# Table of Contents

## Overview
- Architecture Overview  
- Frontend Layer  
- Business and Persistence Layer  
- Database & Application Server Environment  
- SAP Hybris Commerce: Best of Both Worlds  

## Development
- Easy to Install and Run  
- Easy to Develop for  
- Easy to Configure  
- Java Frameworks, Libraries and Standards  

## Customization
- Flexible Business Objects Using an MDA Approach  
- Adding or Replacing Services in the Service Layer  
- Creating new Extensions  
- SAP Hybris Commerce Accelerator  
- Events  
- Internationalization  
- Multi-catalog Support  
- Multi-language and Multicurrency Support  
- Cockpit Customization  

## Integration
- Data Validation  
- Data Integration  
- Data Integration via Data Hub  
- 3rd Party System Integration  
- Built-In RESTful Web Services  
- Spring Integration & Java Message Service  

## Operations
- Minimal & Typical System Infrastructure  
- Persistence Cache  
- Typical Oracle RAC / MySQL Setup  
- Session Failover  
- Virtualization  
- SAP Hybris Commerce in the Cloud  
- Multi-core Performance  
- Monitoring the Running System  
- Performance Monitoring using dynaTrace  

## Security
- Role Creation  
- Securing Your Web Application  
- Securing Your Model  
- Securing Your Data  
- Audit Trails  
- Authentication and Authorization with LDAP  
- PCI Security  

## Summary
Executive Summary

Changing customer behavior, rising competitive pressures, emerging technologies and globalization are all forcing organizations to rethink how they serve their customers and do business. Consequently they must adopt more flexible business models, add new sales channels and interact with customers in new ways – all of which increase the complexity and demands of managing businesses successfully.

In order to be able to handle this complexity while at the same time lowering costs, improving customer service and increasing sales, different departments and subsidiaries often start implementing ad-hoc disparate IT solutions. These provide no single view of customers, products, inventories or marketing information, which is scattered across organizations and systems. Missing data, insufficient collaboration, duplications and uncoordinated activities lead to inefficient marketing and revenue losses. The organization struggles to manage their multiple business models, sales channels, customer interactions, transactions and processes. Additionally, their systems are often rigid, preventing customers from quickly delivering innovative features to gain competitive advantage.

Due to the open and extensible architecture of the SAP Hybris Commerce platform, customers are able to easily extend and customize the data model and the business logic to optimally fit their business requirements. Furthermore, using popular technology standards, SAP Hybris enables you to shorten project ramp up times and accelerate project implementations, as there is no need for extensive training of your IT teams. The SAP Hybris Commerce offers single system for managing product content, commerce operations and channels on top of the SAP Hybris Platform. Thus helping retailers, manufacturers and others to create a unified and seamless cross-channel experience for their customers – from online, to in-store, to mobile and beyond.

Relying on the SAP Hybris Platform, our customers are able to support their businesses and easily integrate with their existing systems; they are able to improve functional productivity, reduce costs and increase revenues. Also they have a future-proof technical foundation for growth which helps them in gaining competitive advantage and delivering even more innovative capabilities.

This document will give you a technical overview of the SAP Hybris Platform. We first discuss the key software layers and architecture. Next, we show you how easy it is to get started with our software and what standard software libraries we chose to use. We then introduce the various options for customization. For example, this includes our MDA-approach for the data model and the usage of the Spring Framework for extending or replacing core business services. Integration options with external systems are discussed next. We cover our built-in options such as automatically generated RESTful web services for the data model as well as custom integration options. Finally, we provide several insights into managing operations and securing the SAP Hybris Platform.
Overview

This section will give you a broad overview of the SAP Hybris Commerce architecture. Later we will dive into individual topics, but before we do this we would like you to understand the lean and lightweight approach we have taken for the design of the SAP Hybris Commerce. You will find out that our architecture is based heavily on Spring\(^1\), an open source framework primarily known for its dependency injection (DI) and aspect-oriented programming (AOP) features. The SAP Hybris Commerce extension concept is based on this Spring foundation and allows the SAP Hybris Platform to be highly extensible and flexible as you will see.

Architecture Overview

The execution environment for the SAP Hybris Platform is a Java EE Servlet Container, for example Apache Tomcat\(^2\). The platform and all extensions to it are running within the Spring environment, which allows easy wiring and configuration of each component. It provides generic logic such as security, caching, clustering and persistence. A SAP Hybris Commerce module, such as the Customer Support Module, will typically result in one or more extensions being added to the SAP Hybris Platform. Each extension may add additional services in the form of Spring beans to the global application context or may also choose to overwrite existing functionality. Extensions are either provided by SAP Hybris as part of the purchase of a module or written by yourself, in which case we call this extension a custom extension. An extension may simply provide additional business logic without exposing a visible UI or it may also contain a Java Web Application, that – for example – exposes RESTful interfaces or a HTML-based UI that can be used via a standard web browser.

---

1 Spring Framework, http://spring.io/
Frontend Layer

As we mentioned already, an extension may include a Java Web Application. A natural framework choice to realize this web application in our Spring environment is to use the Spring MVC Framework, but technically any Java Web Framework such as JSF or Struts may be used by our clients.

We distinguish between two types of web applications:

1. Presentation-oriented web applications are geared towards web browsers and will typically generate dynamic HTML markup. If you choose to offer a web-based store, you will either adapt one of our templates (see for example the Accelerator template) or create this web application from scratch. You are free to use whatever markup and combination of technologies you like, for example we recommend Spring MVC, HTML5, CSS3 and the jQuery Javascript library. The client will later use a standard web browser (desktop or mobile) to interact with your web application. The Backoffice Administration Cockpit, a back-end UI used to administrate the SAP Hybris Platform, is a good example of a presentation-oriented web application.

2. Service-oriented web applications typically implement web service endpoints. The clients of these web services are often mobile devices, but may also be other presentation-oriented web applications (e.g. Ruby on Rails-based web applications) or other 3rd party systems. The Omni Commerce Connect (OCC) web services for example offer a RESTful web services API and therefore can be considered a service-oriented web application.

SAP Hybris Platform is completely open to whatever frontend technology you prefer. You may choose any Java-based web application framework which will allow you to directly interface with the business logic APIs (so-called ServiceLayer API), or you may choose to use a non-Java web technology, run your web application on remote systems and integrate these web applications using web services. As Figure 3 shows, web applications can either be developed using Java (Spring MVC, JSF, others) or any other non-Java technology like a Rails-based front end that communicates with SAP Hybris Commerce via web services can be used.

Figure 3: Each extension to the SAP Hybris Platform may include a Java-based web application
Business and Persistence Layer

To interact with the SAP Hybris Platform and all services offered, an extension uses the ServiceLayer API. This API consists of a set of services, managed by the Spring environment and includes all business logic that the SAP Hybris Commerce offers. The SAP Hybris Platform and each extension may add services to the ServiceLayer, in which case all other extensions can use these services. Each service has a clearly defined Java interface and custom extensions can easily add new services to the ServiceLayer or customize and replace existing ones.

Another key feature of any extension is the ability to add or modify business objects to the global SAP Hybris Commerce data model. The data model is defined in XML and can easily be adjusted to your specific needs. Using this Model Driven Architecture (MDA) approach, our customers have full flexibility in designing their data model and do not have the need to wrap meaningful properties of a business object in general-purpose containers. All applications that are built on top of this flexible data model, such as the Product Cockpit that is used to manage product data, automatically adjust to it.

The persistence layer offers a similar concept to Hibernate, in that the mapping between business objects and database tables is taken care of behind the scenes with an ORM (Object-Relational-Mapping) framework. However it is more flexible as it seamlessly and immediately supports all new custom business models that are created by our partners.

The clear separation of the SAP Hybris ServiceLayer API and SAP Hybris PersistenceLayer API is very beneficial when it comes to testing. Each layer can be tested in isolation, which significantly reduces the time and setup cost for the tests.

If an extension also contains a web front end, the web front end will have its own private Spring application context. This allows you to separate the specific concerns of the web application from the rest of the business logic that your extension provides. The web application context of a web application can still access the full ServiceLayer API.

Figure 4: Two key parts of the SAP Hybris Platform are the Service Layer and Persistence Layer
Database & Application Server Environment

When it comes to the overall execution environment for the SAP Hybris Platform, our product is shipped with preconfigured Tomcat server. This allows our partners to quickly setup new systems and is often also the choice for production systems. The SAP Hybris Commerce build process will create the required Enterprise Application Archive (EAR) for you.

The usage of Java and the JVM implies that these containers are able to run on any operating system, such as Microsoft Windows, Apple Mac OS X Server or various other Unix-based operating systems.

When it comes to the database management system (DBMS) SAP Hybris actively supports Oracle\(^4\), MySQL\(^5\), Percona XtraDB Cluster, SAP HANA DB or Microsoft SQL Server\(^6\). The SAP Hybris persistence layer is designed in a way that it does not use vendor-specific features such as triggers or stored procedures. This is very beneficial in case a client would like to switch to another supported database.

---

\(^5\) MySQL, http://www.mysql.com/
SAP Hybris Commerce: Best of Both Worlds

You typically need to decide whether you want to realize your product using green field development or want to buy an out-of-the-box solution. Both choices have advantages and disadvantages. Green field development typically involves high risk, high cost and will take considerably longer than an out-of-the-box solution. Yet out-of-the-box solutions often are too inflexible and have limited extensibility. In addition, they tend to result in a lot of vendor lock-in.

SAP Hybris offers a solution, that tries to minimize the disadvantages of both solutions while maximizing the advantages by using our carefully chosen design approach. The SAP Hybris Platform and its modules are built for extensibility and flexibility. Existing features can easily be customized or completely replaced. Our solution offers you a shorter time to market and is based on standard technologies for which it is easy to find developer support.

SAP Hybris solution will finally result in a lower total cost of ownership (see Figure 7). Compared to a project by one of our competitors, the implementation cost will already be lower due to our highly flexible and extensible architecture. As our software makes better use of the provided hardware, overall lower operational costs occur over the years after implementation. Finally, upgrading to a new major release with SAP Hybris Commerce is easier and can often be completed within days rather than weeks.

To make things even more simple, we introduced the SAP Hybris Commerce Accelerator, which helps our partners to further reduce the time and cost required for ecommerce projects. A SAP Hybris Commerce Accelerator application is a “best-practices” reference implementation and allows our partners to reduce the implementation time to 3 – 4 months for a typical client project.

Figure 7: Total cost of ownership
Development

Easy to Install and Run
SAP Hybris Commerce software is easy to install and run. SAP Hybris Commerce installation is a ~2.0GB file. Once unzipped into a local directory, the only requirement to run our software is an up-to-date Java installation (currently Java 8 is required). Without any other configuration, the system will start using a file-based DBMS (HSQLDB\(^7\)), which is great for the development phase. Later, in production, the same code that runs on developer machines will also run on the server machines, but in this case of course with a DBMS solution like SAP HANA DB\(^8\). A typical startup of the preconfigured SAP Hybris Tomcat server takes less than 30 seconds.

Easy to Develop for
We do not believe that special tools and IDEs should be required to customize our software. Therefore, any Java IDE can be used for development. We recommend the open source and free SpringSource Tool Suite\(^9\), but developers are free to choose other IDEs (IntelliJ IDEA, Netbeans, or specialized versions of Eclipse).

Easy to Configure
Our platform and all extensions are configured via text-based properties files (one main properties file for the platform and core extensions) and XML-based Spring configuration files. Instead of using XML-based Spring configuration, you may choose to use annotations for your custom extensions. One area where this is particularly well suited is the frontend layer of a Spring MVC-based web application.

While we encourage our partners to use annotation-based Spring configuration, the SAP Hybris Platform itself uses XML-based Spring configuration. This provides benefits for partners as the configuration required is more centrally located and can be identified and changed more easily.

Java Frameworks, Libraries and Standards
The SAP Hybris Commerce builds upon many well proven and widely adopted frameworks most notably Spring, ZK Framework, Angular JS and JUnit. Basing our architecture on such solid foundations brings several benefits: our partners are already familiar with many of these and thus avoid steep learning curves, SAP Hybris can focus on its areas of specialty rather than reinventing the wheel, SAP Hybris can incorporate the numerous latest advances from each framework to bring the best products quickly and efficiently to market.

Many of the frameworks, libraries and standards that we use can be seen in Figure 8.

---

\(^7\) HSQLDB, http://hsqldb.org/
\(^8\) SAP HANA, https://hana.sap.com/abouthana.html

Figure 8: Frameworks, libraries and standards used in the SAP Hybris Commerce.
Customization

Many parts of the SAP Hybris Commerce can be deeply customized, if you wish to. The business model for example is very flexible and can be tailored to your exact needs using an MDA approach. Each service that is exposed in the ServiceLayer can be customized or even completely replaced. Our partners use the same extension concept that we use internally to extend our platform to build the omni-channel solution for their customers. Our own extensions make use the Spring Event System, which makes it easy for partners’ extensions to be notified about what is happening in the platform.

For a partner, to create a new custom extension, SAP Hybris offers easy to use tools that will generate all boilerplate code required for an extension.

Of course SAP Hybris is also strong in internationalization. The default business model supports multiple languages, currencies and sites. Even our built-in back end UIs – cockpits – can be customized to fit your own specific needs.

Flexible Business Objects Using an MDA Approach

SAP Hybris uses an MDA (Model Driven Architecture) approach when it comes to modeling the business objects. The result is a business model that exactly fits your needs and does not require you to wrap any relevant data in meaningless container structures. According to the defined business model, SAP Hybris Platform will automatically generate the business objects and any back-end application like the Backoffice Administration Cockpit will adjust to it.

The business objects can be specified either via UML or XML and will then be automatically generated during the build process. All required ORM setup for managing these objects is generated too, as well as the database schema needed to store it. Business objects are simple POJOs which automatically benefit from the SAP Hybris Platform core services such as caching, clustering, personalization, and internationalization support, and are accessible via the SAP Hybris back-end applications like Backoffice Administration or Customer Support Cockpit.

The following code snippet is an excerpt of an items.xml file. It shows the XML-based format in which business objects are defined.

```
<itemtype code="Product" extends="GenericItem">
  <deployment table="Products" propertytable="ProductProps"/>
  <attributes>
    <attribute qualifier="code" type="java.lang.String" generate="true">
      <persistence type="property" qualifier="Code"/>
      <modifiers read="true" write="true" search="true" initial="true" optional="false" unique="true"/>
    </attribute>
    ...  
  </attributes>
</itemtype>
```

Adding or Replacing Services in the Service-Layer

As SAP Hybris uses the Spring Application Framework, it is very easy to add or replace existing services in the ServiceLayer. By replacing default services, you are free to use any specific implementation.

All a developer needs to do is follow these easy steps to replace an existing service in the ServiceLayer:

1. Find the service interface and Spring bean definition you wish to replace
2. Implement the new service using the same Java interface. Your new implementation can internally forward some of the method calls to the old, still available service or replace the logic completely.
3. Replace the alias of the service in the Spring configuration file and point to the new service implementation bean.

Creating New Extensions

To create new extensions, hybris includes a command-line tool called extgen. It can be used to create completely empty extensions or to create template extensions that already offer certain functionality. These templates currently include:

- **yempty** – a default template for any type of extension
- **ybackoffice** – a template for a backoffice extension
- **ycockpit** – a basic template for a cockpit extension
- **yaddon** – a basic template for an AddOn extension
- **ycommercewebservices** – a basic template for an extension that uses the Omni Commerce Connect web services
- **SAP Hybris Commerce Accelerator templates** – a set of reference extensions for Accelerator applications (see the next section for more details)
Once an extension has been created, it needs to be included into the SAP Hybris Commerce build cycle and also loaded during server startup. To achieve this, the extension is simply added to the `localextensions.xml` file, which is part of the SAP Hybris Platform.

Figure 9 shows an example `localextensions.xml` file and how it maps to a directory structure in the file system.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<extensions>
  <extension dir="${HYBRIS_BIN_DIR}/ext-channel/cscockpit"/>
  <extension dir="${HYBRIS_BIN_DIR}/ext-channel/instore"/>
  <extension dir="${HYBRIS_BIN_DIR}/ext-channel/mobileoptionals"/>
  <extension dir="${HYBRIS_BIN_DIR}/ext-channel/mobileservices"/>
  ...
</extensions>
```

SAP Hybris Commerce Accelerator

The SAP Hybris Commerce Accelerator is a standard solution that integrates best-practice omni-channel commerce capabilities.

It is a ready-to-use framework built on top of the SAP Hybris Platform and it includes numerous extensions from the SAP Hybris Commerce Suite. It can easily be used by our partners using the extgen commandline tool. The hybris Commerce Accelerator applications are a best-practice reference implementations and allow our partners to reduce the implementation time to 3–4 months for a typical client project. We’re constantly extending the range of our Accelerator portfolio for different industries and verticals. For example, there are ones for B2B and B2C sectors; as far as industries are concerned, you can choose from Travel, Telco, Financial Services applications.

Figure 10:
One of the Commerce Accelerator storefronts
Each of the Commerce Accelerator applications include extensions that offer ready-to-use storefronts and use state-of-the-art web technologies such as Spring MVC 3, the Blueprint CSS framework and the jQuery JavaScript library.

The comercefacades extension that is included in the Accelerator applications comes preconfigured with features such as internationalization and Apache Solr Search. It provides a coarse-grained interface which is ideal for creating custom web services on top, e.g. for mobile clients. Our partners have full access to the source code to customize this template to achieve the final result.

**Events**

SAP Hybris uses Spring event handling support to publish and receive various Spring-related and custom events. As each extension becomes part of a global application context, an extension can easily listen to all these events or publish its own, custom events.

SAP Hybris uses this observer-style event handling throughout the platform, for example session or model lifecycle events are being published. In a cluster setup, these events can be also propagated to other cluster nodes.

When you create your custom extensions, the built-in event handling support is a powerful feature enables you to react to activities happening in the system or to inform other components via custom events.

**Internationalization**

The SAP Hybris Commerce provides numerous ways for you to localize and personalize your application including:

- multi-catalog support,
- multi-language and multicurrency support,
- and multi-site support.

**Multi-catalog Support**

Many of SAP Hybris customers deal with multiple catalogs, which share much common data yet have their own focus, target audience and language. SAP Hybris Commerce provides powerful support for such cases by allowing customers to specify multi-level hierarchies of catalogs in which the child catalog(s) will inherit the parent’s settings by default, all of which can be overridden if desired (similar to inheritance in Object Oriented Programming). This is a huge time saver in the content management domain as it allows our partners to specify shared or common data in the ‘parent’ catalog, and specify just the changes in the child catalog(s). The inheritance rules from a parent to child catalog are specified in SAP Hybris Commerce synchronization rules, which can also be used for synchronizing staged catalogs with their online counterparts. A customer can for example update and modify a staged catalog until it is ready for release, and then apply synchronization rules to update its respective online version. It is also common for our customers to run these synchronizations as cron jobs, an approach that is fully supported by SAP Hybris Commerce.

**Multi-language and Multi-currency Support**

The majority of SAP Hybris customers require extensive internationalization support, not just for their presentation layer and printed products but also for their business objects. At the last count more than 80% of our customers support more than one
language or currency in their SAP Hybris Commerce implementation. We support and extend the native Java internationalization framework allowing complete control over the localization of business objects, currencies, numeric fields, dates etc. Advanced options include language-fallback mechanisms, which offer greater flexibility than Java’s own fallback logic. We can for example specify that if a text variable is not available in German, then we should fallback to US English.

Internationalization has not been patched as an afterthought to our business model design, rather it has been integrated from conception, and it is consequently completely embedded into the business models. Each attribute of a business object – be that one from SAP Hybris Commerce, or one added by a customer – is therefore localizable should you wish. In conjunction with the synchronization rules mentioned earlier, our customers can easily and effectively support and synchronize multiple catalogs in multiple languages and locales.

Cockpit Customization

Based on the modules you have purchased, your SAP Hybris Commerce will include extensions that offer various front ends to the backend user. We will introduce only some of them here to give you an idea:

→ **Product Cockpit Module** enables cockpit end users to manage and structure product information and catalogs
→ **Customer Experience** enables the end users to manage website pages, providing them with intuitive graphical way of data presentation and management.
→ **Backoffice Administration Cockpit** is the graphical user interface of SAP Hybris Commerce and offers finer-grained control over the user’s data.

The Product Cockpit is shown below. All SAP Hybris Commerce cockpits are based on the ZK framework and Angular JS can be customized at various levels by our partners. The Product Cockpit allows users to browse, modify and add products to the system. Should the user be assigned administration rights, they are also able to modify the configuration of the cockpit itself, changing for example the attribute groupings, allocating new attributes to a group or reordering attribute lists directly within the cockpit.

While the Product, WCMS, and Print Cockpits offer intuitive and high-level control of their data within the system, the Backoffice Administration Cockpit offers finer-grained control of all data and operations, relating to all areas of the SAP Hybris Commerce. On the left of the Backoffice Administration Cockpit is an (XML-configured) tree providing access to all areas of the SAP Hybris Commerce. One can for example examine and modify cron jobs, import and export data, modify facet searches, all from within the Administration Cockpit, and all in a consistent and intuitive manner. Should a partner introduce their own cockpit, the Administration Cockpit provides complete control over its layout and functionality. Finally, the layout of the Administration Cockpit itself is guided by an XML file, and by modifying this (either within or outside of the Administration Cockpit) one can totally control what should be displayed where and how within the Administration Cockpit itself.
Integration

Integration challenges typically include bulk data import/export operations and 3rd part/remote systems integration. In this section we will present how SAP Hybris partners typically solve these integration challenges and what SAP Hybris offers out of the box to enable integration.

Before we dive into these challenges though, we will explain SAP Hybris validation support. In the end, both data and process integration may change the data stored in the SAP Hybris Commerce. It is of great importance to keep this data consistent and valid, which is why the hybris ServiceLayer includes a customizable ValidationService.

Data Validation

Implicit data validation is built into any SAP Hybris Commerce business object via the ServiceLayer APIs. This validation support is based on JSR 303 (Bean Validation) but adds run-time support for managing the validation constraints. This makes it possible to define the validation constraints for each business object in the Backoffice Administration Cockpit and have them immediately take effect in the Product Cockpit without restarting the SAP Hybris Platform. You can see how validation errors are visually highlighted in the Product Cockpit in Figure 14.

Figure 13: Common integration challenges include bulk data and 3rd party systems integration

Figure 14: Data Validation in the Backoffice Administration Cockpit
The validation constraints can be grouped into attribute-level, type-level and dynamic constraints. Attribute-level constraints are attached to single attributes, such as a credit card number field. Type-level constraints allow you to validate multiple attributes at the same time; an example would be the validation of the postal code and city field for a customer’s address business object. In this case a type-level constraint can ensure the postal code entered fits to the city name and vice versa. Finally, the dynamic constraints allow our partners to validate based on the outcome of a BeanShell script which completely opens up the logic used.

Data Integration

To import and export data SAP Hybris Commerce offers the ImpEx engine. ImpEx is capable of importing bulk text data files. If the format available differs from the CSV-based ImpEx format, open source ETL tools such as Talend can be used to first transform the data into ImpEx format. ImpEx will finally interact with the service and persistence layer to write the data to the database. This workflow is shown in Figure 15.

Data Integration via Data Hub

As companies embrace customer-centric, omni-channel strategies, many organizations continue to be met with major obstacles when it comes to integrating disparate systems and more importantly the data they share. Executing a comprehensive omni-channel strategy requires companies to assimilate massive amounts of data – data that up to now would be stored in multiple external systems with varying data models – into a single platform.

Traditional data imports between third-party systems, such as ERP, and commerce solutions often require custom-built integrations that are costly both to develop and to maintain. New models for customer engagement and omni-channel commerce require new solutions that reduce time-to-market and total cost of ownership, and allow for greater flexibility.

The Data Hub for SAP Hybris Commerce provides a flexible, scalable, and service-oriented solution to simplify data integration efforts (import and export) between a SAP Hybris system and external data storage solutions and systems, such as ERP, PLM, CRM, and so on. Data from various sources can be consolidated and prepared for loading into SAP Hybris Commerce.

Data fragments can easily be consolidated and grouped. Also, users can create categories and assign data accordingly. The Data Hub is based on RESTful web services that allow organizations to run as many Data Hub servers as they like. The Data Hub recognizes and logs incorrect and incomplete data and returns it to its data pool for repair and resubmission.

The Data Hub supports batch and message driven integra-

![Figure 15: Importing and exporting of data using ImpEx engine](image-url)
tion and uses Spring Integration for message handling and supporting all Spring transport modes. It transforms data from the Raw Model (simple, flattened view, matching inbound data; schema-less definitions; fragmented data) to a Canonical Model (ideal representation of domain object like "customer"; independent of source and target structures) to the target model (in this case: the SAP Hybris Commerce data model).

The Data Hub Adapter serves as the central interface between the SAP Hybris Platform and the Data Hub.

With the Data Hub being fully extensible, SAP Hybris partners can quickly and easily build their own extensions and can manually define data models, model transformations, grouping patterns, import and publication interfaces.

Figure 16:
The Data Hub imports and exports data between SAP Hybris Commerce and external data storage systems.

3rd Party System Integration

To connect with external systems (e.g. ERP, PLM, CRM systems) there are 3 common options:

→ SOAP/RESTful web services
→ Spring Integration
→ Java Message Service

Spring Integration offers a wide array of integration options and is our preferred solution when it comes to asynchronous integration. JMS is another popular solution for asynchronous integration.

Synchronous integration is typically achieved via SOAP/RESTful web services. While partners are free to build their own web services in custom extensions, the SAP Hybris Platform automatically generates RESTful resources for each business object, which is often an excellent integration option.
Built-In RESTful Web Services

Built-in RESTful web services are generated transparently with each build process. Each single business object defined in the XML-based definition files [items.xml] can automatically be accessed via the SAP Hybris Commerce WebService API. Our WebService API includes CRUD access for all business objects, support for collection paging and attribute selection. The API uses the standard Accept header to enable content negotiation and uses XML and JSON resource representations. The SAP Hybris role-based security mechanism is also used for the API access, which means that only API calls that adhere to a certain user group will be granted access. To reduce bandwidth, the API also supports ETag-based caching.

The following simplified HTTP request and response shows how all User business objects can be requested via the RESTful WebService API:

```
GET /ws410/rest/users HTTP/1.1
HOST: localhost
Accept: application/json

 "user" : [ { "type" : "employeeDTO",
 "uid" : "admin",
 "pk" : "8796093054980",
 "loginDisabled" : false },
 { "type" : "customerDTO",
 "uid" : "anonymous",
 "pk" : "8796093087748",
 "loginDisabled" : false } ]
```

Spring Integration and Java Message Service

Spring Integration provides an extension to the Spring programming model to support the well-known Enterprise Integration Patterns. It enables lightweight messaging within Spring-based applications and supports integration with external systems via declarative adapters. Those adapters provide a higher-level of abstraction over Spring support for remoting, messaging, and scheduling.

To make it as easy as possible to use Spring Integration, the SAP Hybris Platform already includes all required libraries. In addition, SAP Hybris provides special support classes for sending Spring events to integration channels and for triggering processes based on incoming messages.

In addition to Spring Integration, a SAP Hybris Commerce extension may also use Java Message Service (JMS) to connect with external systems. JMS is a proven technology for sending messages to one or multiple clients.
**Operations**

In this section, we will discuss various topics that deal with or affect the running of SAP Hybris Commerce software. We will first outline typical system infrastructures. Next the SAP Hybris Commerce persistence cache is explained. We present common database setups, including MySQL and Oracle RACs. Further, we will discuss system failover behavior for clustered infrastructures and the possibility to run SAP Hybris commerce products in the cloud as well as virtualized environments. Finally we will look at performance benchmarks and explain how a running SAP Hybris Commerce system can be monitored.

**Minimal & Typical System Infrastructure**

As seen in Figure 19, the minimal system infrastructure involves a web server, application server and a database engine. An appropriate firewall needs to be setup to safeguard these systems from attacks.

The primary purpose of the web server (often Apache HTTPD) is to serve all static content and redirect requests for dynamic content to the application server. Often, the web server is also configured to provide other functionality such as logging.

Requests for dynamic content will be forwarded to the application server (e.g. Apache Tomcat). It is the application server where the SAP Hybris Commerce is being installed and where all business processes are running.

The persistence layer of SAP Hybris Commerce will use the configured DBMS to persist and retrieve all data.

All these components (web server, application server and database) can initially be executed on the same physical hardware. Of course, this setup would not include any redundancy and is therefore not suitable for production but rather a theoretical setup used to introduce the various system components. As with any multi-tier architecture, it allows to scale in a relatively simple way. The components can be split out to individual servers and later multiple instances of each component can run in a clustered setup, which we will look at next [Figure 19].

The application servers are responsible for all dynamic content generation and business processes and therefore consume most processing power. To enable the SAP Hybris Platform to scale in such a scenario, SAP Hybris provides the Cluster Module.

Before new instances for web servers, application servers and database servers are added, a load balancer needs to be configured and will later interface all web request. The load balancer will use sticky sessions to direct all web traffic of a user session to the same set of web and application servers. This guarantees that the in-memory state of each application server does not need to be persisted in the database, which would result in significant additional traffic to the database.

The cluster setup is completely transparent to the developer and integrated seamlessly with the native SAP Hybris caching system. If the instances are installed in a local network, UDP
broadcasts are used to communicate among the nodes. If instances of SAP Hybris Commerce are run in the cloud, UDP broadcasts will be blocked and therefore TCP will be used to communicate among the nodes. The information that needs to be exchanged among the nodes is cache invalidations for business objects that are stored in the local caches of each instance. Once data on one node has changed, this information needs to be sent to all other nodes so their cache entries can be invalidated and the data can be reloaded from the database once again required.

The database setup for scalability depends on the DBMS chosen. For Oracle RAC setups, the Oracle JDBC driver will transparently distribute the JDBC calls to the cluster nodes and no special SAP Hybris Platform setup is required. In case of MySQL, a master/slave approach is used and the master and slave databases need to be configured in the SAP Hybris Platform.

The caching mechanism is part of the SAP Hybris Platform persistence layer and reduces the amount of database queries. Besides the overall lean and efficient architecture, it is another reason why SAP Hybris Commerce has greater performance per single server node than our competitors. The persistence cache transparently caches search results and business objects in memory. The SAP Hybris cache can be split into multiple regions and the exact business objects that are allowed to be cached for each region can be specified.

```
<table>
<thead>
<tr>
<th>Cache Region</th>
<th>ENTITIES</th>
<th>TYPES</th>
<th>QUERIES/RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eviction Strategy</td>
<td>LRU</td>
<td>FIFO</td>
<td>LRU</td>
</tr>
<tr>
<td>Size</td>
<td>100 000</td>
<td>50 000</td>
<td>20 000</td>
</tr>
</tbody>
</table>
```

Figure 21: SAP Hybris cache regions can be flexibly configured.
This allows our partners to fine-tune a running system and to make sure that certain business objects are cached for a longer time while other objects will be removed more quickly due to a limited cache size. The eviction strategies that we support include least recently used (LRU), least frequently used (LFU) and first in first out (FIFO). The default cache implementation is using the open source Ehcache project.

**Typical Oracle RAC / MySQL Setup**

A common database setup is the Oracle Real Application Clusters (Oracle RAC). The JDBC URL that needs to be configured in SAP Hybris Platform needs to include all RAC nodes that should be used. The Oracle JDBC driver will then transparently distribute all database calls (Figure 22).

If multiple database instances are required and MySQL is chosen as the DBMS, both MySQL and the SAP Hybris Platform need to be configured in a master/slave mode. The configuration, which can be enabled with a couple of lines in the SAP Hybris Platform text-based configuration files, will make sure that all write operations will be delivered to the configured master database. The MySQL master will then propagate all changes to the slave databases. The slave databases are only used for read operations, which for a typical application is the majority of all operations. Figure 23 shows the MySQL Master/Slave setup.
Session Failover

SAP Hybris Commerce installations can be configured for partial or full session failover. The result is that other cluster members will be able to serve data in the event that the original cluster node fails. Partial session failover will use a persistent Cart business object. In case of a failover, the cart and all associated business objects (cart content, user object, etc.) can be restored from the database and another cluster node can serve the requests. It is unlikely, that this failover behavior will be recognized by the user, as other state such as the order progress cannot be stored in the database.

To achieve full session failover, Oracle Coherence or Tomcat Session replication must be used. For instance, Oracle Coherence will distribute all session information to the other nodes so that another node can fully take over after a failover. While this represents the best behavior possible, it means additional hardware requirements (memory) and cost (license cost).

Virtualization

With the emergence and wide-scale adoption of infrastructure virtualization, SAP Hybris has ensured that its solutions are fully compatible with this paradigm, and includes for example tests against VMware and Microsoft HyperV in its continuous integration test suites.

By moving one’s infrastructure from real hardware to a virtual infrastructure, partners can change infrastructure within minutes rather than days or weeks. This proves invaluable for running and testing SAP Hybris Commerce in differing topologies quickly and flexibly.

SAP Hybris Commerce is actively testing and supporting the VMware (vSphere 5) and Microsoft Hyper-V (2008 R2) virtualization environments. Based on our load tests, we recommend to use VMware, which results in ~10% overhead.

SAP Hybris Commerce in the Cloud

With the SAP Hybris Commerce Cloud, a fully functional omni-channel commerce platform that can be deployed in the highest quality private cloud environment, you have the flexibility to run any business model, and easily scale to support your transaction goals while meeting the highest traffic peaks. We take care of the hardware, software, and all operations. We help you improve operational efficiency and scalability and maintain high availability and security. Additional services will help you cope with growing traffic during surge events, so if you plan to execute large campaigns, for example, the solution can adjust and scale horizontally to accommodate the increased traffic.

The natural successor to virtualization on single computers has been the emergence of the cloud paradigm – the offering of virtualized massively scalable and complete infrastructures, by vendors such as VMWare, Amazon, Google and Microsoft, configured and accessed over the internet. Amazon has adopted the term “elastic compute cloud – EC2” to emphasize how easy and fast it is to grow or shrink ones infrastructure in this fashion.

In a virtual cloud, one can easily scale the configuration horizontally to cope for example with heavy traffic over Christmas, and then just as easily reduce it once traffic has again decreased. An increasing number of diverse pricing schemes are also emerging for cloud usage, for example paying on an as-use basis, paying to reserve computer space on the cloud for the next year, or bidding for computer power on the cloud at some point in the next week.

Two main causes of concern among companies new to the cloud are security and assurances of quality-of-service. As their existence depends on it, companies that implement clouds have aggressively addressed the issue of security, and one can now deploy to a cloud in the knowledge that their security is better than your own. With regards to quality-of-service, Amazon EC2 for example has a Service Level Agreement of 99.95%.

![Virtualization overheads](image)
SAP Hybris Commerce is fully tested in cloud scenarios, and has Presales and Sales Demos for this purpose, so partners can be assured that the journey to the cloud with SAP Hybris Commerce is an easy and proven one. These demos are built using our “turn-key” cloud instances, which can be quickly configured and deployed, thus setting up a complete SAP Hybris Commerce configuration in minutes.

SAP Hybris current cloud-based solutions see the cloud simply as a virtual infrastructure, an approach called IaaS – “Infrastructure as a service”, which is to be contrasted with PaaS and Saas – “Platform as a service”, and “Software as a service”. SAP Hybris will continue to expand its cloud-based solutions across all three areas.

Multi-core Performance

Critical areas of SAP Hybris software are carefully designed to enable full use of multi core systems. For example catalog synchronization or data import via the ImpEx engine will automatically be run multithreaded. Multithreading is completely transparent to the developer or system integrator. We have optimized and tested our software to take best advantage of modern multi-core CPUs.

Monitoring the Running System

To monitor a running system, SAP Hybris Commerce supports JMX (Java Management Extensions) and exposes several beans by default. This includes beans to monitor the cache, running cron jobs, data sources and sessions. You can use any standard JMX client such as the Oracle JConsole to monitor the system.

Besides supporting JMX, we also recommend to use hyperic for infrastructure and application monitoring.

Performance Monitoring using dynaTrace

Each SAP Hybris Platform ships with a limited license for dynaTrace11 which allows our partners to make performance profiling an essential part of their development process and save time due to the early detection of performance issues. dynaTrace can also be integrated when running continuous or nightly builds and is an excellent solution for monitoring the running system. Once activated in the SAP Hybris Commerce configuration, you can directly launch dynaTrace dashboards from the SAP Hybris Administration Console. You can either view the reports in the browser or start a Java Web Start client.

![Figure 26:](image) Java Monitoring and Management Console

![Figure 27:](image) dynaTrace performance monitoring

---

11 dynaTrace for Java, https://www.dynatrace.com/technologies/java-monitoring/
Security

Security is of paramount importance for any software product and in particular for web-based software that is available to anyone with an internet connection. SAP Hybris places a huge emphasis on securing its software and addresses a number of areas including those illustrated above.

Role Creation

The foundation upon which all security decisions are made is user roles, one or more of which can be assigned to each user to best match your business model and business process. Roles can include for example “Sales department”, “Corporate marketing”, “Customer”. The role-user mapping is often stored on a partner’s LDAP server but could also be contained for example in their SAP Hybris Commerce persistence model or a service in their Service-Layer. Using the Backoffice Administration Cockpit, SAP Hybris partners can create a variety of hierarchical roles each with particular security clearance levels. The security framework will then check that a user/entity is allowed to perform an operation before executing it, as described next.

Securing Your Web Application

With a user’s role(s) in hand, Spring security framework can be used to determine who is allowed what level of access to various parts of the web layer, and also to control coarse and fine grained access to the web services. If you are using Spring MVC you can easily control access to sub-sections of a web page with Spring authorization tags.

Securing Your Model

Similarly, once we have access to the user’s role(s), SAP Hybris system will control the Create, Read, Update and Delete operations on the underlying data models. This is performed not only by using Spring security mechanisms but also by using SAP Hybris Platform FlexibleSearch filters.

The FlexibleSearch service is a feature of SAP Hybris own ORM framework and can convert a high-level query statement involving business objects to a lower level SQL statement. It is possible to set up a so called FlexibleSearch filter that specifies a predicate to be appended to the “WHERE” part of the resulting SQL statement when executed by users with certain user roles. This then ensures that users belonging to those roles are incapable of retrieving, modifying or deleting any objects in the database that do not satisfy that predicate.

Securing Your Data

Data such as credit card numbers and passwords should of course never be stored in clear text in the database and SAP Hybris Commerce supports the standard hashing and encryption practices to avoid this. But in addition, SAP Hybris can transparently encode certain highly-sensitive data before it even crosses the DAO layer to the database, with Advanced Data Security option. This contains a feature called Transparent Attribute Encryption that encrypts and decrypts data between the database and application server transparently, using keys up to 256 bits that are managed from within the Backoffice Administration Cockpit. This is an option for partners who are particularly concerned about securing data at the maximum level possible, and is a necessary step towards PCI compliancy.
Audit Trails

Also related to security requirements is SAP Hybris Commerce support for audit trails. All changes to information in our system are recorded via our audit-trail functionality. This includes the type of change, the user/entity that made this change, the time it occurred, and the old and new values.

The audit trail also provides information that can be used to monitor resources, system break-ins, failed logins and breach attempts. It also helps identify security loopholes, violations, spoofing and those users who are attempting to circumvent security, either intentionally or unintentionally.

Authentication and Authorization with LDAP

SAP Hybris Commerce has an optional LDAP extension that allows authentication of users listed in LDAP and Active directories. In particular the LDAP extension allows:

→ SAP Hybris Commerce to verify the identity of user accounts against an LDAP or active directory server,
→ the chance to implement a Single-Sign-On concept,
→ the ability to import LDIF (the LDAP query language) files and search results.

PCI Security

While our partners are certainly able to create business models and logic to support PCI (Payment Card Industry – i.e. credit card transactions), SAP Hybris recommends strongly that you do not store credit card information on your own database. To gain PCI compliance as a merchant, you must fulfill several rigorous requirements13. These are rather complex, time consuming and expensive, and we consequently recommend that you use a Payment Service Provider (PSP) that is already compliant to this standard. Many SAP Hybris Commerce installations are already running with this kind of integration, which allows a comfortable one-click shopping experience. In this scenario, SAP Hybris Commerce system stores a hash-code per user that is itself non-critical, yet provides unique identification with which both SAP Hybris product and the PSP can handle a user’s complete shopping experience.

However if our partner wishes to perform PCI within their system, the option for added data encryption mentioned earlier (“Securing your data”) will prove useful. This will ensure that sensitive data does not cross the DAO layer thus minimizing risks of security leaks. However this is only the first of many obligatory steps that must be followed should a partner wish to support PCI compliance themselves.

Summary

Today’s commerce ecosystem demands a new breed of retailer: one that exhibits flexibility and integration across multiple channels. Companies need to develop flexible business models to meet the demands of their customers, increasing the complexity of their solutions to stay competitive.

This white paper provided a technical overview of the SAP Hybris Commerce and its ability to meet the evolving needs of retailers, including key software layers and architecture, customization options, built-in features, and managing operations.

SAP Hybris offers an agile and extensible commerce solution that allows retailers to extend and customize their user experiences. A single system to manage product content, commerce operations and channels, SAP Hybris Commerce enables a frictionless, consistent experience across purchase process, improving back- and front-end functionality, improving customer satisfaction, reducing costs and in turn increasing revenue.

The model-driven architecture employed by SAP Hybris Commerce is easy to install, develop and customize, and results in a future-proof commerce solution that fits your business needs exactly, without force-fitting.

About SAP Hybris

SAP Hybris enables businesses to transform how they engage with customers, innovate how they do business, and simplify their technology landscape. With a comprehensive approach to customer engagement and commerce, our solutions unlock opportunities to optimize your customers’ experience and transform your business. We help you drive relevant, contextual experiences across all of your customer touch-points in real-time, so that you can create strong differentiation and build competitive advantage in the Digital Economy.

SAP Hybris has helped some of the world’s leading organizations transform themselves in response to changing market conditions and customer expectations – delivering exceptional experiences, adding new channels, evolving their business models, and entering new markets. How can we help you?

Explore SAP Hybris solutions today. For more information, visit www.hybris.com.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE’s or its affiliated companies’ strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other product and service names mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See http://global.sap.com/corporate-en/legal/copyright/index.epx for additional trademark information and notices.

13http://www.pcicomplianceguide.org/