   The following scores belong to this group: *Activity Score*, *Average Purchase Volume*, *Sales Volume*, and *Store Online Ratio*. Each of these entities provides four attributes represented by *Level 1-4*. By their values a user specifies the boundaries to distribute the actual values in. This means that all values greater or equal than the concrete value specified in *Level 1* and less or equal than the value specified in *Level 2* minus 1 will be displayed as first level on the UI and so on, besides the last level. In the last level, all values greater than the specified value will be displayed as the last level. The values are expected in ascending order.
2. Descending boundaries
   Only one tile shows the four level values in descending order: the *Last Purchase Date* tile. *Level 1* contains the greatest value. All values less or equal than the concrete value specified in *Level 1*, and greater or equal than the value specified in *Level 2* plus 1 will be displayed as first level on the UI and so on, besides the last level. In the last level, all values less than the value specified will be displayed as the last level. The values are expected in descending order.

Customer Scores

You can define scores in SAP Hybris Marketing. Scores related to the object type `Customer` can be directly referenced via the attribute `Customer Scores`, where the selection is supported by an input help. For other object types, see section *Extensibility*. Any change of the attributes can directly be reviewed on the original customer profile page.

Extensibility

In a customer project, you can add or remove the tiles of the customer profile. This is supported by the `yinstorecsfrontendaddon` extension. Its design allows adding or removing your extension data providing objects and simple UI manipulations for the best representation of your customer data.

In-Store Stock Level Quick Check for Product

On the product detail page, the activated button *Take One Home Now* indicates that this product is on stock at the store associate's location. Additionally, the stock amount is shown below. Clicking this button will transfer exactly one product to the shopping cart.

Note

The product detail page typically offers two more buttons to transfer the product to the shopping cart (*Add to Cart* and *Pick Up in Store*). Any combination of take-one-home-now items with other items (add to cart or pick up) in one order is not possible without the extension of the functionality during customer implementation. In your customer implementation, you should decide whether or not you want to allow orders with different shipping methods for individual items and if so, how to handle these combined orders.

Handover of Prepared Deliveries

When a customer has ordered products to be picked up in the store, the customer comes to the store when the delivery is ready to be picked up. To find the correct pick-up order, the store associate uses either the customer ID, customer name, customer e-mail address, or the order number in the ASM header.

When handing over the package to the customer, the store associate clicks the *Hand Over* button on the order overview page. In Hybris, the order status is set to *completed*, consignment status changes to *shipped*. In the SAP ERP back end, the goods issue and invoice are created. The invoice for the pick-up order is displayed automatically in PDF format on a separate browser tab. If the customer has not paid for the products yet, the store associate can print the invoice, and ask the customer to pay at the point-of-sale.

Any messages that occur in the system during the handover process are written to the Hybris log. If an error occurs during the handover process, the Hybris order status is set to `process_error`.

Note

The language of the invoice is controlled by SAP ERP output control, which is typically the language of the business partner in its role as bill-to party.

Order Adjustments for Cross- and Up-Sell

To enable the store associate to modify an order ready for handover, if the customer requests this, the order change functionality is available. The related *Change Order* button is shown on the *Order Details* page, if the application runs in ASM mode and if the order is ready for handover. If the store associate clicks the button, the order entries are added to the shopping cart. Only the following attributes of the entries are copied:

- Product
- Quantity
- Unit of measurement
- Point of sale
- Configuration
- Give-away indicator

Finally, the Hybris order is cancelled; the related SAP ERP will not be changed. If errors occur, the application will revert the changes in the shopping cart and restore the initial state.

Invoice Document Review

To enable the store associate to review an existing invoice later on, a link is offered on the order detail page to open this invoice in PDF format.

5.1 Configuration

Before in-store customer engagement can be used in an integrated landscape with SAP ERP and SAP Hybris Marketing, several configuration settings need to be applied in the Backoffice Administration Cockpit and in the back-end system.

Since in-store customer engagement is based on the Assisted Service Module (ASM), you need to consider the ASM-related documentation to set up an appropriate ASM user. For more information about how to set up an ASM user, see section *Usergroup* in assistedservicestorefront AddOn.

To configure the landscape in the Backoffice Administration Cockpit, proceed as follows:

1. Start the Backoffice Administration Cockpit and choose *Administration* > *SAP Integration* > *SAP Base Store Configuration* .
2. Choose the *In-Store Customer Engagement* tab in the SAP base store configuration.

Entering HTTP Destination to Connect to SAP HANA DB of SAP Customer Activity Repository

In the *In-Store Customer Engagement* tab, you can find the HTTP destination settings to connect to the SAP HANA database of the SAP Customer Activity Repository (CAR).

For the sales-related data on the *Customer Profile* page, the following OData service endpoint should be used: `/sap/hba/t/rtl/car/int/odata/CARServices.xsodata`. This OData service returns sales information for various channels, which can be customized.

Note

The path to the OData service endpoint can be defined in Customizing of the SAP Customer Activity Repository system. To get the right OData service path, contact your system administrator for SAP Customer Activity Repository.

To identify the right channels, enter the following identifiers into the following fields. If more than one identifier is required, they can be separated by a comma:

Table 47: Channel Default Values

Field Name	Default Value	Description
POS Order Channels in SAP Customer Activity Repository	`07`	Identifies the point-of-sales-related (POS) customer purchase transactions
Online Order Channels in SAP Customer Activity Repository	`03`	Identifies web-shop-related customer purchase transactions

To limit the number of transactions shown in the purchase history on the *Customer Profile* page, adjust the value in the *Maximum Number of Displayed Customer Orders in Purchase History* field accordingly.

This connection information is used to build a request URL such as `http://<SAP_CAR_HANA_SERVER>:<HANA_SERVER_PORT>/sap/hba/t/rtl/car/int/odata/`

`CARServices.xsodata/MultiChannelSalesOrdersQuery(P_SAPClient='<CLIENT>')/Results?&$filter=CustomerNumber eq '<CUSTOMER ID>'&$top=2&$format=xml`.

To look up the customer ID, choose *Administration* > *User* > *Customers* > *<find your customer>* in the Backoffice Administration Cockpit. You can find this information in column *Customer ID*, in the search result list .

For more information about the OData service, see Multichannel Sales.

Entering OData Destination to Connect to SAP Hybris Marketing

In the *In-Store Customer Engagement* tab, you can find the OData destination settings to connect to the SAP Hybris Marketing system.

This connection information is used to build a request URL such as `https://<HYBRIS_MARKETING_SERVER>:<HYBRIS_MARKETING_SERVER_PORT>/sap/opu/odata/sap/CUAN_ISCE_COMMON_SRV/ContactPersons?$filter=EMailAddress eq '<unique email of business partner>'`.

To look up the e-mail address, choose *Administration* > *User* > *Customers* > *<find your customer>* in the Backoffice Administration Cockpit. To find the contact e-mail, choose the *Administration* tab in the customer details.

For an appropriate system user setup, see SAP Help Portal at *SAP Hybris Marketing 1608 (and higher)* > *Security Information*. Search for user roles `SAP_CEI_SCI_ISCE` and `SAP_MARKETING_DATA_MANAGEMENT`, which are required to properly execute the OData service.

Activating OData Service RETAILSTORE_CUST_ENG_SRV in the SAP ERP Retail Back-End System

The OData service `RETAILSTORE_CUST_ENG_SRV` transfers transactional data for order fulfillment to the back-end system. You can find the corresponding HTTP destination in the Backoffice Administration Cockpit at *Administration* > *SAP Integration* > *SAP Global Configuration*. In the *Administration* tab, choose *SAP HTTP Destination*.

Note

The OData service uses the same SAP ERP HTTP destination as the Data Hub for the IDoc integration.

To activate the OData service, proceed as follows:

1. Log on to the gateway system.
   NOTE: The gateway system might be the same as the back-end system.
2. Start *Activate and Maintain Services* (transaction `/n/IWFND/MAINT_SERVICE`).
3. Choose *Add Service*.
4. Enter the system alias of the source system, the technical service name (can be defined according to the naming rules in your system), and the external service name `RETAILSTORE_CUST_ENG_SRV` (name of the service in the source system).

NOTE: To search for the service, you need a user in the source system.
5. Choose *Get Services* to search for the OData service.
6. Select the service from the search result list and choose *Add Selected Service*.
7. Save your entries.
8. Start *Define Services* (transaction `SICF`), choose the service `RETAILSTORE_CUST_ENG_SRV`, and enter the appropriate client in the *Logon Data* tab.
NOTE: The assigned user of the HTTP destination needs to have the user authorizations to post goods issue for an outbound delivery and to create an invoice.

5.2 instorecsbackoffice Extension

The `instorecsbackoffice` extension extends the SAP base store configuration for in-store customer engagement. It allows to maintain in-store-related properties on the *In-Store Customer Engagement* tab in an SAP base store configuration, in the Backoffice Administration Cockpit.

Property Description

Table 48: Property Description

Property	Description
`HTTP Destination for SAP Customer Activity Repository`	HTTP destination for calling the OData service in SAP Customer Activity Repository
`Name of OData Service in SAP Customer Activity Repository`	Name of OData service to be called in SAP Customer Activity Repository
`Client in SAP Customer Activity Repository`	Client for calling the OData service in SAP Customer Activity Repository
`POS Order Channels in SAP Customer Activity Repository`	Comma-separated list of order channels for point-of-sale (POS) transactions in SAP Customer Activity Repository
`Online Order Channels in SAP Customer Activity Repository`	Comma-separated list of order channels for online transactions in SAP Customer Activity Repository
`Maximum Number of Displayed Customer Orders in Purchase History`	Maximum number of displayed transactions (point-of-sale (POS) and online order channels) in SAP Customer Activity Repository
`HTTP Destination for SAP Hybris Marketing`	HTTP destination for calling the OData service in SAP Hybris Marketing

5.3 instorecsfacade Extension

The `instorecsfacade` extension provides data access facades for 360 degree customer profile, in-store stock level for a product, handover of prepared deliveries, order adjustments for cross- and up-sell, invoice document review, and order item creation tracking.

Dependencies

The `instorecsfacade` extension depends on the following extensions:

Table 49: Dependencies

Extension	Description
`assistedservicefacades`	Assisted service facade extension
`instorecsservice`	In-store service extension
`sapinstorecsservice`	In-store SAP-related service extension
`sapmodel`	SAP Hybris Commerce SAP integration model extension

5.4 instorecsservice Extension

The `instorecsservice` extension provides data access services for 360 degree customer profile, in-store stock level for a product, handover of prepared deliveries, order adjustments for cross- and up-sell, invoice document review, and order item creation tracking.

Dependencies

The `instorecsservice` extension depends on the following extensions:

Table 50: Dependencies

Extension	Description
`basecommerce`	Standard service extension
`sapcoreodata`	SAP Hybris Commerce integration OData service extension

Extension	Description
sapcoreconfiguration	SAP Hybris Commerce integration configuration service extension
sapinstorecsservice	In-store SAP-related service extension

Data Model Modification

The `AbstractOrder` data model was modified. New fields for the `EmployeeID` and the flag `createdInASMMode` were added to the standard data model, in the `instorecsservices-items.xml` file. This information allows the assignment of order items to a particular employee for better tracking.

```
<itemtypes>
        <itemtype code="AbstractOrderEntry" autocreate="false"
            generate="false">
            <attributes>
                <attribute qualifier="employeeId" type="java.lang.String">
                    <description>Store Associate User Id</description>
                    <modifiers read="true" write="true" search="true"
                        optional="true" />
                    <persistence type="property" />
                </attribute>
                <attribute qualifier="createdInAsmMode" type="java.lang.Boolean">
                    <description>Store Associate User Id</description>
                    <modifiers read="true" write="true" search="true"
                        optional="true" />
                    <persistence type="property" />
                </attribute>
            </attributes>
        </itemtype>
    </itemtypes>
```

5.5 sapinstorecsservice Extension

The `sapinstorecsservice` extension provides the SAP-specific data access services for handover of prepared deliveries and invoice document review.

Dependencies

The `sapinstorecsservice` extension depends on the following extensions:

Table 51: Dependencies

Extension	Description
`sapcoreodata`	SAP Hybris Commerce integration OData service extension
`sapcoreconfiguration`	SAP Hybris Commerce integration configuration service extension

5.6 yinstorecsfrontendaddon AddOn

This is an AddOn for the `yacceleratorstorefront` (B2C storefront) template extension. It adds in-store-specific functionality to the B2C accelerator, such as the *Take One Home Now* action in the catalog.

Definition

Table 52: Definition

Feature	Description
Name	`yinstorecsfrontendaddon` AddOn
Description	Provides UI enhancements for in-store scenarios
Requires	• `addonsupport` extension • `assistedservicestorefront` extension • `instorecsfacade` extension
Author	SAP
Impex Sample Data	• electronics • apparel-uk • apparel-de

Supported Market and Channels

Table 53: Supported Markets and Channels

Supported	B2C Commerce	B2B Commerce	Telco Commerce
Market	✓	⛔	⛔
Channel	Table 54:	Table 55:	Table 56:

Table 54:

Desktop	Mobile
✓	✓

Table 55:

Desktop	Mobile
⛔	⛔

Table 56:

Desktop	Mobile
⛔	⛔

Installation

1. Add the `assistedservicestorefront` AddOn to your `localextensions.xml` file, as listed below:

```
<extensiondir="${HYBRIS_BIN_DIR}/custom/yinstorecsfrontendaddon"/>
```

2. Navigate to the `hybris/bin/platform` directory and call the following `ant` command:

```
ant addoninstall -Daddonnames="yinstorecsfrontendaddon"-
DaddonStorefront.yacceleratorstorefront="yacceleratorstorefront"
```

   This registers the `yinstorecsfrontend` AddOn to the storefront and generates the correct properties in the `project.properties` file.
3. Build the platform, if you have not built it yet.
4. Initialize the platform and start the Hybris server. For more information, see B2C/B2B Accelerator Quick Installation.

Feature Description

Customer Profile Data

The customer profile data is part of the 360 customer view on the B2C/B2B Accelerator Quick Installation Guide *Customer Profile* tab. This tab is opened by default when using the 360 customer view.

To add or remove tiles, extend the `yinstorecsfrontendaddon` AddOn and re-list the data containers required according to your needs on an customer-owned spring bean for name The following data is read from a connected SAP Hybris Marketing system and matches to the SAP Hybris Marketing customer factsheet data.`CMSISCECustomer360ComponentController`. Reuse the default spring implementation `com.sap.retail.isce.frontend.controllers.cms.CMSISCECustomer360ComponentControllercom.sap.retail.isce.container.DataContainerOData` for OData-based ones or `com.sap.retail.isce.container.DataContainerSmart` for data related to SAP Hybris Commerce. To simplify the implementation, either use base class `com.sap.retail.isce.container.impl.DataContainerSmartDefaultImpl` or `com.sap.retail.isce.container.impl.DataContainerODataDefaultImpl`. Re-implement the related

JSP `cmsiscecustomer360component.jsp` to show the required tiles according to your needs. Your new JSP must be referenced by an approriate CMS component.

The following data is read from a connected SAP Hybris Marketing system and matches with the SAP Hybris Marketing customer factsheet data:

- Age
- Gender
- Marital status
- Newsletter
- Sentiment score
- Activity score
- Items of interest

Note

This implementation does not support the shared e-mail feature of SAP Hybris Marketing. It requires to uniquely identify the customer via e-mail.

The following data is read from a connected SAP Customer Activity Repository system.

- Sales volume
- Last purchase
- Store/online ratio
- Average purchase volume
- Purchase history

The following data is read from SAP Hybris Commerce:

- Name
- Address

Quick Check on In-Store Product Stock Level

On the product detail page, the activated *Take One Home Now* button indicates that this product is on stock at the store associate's location. Additionally, the stock amount is shown below. Clicking this button transfers exactly one product to the shopping cart. If a product is not on stock, the button is deactivated and not clickable.

Handover of Prepared Deliveries

The *Hand Over* button on the order detail page triggers the goods issue and invoice creation in SAP ERP. The button is available if the order is in status `Created` or `Checked valid`, if the delivery mode is `pickup`, and if the assigned consignment is in status `waiting`. This status check is in synch with the status handling of the Hybris SAP Integration Asynchronous Order Management. To prevent wrong actions, the point of service of the store associate must be the same as the delivery point of all order items. For more information, see Asynchronous Order Management.

Order Adjustments for Cross- and Up-Sell

The *Change Order* button on the order detail page supports the cross- and up-sell process by cancelling the current Hybris order (the related SAP ERP order is not changed) and transferring the items into the shopping cart. The button is available if the order is in status `Created` or `Checked valid`, if the delivery mode is `pickup`, and if the assigned consignment must is in status `waiting`.

The new shopping cart will have a reference information to the former Hybris order. This reference is stored in the desription attribute of the shopping cart in the format `[Created via sales order {4711}]` while `4711` represents the former order number.

> **Recommendation**
>
> For customer projects, it is recommended to add functionality to this button, which triggers the deletion of the related delivery. Since no follow-up documents exists anymore, the Hybris SAP Integration Asynchronous Order Managment will cancel the related SAP ERP order. This functionality should be added by implementing a customer-owned spring bean for name `defaultOrderChangeFacade`. Extending the default implementation `com.sap.retail.isce.facade.impl.OrderChangeFacadeDefaultImpl` and overwriting method `cancelOrder` allows to execute the project-specific coding before executing the standard implementation via `super.cancelOrder()`.

Invoice Document Review

To enable the store associate to review an existing invoice later on, a link is offered on the order detail page to open this invoice in PDF format. This link is only be shown after a handover took place.

Responsive UI

The `yinstorecsfrontendaddon` AddOn supports the responsive UI. To force the responsive behavior, change the `{HYBRIS_CONFIG_DIR}/local.properties` file and add the following property:

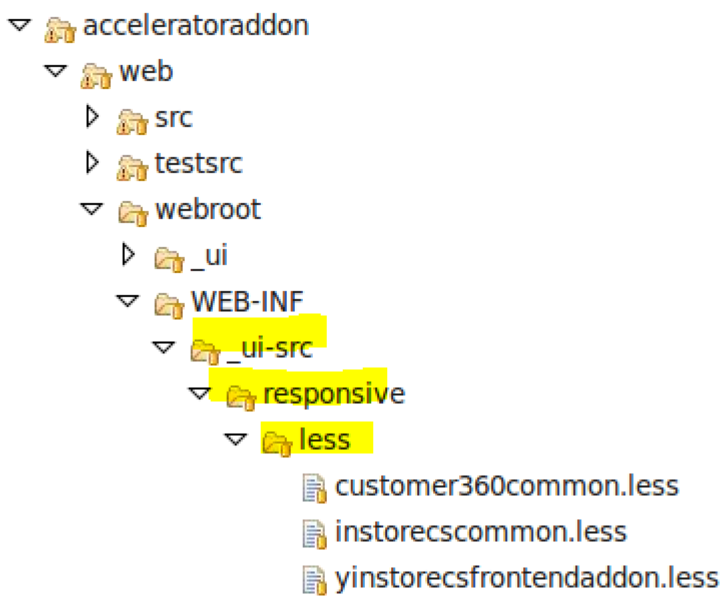
```
commerceservices.default.desktop.ui.experience=responsive
```

LESS Technique

In the SAP Hybris Commerce accelerator, two themes are available: black and blue. By default, the black theme is used for the B2B storefront, and the blue theme is used for the B2C storefront. Colors that are used in both themes are defined in the `variables.less` file, located in the `<${HYBRIS_BIN_DIR}>/ext-template/yacceleratorstorefront/web/webroot/WEB-INF/_ui-src/responsive/lib/ybase-0.1.0/less` directory. Colors that are used in specific themes are defined in the `theme-variables.less` file, located in the `<${HYBRIS_BIN_DIR}>/ext-template/yacceleratorstorefront/web/webroot/WEB-INF/_ui-src/responsive/themes/<theme name>/less` directory.

The main `style.css` is generated, based on the file `style.less`. This less file contains references to several other less files such as `@import "../../../lib/ybase-0.1.0/less/`*addons.less*`"`; this `addons.less` file stores references to `<addon name>.less` files of any installed AddOns. This file is initially empty, but is populated when AddOns are installed that have a corresponding `<AddOn name>.less` file.

Therefore, the missing folder structure under the `yinstorecsfrontendaddon` AddOn is available, and the `yinstorecsfrontendaddon.less` file contains the hardcoded reference to both less files: `customer360common.less` and `instorecscommon.less`.


```
acceleratoraddon
  web
    src
    testsrc
    webroot
      _ui
      WEB-INF
        _ui-src
          responsive
            less
              customer360common.less
              instorecscommon.less
              yinstorecsfrontendaddon.less
```

The `customer360common.less` and `instorecscommon.less` files contain the prepared less variables, mainly based on existing theme-variables already available. These less variables are prefixed with `isce-`.

Related Information

Responsive Themes
Responisve Website Build Process

5.7 Security Information

For in-store customer engagement, the following security information applies.

This document describes the security-relevant information that applies to the In-Store Customer Engagement (ISCE) module.

Preventing Information Disclosure

The `instorecsservice` extension throws exceptions of type `DataContainerRuntimeException` that are not caught by the application.

Recommendation

To avoid revealing sensitive data, it is strongly recommended to define an error page in the `web.xml` file of the web container. For the Hybris Tomcat implementation of the `yacceleratorstorefront`, an example can be found in `hybris/bin/ext-template/yacceleratorstorefront/web/webroot/web-inf/web.xml`.

Users and Authorizations

The In-Store Customer Engagement module needs a specific user setup. For more information, see Configuration [page 120].

Important Disclaimers and Legal Information

go.sap.com/registration/contact.html