



Platform, Services, and Utilities

Generated on: 2024-11-07 21:29:05 GMT+0000

SAP Commerce | 2205

PUBLIC

Original content: https://help.sap.com/docs/SAP_COMMERCE/d0224eca81e249cb821f2cdf45a82ace?locale=en-US&state=PRODUCTION&version=2205



Warning

This document has been generated from the SAP Help Portal and is an incomplete version of the official SAP product documentation. The information included in custom documentation may not reflect the arrangement of topics in the SAP Help Portal, and may be missing important aspects and/or correlations to other topics. For this reason, it is not for productive use.

For more information, please visit the <https://help.sap.com/docs/disclaimer>.

Platform, Services, and Utilities

SAP Commerce is built on a common platform that provides key services and models that are tailored for commerce, and available to all functional modules. Additional modules provide enabling services and utilities such as the API registry, and search and navigation support.

The Modules	Technical Topics
	
<p>Delve into the Platform and related modules.</p> <p>Platform</p> <p>API Registry Module</p> <p>Search and Navigation Module</p> <p>Hybris Utilities Module</p>	<p>Dive deeper into the technical side of Platform.</p> <p>Build Framework</p> <p>Clustering</p> <p>Application Performance and Monitoring</p> <p>Testing</p> <p>Logging</p> <p>OAuth 2.0</p>

Platform Features

Platform provides a range of features related to the main functionality of a SAP Commerce installation, that is, containerization, data management, synchronization, security, or localization. For example, you can use the data report feature to collect raw data stored in the SAP Commerce database and present it in a form of report.

[Build Framework](#)

SAP Commerce comes together with a build framework responsible for many tasks, for example for code generation. You can extend the framework and also use JRebel to help you avoid wasteful builds during the development phase.

[Business Process Management](#)

A business process describes a sequence of steps or activities that is followed repeatedly. Business process management identifies, defines, documents, controls, and optimizes these processes, which typically integrate mechanistic and human-driven processes, these are processes in which human interaction takes place.

[Caching](#)

The SAP Commerce Cache is a part of the SAP Commerce persistence layer. It improves the performance of a single server node by reducing the number of database queries. It transparently stores search results, item attributes, and item instances in memory.

[Clustering](#)

The SAP Commerce Cluster is a number of individual SAP Commerce installations using a common set of data on one database. The cluster functionality offers you a wealth of configurable options.

[Containerization](#)

Containerization allows you to build Docker images and run them as software instances in a different environment. Use containerization to deploy a desired instance of your system in your local environment without compatibility issues.

[Data Retention](#)

SAP Commerce Platform provides features that cover the concept of data retention. With the Data Retention Framework, you can decide to retain specified data until it is cleaned up according to a strategy you define. With the Item Locking Service, you can lock your data against modification or removal.

[Data Validation](#)

SAP Commerce is shipped with a data validation framework that helps you to validate your data before it is persisted. It is based on the widely adopted JSR 303 Java validation specification. In addition, it is configurable at runtime. However it is adjusted in a way that it better suits user needs.

[Digital Asset Management](#)

SAP Commerce includes media conversion and management tools to simplify the management of every aspect of your digital assets.

[Generic Data Report and Audit](#)

The Generic Data Report allows you to collect raw data stored in the SAP Commerce database and present it as a report that you can easily read and understand. Such a report reflects the current state of data. To create a historical report, use the generic audit feature.

[ImpEx](#)

SAP Commerce includes a text-based import and export functionality called ImpEx. The ImpEx engine allows creating, updating, removing, and exporting data items such as customer, product, or order data to and from comma-separated value (CSV) data files.

[Internationalization and Localization](#)

SAP Commerce provides support for internationalization and localization. It allows you to adapt the system to multiple languages and different requirements that depend on a given region.

[Java Message Service \(JMS\)](#)

Java Message Service (JMS) offers an **asynchronous** approach to (possibly remote) method invocation, that complements the **synchronous** solutions offered by RMI, Hessian/Burlap, Spring HTTP Invoker, and Web services.

[Logging](#)

SAP Commerce provides logging options that allow you to configure how your logs are formatted, sort log messages according to message type and level, or control at runtime where they're reported.

[Multitenancy](#)

SAP Commerce can be run in a multi-tenant mode. In this mode, several distinct sets of data are maintained on one single SAP Commerce installation. The effect is that you can have several logical SAP Commerce instances running, for example, to host online shops for different customers using one SAP Commerce installation.

[OAuth 2.0](#)

The OAuth 2.0 authorization framework is the default authorization framework for the commerce driven OCC (Omni Commerce Connect) Web Services.

[Ordering, Payment and Pricing Standards](#)

The Platform offers a basic implementation for processing orders, payment methods for transactions, pricing system, and organizing countries and regions for relating shipping costs to them. Moreover, there are a number of services for managing orders.

[Performance and Monitoring](#)

SAP Commerce supports a variety of methods for monitoring application performance allowing you to fine-tune many aspects of your installation.

[Polyglot Persistence](#)

The polyglot persistence provides a way to store specified data types in an alternative storage, for example document-based storage. Use polyglot persistence to relieve the load of the main database or to provide a non-SQL storage for some data.

[Primary Keys](#)

SAP Commerce provides methods that you can use to filter database data by primary keys.

[Product and Data Modeling](#)

Whenever you start developing and customizing SAP Commerce, you need to prepare a business model of your application. Here you will find help with setting up your business models.

[Product Content and Catalogs](#)

The Product Content and Catalog features of Platform provide functionality for holding, structuring, and managing products and product information.

[Search](#)

There are two search mechanisms implemented in Platform and available without any additional modules. These mechanisms are FlexibleSearch, and GenericSearch. You can also use ViewType, which is a representation of a database view.

[Secure HTTP Transactions](#)

SAP Commerce uses Charon to ensure secure HTTP transactions.

[Security and User Management](#)

SAP Commerce provides tools for security and user management including access control and data encryption.

[ServiceLayer](#)

The SAP Commerce ServiceLayer is an **API** for developing services for SAP Commerce. It provides a number of common services, while allowing you to extend these or develop your own.

[ServiceLayer Direct](#)

ServiceLayer Direct allows you to read and persist data in the database and completely skip the Jalo layer. With ServiceLayer Direct in place, you still use `ModelService` to manage models.

[Workflow and Collaboration](#)

Platform workflow and collaboration tools make defining complex organizational processes easier and more transparent.

Build Framework

SAP Commerce comes together with a build framework responsible for many tasks, for example for code generation. You can extend the framework and also use JRuby to help you avoid wasteful builds during the development phase.

SAP Commerce uses the framework to automatically run tasks such as extensions compilation, code generation, and copying, parsing, and compiling files.

Related Information

[Building SAP Commerce](#)

Business Process Management

A business process describes a sequence of steps or activities that is followed repeatedly. Business process management identifies, defines, documents, controls, and optimizes these processes, which typically integrate mechanistic and human-driven processes, these are processes in which human interaction takes place.

The SAP Commerce **processengine** functionality makes it possible to integrate your business processes seamlessly into the SAP Commerce product family.

About Business Process Management

The purpose of the business process management is to help companies model, support, and monitor their business processes online. A business process is usually modeled as a flowchart, in which there may be loops and parallel paths joining a number of actions. Some of these actions require human interaction, others invoke automated processes. Each action specifies what action should follow it, eventually providing a number of possible actions, the choice of which is based on the output of the preceding action. Manual tasks can be easily specified and incorporated in such a workflow.

Key Features

- Create, manage, and optimize business processes with focus on automated processes
- Process configuration based on XML
- Cluster aware processing of workflows guarantees high availability and speed, and load distribution. See also [Clustering](#).

- Seamless Integration of automated workflows with user interaction based on workflows. See also [Workflow and Collaboration](#).

Benefits

Once a business process is modeled, it can be instantiated and started what is ensured by the business process engine.

- For manual tasks the appropriate user is notified and the business process waits for completion of that task before proceeding.
- The appropriate method call is made for automated tasks.
- The data is shepherded appropriately through the workflow as tasks are performed.
- The business process status is persisted so that a business process will not be lost should a system crash.

A business process administrator is therefore able to easily interrogate the status of all business processes, making it possible for him to:

- Find bottlenecks in order to optimize the process overall.
- Look at the history of data through a business process.
- Cancel a process where required to do so.

The SAP Commerce business process engine is built on a firm architecture, see [The Task Service](#). Find a detailed example in [Example Order Management Business Process](#).

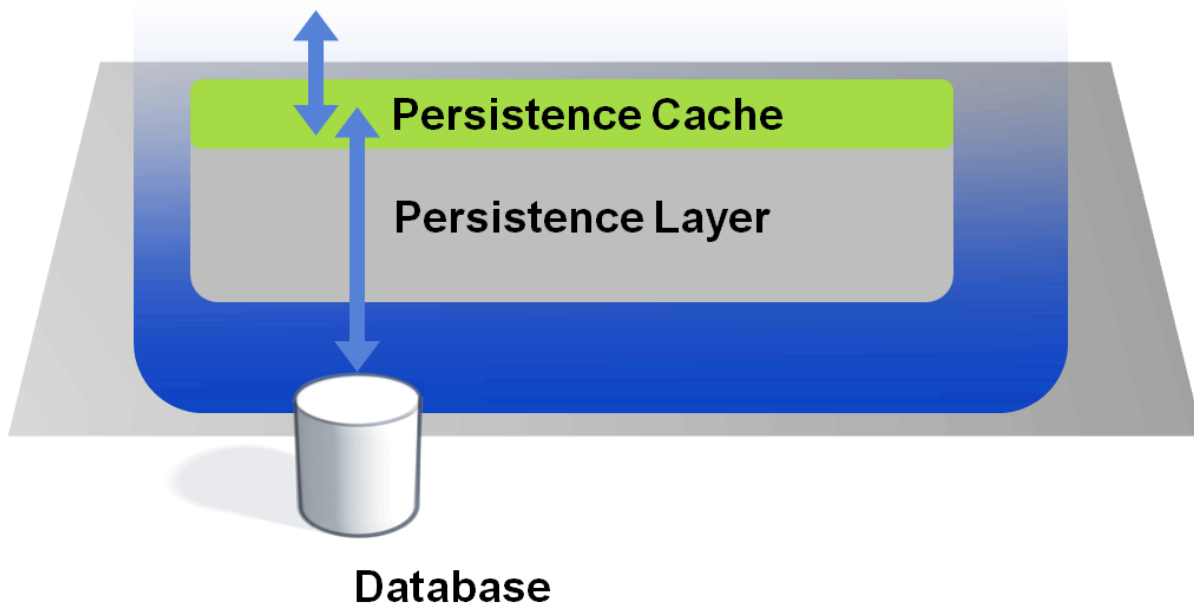
Caching

The SAP Commerce Cache is a part of the SAP Commerce persistence layer. It improves the performance of a single server node by reducing the number of database queries. It transparently stores search results, item attributes, and item instances in memory.

For more information, see [Cache](#) in Platform Module Implementation.

The SAP Commerce Cache reduces the number of database queries, and therefore improves the performance of SAP Commerce. It stores different kinds of data:

- Item instances, for example a Product.
- All item attributes, that is the result of `getAllAttributes()` or `getAttribute(X)` calls.
- Results of FlexibleSearch queries.



SAP Commerce

For more information see, [Cache](#) in Platform Module Implementation.

Clustering

The SAP Commerce Cluster is a number of individual SAP Commerce installations using a common set of data on one database. The cluster functionality offers you a wealth of configurable options.

You may need to have several SAP Commerce installations in a cluster environment. You may use standard or advanced SAP Commerce Cluster functionality depending on whether you want to have complete session failover capabilities.

For more information, see [Clustered Environment](#) in Platform Module Implementation.

Containerization

Containerization allows you to build Docker images and run them as software instances in a different environment. Use containerization to deploy a desired instance of your system in your local environment without compatibility issues.

Use case

An administrator of an SAP Commerce production environment decides to convert the application into containers in order to run it in a different environment without compatibility issues. The Docker image they have created contains a complete filesystem including the application, all its dependencies, and minimal runtime requirements. Packaged into a container, their application becomes independent from the infrastructure they want to run it in.

Features

Platform Containerization Plugin

The Platform Containerization Plugin enhances installer recipes with a domain-specific language (DSL). Use DSL to describe an SAP Commerce deployment structure for your Docker images.

The createPlatformImageStructure Ant Task

With the createPlatformImageStructure Ant Task you can easily create Platform image structures containing Platform binaries, configuration, aspect-specific properties as well as Docker-specific files, such as a Dockerfile.

Memory Configuration in Docker Containers

You can configure SAP Commerce memory settings to prevent Docker containers from exceeding memory limits.

Dependencies

If you want to use containerization, make sure that you have access to Docker.

Related Information

<https://www.docker.com/> 

Data Retention

SAP Commerce Platform provides features that cover the concept of data retention. With the Data Retention Framework, you can decide to retain specified data until it is cleaned up according to a strategy you define. With the Item Locking Service, you can lock your data against modification or removal.

The way the **Data Retention Framework** eventually cleans up your data depends on how you define your retention strategies. In a generic scenario, you can remove some data after it has been stored for a specified period of time. For example, you can remove orders after they have been stored for 10 years, or delete customer accounts that have been inactive for 5 years.

For more information, see [Data Retention Framework](#) in Platform Module Implementation.

The **Item Locking Service** may be really necessary if you have to protect data against modification or removal. For example, you may have to make sure that data related to some customer transactions is protected for some important purposes.

For more information, see [Item Locking Service](#).

Data Validation

SAP Commerce is shipped with a data validation framework that helps you to validate your data before it is persisted. It is based on the widely adopted JSR 303 Java validation specification. In addition, it is configurable at runtime. However it is adjusted in a way that it better suits user needs.

For more information, see [Data Validation Framework](#) in Platform Module Implementation and [validation Extension](#) in Platform Module Architecture.

Benefits

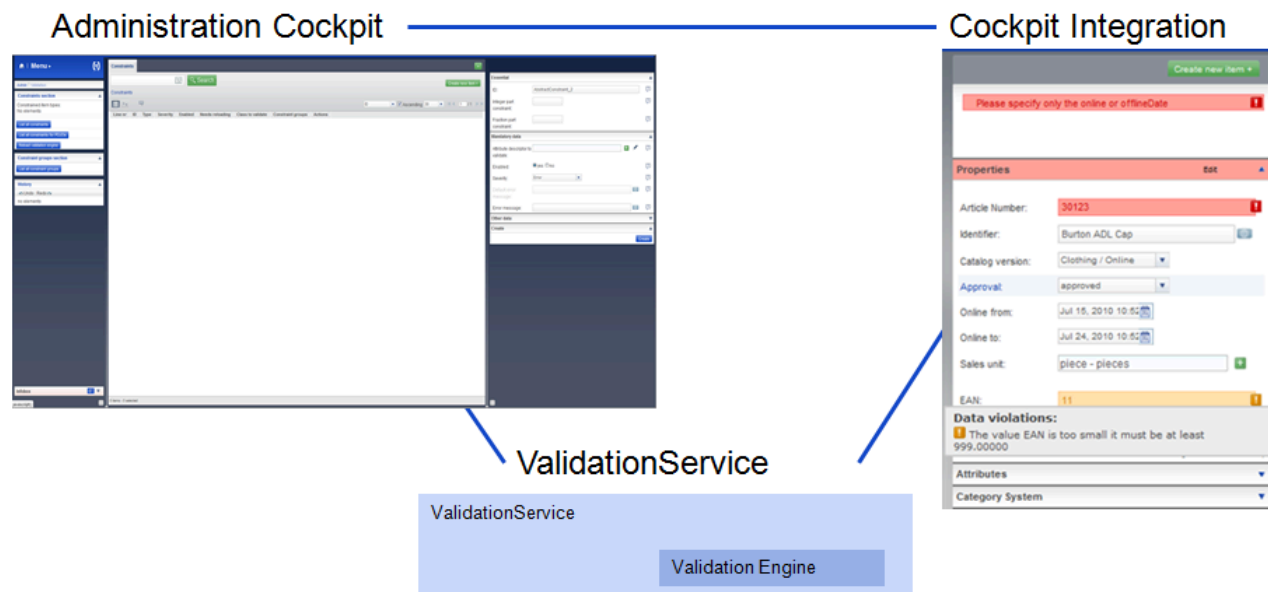
Data validation can be configured in run time and is incorporated in the Administration Cockpit perspective.

For any data-rich application, one of the essential business requirements is to enable data validation before it is persisted. At the front end, you may expect your data to be validated as you type it, or at the latest when you submit it. It should result in user-friendly notifications and guidance to what data needs reentering. At the back end, data validation should ensure that persistence takes place only when the data conforms to given validation constraints. The SAP Commerce Data Validation offers an elegant and extensible solution. For developers, as well as users, it is valuable in ensuring that data is robustly validated before being accepted in the persistence layer.

Key Features

The Data Validation concept concerns the following areas:

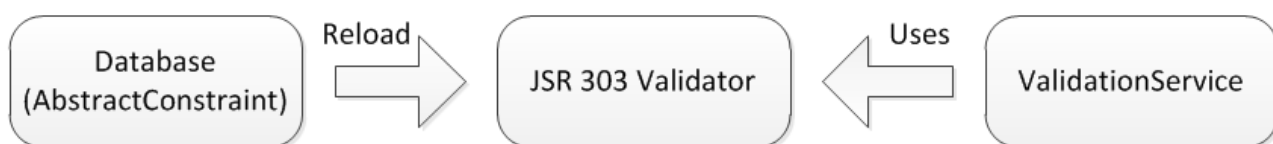
- The **ValidationService**: A new service in the ServiceLayer for performing the data validation.
- The Administration Cockpit perspective: For managing data validation constraints.
- The Cockpit Integration: For providing validation feedback to the user.



Different areas of the Data Validation

The **ValidationService**:

- Loads the validation engine with constraints and invokes its logic.
- Validates data based on a set of validation constraints and returns a list of violations.
- Intercepts all `modelService.save` calls, aborting a save if necessary.
- May also be called explicitly.



Architecture overview

The core of the validation service is a JSR 303-compliant validation engine. It is based upon the Hibernate Validator, which is a comprehensive specification allowing rich validation constraints to be specified. In particular you can create:

1. **Attribute constraints**, for example: Field A of Object O must be between 10 and 20.
 2. **Type constraints**, for example: Only Field A or Field B of Object O must be set otherwise the Object O itself is invalid.
 3. **Dynamic constraints**: Java BeanShell script that returns `true` or `false`.
 4. Constraints with different severity thresholds: Error, Warning, and Info. You may choose, for example, to ignore warning constraints when attempting to submit data.
 5. **Constraint Groups** to which a set of constraints can be assigned, and activated.
- It is possible to create constraint groups during runtime (before, you needed an existing interface and had to restart the VM).

- Default constraint group is always available; it cannot be modified. It shows all unbound constraints.
- There is a flag in `admincockpit` that shows if the current constraint has already been loaded into the framework or not.

i Note

JSR 303 focuses on static data validation constraints: They are encoded in Java or XML files and are loaded once at boot-up. SAP Commerce Data Validation extends this model to allow constraints to be created and modified at runtime. You may define your own constraint types.

The Administration Cockpit perspective enables performing data validation related operations. Creating and managing validation constraints in the Administration Cockpit perspective is intuitive and user-friendly. It enables you to create and manage constraints and constraints groups for the Data Validation. For details refer to [Validation Perspective](#).

Data Validation is integrated into the editing area of the cockpits as illustrated here. All validation problems, for example when adding a new product, are fed back to the user. For details how Data Validation is integrated into the Cockpit Framework go to:

- [Creating Products and Categories](#), *Data Validation* section
- [Product Perspective](#), *Data Validation in the List View Mode* section
- [Product Perspective](#), *Data Validation in the Compare View Mode* section

Related Information

[Data Validation Framework](#)

[Administration Cockpit](#)

Digital Asset Management

SAP Commerce includes media conversion and management tools to simplify the management of every aspect of your digital assets.

An asset is anything that adds value to an item managed in SAP Commerce. For example, if an item is a product, the asset can be product attributes such as description, name, code, approval, or references to other business items like customer reviews and categories. Typical media assets for a product are images, logos, illustrations, audio and video files, presentations, layout page files, office documents, and spreadsheets, CAD files, and other digital files in different formats. All of these can be managed with SAP Commerce's media management tools.

Digital Assets in the Multichannel Environment

Making digital assets available to all channels means managing them in a centralized and consistent manner. Single-sourcing media assets and performing conversion for the differing requirements of various delivery channels is functionality that built in to SAP Commerce.

For more information about how media management is implemented in SAP Commerce, see the related links and the topics that follow.

Related Information

[Digital Asset Management](#)

[Integrating ImageMagick](#)

Media

SAP Commerce supports media files of all sorts. A media can be anything that can be saved on a file system, such as a Flash animation file, a JPEG image, an MPEG video file, a CSV file, a text file, an XML file, and other.

For more information, see [Digital Asset Management](#).

Generic Data Report and Audit

The Generic Data Report allows you to collect raw data stored in the SAP Commerce database and present it as a report that you can easily read and understand. Such a report reflects the current state of data. To create a historical report, use the generic audit feature.

For more information, see [Data Report and Audit](#) in Platform Module Implementation.

You can collect data of **any** item type stored in the database. For example, it can be data about users, or products. The data you can collect about users can include names, credit card numbers, or addresses. Your report is like a snapshot of that data, and reflects the data current state. You can, however, generate historical reports that track changes in data, too. It is possible because the Generic Data Report is aware of **audit records** of the **Generic Audit** feature and it can use **audit records** to generate historical reports (for more information, see [Generic Audit](#)).

The Generic Audit Report is a generic solution. It uses the fact that you can describe dependencies between items and use that description to extract a subset of data from those interconnected items. Generic Audit Report is type system aware, and to support extensibility, it allows every SAP Commerce extension to contribute to a single item-dependency description. A typical item-dependency description starts from a single root type but it can reach other types by following the references known from the type system. Your description serves as a configuration input that a dedicated API can process to generate from it a JSON-like object. This object includes all the data collected for a single instance of a given root type. You can present this data as a report.

Related Information

[Data Report and Audit](#)

ImpEx

SAP Commerce includes a text-based import and export functionality called ImpEx. The ImpEx engine allows creating, updating, removing, and exporting data items such as customer, product, or order data to and from comma-separated value (CSV) data files.

You can use ImpEx to import or export any data from SAP Commerce, both during runtime and during the initialization or update process. For technical details, see [ImpEx API](#) in Platform Module Implementation.

With ImpEx, you can do any of the following:

- Update SAP Commerce data at run time.
- Create initial data for a project, or migrate existing data from one SAP Commerce instance into another during an upgrade, for example.
- Facilitate data transfer tasks such as cron jobs, or synchronization with third-party systems such as an LDAP system, or SAP R/3.

Internationalization and Localization

SAP Commerce provides support for internationalization and localization. It allows you to adapt the system to multiple languages and different requirements that depend on a given region.

If you decide to start supporting multiple languages in your implementation of SAP Commerce, find out about internationalization and localization (I18n). There are special services available and special requirements for characters encoding in the localization files. There is also a mechanism that enables you to easily add new languages to the system.

Related Information

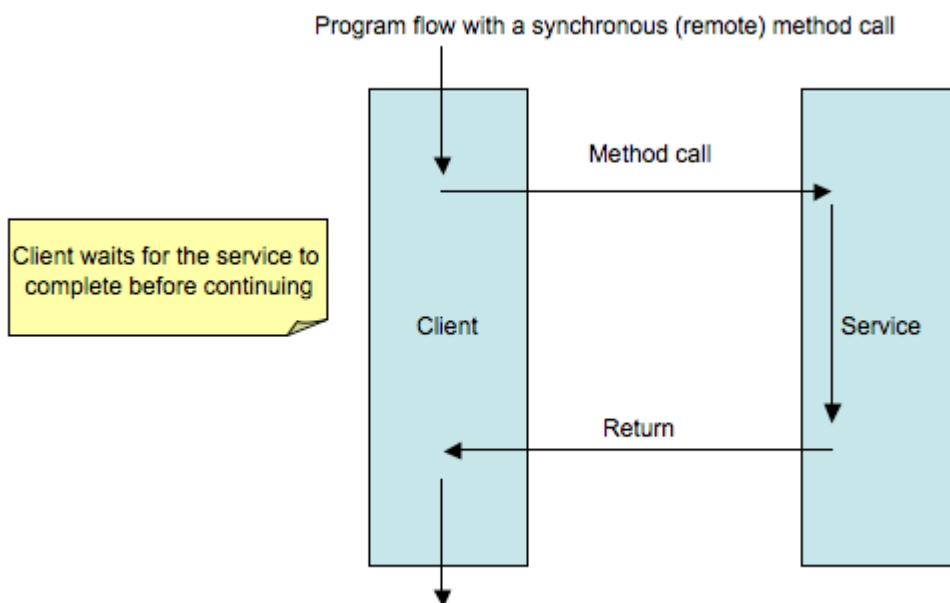
[Internationalization and Localization](#)

Java Message Service (JMS)

Java Message Service (JMS) offers an **asynchronous** approach to (possibly remote) method invocation, that complements the **synchronous** solutions offered by RMI, Hessian/Burlap, Spring HTTP Invoker, and Web services.

About JMS and Synchronous and Asynchronous Method Calls

JMS is a means of invoking method calls in an asynchronous way. This is a different concept to web services, for example, which are invoked in a synchronous way. In a synchronous method call, the client dispatches the call and then waits until the service has completed the task. Graphically, this can be interpreted as follows:



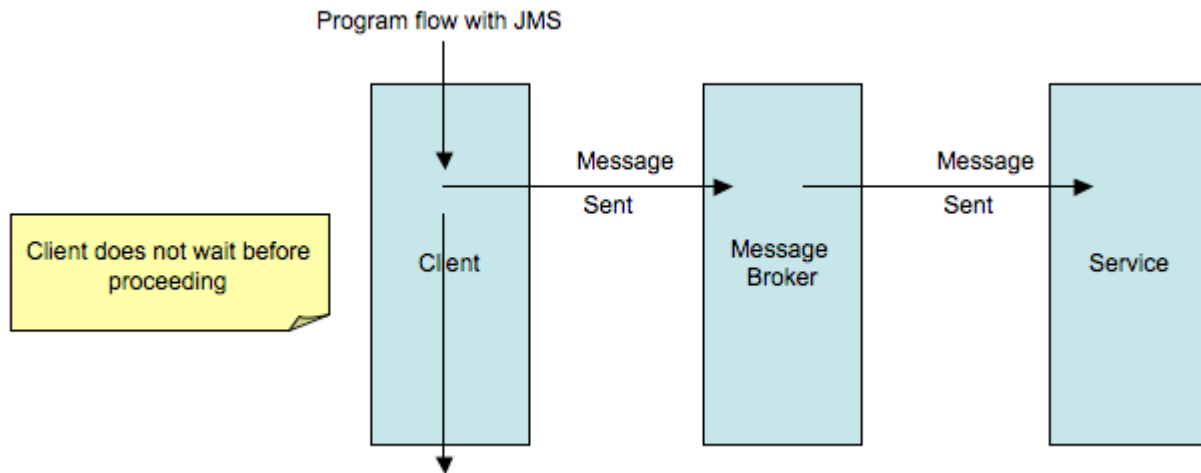
In this scenario when the client makes a call to a service it waits until that service has completed before proceeding further. Often this is the desired behavior but brings with it potential issues:

- Perhaps the service takes a long time to run leading to perceived "hanging" in the client thread.
- Perhaps the client does not really need to wait for the service to end before proceeding.
- The service can be called from any number of clients at the same time, but perhaps the service is CPU intensive and we must ensure that only two such processes can run at any one time on any one CPU.
- Perhaps certain clients should have a higher priority, and be able to trigger their service to run before other clients with lower priority.

Using JMS to Overcome Issues with Synchronous Calls

While these points can all be addressed to some extent with the synchronous approaches, JMS offers a neat solution to all of these points and more.

JMS is based on the notion of messages that are sent from a client to a message broker (for example ActiveMQ). This message broker's job is to then forward those messages to one or more message listeners, which then perform some task as directed by the message's contents. The message broker handles the persistence of messages (so that none is lost if a system crashes) and is transaction aware (so that the service that is called can be run within a transaction if desired).



Once the client has sent the message, it continues rather than waiting for the service to complete. This offers a powerful means for distributing possibly CPU-intensive tasks over both time and over multiple computers.

The message broker can contain many "buckets" into which messages are placed, called Queues and Topics. Queues are used when we would like only one service to act upon a message. Topics are suitable for broadcasting messages to one or more services. Which services receive messages from which Queues and Topics, is specified when configuring the Message Broker.

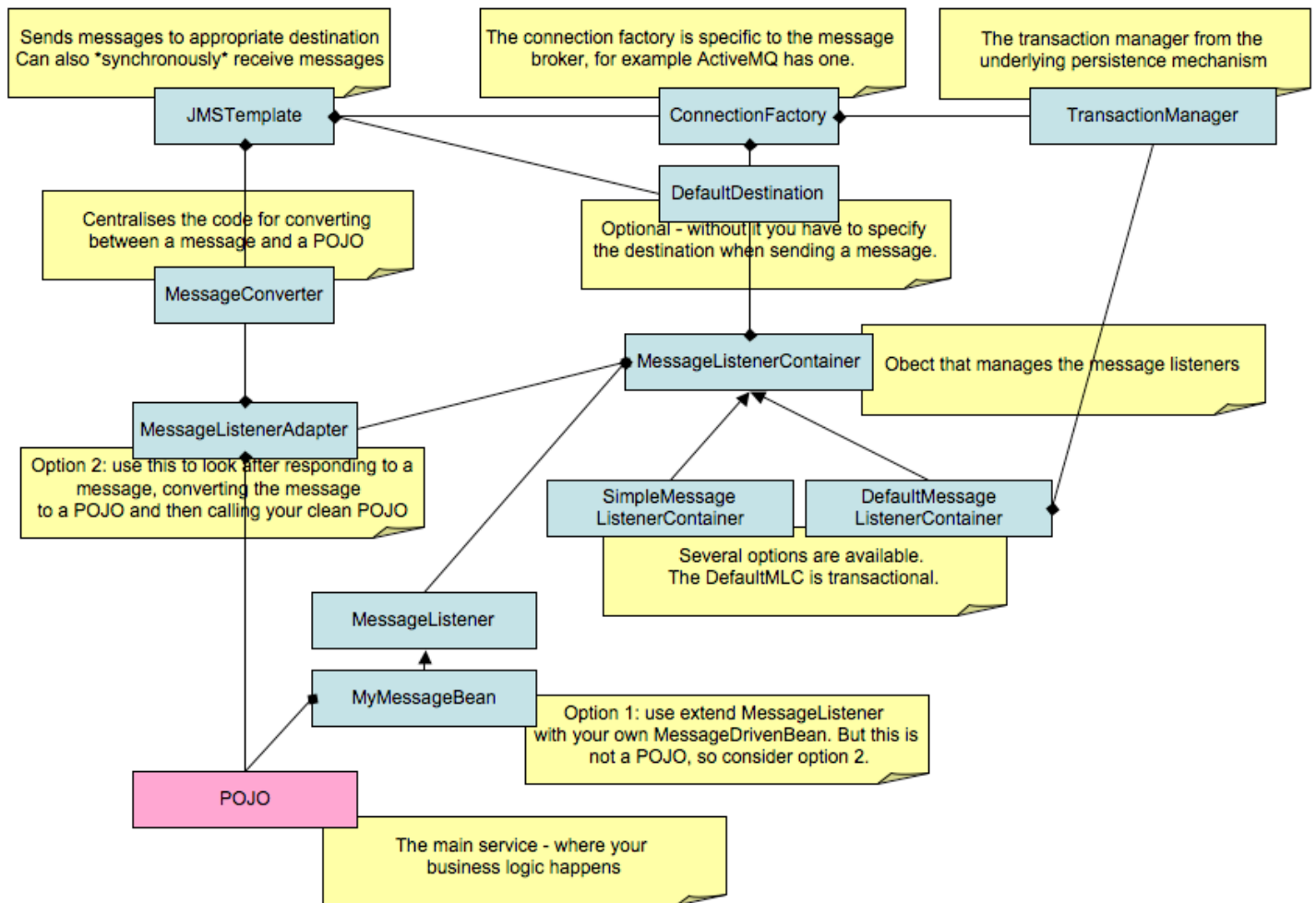
Benefits

JMS has several benefits that complement the problems listed above:

- It does not matter if the service takes a long time to execute, as the client is not waiting for it to complete.
- While the message broker may have received many messages in one go, it passes the message on to the service only when the service is ready. This helps avoid CPU overloading on the service.
- Should a service crash (or be unavailable for what ever reason) the message broker ensures that all messages are persisted and guarantees their delivery when the service is back up.
- It is possible to prioritize messages such that some clients messages can "jump the queue". (While this is not strictly possible with ActiveMQ, there is a way to work effectively simulate it)
- The message broker can vary its output bandwidth, so for example ensuring that a maximum of three service jobs run on one CPU at any one time.
- The message broker is transaction aware so that the service can run within a transaction.
- Should a service throw an exception upon receipt of a message, you can specify how many times the broker should resend that message.
- Should a service repeatedly thrown an exception upon each receipt of one message, we can specify how many retries are allowed before that message is sent to a "Dead letter Queue" where it can be analyzed later by support.

As JMS has been a part of the Java Enterprise Edition for many years, it is fully supported by Spring.

Summary of the JMS Architecture discussed in Spring In Action 2nd Edition



Integration Guide

Every project is different and has individual requirements. Therefore, SAP Commerce cannot provide a customized, out-of-the-box standard solution for JMS integration. The setup you are going to choose for JMS integration depends on system architectures, system landscapes, and other project-related requirements. In the end, you have to define and set up the best-suited JMS integration setup depending on your specific requirements.

To help you getting started, SAP Commerce provides two aspects of JMS:

- SAP Commerce contains the JMS framework for you to use out-of-the-box.

Logging

SAP Commerce provides logging options that allow you to configure how your logs are formatted, sort log messages according to message type and level, or control at runtime where they're reported.

Use Case

Logging is a tool for capturing and persisting data. It allows you to produce log reports that you can analyze in terms of performance results, configuration errors, security failures, or system bugs.

Features

Logging with Log4j

SAP Commerce is shipped with the Apache Log4j 2 logging framework. You can use it directly or with SLF4J.

JDBC Logging in JSON Format

You can produce JDBC log files in JSON format. It allows performance analysis applications such as Kibana and Elasticsearch to parse logs more efficiently.

Dependencies

SAP Commerce comes prebundled with the Log4J logging framework.

Related Information

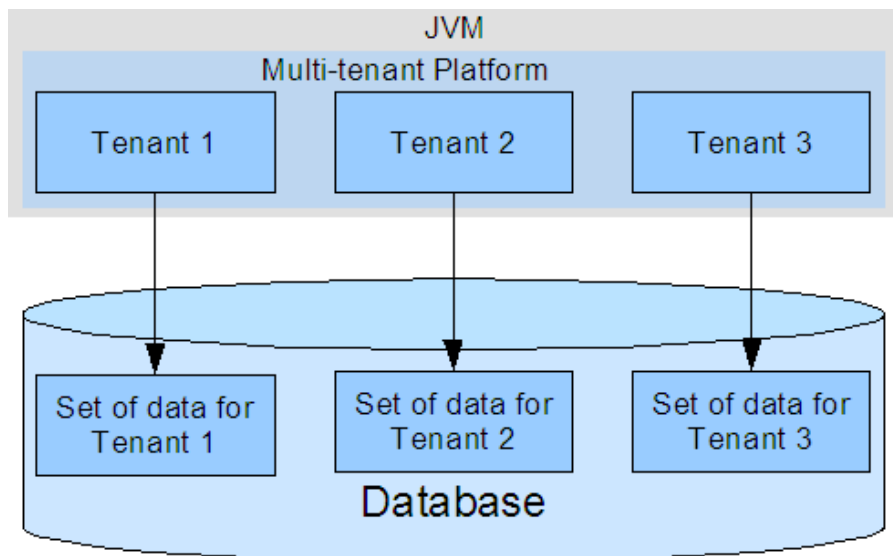
[Logging](#)

Multitenancy

SAP Commerce can be run in a multi-tenant mode. In this mode, several distinct sets of data are maintained on one single SAP Commerce installation. The effect is that you can have several logical SAP Commerce instances running, for example, to host online shops for different customers using one SAP Commerce installation.

Introduction to Multi-Tenant Systems

SAP Commerce allows using individual sets of data across one single database.



Multi-tenant systems are like having several individual SAP Commerce installations with:

- Isolation of data between tenants
- Option of using separate databases
- Option of setting individual time zones and locale settings

Each tenant has a fully individual and isolated set of data.

In general, multi-tenant systems tend to require less system memory than individual SAP Commerce installations. Multi-tenant systems are useful when a SAP Commerce installation is expected to run with several sets of data using the same code base (that is, the same SAP Commerce version), such as:

- Hosting SAP Commerce for several individual customers
- One single, corporate-wide SAP Commerce serving individual countries with individual product and customer data
- Using the SAP Commerce CMS module to power country-specific versions of a company website

i Note

A SAP Commerce Cluster is a concept different from a multi-tenant SAP Commerce. A SAP Commerce Cluster is a number of individual, separate SAP Commerce instances sharing one single set of data, whereas a multi-tenant SAP Commerce is one single SAP Commerce using separate sets of data. For a discussion of running SAP Commerce in clustered mode, refer to the [Cluster - Technical Guide](#) document.

General Overview

Tenants allow customizing:

- Time zones (and, by consequence, the date)
- Database in use
 - Database URL
 - Database driver
 - Database user and password
- Currency
- Currency format
- Date format

By consequence, it's possible for each tenant in one single SAP Commerce installation to use a different database server and a different user account on the database.

It isn't possible to use different versions of SAP Commerce - all tenants necessarily use the same SAP Commerce version (although you're able to specify the extensions available to each tenant). To run different versions of SAP Commerce, you need to use different SAP Commerce installations.

Tenants

The master tenant is the primary tenant of SAP Commerce. The master tenant:

- Uses the database connection specified in the **project.properties** file and no database table prefix
- Is created automatically
- Can't be removed. Even on a non-multi-tenant SAP Commerce, the master tenant is available.

SAP Commerce allows you to use any number of tenants.

Creating New Tenants

Tenants need to be configured in the **project.properties** or **local.properties** file.

To install tenants, the property **installed.tenants** needs to be specified. For example, if the main purpose of your installation is unit testing, you can decide to configure four tenants (besides the default master tenant): junit, foo, t1, and t2.

```
installed.tenants=junit,foo,t1,t2
```

i Note

SAP Commerce comes with the **installed.tenants** property set to **=junit,foo,t1,t2** by default - you don't have to configure it yourself.

Also, for each tenant, a properties file needs to be defined as **tenant_{tenantID}.properties**. The user can configure their own tenant properties files or override the current one, and the files must be put directly under the **config** directory, and the naming convention is **local_tenant_{tenantID}.properties**.

For example, the **tenant_junit.properties** file has the following properties:

```
db.factory=de.hybris.platform.jdbcwrapper.JUnitDataSourceFactory

db.tableprefix=junit_
alt.datasource.AL1.foo=bar
alt.datasource.AL2.foo=bar

slave.datasource.A.foo=bar
slave.datasource.B.foo=bar
slave.datasource.C.foo=bar

hac.webroot=/hac_junit
```

The database prefix can be configured using the **local_tenant_junit.properties**:

```
db.tableprefix=myjunit_
```

Individual tenant data sets are isolated, which means that using another tenant's data is only possible by using exactly the same database connection. Consequently, you need to do an initialization on every individual tenant. Initializing the master tenant doesn't allow any of the other tenants to use the master tenant's data.

i Note

Initialize the master tenant before initializing other tenants.

Production and Development Environments

For faster startup in a production environment, we recommend disabling the junit tenant completely. Set the **installed.tenants** flag to an empty value to have no subordinate tenants. Optionally, you could only add custom partner tenants if there is a need for that.

In a development environment, you may leave the default configuration as it is - **installed.tenants=junit,foo,t1,t2**. It allows you to execute all SAP Commerce tests, also from the SAP Commerce Administration Console test servlet. However, you can always set it up using the configuration recommended for a production environment if you don't need the junit tenant.

Tenants Stay Online

Keeping tenants online at all times is crucial for business operation. We made sure that initializing tenants doesn't cause the master or other tenants to go into a maintenance mode, either within a single or a clustered SAP Commerce installation.

Look at the following scenarios that explain what happens when you're initializing or adding a new tenant.

Scenario with one SAP Commerce instance

The following conditions have been set:

- There's one SAP Commerce instance and it has a master tenant and the **t1**, **t2**, **t3** tenants installed.
- The **t3** tenant hasn't been initialized yet.

While you're initializing the **t3** tenant, all the other tenants always stay online and are available.

If you want to add a new tenant, let's say **t4**, you have to shut down the running server, configure the **t4** tenant, rebuild, and initialize the Platform. As you perform these steps, all the tenants go offline and are unavailable. They're available when the Platform has finished initialization. The only way to keep all tenants available during initialization is to use at least two SAP Commerce instances - see the following scenario.

Scenario with two SAP instances

There are two SAP Commerce instances operating within a cluster, **hybris 1** and **hybris 2**.

- **hybris 1** and **hybris 2** have a master and the **t1**, **t2**, **t3** tenants installed.
- The **t3** tenant hasn't been initialized yet.

While you're initializing the **t3** tenant, either through **hybris 1** or **hybris 2**, all the other tenants always stay online and are available.

If you want to add a new tenant, **t4**, you can do it either through **hybris 1** or **hybris 2**. For a chosen instance, you have to shut down its running server, configure the **t4** tenant, rebuild, and initialize the Platform. In this case, however, all the other tenants stay online and are available despite the fact that you're reinitializing the instance.

Technical Details

SAP Commerce offers the following means to distinguish individual tenants:

- Database table prefix: A prefix added to each database table name for the tenant. Note that some databases are limited as to the maximum number of characters allowed for a table name. The longer the prefix, the higher the probability the parts of the database table names of SAP Commerce get truncated by the database. Refer to the database manufacturer for details.
- Database url
- Tenant username and password

To avoid issues with the length of database table names, the Platform limits the name length of tenants to a maximum of five (5) characters.

Different tenants on a single SAP Commerce installation differ in at least one of these three options. The most common distinguisher is the database table prefix. The following table gives an overview on the effect of a database table prefix.

Tenant	Default table name	Effective table name
(master)	products	products
	productslp	productslp
test	products	test_products
	productslp	test_productslp

Tenant	Default table name	Effective table name
alpha	products	alpha_products
	productslp	alpha_productslp

By default, tenants use the same settings as the master tenant. The default settings for the master tenant are the settings of the Java Virtual Machine the SAP Commerce is running in. In other words: unless explicitly overridden on either the master tenant or other tenants, the tenant uses the Java Virtual Machine's settings for locale and time zone. The settings are used when a session is created by SAP Commerce.

i Note

When specifying an empty table prefix when creating other tenants, make sure that you also specify a database URL. If you don't specify the table prefix and the database URL, the tenant accesses the same database tables as the master tenant, which is likely to cause unwanted side effects.

The fact that each tenant can have individual locale settings has an implication on implementing tenant-related code. The Java.util-related methods and classes retrieve the time zone, date, currency, and date format settings from the Java Virtual Machine by default. As a tenant can optionally have locale settings different from the locale settings of the Virtual Machine, using the Java.util-related methods can cause the tenant-specific settings to be ignored. To avoid this, SAP Commerce offers the locale-specific helper class **Utilities**, located in the **de.hybris.platform.util** package. Refer to the **Utilities** class for a list of available methods. Finally, each tenant has its own core application context, see [Spring Framework in SAP Commerce](#).

Display List of Tenants

To get a list of all tenants available on your system, follow these steps:

1. Open SAP Commerce Administration Console.
 2. Go to the **Platform** tab and select **Tenants** option.
- * The **Tenants Overview** page displays a list of existing tenants.

For more information, see [Administration Console](#)

3. Click the **Update** button.
4. Initialize the tenant. Click the **Initialize** button.
5. The tenant activates and **Initialization** page displays.
6. Click the **Initialize** button.

Registering a Background Thread in a Tenant

In order to have more control over processes running in the background, use the `createAndRegisterBackgroundThread` method available in the Tenant interface. It enables you to register a background thread in a tenant. As a result, before you shut down the Platform, it makes sure that all registered background processes are finished. First, the Platform waits for the time (in seconds) defined by the `shutdown.background.processes.wait.timeout` property (30 seconds by default) for background threads to finish. Next, it interrupts the remaining alive background threads and waits again for the time (in seconds) defined by the `shutdown.background.processes.forced.wait.timeout` property (30 seconds by default). This method prevents problems such as, for example, the task service's polling thread trying to process tasks when master tenant is being shut down, which could have happened in previous versions of the Platform.

Related Information

OAuth 2.0

The OAuth 2.0 authorization framework is the default authorization framework for the commerce driven OCC (Omni Commerce Connect) Web Services.

For more information, see [OAuth2](#) in Platform Module Implementation.

In the traditional client-server authentication model, the client requests an access-restricted resource (in other words, protected resource) on the server by authenticating with the server using the resource owner's credentials. In order to provide third-party applications access to the restricted resources, the resource owner shares its credentials with the third-party application. This creates several issues and limitations:

- Third-party applications are required to store the resource owner's credentials for future use, typically a password in clear text form.
- Servers are required to support password authentication, despite the security weaknesses inherent in passwords.
- Third-party applications gain overly broad access to the owner's protected resources, leaving resource owners without any ability to restrict duration or access to a limited subset of resources.
- Resource owners cannot revoke access to an individual third-party without revoking access to all third-parties, and must do so by changing their password.
- Compromise of any third-party application results in compromise of the end-user's password and all of the data protected by it.

OAuth 2.0 addresses these issues by introducing an authorization layer and separating the role of the client from that layer.

The OAuth 2.0 authorization framework is the default authorization framework for the commerce driven OCC (Omni Commerce Connect) Web Services under the `ycommercewebservices` extension. The key benefit of using OAuth 2.0 (compared to basic authentication, even over HTTPS) is that the API client does not have to save or, in some cases, even obtain the user's credentials. Instead, access tokens are returned to the client that can use refresh tokens to obtain new access tokens once they have expired.

For additional level of security, the client can enable the one-time password functionality for customer login. For more information, see [One-Time Password](#) in Platform Implementation.

Related Information

[About OAuth 2.0](#) ➤

[OAuth 2.0 Authorization Framework](#) ➤

[OCC Calls Security](#)

Ordering, Payment and Pricing Standards

The Platform offers a basic implementation for processing orders, payment methods for transactions, pricing system, and organizing countries and regions for relating shipping costs to them. Moreover, there are a number of services for managing orders.

For more information, see [Ordering, Payment and Pricing Standards](#) in Platform Module Implementation, and [deliveryzone Extension](#) and [paymentstandard Extension](#) in Platform Module Architecture.

Performance and Monitoring

SAP Commerce supports a variety of methods for monitoring application performance allowing you to fine-tune many aspects of your installation.

Related Information

[JMX Monitoring in SAP Commerce](#)

[Performance Tuning](#)

Polyglot Persistence

The polyglot persistence provides a way to store specified data types in an alternative storage, for example document-based storage. Use polyglot persistence to relieve the load of the main database or to provide a non-SQL storage for some data.

Use Case

A technical team managing an SAP Commerce production environment expects a significantly increased year-to-year online store traffic on Black Friday next year. The team members agree that they have to prevent the database from hitting its performance limit on that day. The team decides to protect the main database from some of the expected load and move it outside the database. For that purpose, the team decides to store customer carts in a fast and highly scalable in-memory data grid, and manage all cart-related operations there.

To carry out the project, the team uses the polyglot persistence feature. The team optimizes the data structure of the `Cart` type and its related types as one composed structure (a document) and keeps it in the external in-memory storage.

While shopping, customers add and remove products from their carts multiple times until they place their orders. Those operations don't affect the main database at all because they take place in the external memory. The load on the main database is significantly decreased. It is only when customers place their orders that the main database is affected - orders are persisted there.

Features

Default Storage Implementation

Polyglot persistence offers a default implementation of storage for the `Cart` type.

Unit of Work

A unit of work allows you to cache all modifications performed on a single item, for example on a cart instance, and flush them when a main operation ends.

Dependencies

Polyglot persistence uses the polyglot persistence query language to realize **read** operations from a document storage.

Related Information

[Polyglot Persistence](#)

[ydocumentcart Extension Template](#)

[Polyglot Persistence Query Language](#)

Primary Keys

SAP Commerce provides methods that you can use to filter database data by primary keys.

If you know a primary key (X), you can use a `SELECT * FROM table WHERE PK=X` in such database systems to return a unique result.

Information encoded in each PK includes:

- Creation time (for old UUID-based PKs)
- Counter (for 5.0.0 PKs)
- Typecode

Available Methods

It's possible to:

- Get primary keys by using the following methods:
 - `item.getPK()`
 - `jalosession.getItem(PK)`
- Convert primary keys to a different format:
 - From Strings to PK and vice versa
 - From hex to PK and vice versa
- Extract data from a primary key:
 - `getTypeCode()`
 - `getCreationTime();`

Information Encoded in Primary Key

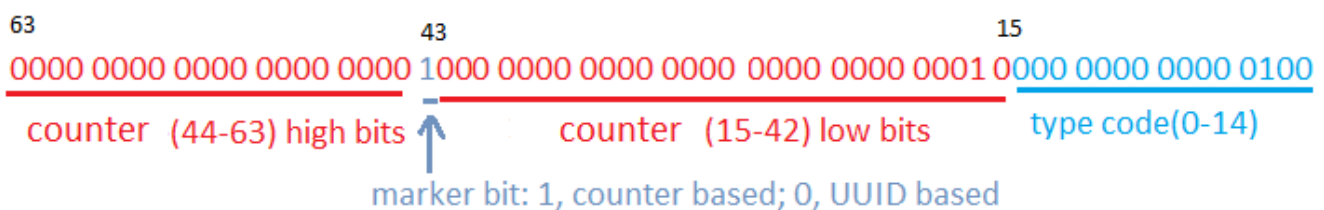
The counter-based PK is used as default, and the UUID PK is deprecated.

That means that there's no information for the creation time included in the PK (be aware that `getCreationTime()` returns the counter for new PKs and the date for old PKs).

Let's have a look at the following primary key:

8796093087748 (decimal format)

0x800000010004 (hexadecimal format)



From the PK we know that it represents a user, and is the second user created in the system. There's no information in the primary key itself about when it was created.

Product and Data Modeling

Whenever you start developing and customizing SAP Commerce, you need to prepare a business model of your application. Here you will find help with setting up your business models.

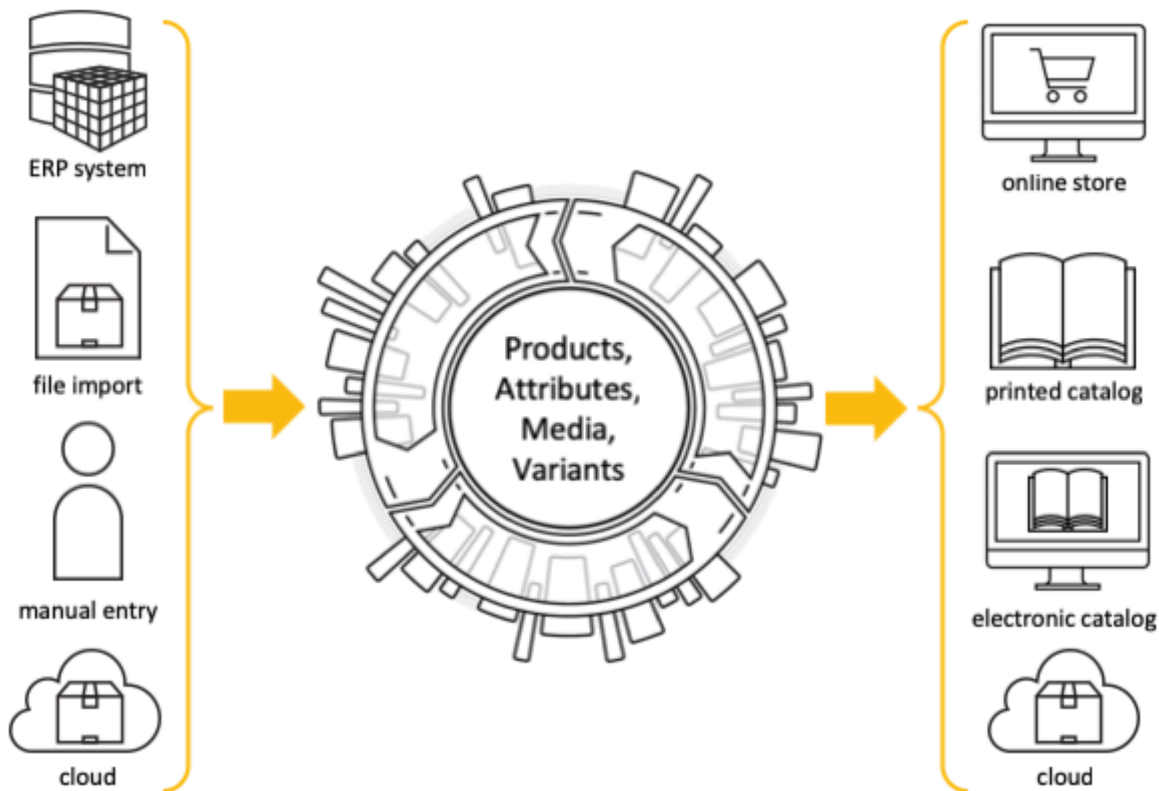
For more information, see [Product and Data Modeling](#) in Platform Module Implementation.

Product Content and Catalogs

The Product Content and Catalog features of Platform provide functionality for holding, structuring, and managing products and product information.

For more information, see [catalog Extension](#) in Platform Module Architecture.

Products can be imported from different sources and organized hierarchically. Through classification, you can define product attributes dynamically. They complement the more static type attributes created by the method of typing products. Catalogs can be exposed to the audience through one or more output channels such as web shops, printed sales catalogs, and electronic catalogs.



Search

There are two search mechanisms implemented in Platform and available without any additional modules. These mechanisms are FlexibleSearch, and GenericSearch. You can also use ViewType, which is a representation of a database view.

For more information, see [Search Mechanisms](#) in Platform Module Implementation.

FlexibleSearch

FlexibleSearch searches SAP Commerce items. It retrieves items in an SQL-based query language. FlexibleSearch search results can be limited by restrictions.

Search Mechanism	FlexibleSearch
Search Base	FlexibleSearch
Used for/Available in	<ul style="list-style-type: none"> the Administration Console FlexibleSearch page the SAP Commerce API
Pro	<ul style="list-style-type: none"> Returns a list of the SAP Commerce items Result list is always up to date Search parameters can be passed by a Java Map at run time
Con	<ul style="list-style-type: none"> Requires basic knowledge of the FlexibleSearch syntax, which in turn requires basic understanding of SAP Commerce type system Errors in built query result in run-time errors

GenericSearch

Generic Search searches SAP Commerce items. It wraps FlexibleSearch into Java Objects. The Backoffice search area uses GenericSearch.

Search Mechanism	GenericSearch
Search Base	FlexibleSearch
Used for/Available in	<ul style="list-style-type: none"> the Backoffice Search Area the SAP Commerce API
Pro	<ul style="list-style-type: none"> Unlike FlexibleSearch statements, errors in built query result in compile-time errors Returns a list of SAP Commerce items Result list is always up to date Search parameters can be passed by a Java Map at run time
Con	<ul style="list-style-type: none"> Even though the FlexibleSearch queries are built via Java objects, GenericSearch still requires a basic knowledge of FlexibleSearch

ViewType

ViewType is the SAP Commerce representation of a database view. It searches on SAP Commerce items. ViewType allows you to conveniently gather attributes from different types, and to define parameters which will be passed to the FlexibleSearch statement at run time. ViewType is mostly used in reporting contexts within Backoffice.

Search Mechanism	ViewType
Search Base	FlexibleSearch
Used for/Available in	<ul style="list-style-type: none"> the Backoffice Report Definitions
Pro	<ul style="list-style-type: none"> Returns a list of SAP Commerce items

	<ul style="list-style-type: none"> • Result list is always up to date • Search parameters can be passed to the ViewType at run time • Allows defining a list of parameters
Con	<ul style="list-style-type: none"> • Requires basic knowledge of the FlexibleSearch syntax, which in turn requires basic understanding of SAP Commerce type system

Related Information

[Restrictions](#)

[workflow Extension](#)

[The Type System](#)

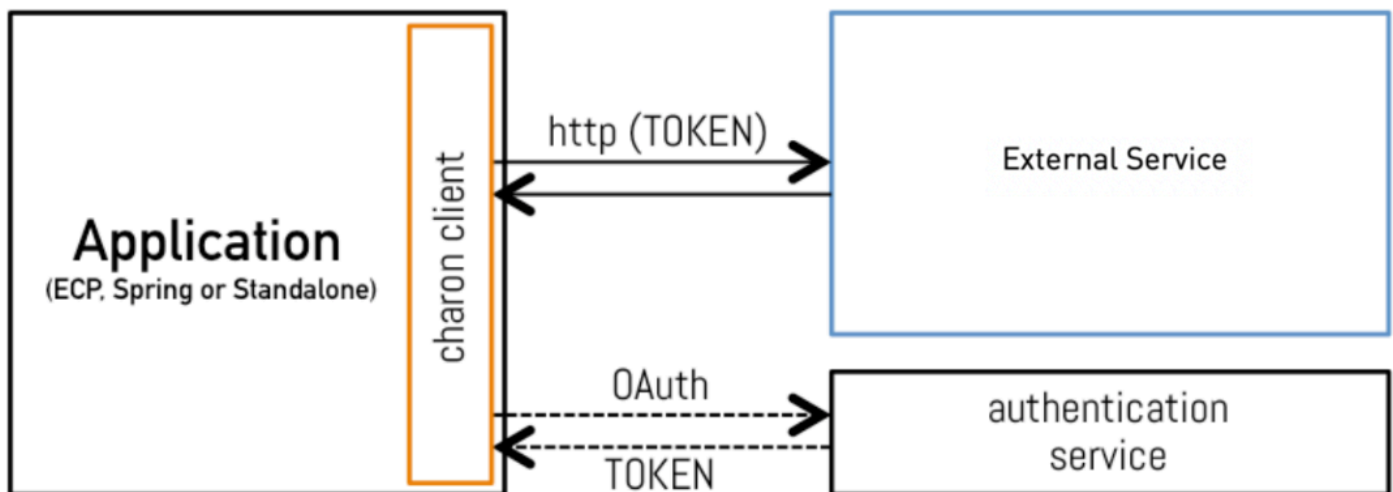
Secure HTTP Transactions

SAP Commerce uses Charon to ensure secure HTTP transactions.

For more information, see [Charon](#) in Platform Module Implementation.

Charon is a library that allows applications to interact with external (micro)services. It may be integrated into either new or existing applications, providing a framework for the composition of asynchronous HTTP transactions. Charon is lightweight, built on Netty, and based on RxJava. As a library for RxNetty, it is entirely generic, meaning it is not bound to the Platform but may be used with any software solution. Charon performs HTTP requests in a declarative and easy way.

In addition to standard HTTP requests, Charon can use OAuth to retrieve a secure token from an external authentication service. This token is used to authenticate subsequent requests on the remote service.



Features of Charon include:

- Asynchronous calls through Java NIO api
- Control over timeouts and retries
- Control over concurrent requests
- Declarative HTTP remote description through JAX-RS 2.0 annotations
- Transparent OAuth authentication with a time-to-live token
- Connection pooling

- Multipart uploads
- Spring integration
- SAP Commerce ECP integration
- SAP Commerce Data Hub integration
- Pluggable configuration

Security and User Management

SAP Commerce provides tools for security and user management including access control and data encryption.

After installing SAP Commerce, it's best practice to look at all aspects of security. Appropriate access rights, user management, and user groups ensure that items are accessible only to authorized users. Your security policy can also include data encryption, authentication protocols such as LDAP, or the Spring Security framework.

Related Information

[Access Rights](#)

[SAP Commerce Security](#)

ServiceLayer

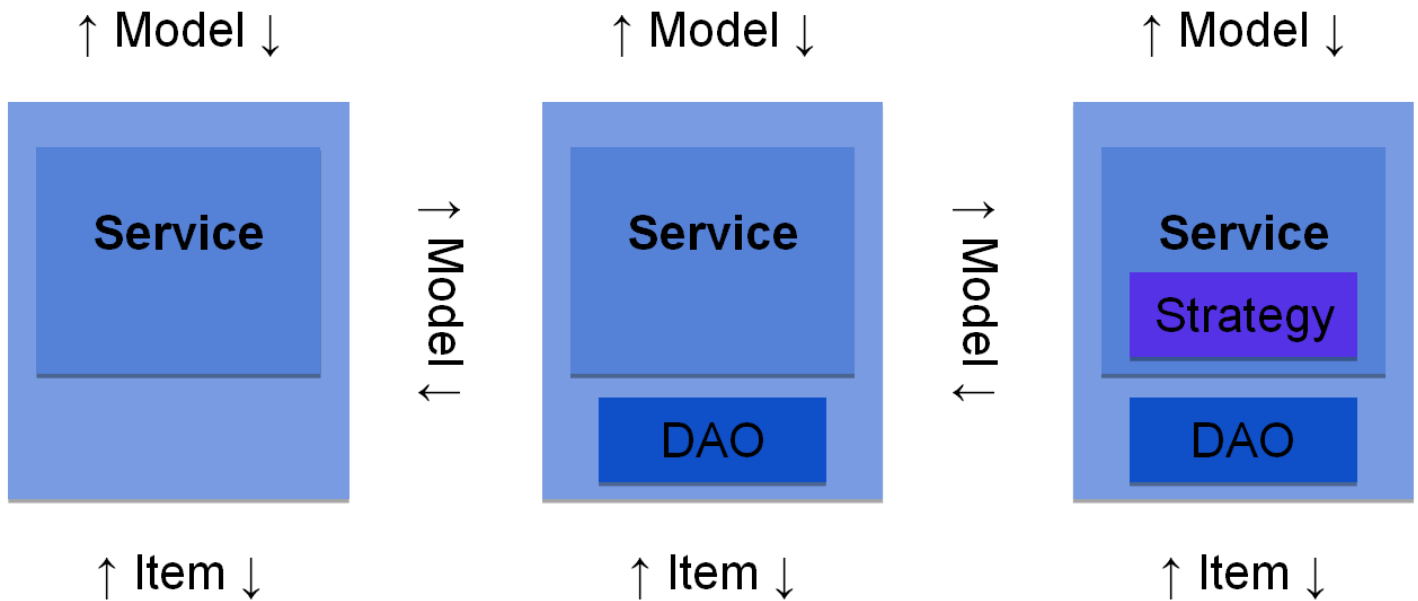
The SAP Commerce ServiceLayer is an **API** for developing services for SAP Commerce. It provides a number of common services, while allowing you to extend these or develop your own.

For more information, see [ServiceLayer](#) in Platform Module Implementation, and [platformservices Extension](#) in Platform Module Architecture.

The main characteristics of the ServiceLayer are:

- It is based on a service-oriented architecture.
- It provides a clean separation of business logic and persistence logic.
- It provides a number of services, each with its well-defined responsibilities.
- It provides a **framework** to develop your own services and to extend existing ones.
- It is heavily based on the **Spring** Framework.
- It is based on **common patterns**, such as interface-oriented design and dependency injection.
- It is the layer where partners should implement their business logic.
- It provides hooks into **model life-cycle events** for performing custom logic.
- It provides hooks into system event life-cycle events such as **init and update** process.
- It provides a framework for publishing and receiving **events**.

Client



Persistence Layer

Overview of the integration of the SAP Commerce ServiceLayer.

For a discussion of the components, please refer to [ServiceLayer Architecture](#).

The objectives of the ServiceLayer are to be:

- Consistent
- Easily approachable
- Comprehensive
- Adaptable
- Extensible
- Flexible

→ Tip

If you are new to the ServiceLayer, read the following documents. This will help you get started and allow you to work with the ServiceLayer more effectively:

- Essential knowledge, overall:
 - Spring Framework: <http://static.springframework.org/spring/docs/2.5.x/reference/index.html> 📖
 - Especially the chapter about the IoC container: <http://static.springframework.org/spring/docs/2.5.x/reference/beans.html> 📖
- Essential knowledge, specific for SAP Commerce:
 - [Spring Framework in SAP Commerce](#)

Related Information

[ServiceLayer](#)

ServiceLayer Direct

ServiceLayer Direct allows you to read and persist data in the database and completely skip the Jalo layer. With ServiceLayer Direct in place, you still use `ModelService` to manage models.

Configuration

To enable ServiceLayer Direct globally, set the `persistence.legacy.mode` property to `false`. This setting enables writing and reading operations through the new layer. You can enable ServiceLayer Direct just for a current session context, but have it disabled globally. See the implementation example:

```
PersistenceUtils.doWithSLDPersistence(() -> {
    final TitleModel title = modelService.create(TitleModel.class);
    title.setCode("foo");
    modelService.save(title);

    return title;
});
```

Limitations

Because some code may still have business logic in the old Jalo classes in place (especially in Jalo attributes), we don't recommend switching on ServiceLayer Direct. In the following cases, you cannot use ServiceLayer Direct:

- Item attribute is configured as Jalo in `items.xml`. It means there is a business logic in a Jalo class (Item or Manager) for a getter or setter.
- Item attribute is configured as a property in `items.xml` but it has an overridden getter or setter in a Jalo class
- Item has a custom logic in `createItem(...)` protected method (usually for validation or preparation purposes)

There is an easy way to migrate such a code and make it ServiceLayer Direct enabled. However, treat each case individually. There isn't one correct way to migrate. Migration would usually consist in changing Jalo attributes into dynamic attributes, and moving the original logic into a dynamic attribute handler, or creating proper interceptors (prepare, validation, or remove interceptors).

If for some reason it is not possible to migrate Jalo logic, mark an appropriate method or type with the `ForceJALO` annotation. Such an annotated type is always persisted and read using the Jalo layer. Such annotations have informational value and help you quickly locate a problematic code for further migration.

There is also the `SLDSafe` annotation that has an opposite meaning to `ForceJALO`. Use `SLDSafe` to provide a hint that an item is fully migrated and is safe to persist and read through ServiceLayer Direct.

Another way to force a particular type through the Jalo layer is to mark it directly in `items.xml` as follows:

```
<itemtype code="Foo">
  <custom-properties>
    <property name="legacyPersistence">
      <value>java.lang.Boolean.TRUE</value>
```

```

</property>
</custom-properties>

```

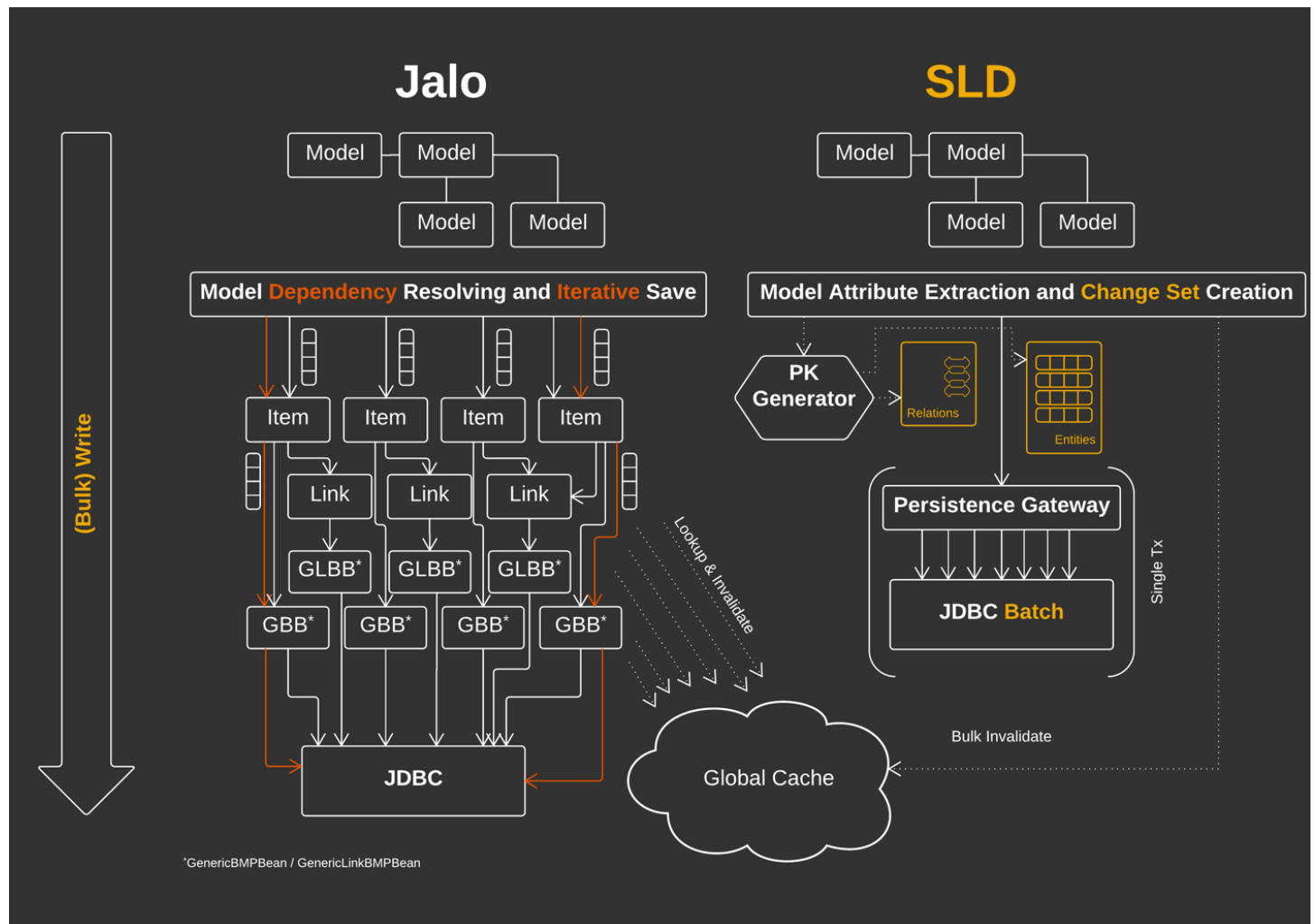
```

...
</itemtype>

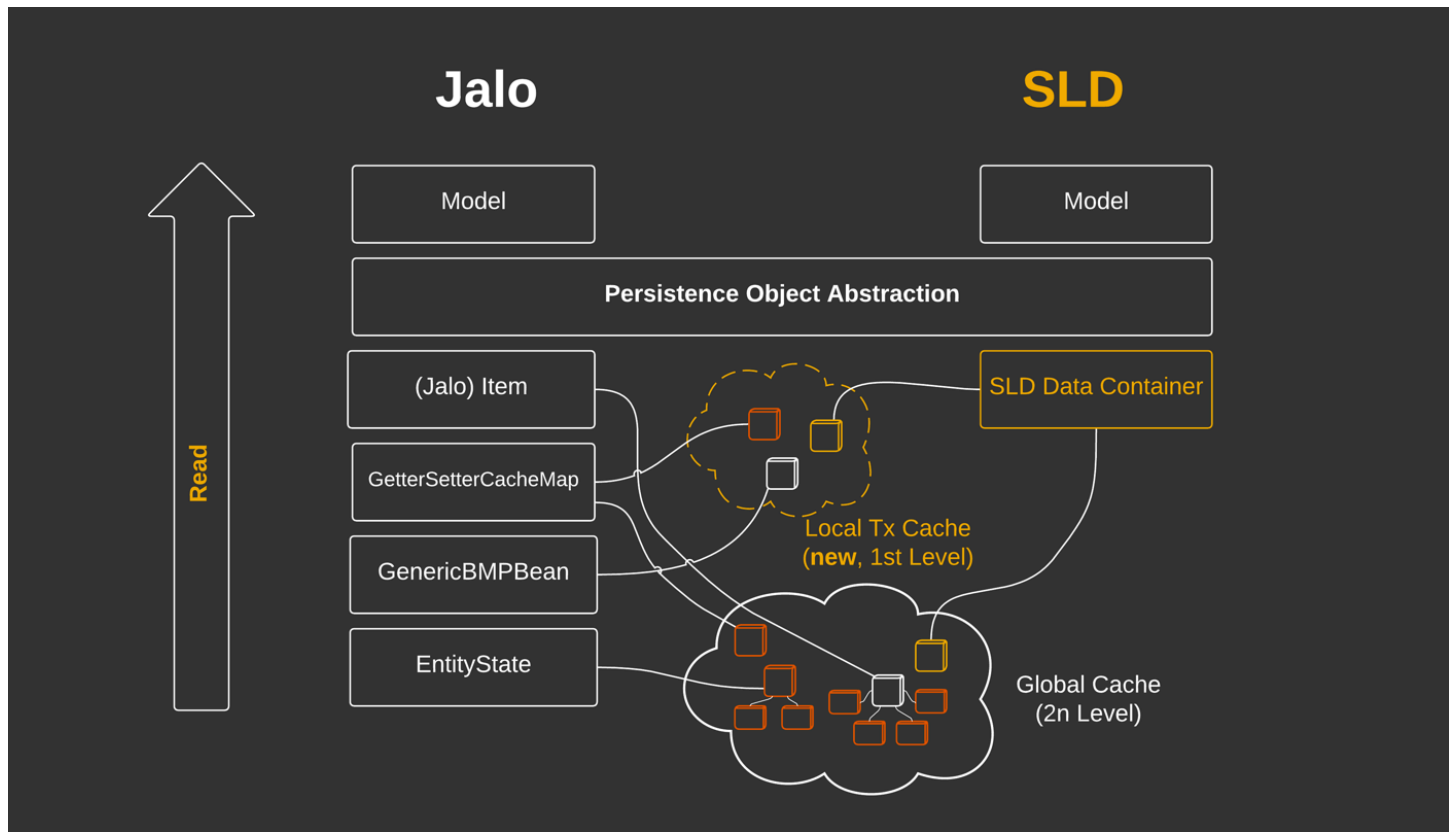
```

Architecture Diagrams

Here is a comparison of **write operations** in ServiceLayer Direct and Jalo:



Here is a comparison of **read operations** in ServiceLayer Direct and Jalo:



Related Information

[Jalo Layer](#)

[Jalo-Logic-Free Extension](#)

Workflow and Collaboration

Platform workflow and collaboration tools make defining complex organizational processes easier and more transparent.

For more information, see [Workflow and Collaboration](#) in Platform Module Implementation.

Workflow and collaboration business benefits include:

- Support of complex processes in company environment
- Support of single and easy tasks in company environment
- Easy handling and storage of information given by employee for any SAP Commerce item, for example: products or medias

With workflow and collaboration you can perform the following tasks:

- Track a workflow
- Start a workflow via SAP Commerce CronJobs
- Create workflow templates
- Design human-based and system-based workflows
- Allocate users and user groups to certain tasks
- Get real-time notification via e-mail
- Comment on the task
- Add task-relevant attachments

The collaboration functionality is also included in the `comments` extension of the Platform. For more information, see [Comments Integration](#).

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

Key Aspects

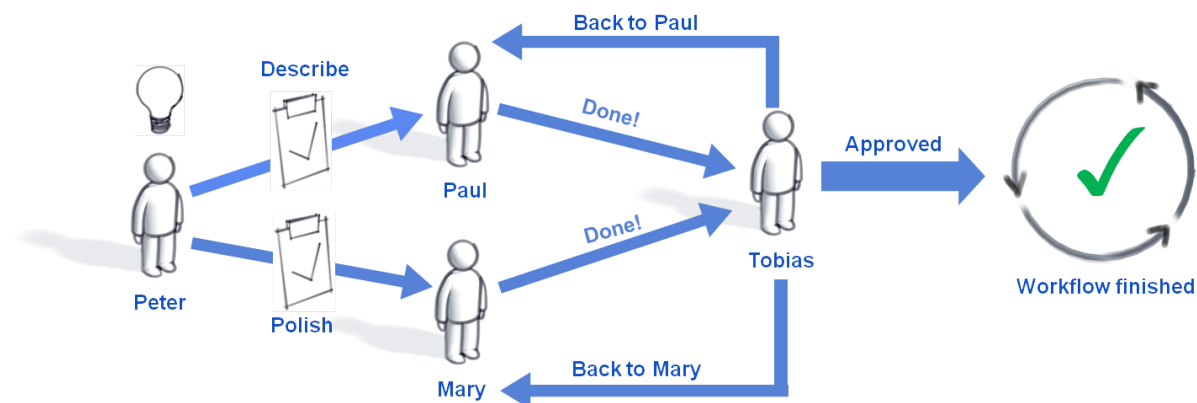
Workflow & collaboration provides you with the following functionality:

- User-defined workflow triggering: create, extend, start, rename, delete, terminate
- Two types of workflows
 - Predefined workflows for complex processes
 - Ad-hoc workflow for one single task
- The workflow section is visible only for users with the assigned workflow templates

Example of Usage

Peter is a purchase manager and has just created some new products.

1. Peter wants the product manager Paul to describe the products
2. And the designer Mary should polish the product images
3. As the last step the products information needs to be approved by marketer Tobias - but only if both previous steps are finished
4. Tobias should also be able to reject the workflow task back to Paul or/and Mary if necessary
5. Lucky Peter - with SAP Commerce he will just have to trigger a predefined customized workflow!



Related Information

[Workflow Integration](#)

[Workflow Overview](#)

[workflow Extension](#)

[Comments Integration](#)

Workflow Overview

You can model in-house business processes within an SAP Commerce workflow.

Typical business process workflows include the following:

- Handle customer complaints
- Create and go live with a website paragraph
- Rewrite and review a product description

Key Concepts

The following terms relate to the SAP Commerce workflow extension:

Key Concept	Description
Workflow Template	A template for a workflow. It consists of workflow action templates and defines the basic sequence of a workflow in the form of options.
Workflow Action Template	A template for a workflow action. Logically, a definition of a step in a workflow. It is linked to other workflow action templates via options.
Option	Defines a possible result of a workflow action template, that is, is a template for a decision. It also defines which steps might follow after the current workflow action Template.
Workflow	Represents an actual workflow. It is created by copying from a workflow template.
Workflow Action	Represents an individual step within a workflow. It is copied from a workflow action template in the process of copying a workflow from a workflow template.
Decision	The actual choice of the options selected. A decision stores the result of the action.
Comment	A note left on a workflow action. For example, to leave a piece of information about how to tackle the upcoming workflow action, or to keep track of how somebody worked on the current workflow action.
VisibleForPrincipals	An attribute of a workflow template, used for assigning users to a particular workflow template in order to enable users to see and create new workflows. If the user has no template assigned, then the workflows templates section is not shown at all.

User Roles

The SAP Commerce workflow extension differentiates between three different user roles:

User Role	Description
Workflow Modeler/Workflow Designer	Sets up and defines workflow templates.
Workflow Manager	Creates actual workflows from workflow templates.

User Role	Description
Workflow Assignees	In-house employees who perform individual steps on a workflow.

Basic Principles

Workflows and Workflow Templates

With the workflow extension, you can do the following:

- Define workflow templates
- Create workflows from workflow templates
- Process workflows

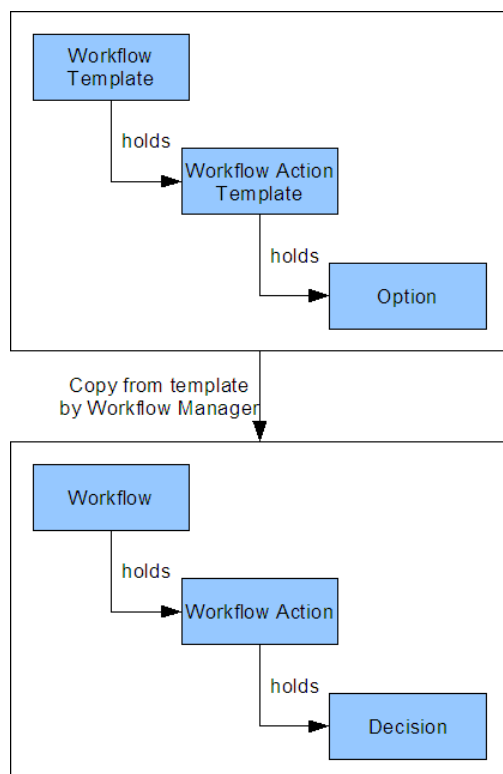
You create a workflow by copying a workflow template into a workflow. The workflow template is persistent and you can reuse it to create any number of workflows. However, workflows created from a template always have the same functionality unless explicitly modified. To use two different workflows, you create two different templates.

You can create a workflow from a workflow template in one of the following ways:

- Manually, using the Backoffice or ImpEx script.
- Scripted, using an activation script defined on the workflow template. The workflow extension comes with a scripting extension. The scripting extension automatically creates a workflow from a workflow template based on an activation script.

Actions and Workflow Action Templates

A workflow consists of steps called workflow actions. A workflow template defines the sequence of workflow actions in the form of workflow action templates. The link between workflow action templates defines the basic sequence of the workflow. Just as you create workflows from workflow templates, you create workflow actions from workflow action templates which are defined on the workflow template.



Once created from a workflow template, the basic sequence of a workflow cannot be changed. You can set another assignee, another status, and determine whether a certain action begins or completes the workflow. The sequence of individual actions, however, cannot be changed afterwards.

A workflow action is always in one of the following statuses:

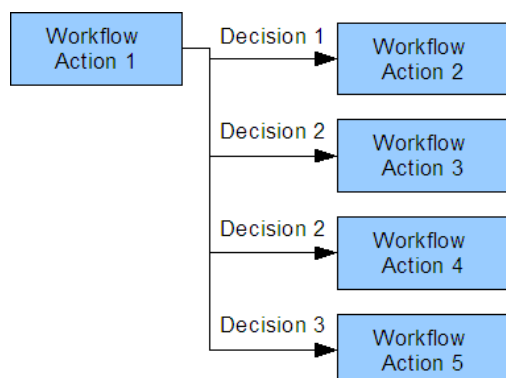
Status	Description
pending	Indicates that the action is on hold because at least one of its preceding actions has not been completed.
in progress	Indicates that this action is unlocked and can be worked on. During the creation of a workflow from a workflow template, every workflow action with its type set to start will be set to this status automatically unless it was created through one of cockpits, in that case workflow has to be started manually.
completed	Indicates that the person who worked on the action thinks that the action has been dealt with. Any actions subsequent to this one will be set to in progress automatically. During the creation of a workflow from a workflow template, every workflow action with no subsequent action will be set to this status automatically.
disabled	Deprecated status, used for backwards compatibility only.
ended through end of workflow	Indicates that an action marked as end has been reached and, by consequence, that the workflow has completed before the action began.

Processing a Workflow

Workflow action templates are connected by options. An option defines that an workflow action template can be followed by another workflow action template. On a workflow, the selected results of workflow actions are saved in the form of decisions. Decisions are working copies of options.

When a decision has been selected for a certain action, the status of that action is set to **completed**. If the decision leads to any subsequent actions, those actions are activated and set to **in progress**. Only those actions specified by the decision are activated. If an action has different potential subsequent actions, the selected decision determines which of the potential actions is activated.

For example, the diagram below shows one workflow action that has three potential options. In an actual workflow, the chosen option is stored as decisions. Selecting **Decision 1** activates **Workflow Action 2**, **Decision 2** activates **Workflow Action 3** and **Workflow Action 4**, while **Decision 3** activates **Workflow Action 5**.



In addition, it is possible to create comments on workflow actions. The `workfLow` extension automatically creates comments, for example when a decision has been selected. But users can also manually create comments; for example, to keep notes of how they worked on a certain workflow action or to give the subsequent assignee tips or instructions.

A user account that is the new assignee of an action receives a message in the account's inbox in Backoffice. In addition, the `workfLow` extension can optionally send an email to the assignee for information purposes.

For example, the `workfLow` extension allows creating a quality control cycle in which the first action item is to write a certain text. Once satisfied with the result, the writer marks the action item to be completed and automatically triggers the subsequent action item.

The subsequent action item is to review the written text. If the text is accepted during the review, the text can be processed further (that is, the action item subsequent to the review is activated). If the text is rejected during the review, the text is re-written—that is, the first action item is activated again.

Workflow Procedures

Fundamental workflow procedures include creating a workflow template, creating a new workflow from a template, and processing a workflow.

Each of the fundamental workflow procedures are discussed in turn in the following sections.

Creating A Workflow Template

i Note

Sketching workflows recommended

As workflow models might end up being rather complex, it is easy to create workflow templates from memory. It is very useful to sketch down the workflows you want to model. This makes modeling easier: you only have to transfer the sketches into the `workfLow` extension-related objects.

Workflow designers and modelers can create workflow templates. A workflow template consists of templates for individual workflow actions and connections between the individual workflow actions. The connections are called options and also define the potential results of an action. Which action is triggered after a given action depends on the option selected. The workflow template defines the basic sequence in which individual actions have to be processed via options. In short, the `workfLow` extension uses two kinds of templates to represent workflow models:

- Workflow action templates, which represent the individual steps of a workflow.
- Options, which represent the connections between individual steps.

The `workfLow` extension also allows you to define conditions on options. If two or more options link to the same workflow action template, you can set the incoming connection to be OR-connected or AND-connected.

- On OR-connected options, the target workflow action template is activated if at least one of the incoming options is set to the respective value later on.
- On AND-connected options, the target workflow action template is activated only if all of the incoming options are set to the respective value later on.

A workflow action template can also be set to send an email notification: if the action, which is copied from the workflow action template, is activated (that is, the action's preceding actions are completed), then an email is sent to the assignee (using the SAP Commerce email settings). Whether or not an email is sent can be set for the action at any time.

The `workflow` extension also distinguishes between three different types of workflow action templates in a workflow template. Each workflow action template is marked as one of three possible states: start, normal, and end.

Workflow action templates marked as start represent the beginning points of a workflow. Workflow action templates marked as end represent the completion points of a workflow. Workflow action templates marked as normal represent actions that are neither beginning points nor completion points of a workflow.

Type of Workflow Action Template	Description	How to identify in an Activity Diagram?
start	Beginning points of a workflow.	<ul style="list-style-type: none"> Every step that is in the first row of a workflow Every step whose direct ancestor is an initial node
normal	Neither beginning points nor completion points of a workflow.	<ul style="list-style-type: none"> Every step that is neither in the first row of a workflow nor in the last row of a workflow Every step whose direct ancestor is not an initial node and whose direct successor is not an activity final node
end	Completion points of a workflow.	<ul style="list-style-type: none"> Every step that is in the last row of a workflow Every step whose direct successor is an activity final node

- If a workflow action template is marked as start, then the status of the workflow action copied from this workflow action template is set to in progress automatically. In addition, every workflow action with no subsequent workflow action is automatically set to completed.
- If no workflow action template of a workflow is marked as start, then the status of all workflow actions is automatically set to pending.
- If a workflow reaches any workflow action marked as end, every single workflow action in the workflow is marked to be ENDED THROUGH END OF WORKFLOW. This means that the entire workflow is completed once the first workflow action marked as end is reached.

→ Tip

Define all Workflow Action Templates before creating Options

Interconnections between individual workflow action templates (and, by consequence, between workflow actions later on) are specified by the options set for individual workflow action templates. For an individual option, you are only able to select workflow action templates that already exist. It is not possible to create a workflow action template from an option. Therefore, SAP Commerce recommends creating all of the workflow action templates first and then the interconnections.

Creating A Workflow From A Workflow Template

Workflow managers can create a wWorkflow from a workflow template manually or have copies created automatically via the activation script. A workflow template is used as a blueprint for an actual workflow. The workflow template is copied into a workflow, creating copies of all related objects in the process (such as actions, options, etc). In essence, the workflow is a working

copy of the workflow template, and all objects referenced by the workflow template are working copies, too. It is not possible to change the basic action sequence pre-defined by the workflow template.

In addition, attachments can be added to the workflow. An attachment is a reference to an item in the SAP Commerce that is intended to be reviewed in the course of the workflow--a modified product or category, for example.

A workflow action has a status indicating its processing state. If a workflow action template is marked as **start**, the status of the workflow action copied from this workflow action template is set to **in progress** automatically. In addition, every workflow action with no subsequent workflow action is automatically set to **completed**.

If no workflow action template of a workflow is marked as **start**, then the status of all workflow actions is automatically set to **pending**.

For information about available statuses, see [Workflow Actions](#)>

Processing A Workflow

An action can be assigned to a user or user group. The user or any member of the user group can then work on the action. By selecting an option from the list of potential decisions (the result of the action, so to speak), the user triggers the next action. Which action that is depends on the sequence of actions as predefined by the workflow template and the selected option.

Generally, actions are processed from an action marked as **start** to an action marked as **end** via any number of actions marked as **normal**. This does not have to be a direct or uninterrupted path; some actions may be processed several times, while other actions may not be processed at all.

If a workflow action is assigned to a user account to process, the workflow action shows up in the user account's Inbox in the Backoffice. The Inbox allows a user to quickly see the workflow actions assigned to them. In addition, a workflow action can also be set to send an email notification: if the action is activated (that is, the action's preceding actions are completed), then an email is sent to the assignee (using SAP Commerce email settings). Whether or not an email is sent can be set for the action at any time.

Related Information

[Workflow and Collaboration](#)
[workflow Extension](#)

Workflow Integration

The SAP Commerce workflow extension enables the modeling and processing workflows mainly based on the user interaction (like batch cards).

The SAP Commerce workflow extension is intended to provide an easy way of modeling these workflows by using a template mechanism as well as a comfortable way of processing the actions of a workflow from the assignee's point of view.

The functions of the extension can be summarized in the categories modeling and processing.

Backoffice offers an advanced Workflow managing system. For details, see [Collaboration Center in Backoffice Framework](#).

Modeling

A workflow instance is always based on a template which is reusable for other workflow instances:

- Create your own workflow template per click
- Use a template to easy creating instances
- To model automated workflow actions without user interaction, write your own **Job** definitions
- Marker for start and end actions (for automatic start and end)
- Multiple decision options for each action
- Logical couplings of actions (Incoming transitions can be marked with AND or OR where all transitions with an AND have to be fired or one of the transitions with OR marker.)
- Building iterations is possible.

Processing

The processing of an interaction-based workflow action is quite easy. It provides:

- An inbox where all you can access all your active actions, as well as the processed and queued ones
- An e-mail notification if configured for the actions
- Attachments for each action to mark the items to process related with action
- List of comments. A comment is added to the history of an action if the action gets activated and consists of a generated message containing the activators.

Limitations

Note these limitations to the SAP Commerce workflow extension.

- The workflow extension supports simple and moderate workflows without logical checks. However, you can use tasks represented by textual directions and related by basic logic functionality, parallel execution, and iterations.
- Because there is currently no graphic modeling tool available, modeling of complex workflow is not as well supported as you may expect it.
- By default, the SAP Commerce workflow extension does not support a combined usage with other workflow engines. This is because it is based completely on the SAP Commerce type system and does not provide standardized interfaces like BPEL. For details contact Professional Services.

Related Information

[workflow Extension](#)

[Workflow and Collaboration](#)

Comments Integration

SAP Commerce provides you with great tools to support collaboration in your working environment: comments and workflow functionality.

The comments functionality offers the easy handling and storage of information given by employee for any SAP Commerce item, such as products or medias.

Key Aspects

- You can comment every SAP Commerce item

- Item-specific comments are shown in its own section in the editor area
- Preview of comment on rollover in the editor area
- You can attach items to comments
- "Easy reply" on comments in the editor area
- Detailed view on comments in the browser area
- Commenting the multiple items at the same time
- Commenting the attachments of a task

Comments			
We will have additional variants	Paul Product	19.07.	↗
▶ This is my 2nd hat of this type.	Peter Purchase	19.07.	↗
This seems like a good knife, e:	Tobias Marketing	19.07.	↗
▶ As I have just heard this produ	Paul Product	19.07.	↗
Hi, I found some bad translation	Mary Me	19.07.	↗
▼ Hey all, We need to change des	Peter Purchase	19.07.	↗
▼ When shall I start with work	Mary Me	19.07.	↗
You will have to start on a	Tobias Marketing	19.07.	↗
Yes do it!	Product Manager C	19.07.	↗
▶ When shall I start with work	Mary Me	19.07.	↗
Just testing!	Administrator	19.07.	↗
▶ Please check the categories!	Product Manager C	19.07.	↗
▶ This product has an awful produ	Product Manager C	19.07.	↗

Figure: The comments in the editor area.

Communication Center

All products

Search

Communication Center

Time modified 30

Paul Product, 7/19/10: Product variants

Peter Purchase, 7/19/10: Try out!

Tobias Marketing, 7/19/10: Great product! Tested it :)

Paul Product, 7/19/10: Topseller Product

Mary Me, 7/19/10: Wrong translation to german

Peter Purchase, July 19, 2010
Need to change description

Comment Details

Hey all, We need to change description of this product because of our new launch plan for topsellers. Also the assets should be changed. They are really old fashioned and don't have our companys' spirit anymore. We must initiate new shots from the photograph. And don't forget to tell him about our new CI we created last year.

Mary Me, July 19, 2010
RE: Need to change description

When shall I start with work on this product? Wanted to plan my vacations.

Attachment: no attachments available

Tobias Marketing, 7/19/10: RE: Need to change description

Product Manager Demouser, 7/19/10: Need to change description

Mary Me, 7/19/10: RE: Need to change description

9 items - 0 selected

Figure: Detailed view of comments in the browser area.

Related Information

[comments Extension](#)

[Workflow and Collaboration](#)

Platform Architecture

Platform is a set of extensions providing the main functionality of a SAP Commerce installation.

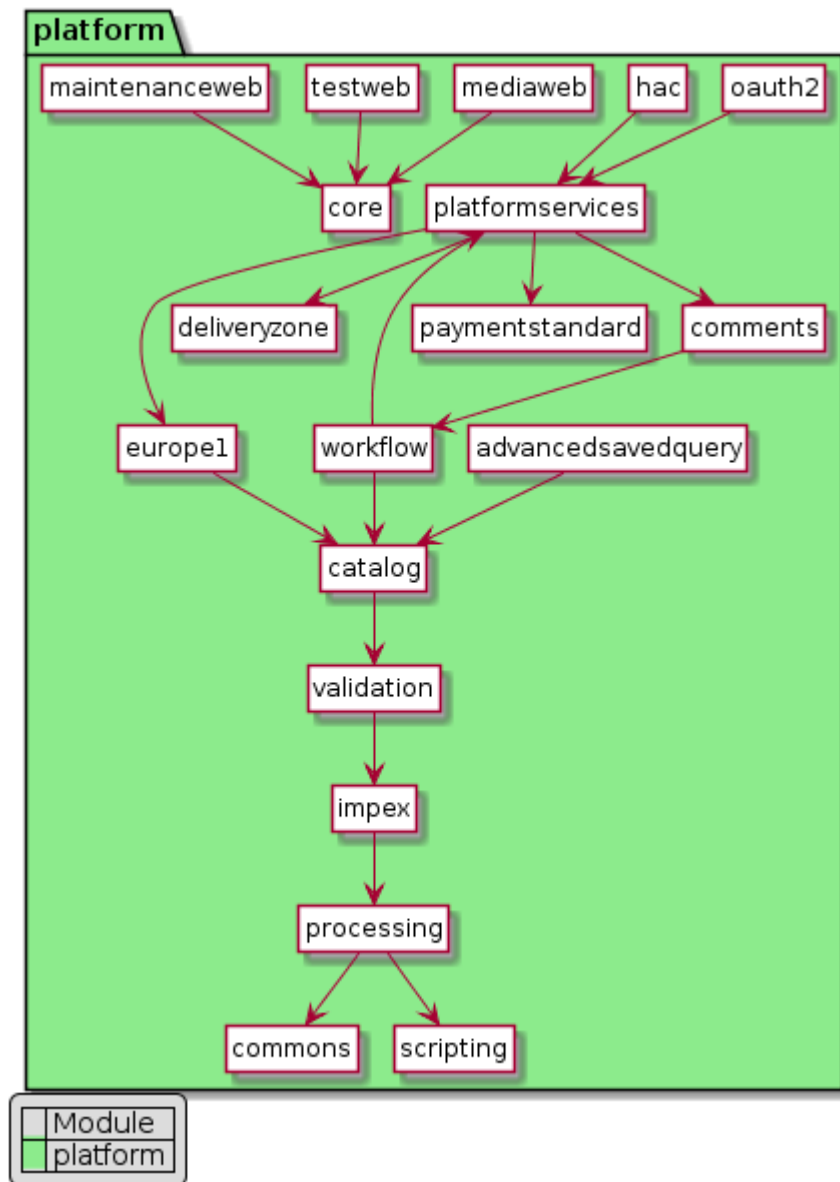
Dependencies

The Platform module is not dependent on any other module, and may be installed as a stand-alone component.

The Platform module consists of core and optional extensions.

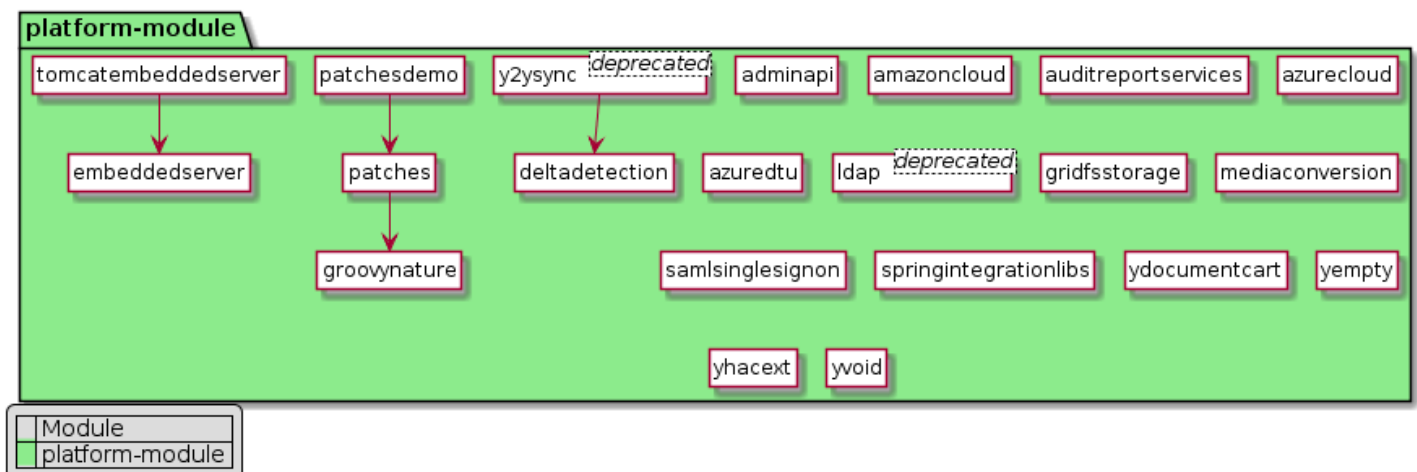
Core Extensions

The core extensions are autoloaded and they are located in `<HYBRIS_BIN_DIR>/platform/ext`. Each of the core extensions depends on the core extension. See the diagram for the internal dependencies between the core extensions. The dependency on the core extension isn't shown for clarity.



Optional Extensions

Each optional extension depends on the core extensions. The optional extensions are located in `<HYBRIS_BIN_DIR>/modules/platform`. The diagram shows internal dependencies between the optional extensions. The dependency on the core extensions is not shown for clarity.



For more information, see [SAP Commerce Directory Structure](#).

Recipes

For a complete list of SAP Commerce recipes that may include this module, see [Installer Recipes](#).

For a complete list of the SAP Commerce Cloud, integration extension pack recipes that may include this module, see [Installer Recipe Reference](#).

Extensions

Platform consists of the following extensions:

[advancedsavedquery Extension](#)

The `advancedsavedquery` extension extends the Hybris Management Console's (HMC) SavedQuery functionality.

[catalog Extension](#)

The `catalog` extension provides catalog, synchronization, classification, and category functionalities.

[comments Extension](#)

The `comments` extension provides the comments functionality within the workflow & collaboration features of Platform.

[commons Extension](#)

The `commons` extension provides core helper methods for templates. These methods are used for media neutral handling of content to be exported.

[core Extension](#)

The `core` extension is one of the most basic, technical foundation pieces of SAP Commerce. Without the `core` extension, SAP Commerce is inoperable.

[deliveryzone Extension](#)

The `deliveryzone` extension organizes countries and regions for relating shipping costs to them.

[europa1 Extension](#)

The `europa1` extension is the pricing system of SAP Commerce and is SAP Commerce's default PriceFactory. As well as storing and retrieving price information, it matches those prices, taxes and discounts that apply specifically to a given product or customer, thus arriving at the correct final price to display.

[hac Extension](#)

The `hac` extension is the default administration web application of SAP Commerce. It provides the SAP Commerce Administration Console, which offers functionality for administration, monitoring, and configuration of SAP Commerce.

[impex Extension](#)

The `ImpEx` extension allows you to create, update, remove, and export platform items such as customer, product, or order data to and from Comma-Separated Values (CSV) data files both during run time and during SAP Commerce initialization or update process.

[maintenanceweb Extension](#)

The `maintenanceweb` extension provides functionality for a web maintenance page to be displayed in case of unavailability of a web front end.

[mediaweb Extension](#)

The `mediaweb` extension has two purposes: to define where media files are located and to redirect HTTP and HTTPS requests to them.

[oauth2 Extension](#)

The `oauth2` core extension has replaced the `webservicescommons/oauthauthorizationserver` extension. It exposes the HTTP endpoint as an authorization server. It doesn't introduce any new significant functionalities.

[paymentstandard Extension](#)

The `paymentstandard` extension manages payment methods for transactions. The **PaymentStandard** extension allows you to set up payment modes for SAP Commerce.

[platformservices Extension](#)

The `platformservices` extension forms a part of the SAP Commerce ServiceLayer and comprises the functional and business services.

[processing Extension](#)

The `processing` Extension provides three areas of functionality: The CronJob Service, The Task Service, and the Process Engine.

[scripting Extension](#)

SAP Commerce scripting engine support allows you to execute logic written in scripting languages in run time without restarting the SAP Commerce Server AddOn enables the integration between. The scripting engine thus saves time and makes it possible to improve existing logic like cron jobs, the task engine, and ols.

[testweb Extension](#)

The `testweb` extension provides the Testweb user interface that allows you to run tests and test suites.

[validation Extension](#)

The `validation` extension enables you to easily and intuitively define data validation constraints, validate data before it is saved and notify the caller of any validation violations should they occur.

[workflow Extension](#)

The `workflow` extension provides a concept for modeling and processing of batch card-like processes.

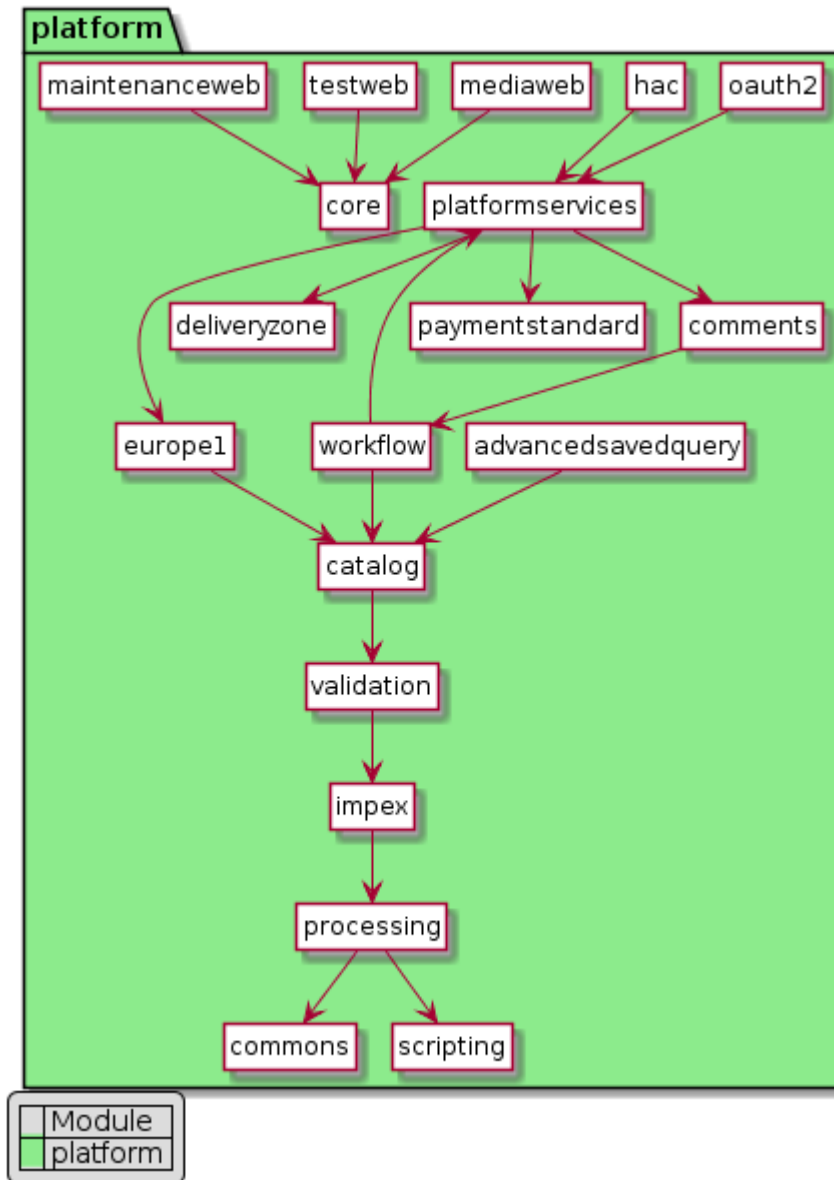
advancedsavedquery Extension

The `advancedsavedquery` extension extends the Hybris Management Console's (HMC) SavedQuery functionality.

About the Extension

Name	Directory	Related Module
advancedsavedquery	hybris/bin/platform/ext	Platform Architecture

Dependencies



Dependencies

catalog Extension

The catalog extension provides catalog, synchronization, classification, and category functionalities.

- The catalog functionality allows you to hold, structure, and manage products, and product information.
- The synchronization functionality allows you to synchronize catalog versions in a one-directional way for updating a catalog version according to another.
- The classification functionality allows you to define product attributes by using the classification methods.
- Variant functionality and the data model for product variants allow you to create products that differ in some aspect from one another, but are based on the same basic model. An example of variants is color for t-shirts. The base product is a t-shirt, the variants are a red t-shirt or a blue t-shirt.
- The category functionality allows you to build a hierarchical product structure, where products can be kept in categories.

For more information, see [Product Content and Catalogs](#).

i Note

The `comments` extension provides the comments functionality within the workflow & collaboration features of Platform.

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

Workflow and Collaboration

[illegible]

Configuring in project.properties File

In the `project.properties` file, you can:

- Configure the domain, component, and comment type.
- Define the maximum level of replies displayed in the comment section of the editor.
- Enable the comments--if the comment section and the comments in the inbox are shown.

These configurations can be overridden in each cockpit.

Because you can use comments in different scopes, each comment has a domain, component, and comment type in order to differentiate between the comments:

- Domain: Is the bigger scope of the comment, this could be for example cockpit or customer review.
- Comment type: Is the type of the comment the doesn't need to be a comment, but for example, a message or an action. Each comment type belongs to a domain and a component.
- Component: Each domain of a comment can have multiple components, for example, the domain cockpit could have the components `printcockpit`, `productcockpit`, and `cmscockpit`.

Beans

In the `cockpit-web-spring.xml` file, the most important classes used for working with comments in the cockpit are defined as beans.

The following beans are used for the comment section in the editor area:

- `CommentsSection`
- `CommentsSectionRenderer`
- `CommentActionColumnsEditorArea`

The following beans are used for the communication center in the browser area:

- `CommunicationBrowserModel`
- `CommentActionColumnsCommunicationBrowser`

In the `CommentsSectionRenderer` bean, you can configure the widths of the columns displayed in the comment section of the editor area.

All comment actions shown in the editor area and the browser area are configured via the `cockpit-web-spring.xml` Spring file.

The following actions available for comments:

- `AnswerCommentAction`
- `DeleteCommentAction`
- `EditCommentAction`
- `AddAttachmentCommentAction`.

Services

For handling comments there are two services available: `commentService` and `cockpitCommentService`.

Use the `cockpitCommentService` if develop cockpit functionality. The `cockpitCommentService` provides methods for creating a comment or reply, getting comment, replies, attachments and setting a comment to read. This service has methods to get search queries for comments on an item and comments a user created which are used as a filter in the comment browser.

The `commentService` provides various methods for getting comments, replies, comment type, domain, component and user settings.

Related Information

[Comments Integration](#)

[Cockpit Framework Overview](#)

[Workflow and Collaboration](#)

[Workflow Integration](#)

commons Extension

The **commons** extension provides core helper methods for templates. These methods are used for media neutral handling of content to be exported.

Included are several converters and the following frameworks:

- **Formatter** framework for converting item content to PDF, for example for converting an order to PDF.
- **Translator** framework mainly used for converting of HTML documents to another format, for example to Adobe InDesign.
- **Renderer** framework used for e-mail templates

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

Items

The following types are defined in the **commons** extension:

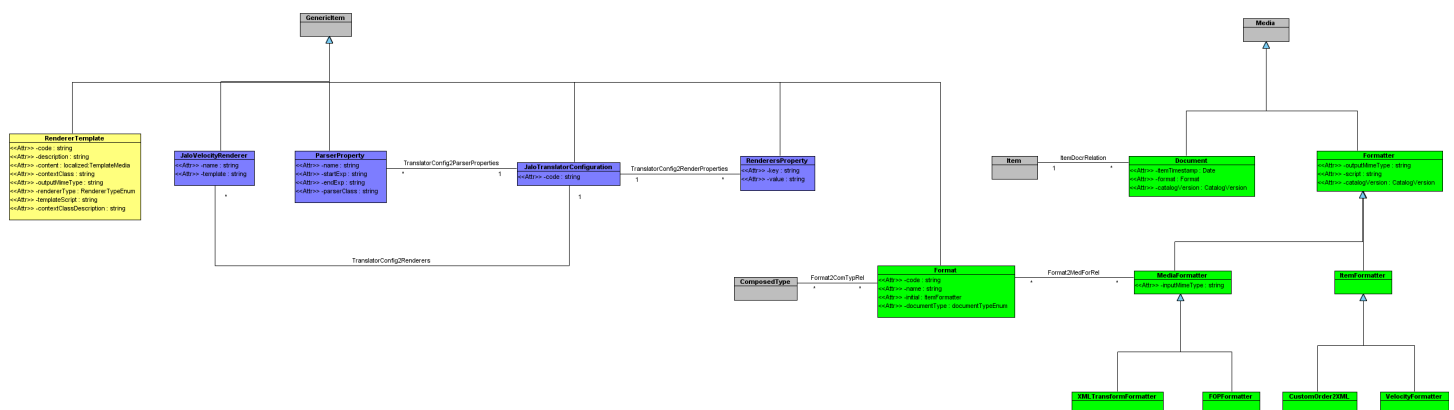


Figure: Commons items UML diagram presents the items and relations without metadata.

Type	Framework	Description
Document	Formatter	Contains the created result document, for example a PDF document.
Format	Formatter	Main class which covers the creating of the document.
Formatter	Formatter	Base item for every items of this framework.
ItemFormatter	Formatter	Base item for first step: item → media.
CustomOrder2XML	Formatter	Example of creating XML from the order.
VelocityFormatter	Formatter	Item for converting of item into the XML using Velocity script.
MediaFormatter	Formatter	Base item for second step: media → document (media).
FOPFormatter	Formatter	Item for converting of PDF document using FOP framework.
XMLTransformFormatter	Formatter	Item for converting of XML to the media using XSL script.
JaloTranslatorConfiguration	Translator	Contains settings that are used during converting from HTML into another format.
JaloVelocityRenderer	Translator	Velocity renderer.
ParserProperty	Translator	Contains the properties (HTML tags/elements) for parser.
RendererProperty	Translator	Contains the properties (available by key in the Velocity scripts) for renderer.
RendererTemplate	Renderer	Template for rendering of documents like e-mails.

Formatter Framework

The **Formatter** framework creates PDF documents by using Apache FOP.

You can use this framework as a starting point for extending, in order to create documents of another format supported by Apache FOP. The **Formatter** framework creates PDF documents by using Apache FOP.

You can use this framework as a starting point for extending, in order to create documents of another format supported by Apache FOP.

There is a ComposedType **Format** that uses an **ItemFormatter** and one or more **MediaFormatters**.

The converting is done step-by-step:

1. Item → Media: The **ItemFormatter** is a class that transforms a given item (whose content should be displayed in the result document) to an output that is used as input for **MediaFormatters**.
2. Media → result Media: The **MediaFormatters** now use this created input and transform it one by one to the result document.

There is an example in the sample data that creates PDF documents from orders, that works as follows:

1. The **Format** named **order2pdf_format** holds an **ItemFormatter** of type **CustomOrder2XML** named **customorder2xml**. This **ItemFormatter** transforms the given Order object to an XML structure.
2. This XML structure is used as input for a **MediaFormatter** of type **FOPFormatter** named **fopformatter** which creates the resulting PDF file. The **FOPFormatter** uses FOP and a stored XSL template (bin\platform\ext\commons\resources\commons\unittest\quotation.xsl) for creating the PDF document.

The XSL template file can be created by using [XSLfast](#) . XSLfast seems to be one of the best solutions for creating XSLT files because it is easy understandable for non-technical users and creates the XSL templates.

Java Classes

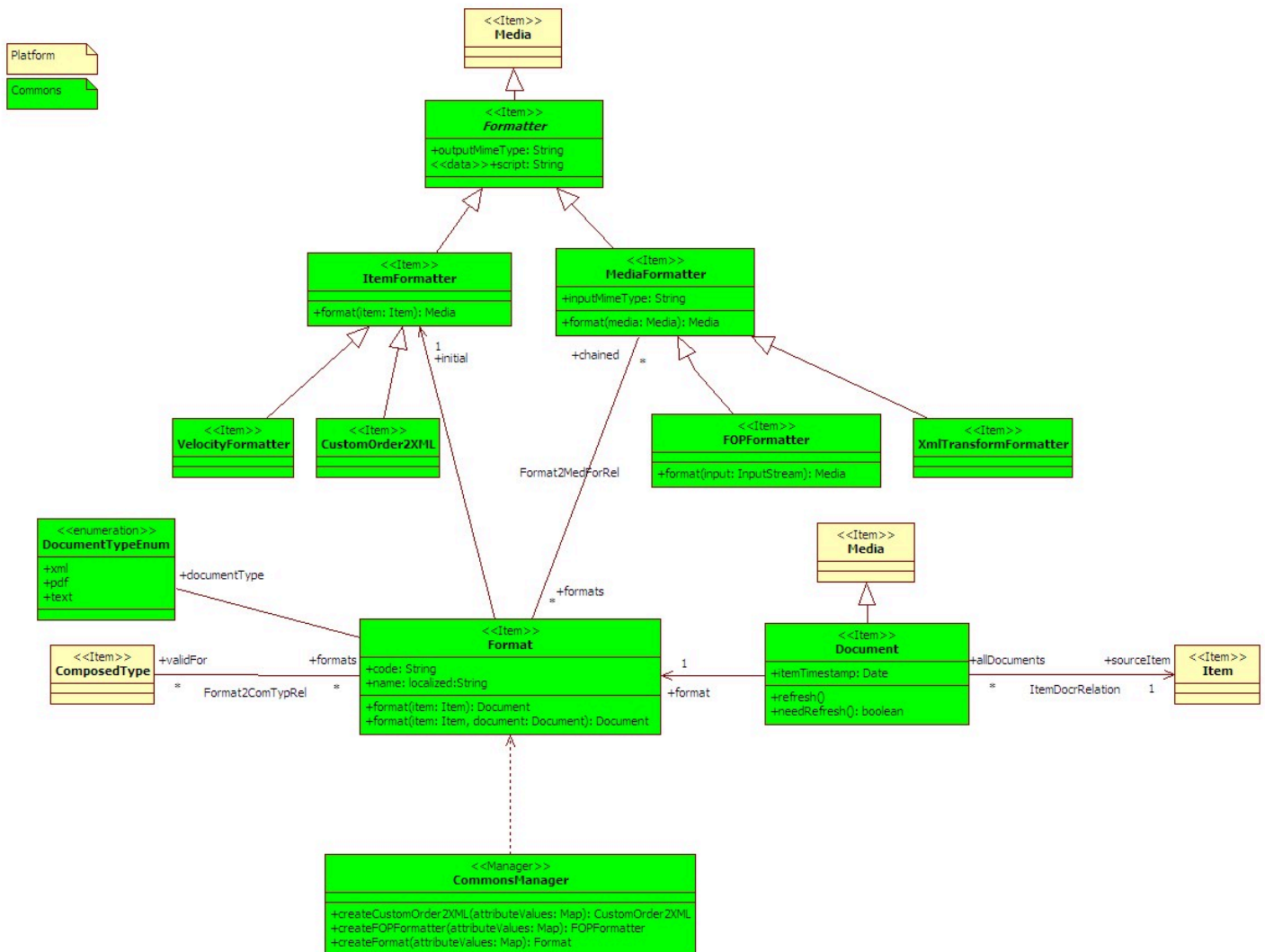


Figure: Java classes of the **Formatter** framework.

Usage

The following code presents the usage of the **Formatter** framework:

CommonsTest.java

```

@Test
public void testCustomOrder2PDF() throws ConsistencyCheckException, JaloBusinessException, IOException {

```



```

    final CustomOrder2XML co2x = CommonsManager.getInstance().createCustomOrder2XML(
        Collections.singletonMap(CustomOrder2XML.CODE, "test_smallchain_1"));
    assertNotNull(co2x);

    final FOPFormatter fopf = CommonsManager.getInstance().createFOPFormatter(
        Collections.singletonMap(FOPFormatter.CODE, "test_smallchain_2"));
    assertNotNull(fopf);

    fopf.setInputMimeType("text/xml");
    fopf.setOutputMimeType("application/pdf");
    fopf.setData(new DataInputStream(CommonsTest.class.getResourceAsStream("/commons/unittest/c
        "quotation.xml", "text/xml"));

    final Format format = CommonsManager.getInstance().createFormat(Collections.singletonMap(F
    assertNotNull(format);

    format.setInitial(co2x);
    format.setChained(Collections.singletonList((MediaFormatter) fopf));

    final Document ret = format.format(testorder);

    final String mimetypetest = MediaUtil.guess(ret.getDataFromStream());
    assertEquals(mimetypetest, "application/pdf");

    final InputStreamReader streamReader = new InputStreamReader(ret.getDataFromStream());
    final BufferedReader bufferedReader = new BufferedReader(streamReader);
    String line = null;

    line = bufferedReader.readLine();
    assertEquals(line, "%PDF-1.4");
    line = bufferedReader.readLine();
    line = bufferedReader.readLine();
    assertEquals(line, "4 0 obj");
    bufferedReader.close();
}

```

Translator Framework

The translator can be divided into the following main parts:

- **Corrector** corrects the incoming HTML, for example for replacing of the unclosed tags.
- **Parser** converts the HTML document into the Java Objects (Nodes)
- **Renderer** converts the parsed Java Objects (Nodes) into the output format using the Velocity scripts. Output format is the InDesing format only.

The embedded implementation is the **HTMLCorrector**, precisely the **HTMLCorrectorReplaceStrategy** which replaces the HTML tags upon the defined in `resources\commons\corrector\brreplacestrategy.properties` conditions.

Java Classes

The following UML diagrams show the Java classes of the **Translator** framework:

Items	
Corrector	
Parser & Renderer	

Renderer Framework

The **Renderer** framework is used for creating of the e-mail template by using placeholders. A sample use case is creating of an e-mail after placing an order.

It provides a new type **RendererTemplate**, which can be managed with Backoffice. To do so, choose **System > Output Documents > Communication Template** in the explorer tree. An instance of this type holds a localized template script where placeholders can be used. Furthermore the type of template is defined, which currently is **Apache Velocity** only.

For more information, see <http://velocity.apache.org/> : Apache Velocity Project.

The needed template instance can be configured via Backoffice and identified via the code attribute. The context object with the values for the placeholder has to be defined and you can simply call the `render` method of the **RendererService** to get the rendered text. The interface with full package-name (not the **Impl-class**) has to be specified at the template instance using the **contextClass** attribute.

Java Classes

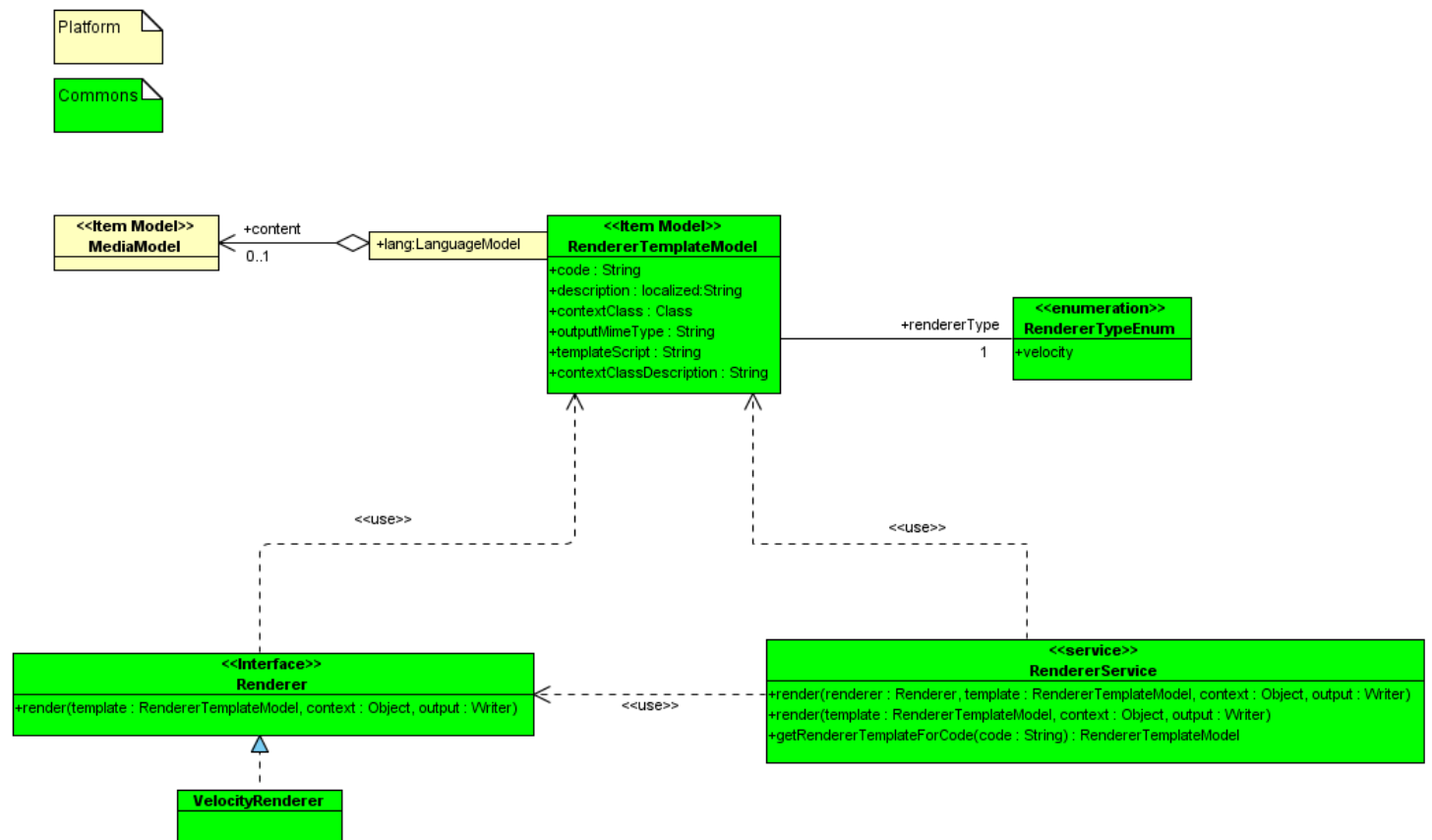


Figure: Example Java classes of the Renderer framework from an earlier SAP Commerce version.

The following table gives you an overview on the involved components of SAP Commerce templating management:

Component	Type	Description
RendererTemplateModel	Model	<ul style="list-style-type: none"> Holds the text template, which is a plain text with placeholders. Defines the context class. Defines the renderer type.
Renderer	Interface	<ul style="list-style-type: none"> The renderer type that, similar an engine, renders the output and replaces placeholders (interface <code>de.hybris.platform.commons.renderer.Renderer</code>) Currently we deliver only one implementation type, the <code>de.hybris.platform.commons.renderer.impl.VelocityTemplateRenderer</code>
render method from RendererService	Method	<ul style="list-style-type: none"> Renders template using default renderer or renderer passed as a parameter
Context	Interface + implementation	<ul style="list-style-type: none"> Used for RendererService as interface for defining the placeholders. (Each getter is a placeholder.) Example: <code>getName()</code> is available as placeholder <code>\$ctx.name</code>. Used for Renderer as implementation for providing the placeholder content.

Usage

The following code presents the usage of the **Renderer** framework:

```
//instance of de.hybris.platform.workflow.mail.WorkflowMailContext,
//defined as bean with id 'new.workflow.mail.context'
WorkflowMailContext ctx;

...

//gather needed template instance with code "testTemplate" ..
RendererTemplate template = rendererService.getRendererTemplateForCode("testTemplate");

ctx.setToEmailAddress( "test2@test.de");
ctx.setAssigneeName( "testUser" );

//Create mail instance using commons-mail and the hybris MailUtility class
//with set mail properties read from project.properties
HtmlEmail htmlEmail = (HtmlEmail) MailUtils.getPreConfiguredEmail();
htmlEmail.addTo("test@test.de");
htmlEmail.setSubject("test");

//Create a writer where the rendered text will be written to
StringWriter mailMessage = new StringWriter();

//Render the template using the context object
rendererService.render( template, ctx, mailMessage );

//set rendered text to mail and send it
htmlEmail.setHtmlMsg( mailMessage.toString() );
```

core Extension

The **core** extension is one of the most basic, technical foundation pieces of SAP Commerce. Without the **core** extension, SAP Commerce is inoperable.

The **core** extension handles basic functionality such as persistence handling, the ordering process and contains the service layer functionality.

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

deliveryzone Extension

The **deliveryzone** extension organizes countries and regions for relating shipping costs to them.

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

A delivery zone in SAP Commerce is related to countries. A delivery zone is independent of the regions of a country, and it is possible to add several countries to one delivery zone.

Name of Type	Type Localization in the Backoffice	Sample Data (Excerpt)	Description
ZoneDeliveryMode	Zone Delivery Mode	DHL, Fedex, UPS, Deutsche Post	The representation of a logistics company, allowing to specify delivery fees and payment modes. Holds ZoneDeliveryModeValues to specify delivery fees. Also specifies the payment modes allowed.
ZoneDeliveryModeValue	Delivery Costs	<ul style="list-style-type: none"> • DHL <ul style="list-style-type: none"> ◦ 6 EUR from 0 EUR order total for delivery in de ◦ 4 EUR from 20 EUR order total for delivery in de • Fedex <ul style="list-style-type: none"> ◦ 5 EUR from 0 EUR order total for delivery in de ◦ 6 EUR from 0 EUR order total for delivery in europe 	<p>Represents a single potential delivery fee and also specifies the Delivery Zone to which the fee applies.</p> <p>Similar to the price row and tax row calculation of the Europe1 PriceFactory, the actual delivery fee is selected from all applicable delivery fees.</p>
Zone	Delivery Zone	de, europe, world	Allows specifying the countries for which this delivery zone applies.

ZoneDeliveryMode

A ZoneDeliveryMode

- holds any number of **ZoneDeliveryModeValues**,
- and also holds any number of **StandardPaymentModes**.

Attribute Name	Mandatory	Description
active	yes	Whether the ZoneDeliveryMode is enabled (true) or disabled (false). Disabled ZoneDeliveryMode s will not be available for use (in the web shop, for example).
code	yes	The identifier of the ZoneDeliveryMode .
description	no	A localized description of the ZoneDeliveryMode .
name	no	A localized identifier.

Attribute Name	Mandatory	Description
net	yes	Whether the ZoneDeliveryModeValues referenced by the ZoneDeliveryMode are given as a gross (false) or as a net (true) value.
propertyName	yes	If set to the default of delivery.zone.price , the ZoneDeliveryMode uses the default price determination implementation.
supportedpaymentmodes	no	A Collection of the StandardPaymentModes allowed for this ZoneDeliveryMode .
values	no	A Collection of ZoneDeliveryModeValues allowed for this ZoneDeliveryMode .

ZoneDeliveryModeValue

A price setting for a **ZoneDeliveryMode**.

Attribute Name	Mandatory	Description
currency	yes	The currency in which the ZoneDeliveryModeValue is specified. A ZoneDeliveryModeValue is always an absolute value (such as 1.50 EUR), relative delivery costs are not supported.
deliverymode	yes	Specifies the ZoneDeliveryMode to which this ZoneDeliveryModeValue is assigned. This attribute cannot be changed after creating the ZoneDeliveryModeValue (initial attribute).
minimum	yes	The minimum order total for which this ZoneDeliveryModeValue applies. The ZoneDeliveryModeValue will not be eligible for an order total of less than this value.
value	yes	The value of the delivery cost.
zone	yes	Holds a Zone to specify the area for which the ZoneDeliveryModeValue applies.

Zone

A collection of countries to which a given **PaymentMode** delivers to. This determines the delivery modes available to a customer by the country the customer.

Attribute Name	Mandatory	Description
code	yes	The identifier of the Zone.

Attribute Name	Mandatory	Description
countries	no	A Collection of the countries assigned to the Zone.

Related Information

[paymentstandard Extension](#)

europe1 Extension

The **europe1** extension is the pricing system of SAP Commerce and is SAP Commerce's default PriceFactory. As well as storing and retrieving price information, it matches those prices, taxes and discounts that apply specifically to a given product or customer, thus arriving at the correct final price to display.

Throughout your business life cycle you will use various pricing strategies to encourage your customers to buy products. The **europe1** extension enables you to implement your own unique pricing strategy and facilitate processes related to managing orders by realizing the following functions:

- It stores, retrieves and matches prices, discounts and taxes to products and customers.
- It offers solutions for managing prices, taxes or product quantities more flexibly:
 - Per line item - the base price is provided as well as taxes and discounts (even multiple taxes / discounts combined).
 - Per order - global discounts are possible (even multiple ones) applying to the order subtotal.
 - Multiple units - Products can be sold in different (packaging) units, for example selling a pack of 10 printer cartridges as well as selling 10 cartridges individually.
 - Scale prices - You may wish to offer reduced prices when the number of units ordered exceeds a predefined threshold. For example, if a customer buys up to 9 bottles of wine, the price is €5 per bottle but above that number the price drops to €4.
 - Date ranges - You may want to offer special prices or discounts for specific time periods, for example, lower prices set for Mondays might encourage customers to buy more on this day.
 - Grouping per customer and / or product - You may offer your customers lower prices as a result of business negotiations. When a customer is logged in, the system will automatically display the negotiated prices. You are able to set up price categories and put in them products that match these categories instead of assigning a price to each product individually.
 - Net / gross prices - It is possible to store both gross and net prices individually, so that they appear more attractive, for example €2,99. If only net **or** gross is available, prices are converted on-the-fly if the order requires so.

i Note

The europe1 PriceFactory makes up only a part of the ordering process. The ordering process is part of SAP Commerce's core and is not integrated within the **europe1 PriceFactory**. Within the Ordering Process (see [Ordering Process](#), the europe1 PriceFactory determines prices, taxes and discounts of a given order.

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

Related Information

[Decoupling PDTRows from Product](#)

hac Extension

The hac extension is the default administration web application of SAP Commerce. It provides the SAP Commerce Administration Console, which offers functionality for administration, monitoring, and configuration of SAP Commerce.

Among other functions, it enables you to check and monitor general settings and initialize or update SAP Commerce. In addition, it offers links to documentation resources.

You can access it by navigating to `http://localhost:9001`. By default, the Platform port is for HTTP is 9001 and for HTTPS is 9002. So if you do not change the default configuration, you can access the SAP Commerce Administration Console by navigating to the Platform URL and port 9001 or 9002, respectively.

For example by navigating to `http://localhost:9001` or `https://myserver.mycompany.com:9002`.

You can run the SAP Commerce Server in the Administration Console-only mode. To do this, run `adminserver.bat` instead of `hybrisserver.bat`. As you can see, web contexts are loaded for Administration Console only. You can therefore use this node strictly for administration purposes.

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

Related Information

[Administration Console](#)

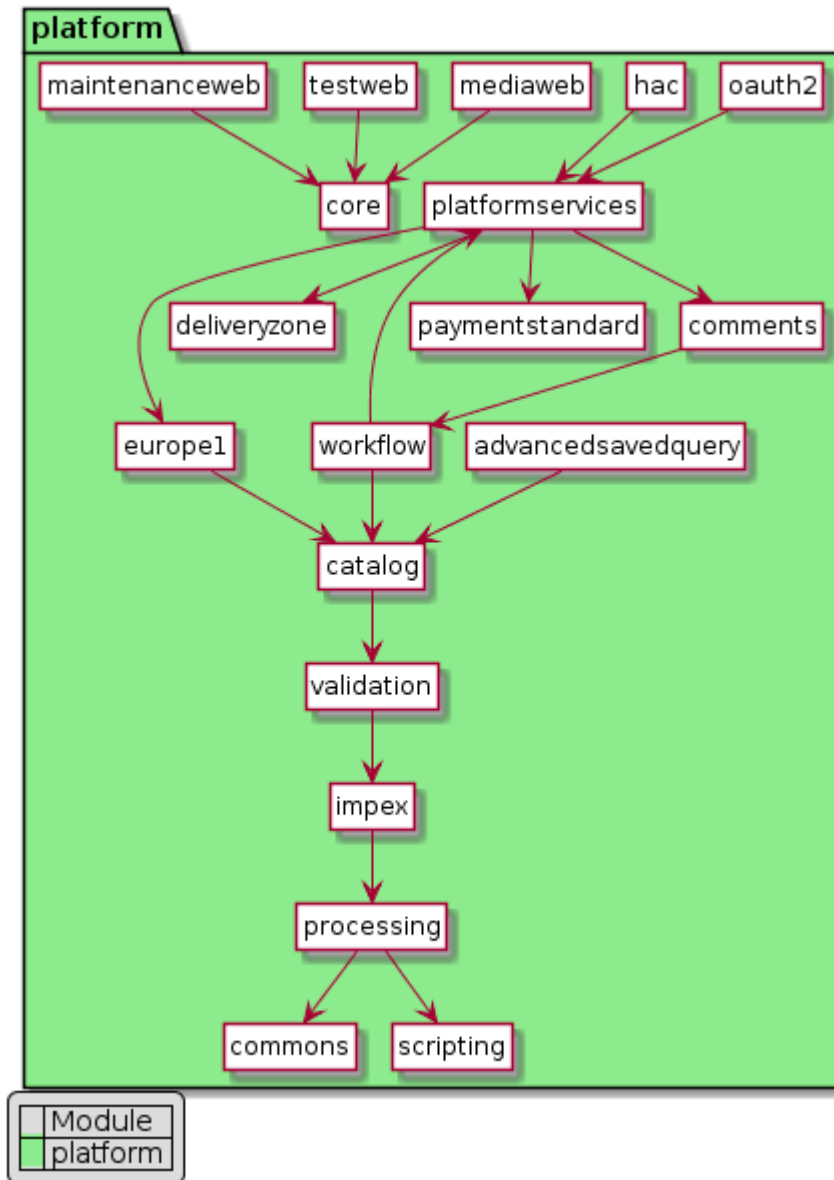
impex Extension

The ImpEx extension allows you to create, update, remove, and export platform items such as customer, product, or order data to and from Comma-Separated Values (CSV) data files both during run time and during SAP Commerce initialization or update process.

About the Extension

Name	Directory	Related Module
impex	hybris/bin/platform/ext	Platform Architecture

Dependencies



Dependencies

Related Information

[ImpEx Syntax](#)

maintenanceweb Extension

The `maintenanceweb` extension provides functionality for a web maintenance page to be displayed in case of unavailability of a web front end.

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

mediaweb Extension

The `mediaweb` extension has two purposes: to define where media files are located and to redirect HTTP and HTTPS requests to them.

- Redirect Servlet:

The `mediaweb` extension contains a Java Servlet that evaluates the path to where the Media files are located and makes sure that HTTP and HTTPS requests are modified to match that path.

- Media File Storage Location:

The `mediaweb` extension is the factory default location where files of Media items are stored, such as images, CSV files, or video clips. For Cluster systems, you need to specify a network share as storage location instead using the media-related properties, such as `media.replication.dirs`.

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

Related Information

[Configuring the Behavior of SAP Commerce](#)

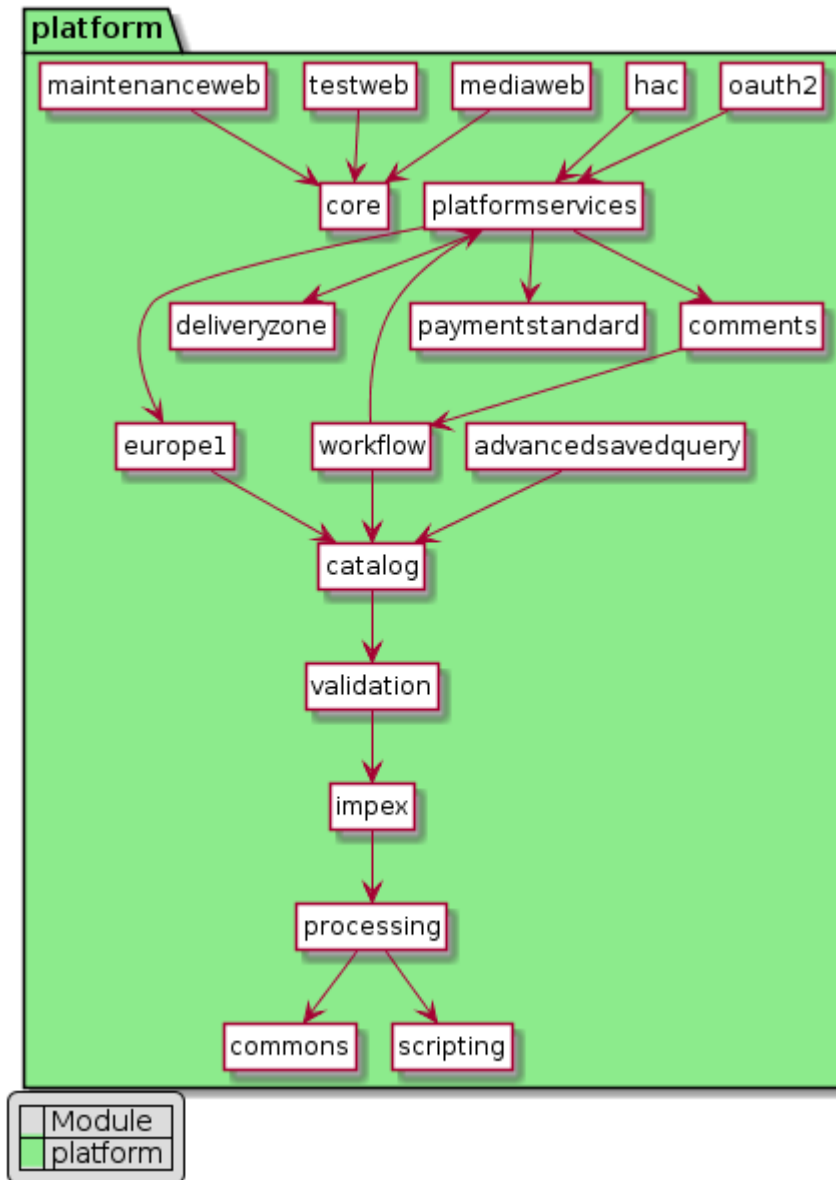
oauth2 Extension

The `oauth2` core extension has replaced the `webservicescommons/oauthauthorizationserver` extension. It exposes the HTTP endpoint as an authorization server. It doesn't introduce any new significant functionalities.

About the Extension

Name	Directory	Related Module
oauth2	hybris/bin/platform/ext	Platform Architecture

Dependencies



Dependencies

Related Information

[OAuth2](#)

paymentstandard Extension

The `paymentstandard` extension manages payment methods for transactions. The **PaymentStandard** extension allows you to set up payment modes for SAP Commerce.

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

By factory default, SAP Commerce comes with these common payment modes:

- credit card,
- invoice,

- advance payment,
- debit.

These payment modes will be available across SAP Commerce.

It is possible to set up transaction costs for the use of a payment mode. For example, some shops may require a transaction cost for credit card use.

The **PaymentStandard** extension does not support relative transaction costs, only absolute transaction costs.

Name of Type	Sample Data (Excerpt)	Description
StandardPaymentMode	creditcard, debitentry, invoice, advance	Representation of a means of payment. Holds StandardPaymentModeValues to specify the transaction cost of the payment mode. Also specifies the delivery modes allowed.
StandardPaymentModeValue	<ul style="list-style-type: none"> • invoice <ul style="list-style-type: none"> ◦ 1.5 EUR ◦ 1.4 USD ◦ 1.5 GBP • advance <ul style="list-style-type: none"> ◦ 0 EUR ◦ 0 USD ◦ 0 GBP 	The cost of using the Payment Mode, depending on the currency.

StandardPaymentMode

A StandardPaymentMode

- references a subtype of **PaymentInfo** to specify the type of payment mode,
- holds any number of **StandardPaymentModeValues**,
- and also holds any number of **ZoneDeliveryModes**.

Attribute Name	Mandatory	Description
active	yes	Whether the StandardPaymentMode is enabled (true) or disabled (false). Disabled StandardPaymentModes will not be available for use (in the web shop, for example).
code	yes	The identifier of the StandardPaymentMode .
description	no	A localized description of the StandardPaymentMode .

Attribute Name	Mandatory	Description
name	no	A localized identifier.
net	no	Whether the StandardPaymentModeValues referenced by the StandardPaymentMode are given as a gross (false) or as a net (true) value.
paymentinfotype	yes	References a subtype of PaymentInfo to specify the type of payment mode.
paymentmodevalues	no	A Collection of the StandardPaymentModeValues allowed for this StandardPaymentMode .
supporteddeliverymodes	yes	A Collection of the ZoneDeliveryModes allowed for this StandardPaymentMode .

StandardPaymentModeValue

A **StandardPaymentModeValue** is a transaction cost and has both a currency and a value.

It is not possible to specify a **StandardPaymentModeValue** to be gross or net. Whether a **StandardPaymentModeValue** is considered to be gross or net depends on the value of the **net** attribute of the **StandardPaymentMode** to which the **StandardPaymentModeValue** is assigned.

Attribute Name	Mandatory	Description
currency	yes	The currency in which the StandardPaymentModeValue is specified. A StandardPaymentModeValue is always an absolute value (such as 1.50 EUR), relative transaction costs are not supported.
paymentMode	yes	Specifies the StandardPaymentMode to which this StandardPaymentModeValue is assigned. This attribute cannot be changed after creating the StandardPaymentModeValue (initial attribute).
value	yes	The value of the transaction cost.

PaymentInfo

PaymentInfo has four subtypes by factory default:

- **AdvancePaymentInfo**
- **CreditCardPaymentInfo**
- **DebitPaymentInfo**
- **InvoicePaymentInfo**

Instances of **PaymentInfo** and its subtypes are individual payment settings. Every **PaymentInfo** instance is a different payment option, such as an individual credit card representation or a bank account representation. If user **demo** has entered data for two bank accounts and one credit card, there will be three different **PaymentInfo** instances for user **demo**.

On every order where a credit card or debit payment mode is selected, SAP Commerce creates a copy of the **PaymentInfo** instance used. This makes sure that the order always is stored with the payment data provided at its creation time.

SAP Commerce does not store individual payment information when an order has been set to use advance or invoice payment. **AdvancePaymentInfo** and **InvoicePaymentInfo** do not require any actual banking or credit card information for safekeeping with the order, so no documentation of the payment data will be necessary.

AdvancePaymentInfo

This type is listed here for completeness; no actual instances of this type will be created automatically.

Attribute Name	Mandatory	Description
code	yes	The identifier of the advance payment data.
user	yes	The User account to whom the advance payment is assigned. This attribute cannot be changed after creating the AdvancePaymentInfo (initial attribute).

CreditCardPaymentInfo

Attribute Name	Mandatory	Description
code	yes	The identifier of the credit card data.
ccOwner	yes	The owner of the credit card. This is a free text field and, unlike the user attribute, does not refer to an actual instance of the User type in SAP Commerce.
number	yes	The number of the credit card.
pin	no	The PIN of the credit card.
type	yes	The type of credit card.
user	yes	The User account to whom the credit card is assigned. This attribute cannot be changed after creating the CreditCardPaymentInfo (initial attribute).
validFromMonth	no	The month of the beginning of the credit card's validity time span.
validFromYear	no	The year of the beginning of the credit card's validity time span.
validToMonth	yes	The month of the end of the credit card's validity time span.
validToYear	yes	The year of the end of the credit card's validity time span.

DebitPaymentInfo

Attribute Name	Mandatory	Description
accountNumber	yes	The number of the bank account from which to charge for the order.
bank	yes	The name of the bank.
bankIDNumber	yes	The bank code number of the bank.
baOwner	yes	The name of the account holder.
code	yes	The identifier of the debit payment data.
user	yes	This attribute cannot be changed after creating the DebitPaymentInfo (initial attribute).

InvoicePaymentInfo

This type is listed here for completeness; no actual instances of this type will be created automatically.

Attribute Name	Mandatory	Description
code	yes	The identifier of the invoice payment data.
user	yes	The User account to whom the invoice payment is assigned. This attribute cannot be changed after creating the InvoicePaymentInfo (initial attribute).

platformservices Extension

The **platformservices** extension forms a part of the SAP Commerce ServiceLayer and comprises the functional and business services.

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

Related Information

[ServiceLayer](#)

processing Extension

The **processing** Extension provides three areas of functionality: The CronJob Service, The Task Service, and the Process Engine.

For detailed information about each of these three areas, see the following:

- [The Cronjob Service](#)
- [The Task Service](#)
- [The SAP Commerce processengine](#)

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

scripting Extension

SAP Commerce scripting engine support allows you to execute logic written in scripting languages in run time without restarting the SAP Commerce Server AddOn enables the integration between. The scripting engine thus saves time and makes it possible to improve existing logic like cron jobs, the task engine, and ols.

Scripts may be stored in various repositories like a database (based on the `Script` model), classpath, file system, or even in a remote repository (for example Gists at GitHub) or can be executed on the fly as a snippet without being stored.

You do not need to add the `scripting` extension to your `localextensions.xml` file. Scripting is autoloaded together with Platform.

Related Information

[Scripting Engine](#)

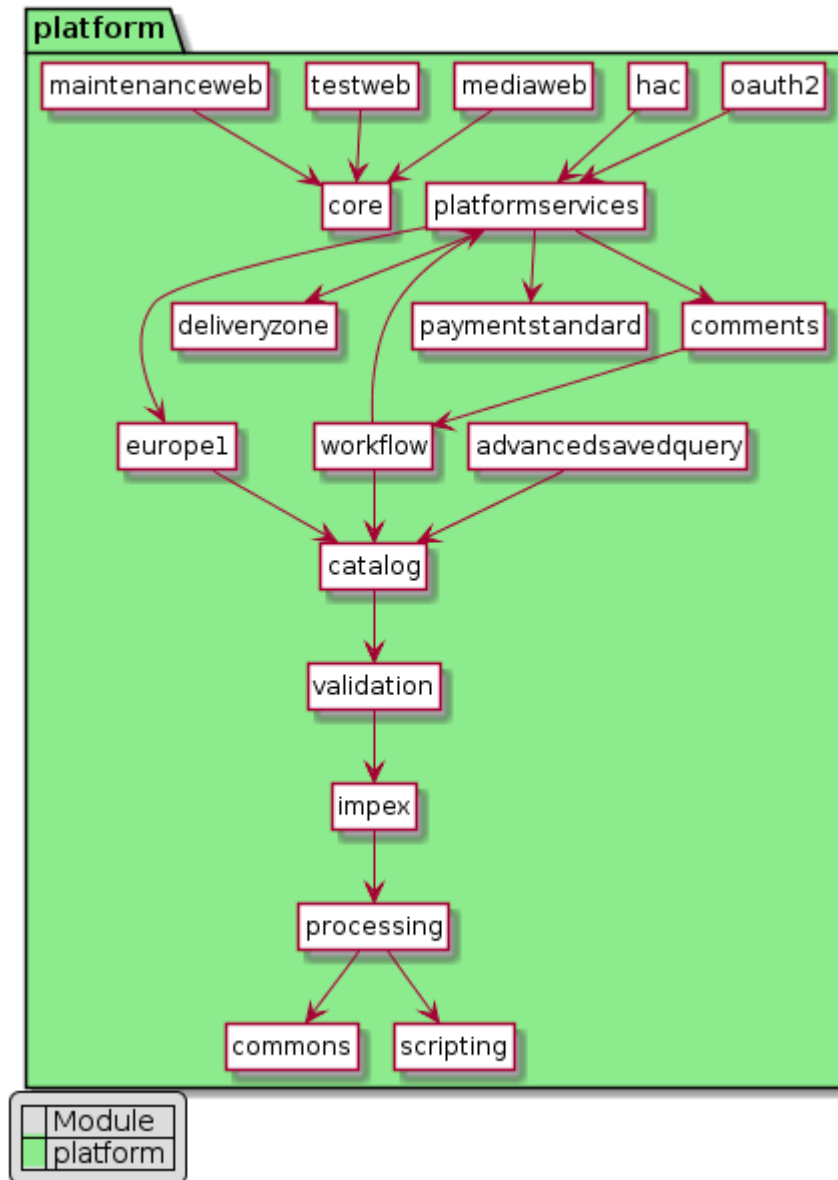
testweb Extension

The `testweb` extension provides the Testweb user interface that allows you to run tests and test suites.

About the Extension

Name	Directory	Related Module
testweb	hybris/bin/platform/ext	Platform Architecture

Dependencies



Dependencies

Related Information

[The SAP Commerce Testweb Front End](#)

validation Extension

The `validation` extension enables you to easily and intuitively define data validation constraints, validate data before it is saved and notify the caller of any validation violations should they occur.

See how it works on the example of data validation for multiple languages -[Validating Data for Multiple Languages](#).

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

Related Information

The workflow extension provides a concept for modeling and processing of batch card-like processes.

For example, think of processes when visiting a government agency or a district recruiting office. There, you get a batch card and you have to visit the offices marked on the card. Once you have visited them in the correct sequence and each office has marked and acknowledged the visit, the process is completed successfully.

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

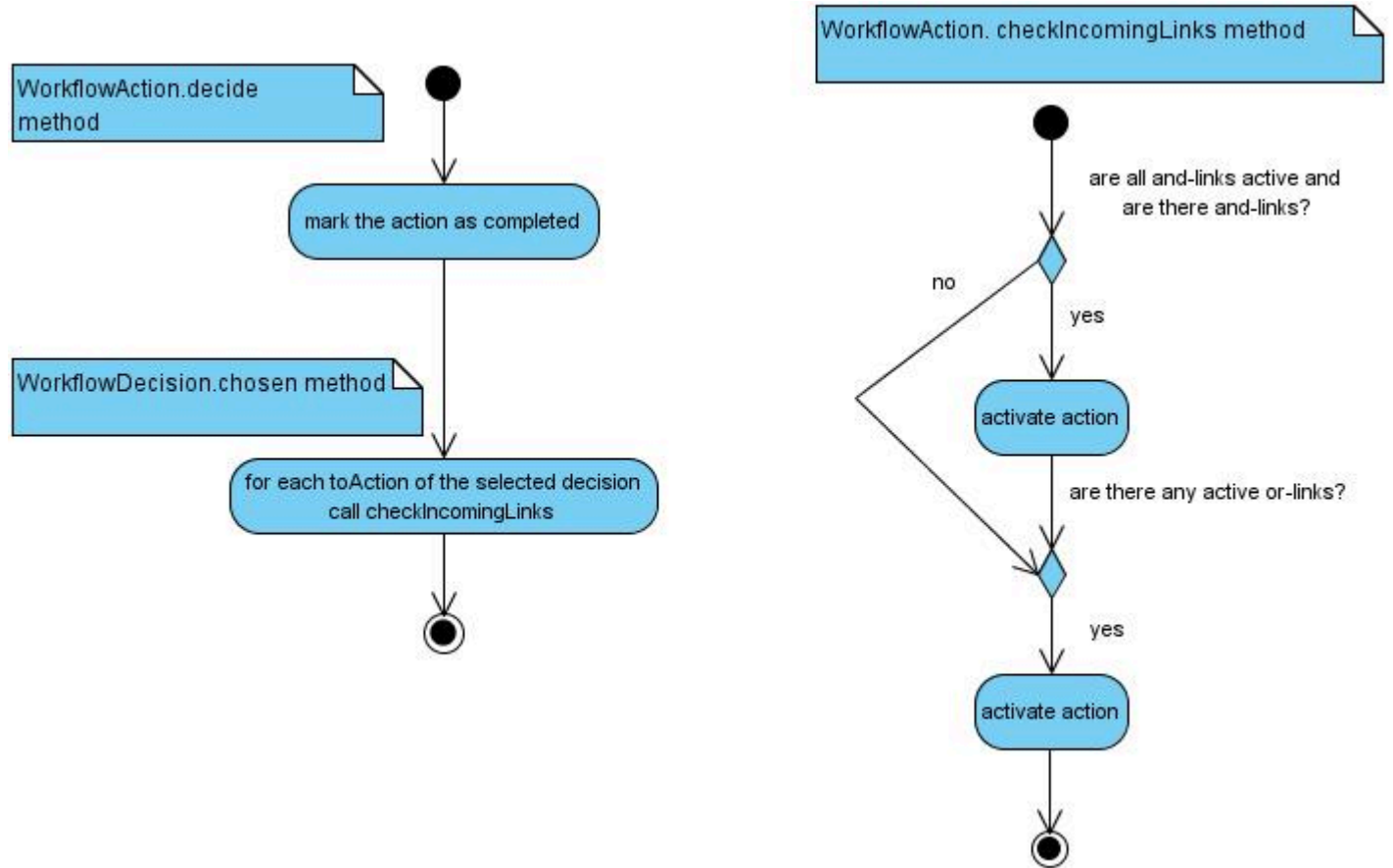
The workflow extension brings several new item types to the platform. These types and their relations to each other are described below.



This is custom documentation. For more information, please visit the [SAP Help Portal](#)

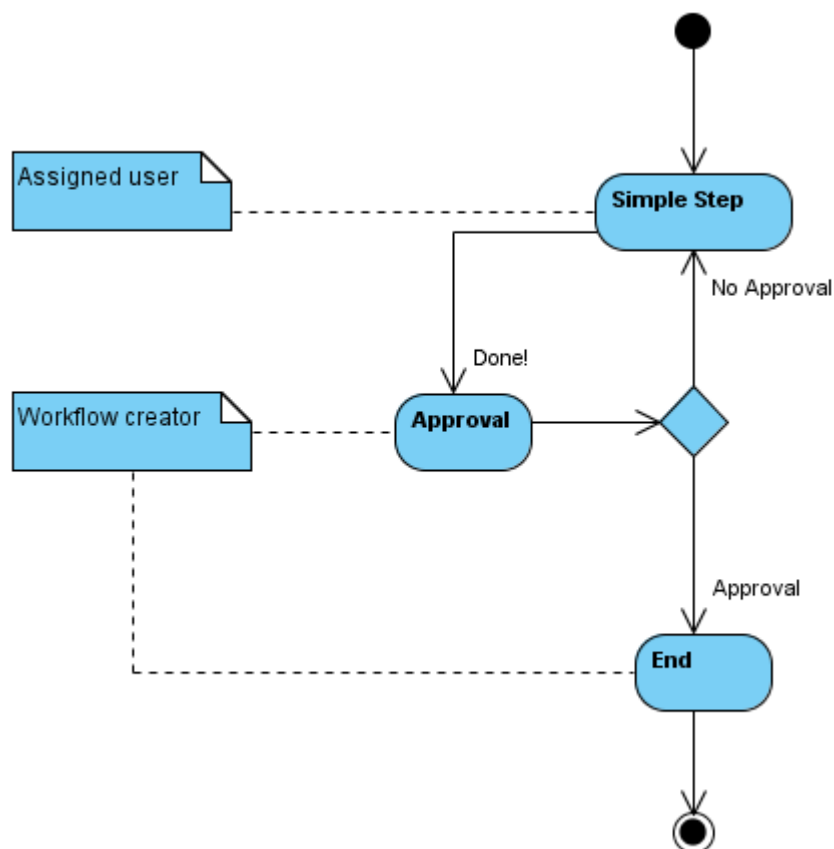
The algorithm for activating actions will be performed with each decision made. After the current action is set to completed, all outgoing actions `toActions` of the decision are checked and depending on the connection set active.

Besides the general attributes, the workflow template also holds a owner reference to a user which defines the responsibility for this template instance. It is also connected to a set of `WorkflowActionTemplates` by a 1:n relation. These workflow action templates are used when creating a workflow instance from the workflow template for creating the action instances. So if you define a workflow template, you already configure the set of workflow action templates and their ordering through the decision. When you create a workflow instance you get the set of related action instances automatically.



Ad-hoc Workflows

Ad-hoc workflows are created out of a special workflow template. The default name of this special template is `adhoctemplate`. If needed, it can be changed in the `project.properties` file of the workflow extension. The template consist of 3 simple linear tasks described in the diagram on the right.



Workflow Activation Script

A workflow can be configured to be automatically activated when certain criteria are met, for example when a new product is created using the Backoffice and the approval status is set to check. This is done by the use of workflow template activation script attribute. The Activation Script is defined for a Workflow Template and is a piece of Java code (a regular expression, for example) that must return a Boolean value. If the Boolean value is evaluated to `true`, then the workflow template is copied to a workflow automatically. Each Workflow Template can have an individual Activation Script.

An activation script is evaluated each time an item is:

- Created
- Saved
- Removed

If the activation script evaluates to `true` a new workflow is created based on the workflow template used.

There are a few variables available to the developer as listed:

Event	action(String)	item(Item)	itemType(ComposedType)	initialValues(Map)	currentValues(Map)
Item created	create	the item	Composed type of the created item	*	attribute values
Item removed	remove		Composed type of the removed item		attribute values
Item saved	save	the item	Composed type of the saved item	old attribute values	new attribute values

* = Available but `currentValues` should be used instead.

Note that the availability of a variable and what it holds depend on the type of event.

Example: The following activation script will create a new workflow when a product is created or saved and its approval status is check:

```
// this script will create a new workflow when a product is created or saved and its approval status is check
(
    // create new item
    (action.equals("create") &&

    // initialValues must be set
    initialValues != null &&

    // approvalStatus must be set
    initialValues.get("approvalStatus") != null &&

    // approvalStatus must be set to "check"
    initialValues.get("approvalStatus").getCode().equals("check"))
) ||

// save item
(action.equals("save") &&

// currentValues must be set
currentValues != null &&

// approvalStatus must be set
currentValues.get("approvalStatus") != null &&

// approvalStatus must be set to "check"
currentValues.get("approvalStatus").getCode().equals("check"))) &&

// created or saved item is a product
((de.hybris.platform.servicelayer.type.TypeService) Registry.getApplicationContext().getBean("typeService").getTypeService().isProductType(item.getTypeCode()))
```

WorkflowActionTemplate Type

A workflow action template is used for describing a specific task when modeling a workflow template.

When creating a workflow from a workflow template, each workflow action template associated to the workflow template is used for creating the respective workflow action. A workflow action template holds a set of general attributes which are used as initial values for the action created. To get a partial order between actions templates in scope of a workflow template, the workflow action template also sets up the basic sequence of actions via decision templates, which define possible options for activating the next action after this action was processed.

Each workflow action template references a principal, which can be a user or a user group. This principal will be used as the initial value for the `principalAssigned` attribute when creating an action using the template. The principal will be responsible for processing and choosing a decision to complete.

AutomatedWorkflowActionTemplate Type

An automated workflow action template is very similar to a conventional workflow action template. The automated action which is created from the automated workflow action template does not require any manual interaction by the user. To run, it needs a job class which implements `AutomatedWorkflowTemplateJob`.

An automated action is very similar to an action. The automated action, which is an action of the type `AutomatedWorkflowActionTemplate`, performs with no user interaction needed. To run, it needs a job class which implements `AutomatedWorkflowTemplateJob`. When the automated action gets activated it calls the `perform` method of the job class. Afterwards the decision returned by the `perform` method is selected.

The following example writes a text to the log and return the first decision of this action. In case that this action has no decisions, for example if its an end action, it returns null.

WorkflowAutomatedAction.java

```
package de.hybris.platform.workflow.jobs;

import de.hybris.platform.workflow.model.WorkflowActionModel;
import de.hybris.platform.workflow.model.WorkflowDecisionModel;

import org.apache.log4j.Logger;

public class WorkflowAutomatedAction implements AutomatedWorkflowTemplateJob
{
    private static final Logger LOG = Logger.getLogger(WorkflowAutomatedAction.class.getName());

    @Override
    public WorkflowDecisionModel perform(final WorkflowActionModel action)
    {
        LOG.info("This will complete the action automatically without any user interaction");

        for (final WorkflowDecisionModel decision : action.getDecisions())
        {
            return decision;
        }
        return null;
    }
}
```

If no decision is returned the action gets completed, no next action is activated. Remember, the template holds the general behavior of an action, so all calls have to be done at the given action object. Furthermore there is no need for overriding the **WorkflowAction** type.

After creating the job class, instead of a usual workflow action template add an automated workflow action template with your job class set to the workflow template. After creating a workflow using the workflow template, your action performs your method when getting activated.

WorkflowDecisionTemplate Type

Decision Templates are outbound links from an Workflow Action Template to another Workflow Action Template. A Decision Template defines which actions might follow a given Workflow Action Template. A Workflow Action Template can only be a follow-up to a Workflow Action Template if linked by a Decision Template. A Workflow Action Template cannot be followed by another Workflow Action Template if no Decision Template exists between the two Workflow Action Templates.

When creating a workflow from a workflow template, each decision template associated to the workflow action templates of the workflow template is used for creating the associated workflow decision. An decision template holds a set of general attributes which are used as initial values when creating an action from it.

Each decision template references the action it belongs to and a set of actions it links to. These actions get activated when the decision is chosen.

Workflow Type

Technically, Workflows are cron jobs. However, Workflows do not run fully automatically by itself. In order to process through a Workflow, a user will have to select Decisions on every Action. The Decision selected determines the upcoming Actions. To process through a Workflow automatically, you will need to use automated workflow actions.

A consequence of the fact that Workflows are cron jobs is that you need to specify a job for a Workflow. By default, the job specified for a Workflow is the Workflow Template from which the Workflow was copied.

A workflow is related to a set of actions where the actions themselves are in a certain sequence. When creating a workflow it will be performed automatically, where the default implementation is to check/switch/activate the state of all related actions as described in the *WorkflowTemplate Type* section above. If not all actions are completed, the workflow will set its status attribute to pause mode. By default, the workflow will only be triggered if a decision for an action is selected. If the status of all actions is completed, the status of the workflow will also be set to completed. The status attribute is inherited from the cron job type as well as the `sendEmail` and `emailAddress` attribute which will be used for sending an e-mail if the workflow has completed.

Formally a workflow is similar to a batch card holding a collection of actions ordered in some way. The workflow itself symbolizes an overall process where the actions are tasks. Such a process has a name as well as a description where the responsible owner can manage its data. It also has a list of attachments, here a collection of `WorkflowItemAttachments`. An example of a workflow would be adding a new product to your catalog where the product is attached to the workflow so that the assigned principals of the actions can access it directly for adding it to categories or for localization of attributes.

WorkflowAction Type

An action represents a task the assigned principal has to do. The action will only get active if a decision (OR-Connection) or rather all decisions (AND-Connection) get chosen which link to this action in previous actions. If the action gets active, the `perform` method of the related template is performed for activating automated logic (no logic implemented by default). Furthermore the principal can be notified by e-mail if the action gets activated. If the principal has finished the task following the description attribute and respecting the attached items, they can finish the action by choosing a decision (can also be performed by the implemented logic). Selecting a decision of an action triggers the linked actions.

AutomatedWorkflowAction Type

An automated action is very similar to an action. The automated action does not require any manual interaction by the user. To run, it needs a job class which implements `AutomatedWorkflowTemplateJob`.

When the automated action gets activated it calls the `perform` method of the job class. Afterwards the decision returned by the `perform` method is selected.

WorkflowDecision Type

Decisions are outbound links from an Action to another Action. A Decision defines which action might follow a given Action. An action can only be a follow-up to an action if linked by a Decision Template. An Action cannot be followed by another Action if no Decision exists between the two Actions.

Each decision references the action it belongs to and a set of actions it links to. These actions get activated when the decision is chosen.

WorkflowItemAttachment Type

An attachment holds a reference to an item of any type needed in the context of the referenced workflow. If this reference is specific for some actions of the workflow, these actions can be referenced, too.

Modeling a Workflow via the API

To learn how to model a workflow through API, see [Modeling a Workflow Using the API](#).

Processing a Workflow via the API

To learn how to process a workflow through API, see [Processing an Action Using the API](#).

Displaying Workflows with Product Cockpit

If you have a set of workflows defined for a particular user, then you can see these from the perspective of Product Cockpit. An additional section **Workflows** is created in the navigation area. This section holds the workflows defined for the current user. Moreover, if the number of defined workflows exceeds a certain defined value, then the total number of workflows is split into sets, where each set holds a configurable amount of workflow entries. This helps in avoiding a long list of workflows in the navigation area while still having quick and easy access to the particular workflow. The number of these paginated workflow entries is easily configurable in the `local.properties` file for the `cockpit` extension.

```
cockpit.navigationarea.workflowtree.pageSize=50
```

Related Information

[Product Perspective](#)

[Users in Platform](#)

[The Cronjob Service](#)

[Workflow Integration](#)

[Workflow and Collaboration](#)

[Workflow Overview](#)

[Collaboration Center in Backoffice Framework](#)

[Workflow Templates](#)

[Workflow and Collaboration](#)