hybris software

THE AGILE RESPONSE

hybris Architecture and Technology

Abstract

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[1].

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 hybris 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, 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 hybris Multichannel Commerce Solution uses the hybris Platform to offer a single system for managing product content, commerce operations and channels. 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 hybris Platform our customers are able to support their businesses and easily integrate with their existing systems; they are able to improve functional productivitiy, 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 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 hybris Platform.

[1] Trend zum Multi-Channel, ChannelPartner.de, March 24, 2011, http://www.channelpartner.de/handel/ecommerce/2383684/index.html

Table of Contents

Overview

This section will give you a broad overview of the hybris 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 hybris Multichannel Suite. You will find out that our architecture is based heavily on Spring[2], an open source framework primarily known for its dependency injection (DI) and aspect-oriented programming (AOP) features. The hybris extension concept is based on this Spring foundation and allows the hybris Platform to be highly extensible and flexible as you will see.

Architecture Overview

The execution environment for the hybris Platform is a Java EE Servlet Container, for example Tomcat[3] 6 or VMware vFabric tcServer[4], which is also based on Tomcat but provides commercial support.
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 hybris Module, such as the PCM Module, will typically result in one or more extensions being added to the 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 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.

Figure 1: Architecture overview of the hybris Multichannel Suite

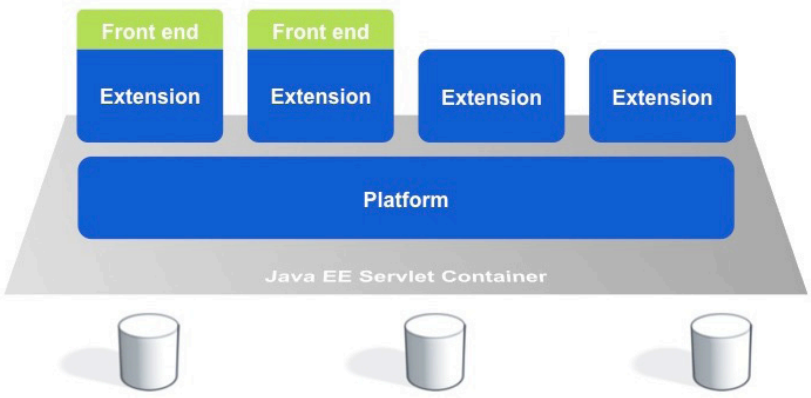
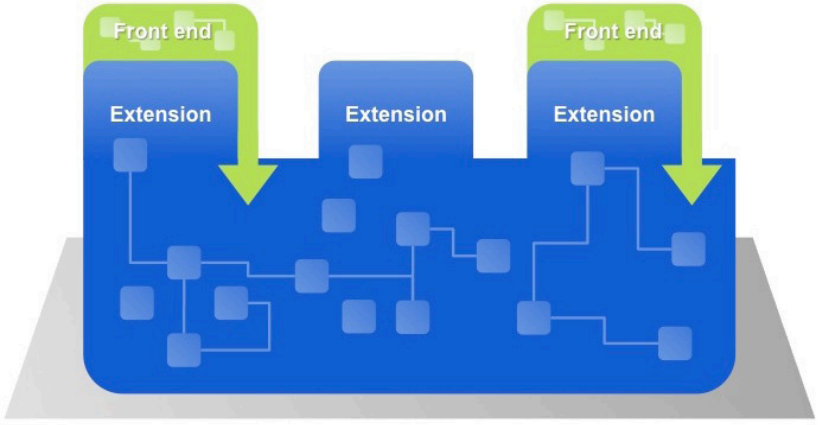


Figure 2: The hybris Platform is based on the Spring Framework



[2] Spring Framework, http://www.springsource.org/about
[3] Apache Tomcat, http://tomcat.apache.org/
[4] VMware vFabric tcServer, http://www.vmware.com/products/vfabric-tcserver/

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:

- 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 hybris Administration Console, an extension used to administrate the hybris Platform, is a good example of a presentation-oriented web application.
- 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 hybris Commerce Web Services for example offer a RESTful web services API and therefore can be considered a service-oriented web application.

hybris 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 hybris 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 hybris via web services can be used.

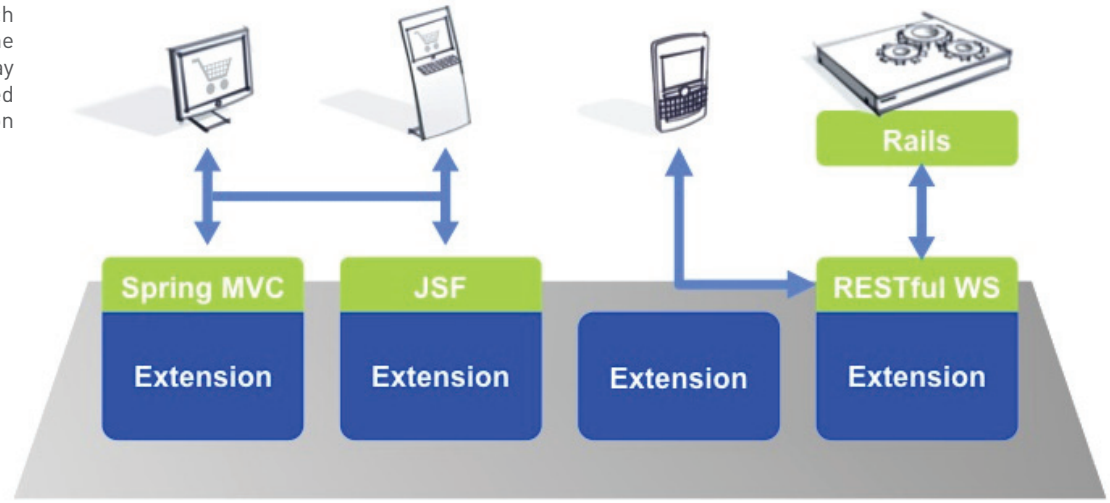


Figure 3: Each extension to the hybris Platform may include a Java-based web application

Business and Persistence Layer

To interact with the hybris Platform and all services offered, an extension uses the hybris ServiceLayer API. This API consists of a set of services, managed by the Spring environment and includes all business logic that the hybris Platform and extensions offer. The 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 hybris data model. The hybris 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 hybris 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 Service Layer (hybris ServiceLayer API) and Persistence Layer (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.

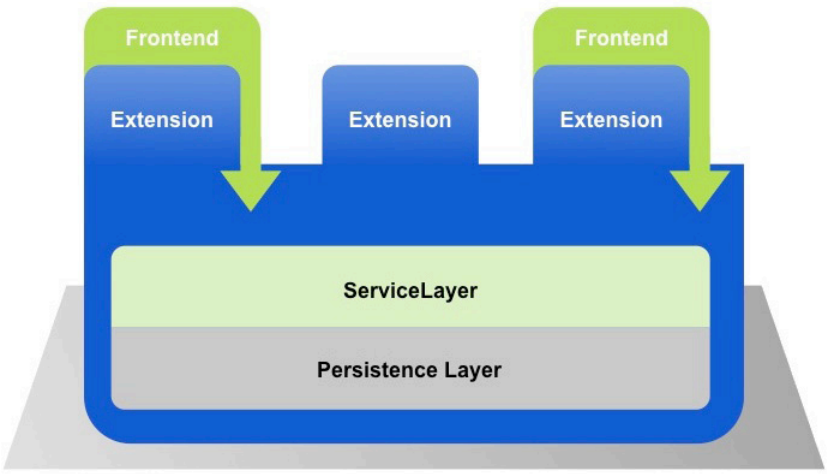


Figure 4: Two key parts of the hybris Platform are the Service Layer and Persistence Layer

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.

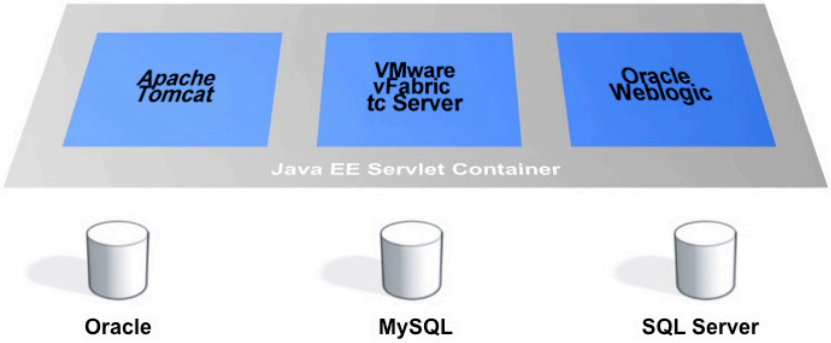
Database & Application Server Environment

When it comes to the overall execution environment for the hybris Platform, hybris ships with preconfigured Tomcat and tcServer servlet containers. This allows our partners to quickly setup new systems and is often also the choice for production systems. Besides running purely on a servlet engine, hybris can also be deployed on Oracle WebLogic[5] . The hybris build process will create the required Enterprise Application Archice (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) hybris actively supports Oracle[7], MySQL[8] or SQL Server[9]. The 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.

Figure 5: hybris supports popular databases and servlet containers



hybris: 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, out-of the box solutions tend to result in a lot of vendor lock-in.

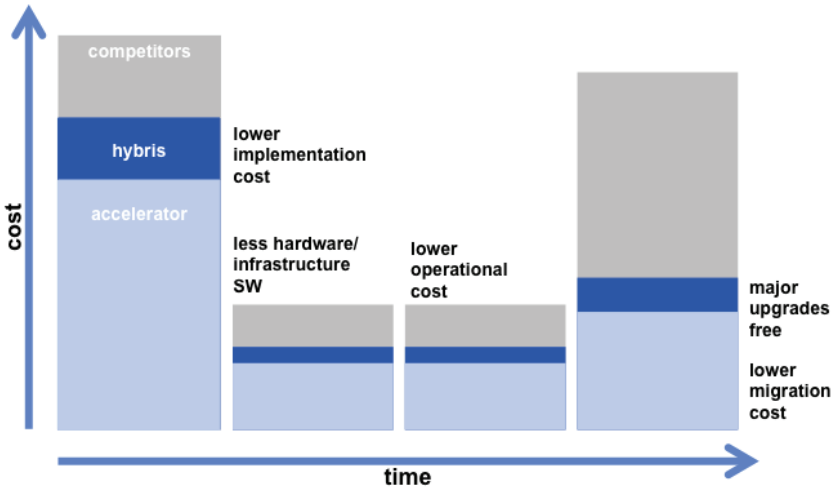
hybris offers a solution, that tries to minimize the disadvantages of both solutions while maximizing the advantages by using our carefully chosen design approach. The hybris Platform and its modules are built for extensibility and flexibility. Existing features can easily be customized or completely 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.

hybris solution will finally result in a lower total cost of ownership (see Figure 6). 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 hybris is easier and can often be completed within days rather than weeks.

[5] Oracle WebLogic, http://www.oracle.com/de/products/middleware/appserver/index.html
[6] IBM WebSphere, http://www-01.ibm.com/software/websphere/
[7] Oracle Databases, http://www.oracle.com/us/products/database/index.html
[8] Oracle MySQL, http://mysql.com/
[9] Microsoft SQL Server, http://www.microsoft.com/sql-server/en/us/default.aspx

Recently, hybris introduced the hybris Multichannel Accelerator, which helps our partners to further reduce the time and cost required for ecommerce projects. The hybris Multichannel Accelerator 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 6: Total cost of ownership



Development

Easy to install and run

hybris software is easy to install and run. The hybris Platform is a ~160MB download. Once unzipped into a local directory, the only requirement to run our software is an up-to-date Java installation (currently Java 6 is required). Without any other configuration, hybris will startup using a file-based DBMS (HSQLDB[10]), 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 Oracle RAC[11]. A typical startup of the preconfigured 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 Eclipse IDE[12], but developers are free to choose other IDEs (IntelliJ IDEA, Netbeans, specialized versions of Eclipse like SpringSource Tool Suite). If the easy installation and consistent configuration of the developers' IDEs is of concern to you, we also offer an Eclipse Pulse download, which sets up an Eclipse Workspace for hybris development. Eclipse is also the IDE hybris has chosen internally for all Java-related development tasks. We recommend to use Eclipse or the Eclipse Pulse[13] download as the provided project settings files can be reused and the time spent to setup your IDE workspace is thereby reduced.

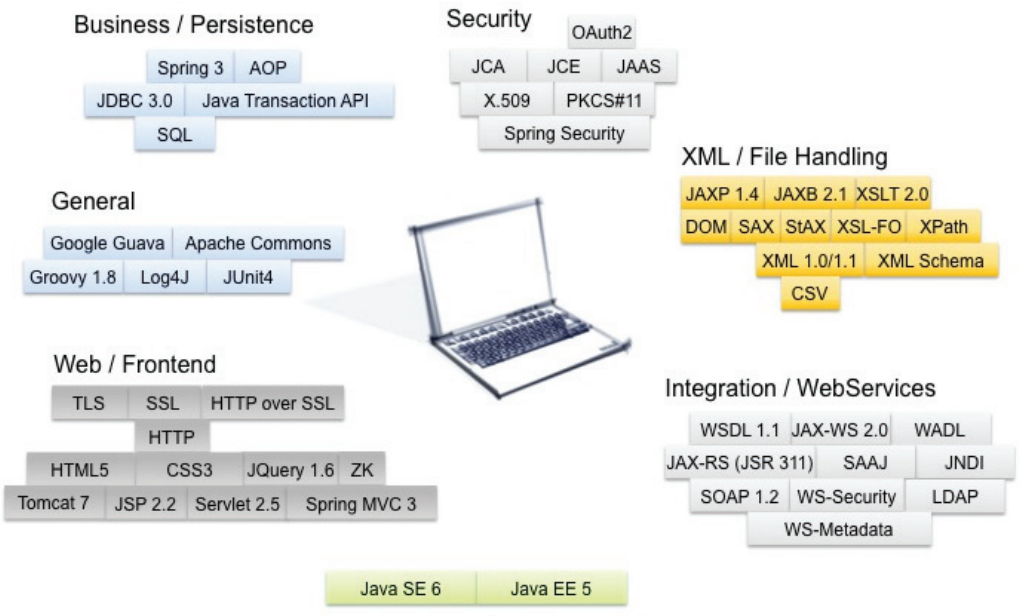


Figure 7: Java Frameworks, Libraries and Standards

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 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.

[10] HSQLDB, http://hsqldb.org/
[11] Oracle RAC, http://www.oracle.com/technetwork/database/clustering/overview/index.html
[12] Eclipse IDE, http://www.eclipse.org/
[13] Eclipse Pulse, http://www.poweredbypulse.com/

Java Frameworks, Libraries and Standards

The hybris Multichannel Suite builds upon many well proven and widely adopted frameworks most notably Spring 3.1, ZK Framework 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, hybris can focus on its areas of specialty rather than reinventing the wheel, 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 7. We are focused on adopting the latest but widely adopted languages and protocols such as Java 7, Groovy 1.8, Spring 3, HTML5: we carefully balance the adoption of latest practices and the learning curves they entail with optimizing productivity and code stability.

Customization

Many parts of the hybris Multichannel Suite 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 hybris 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 Multichannel 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, hybris offers easy to use tools that will generate all boilerplate code required for an extension.

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

Flexible Business Objects using an MDA approach

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, the hybris platform will automatically generate the business objects and any back-end application like the hybris Product 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 hybris core services such as caching, clustering, personalization, and internationalization support, and are accessible via the hybris back-end applications like hybris Administration Cockpit or Product 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 ServiceLayer

As 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 hybris 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
- storetemplate – a basic store template
- springmvcstore – a basic store template realized using Spring MVC
- flexstore – a basic store template using Adobe Flex
- ycockpit – a basic cockpit template
- multichannel accelerator – a best practices reference implementation (see below)

Once an extension has been created, it needs to be included into the hybris build cycle and also loaded during server startup. To achieve this, the extension is simply added to **localextensions.xml** file, which is part of the hybris Platform configuration.

Figure 8 shows an example **localextensions.xml** file and how it maps to a directory structure in the file system.

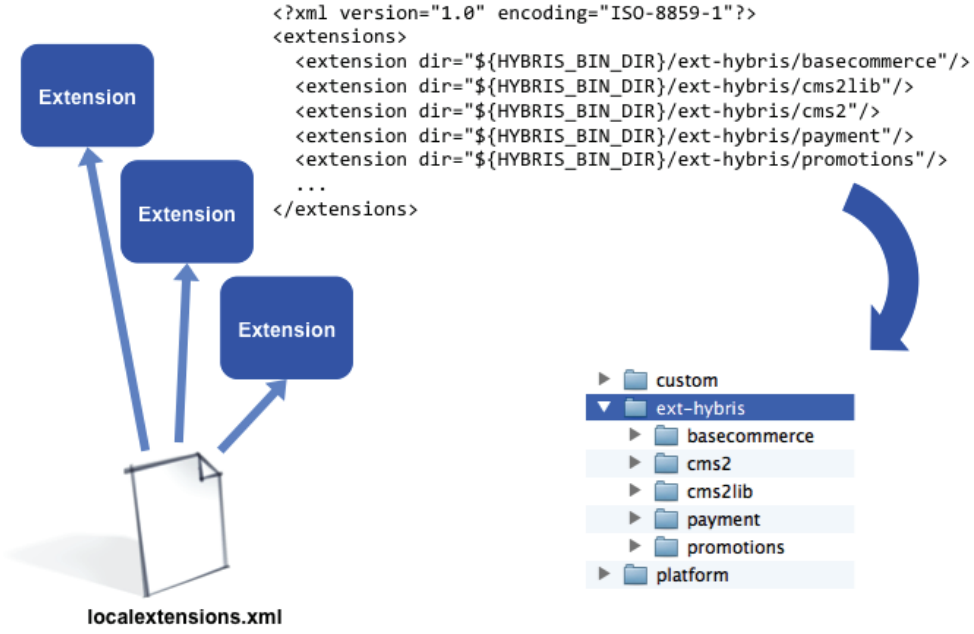


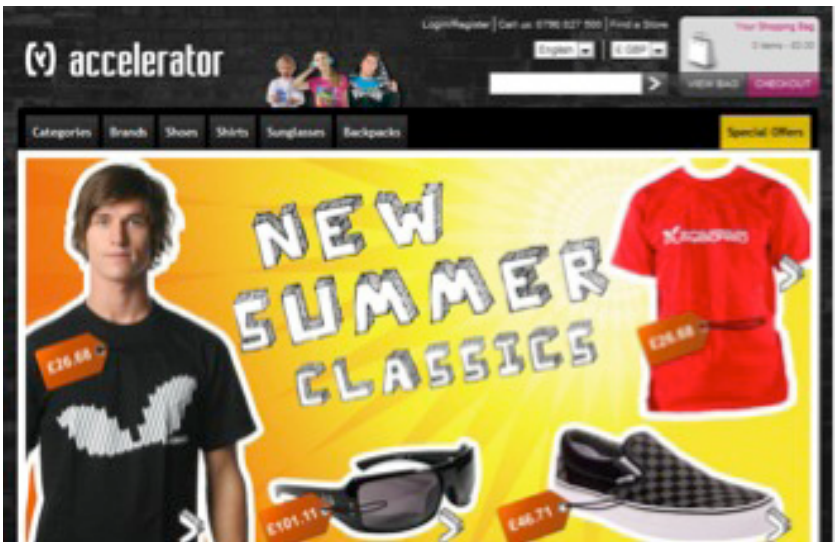
Figure 8: Each extension is referenced from localextensions.xml

hybris Multichannel Accelerator

The hybris Multichannel Accelerator is a standard solution that integrates best-practice Multichannel Commerce capabilities. It is a ready-to-use framework built on top of the hybris Platform and it includes numerous extension from the hybris Multichannel Suite. It can easily be used by our partners using the extgen command-line tool. The hybris Multichannel Accelerator is a "best-practices" reference implementation and allows our partners to reduce the implementation time to 3 – 4 months for a typical client project.

The Accelerator includes extensions that realize 2 storefronts out of the box and uses state of the art web technologies such as Spring MVC 3, the Blueprint CSS framework and the jQuery JavaScript library. The commercefacades extension that is part of the Accelerator comes preconfigured with features such as internationalization and 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.

Figure 9: One of the Multichannel Accelerator store fronts

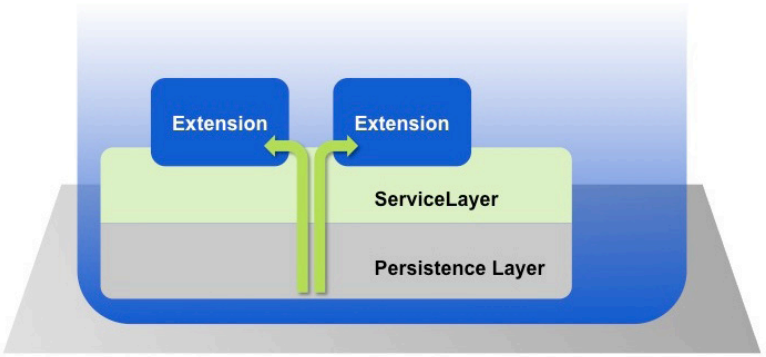


Events

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.

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.
For your own extensions, the event handling support that hybris has built-in is a powerful feature to react to activities happening in the system or to inform other components via custom events.

Figure 10: Spring events are used throughout the hybris Platform



Internationalization

The hybris Multichannel Suite provides numerous ways for you to localize and personalize your application including:

- multicatalog support,
- multilanguage and multicurrency support,
- and multisite support.

Multicatalog support

Many of hybris customers deal with multiple catalogs, which share much common data yet have their own focus, target audience and language. The hybris Multichannel Suite provides powerful support for such cases by allowing customers to specify multilevel 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 hybris 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 hybris.

Multilanguage and Multicurrency support

The majority of 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 hybris product. The hybris Multichannel Suite supports and extends 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 hybris, 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.

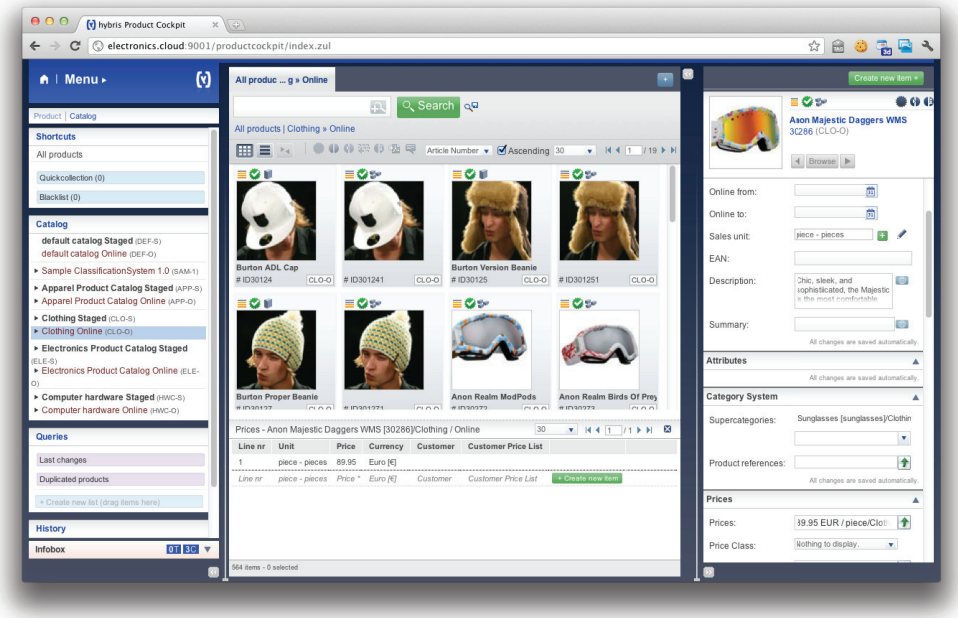
Cockpit Customization

Based on the hybris Modules you have purchased, your hybris Multichannel Suite 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:

- The hybris Product Cockpit Module enables cockpit end users to manage and structure product information and catalogs
- The hybris WCMS Module enables the end users to manage website pages, providing them with intuitive graphical way of data presentation and management.
- the hybris Administration Cockpit is the graphical user interface of the hybris Multichannel Suite and offers finer-grained control over the user's data

The hybris Product Cockpit is shown below. All hybris cockpits are based on the ZK framework and 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.

Figure 11: The hybris Product Cockpit



While the hybris Product, WCMS, and Print Cockpits offer intuitive and high-level control of their data within the system, the hybris Administration Cockpit offers finer-grained control of all data and operations, relating to all areas of the hybris Multichannel Suite. On the left of the hybris Administration Cockpit is an (XML-configured) tree providing access to all areas of the hybris Multichannel Suite. 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 hybris partners typically solve these integration challenges and what hybris offers out of the box to enable integration.

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

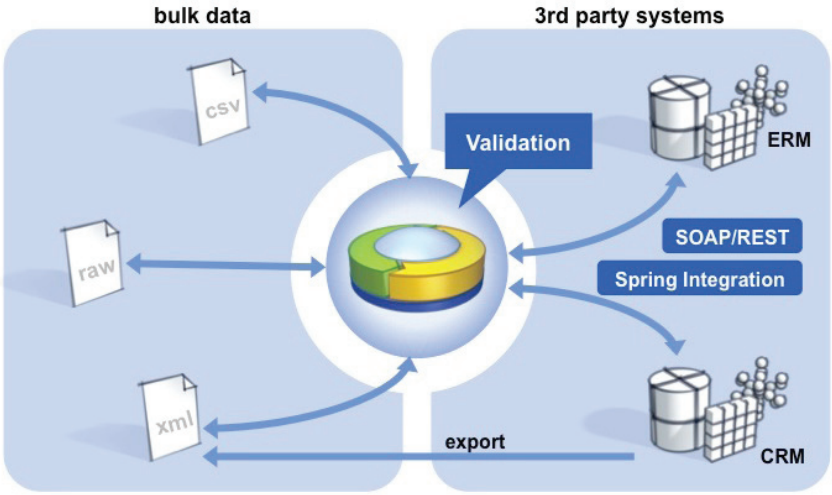


Figure 12: Common integration challenges include bulk data and 3rd party systems integration

Data Validation

Implicit data validation is built into any hybris 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 hybris Administration Cockpit and have them immediately take effect in the hybris Product Cockpit without restarting the hybris Platform. You can see how validation errors are visually highlighted in the hybris Product Cockpit in Figure 13.

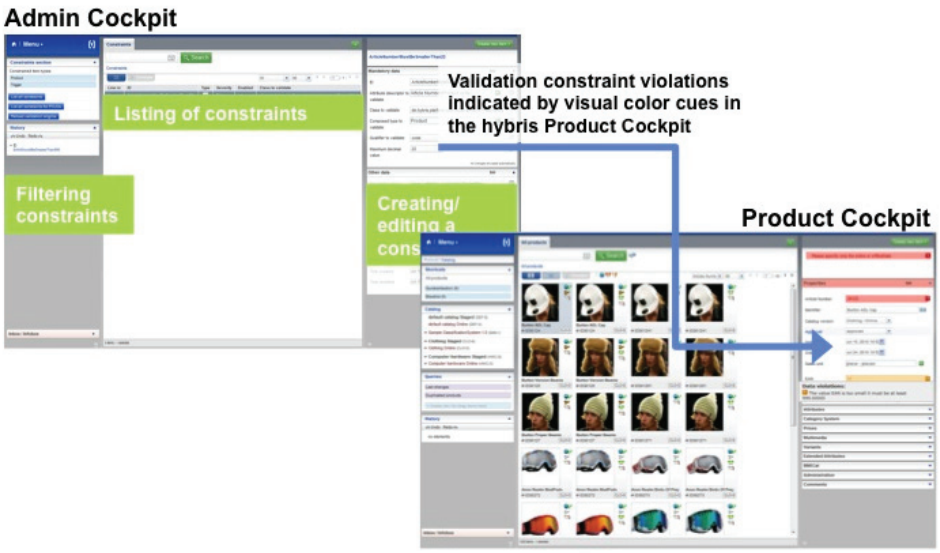


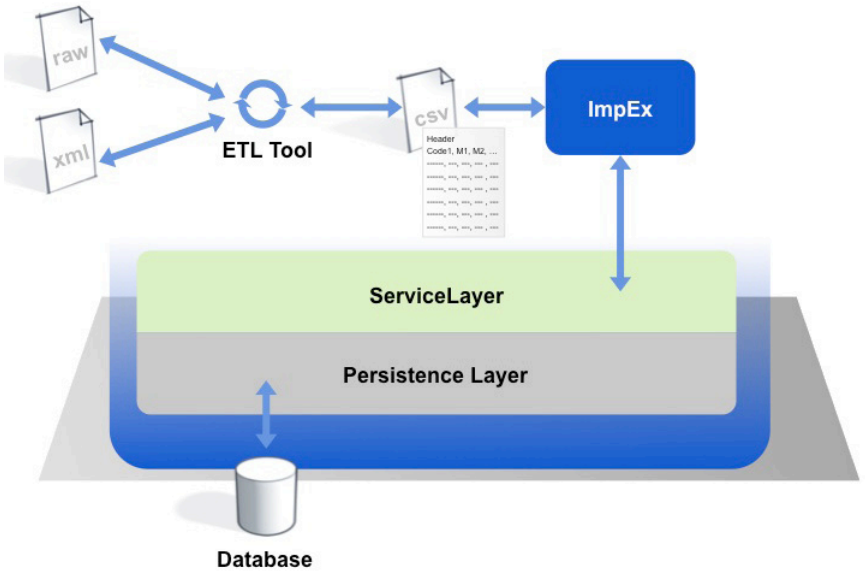
Figure 13: Data Validation in the hybris Administration Cockpit

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 hybris 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 14.

Figure 14: Common integration challenges include bulk data and 3rd party systems integration



hybris Import Cockpit

The hybris Import Cockpit (see Figure 15) allows the user to import data into the hybris Multichannel Suite using the ImpEx Engine in a visual way and without the need of specifying an ImpEx import script. As all hybris cockpits share the same visual elements, a user quickly gets used to it. The Navigation area is used for previewing the import jobs history. The Browser area in the center of the screen is used for browsing import jobs and mappings. Finally, the Editor area on the right side is used for editing the details of an import job. Using the hybris Import Cockpit, you can now perform the import operations in the user-friendly interface of the hybris Import Cockpit.

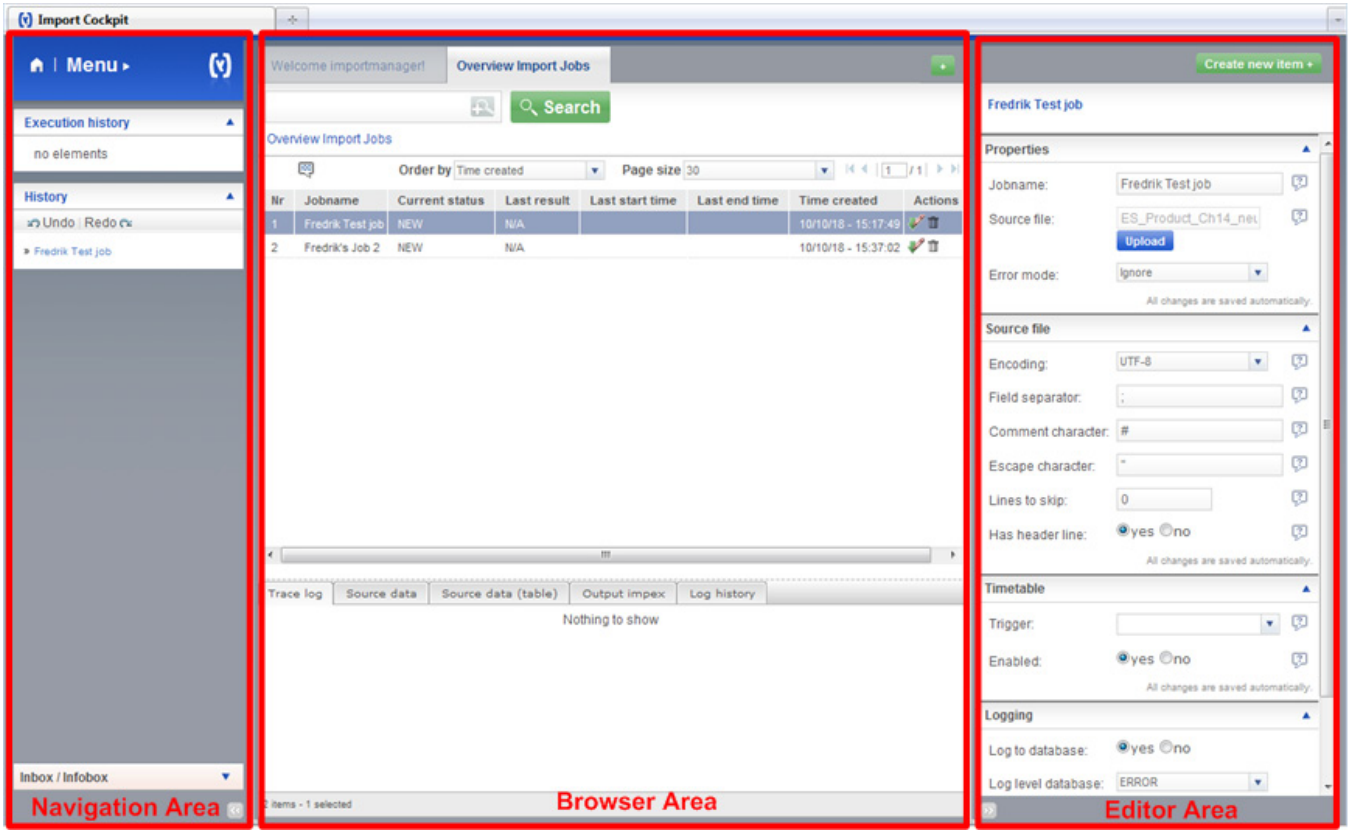


Figure 15: hybris Import Cockpit

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 and hybris offers a demo extension showcasing how JMS can be used.

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

Built-In RESTful Web Services

The hybris 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 hybris 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 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

{
 "uri" : "http://localhost:9001/ws410/rest/users",
 "user" : [ {
  "type" : "employeeDTO",
  "uid" : "admin",
  "pk" : "8796093054980",
  "uri" : "http://localhost:9001/ws410/rest/employees/admin",
  "loginDisabled" : false
 }, {
  "type" : "customerDTO",
  "uid" : "anonymous",
  "pk" : "8796093087748",
  "uri" : "http://localhost:9001/ws410/rest/customers/anonymous",
  "loginDisabled" : false
 } ]
}
```

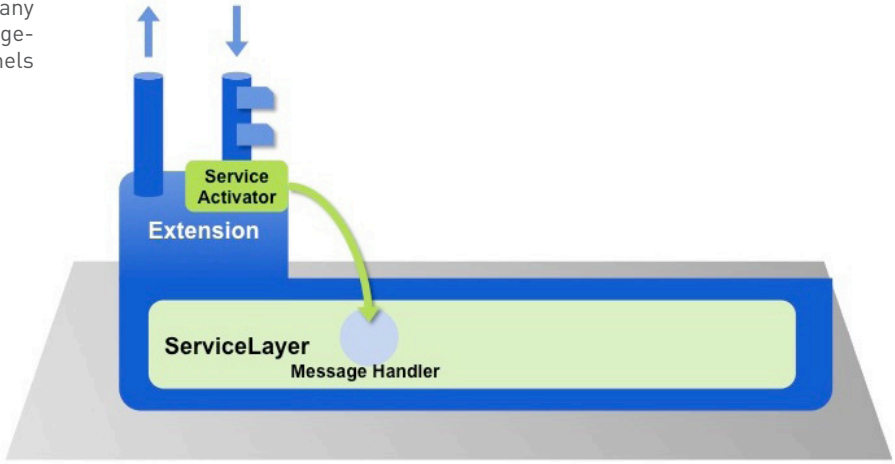
Spring Integration and Java Message Service

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 hybris Platform already includes all required libraries. In addition, 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 hybris 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. A JMS extension is available, demonstrating how JMS can be integrated.

Figure 15: Spring Integration offers many integration options, among them message-based, asynchronous channels



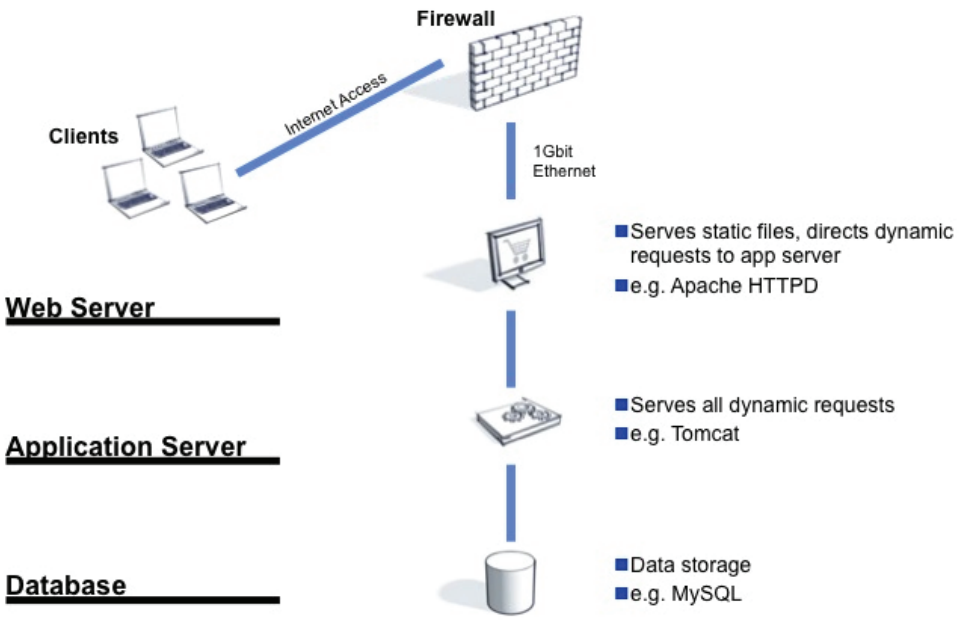
Operations

In this section, we will discuss various topics that deal with or affect the running of hybris software. We will first outline typical system infrastructures. Next the hybris 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 hybris in the cloud as well as virtualized environments. Finally we will look at performance benchmarks and explain how a running hybris system can be monitored.

Minimal & Typical System Infrastructure

As seen in Figure 16, the minimal system infrastructure involves a web server, application server and 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 hybris Multichannel Suite is being installed and where all business processes are running.
The persistence layer of hybris will use the configured DBMS to persist and retrieve all data.

Figure 16: Minimal System Infrastructure



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 17).

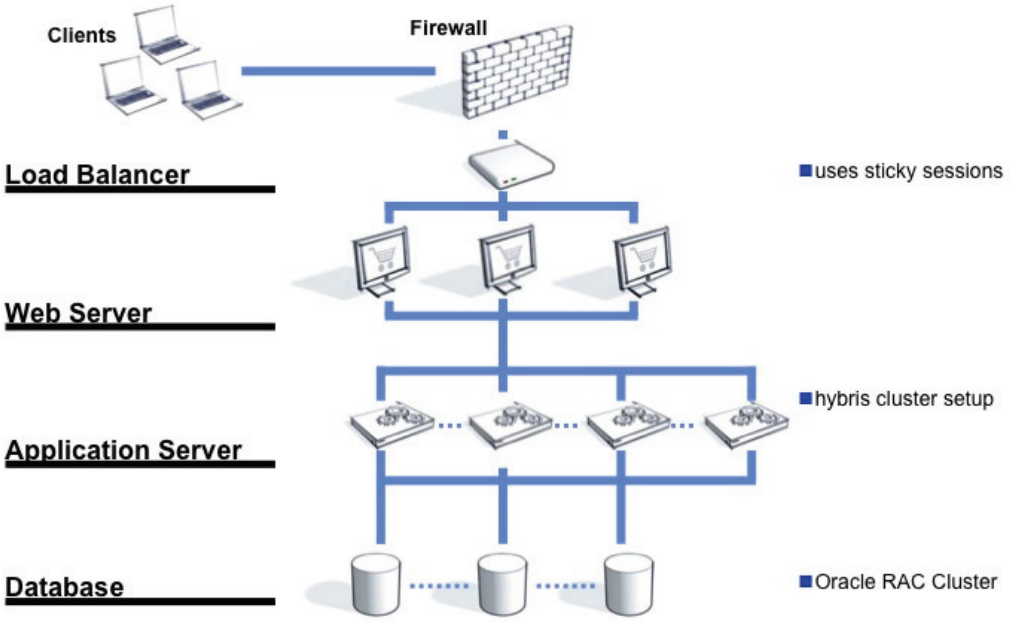
The application servers are responsible for all dynamic content generation and business processes and therefore consume most processing power. To enable the hybris Platform to scale in such a scenario, hybris provides the Cluster Module.

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 hybris Cluster setup is completely transparent to the developer and integrated seamlessly with the hybris caching system. If the hybris instances are installed in a local network, UDP broadcasts are used to communicate among the nodes. If instances of hybris 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 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 hybris Platform.

Figure 17: Scaled System Infrastructure



Persistence Cache

The hybris Cache is part of the hybris persistence layer and reduces the amount of database queries. Besides the overall lean and efficient architecture, it is another reason why hybris has greater performance per single server node than our competitors. The persistence cache transparently caches search results and business objects in memory. The 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. 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.

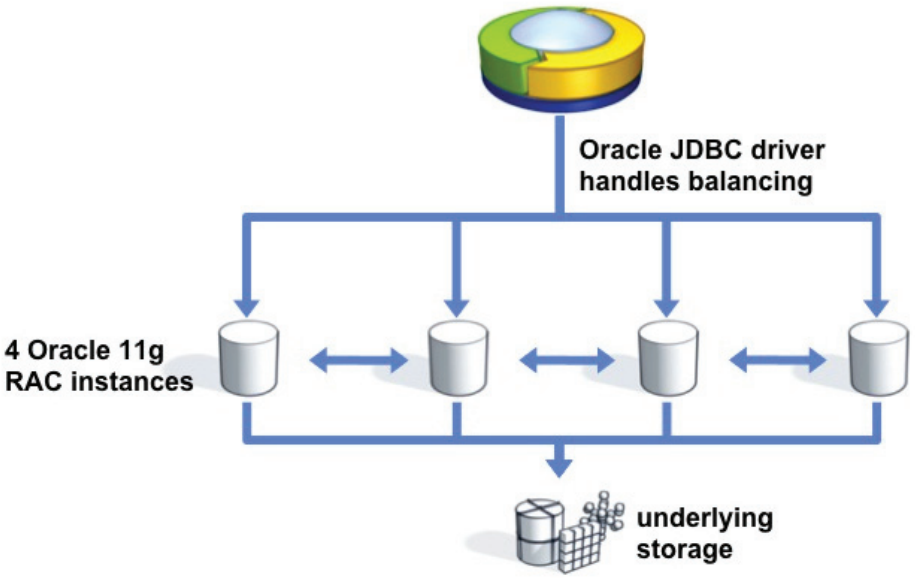
Figure 18: hybris Cache regions can be flexibly configured

Cache Region	Entities	Types	Queries Results
Eviction Strategy	LRU	FIFO	LRU
Size	100 000	50 000	20 000

Typical Oracle RAC / MySQL Setup

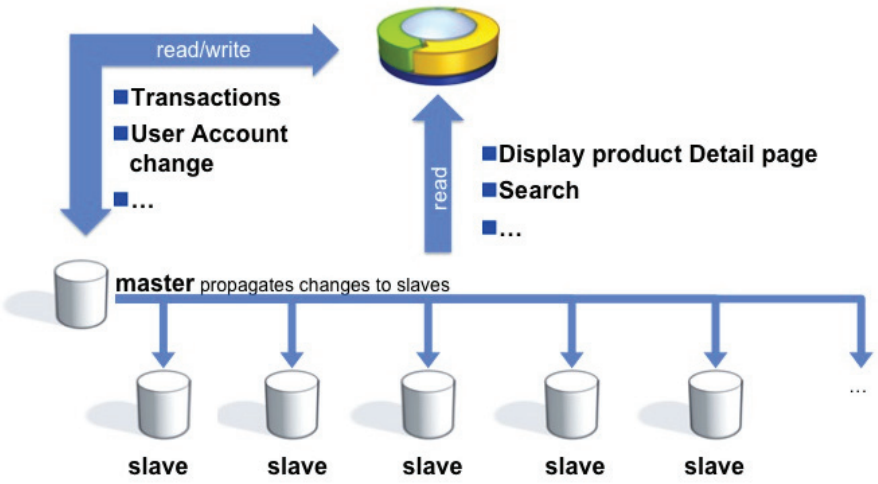
A common database setup is the Oracle Real Application Clusters (Oracle RAC). The JDBC URL that needs to be configured in hybris needs to include all RAC nodes that should be used. The Oracle JDBC driver will then transparently distribute all database calls (Figure 18).

Figure 19: Oracle RAC setup



If multiple database instances are required and MySQL is chosen as the DBMS, both MySQL and the hybris Platform need to be configured in a master/slave mode. The configuration, which can be enabled with a couple of lines in the hybris 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 20 shows the MySQL Master/Slave setup.

Figure 20: MySQL Master/Slave setup



Session Failover

hybris 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[14] 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

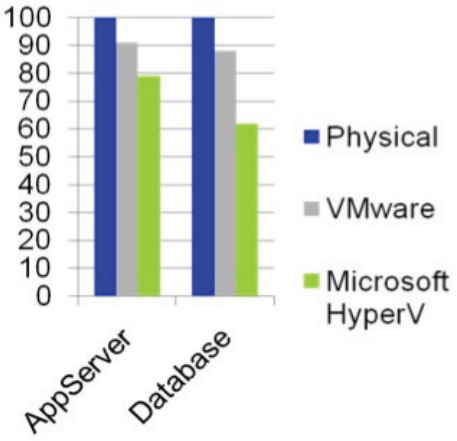
With the emergence and wide-scale adoption of infrastructure virtualization, hybris has ensured that its solutions are fully compatible with this paradigm, and includes for example tests against VMware and Microsoft HyperV in its continuous integration test suites.

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 hybris in differing topologies quickly and flexibly.

[14] Oracle Coherence, http://www.poweredbypulse.com/

hybris 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.

Figure 21: Virtualization overheads



hybris in the Cloud

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 usually 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%.

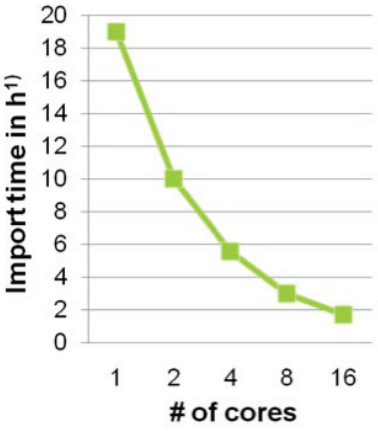
hybris is fully tested in cloud scenarios, and has Presales and Sales Demos for this purpose, so partners can be assured that the journey to the cloud with hybris is an easy and proven one. These demos are built using hybris "turn-key" cloud instances, which can be quickly configured and deployed, thus setting up a complete hybris configuration in minutes.

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". hybris will continue to expand its cloud-based solutions across all three areas.

Multicore Performance

Critical areas of 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.

Figure 22: Multicore performance

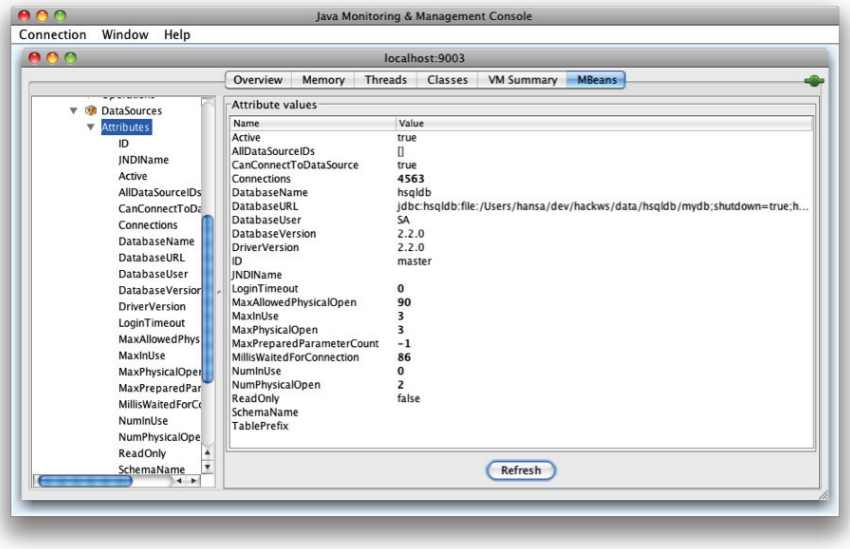


Monitoring the Running System

To monitor a running system, hybris 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.

Figure 23: Java Monitoring and Management Console

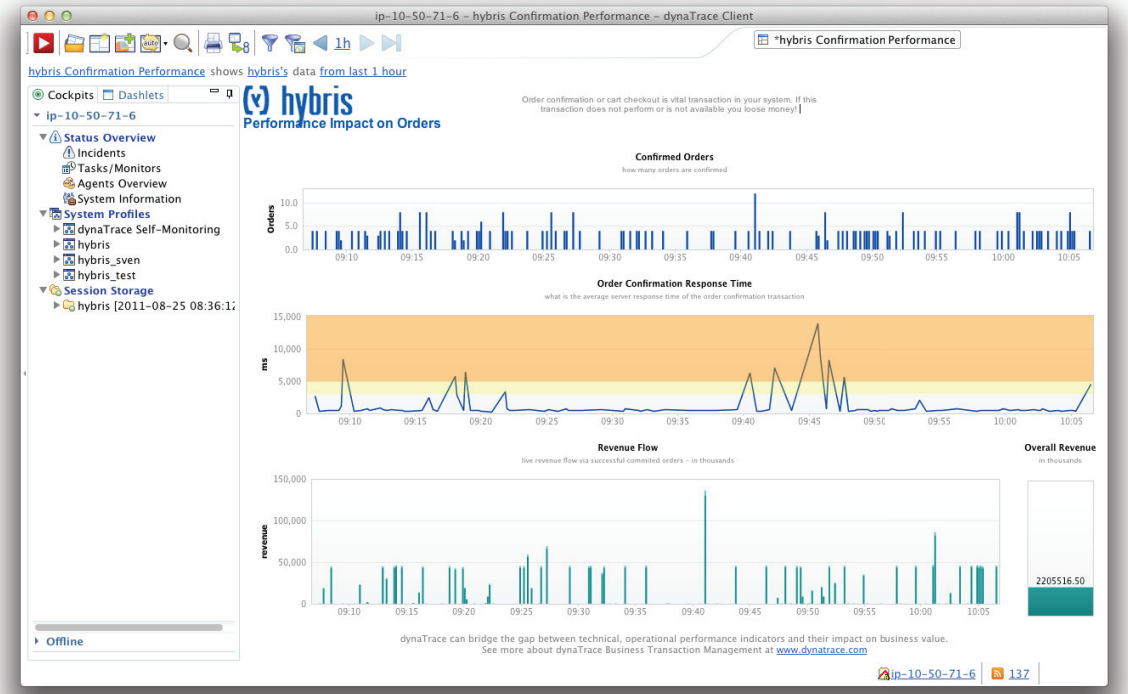


Performance Monitoring using dynaTrace

Each hybris Platform ships with a limited license for dynaTrace[15], 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 hybris configuration, you can directly launch dynaTrace dashboards from the hybris Administration Console. You can either view the reports in the browser or start a Java Web Start client.

Figure 24: dynaTrace performance monitoring



Oracle Exadata Proof of Concept Test

To proof the performance of using hybris with Oracle Exadata, hybris conducted a test that included three scenarios:

1. Mostly read access: this is a common scenario for a typical B2C retail website. Customers search and browse the catalog and finally order. There is a lot of traffic on the page but write access to the database is limited.
2. Mostly write access: this scenario is typical for marketplaces and B2B use cases. Massive PIM data is imported, which causes millions of products data affected.
3. A combination of read and write access: while running under read load, parallel write operations were triggered to measure the effect on performance.

While the full test report including the detailed system architecture is available on our website[16], we present the results of the latter scenario – both read and write access (see Figure 25). While it was possible to write 4.823 articles/s in the write scenario and deliver 1500 page impressions (PIs) in the read scenario, the combined scenario still shows impressive results: while importing a massive amount of data with 3.921 articles/s it was at the same time possible to deliver 1.184 PI/s.

[15] dynaTrace for Java, http://www.dynatrace.com/en/product-platform-java.aspx
[16] hybris Oracle EXADATA report, http://www.hybris.com/hybris/en/campaigns/Exadata.html

The bottleneck that we found was the application server, not the database. The database was still running at 56% CPU load and 550MB/s was transferred to/from the database during this test.

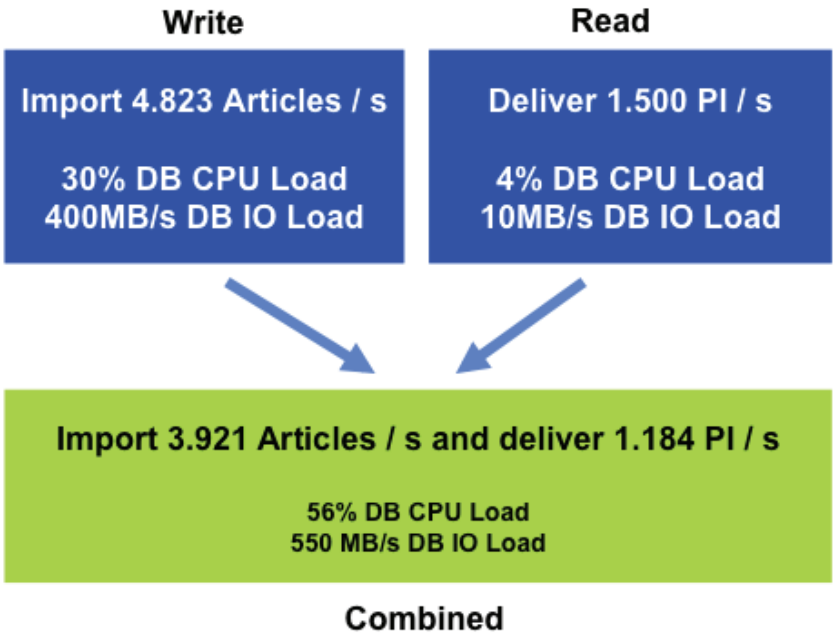
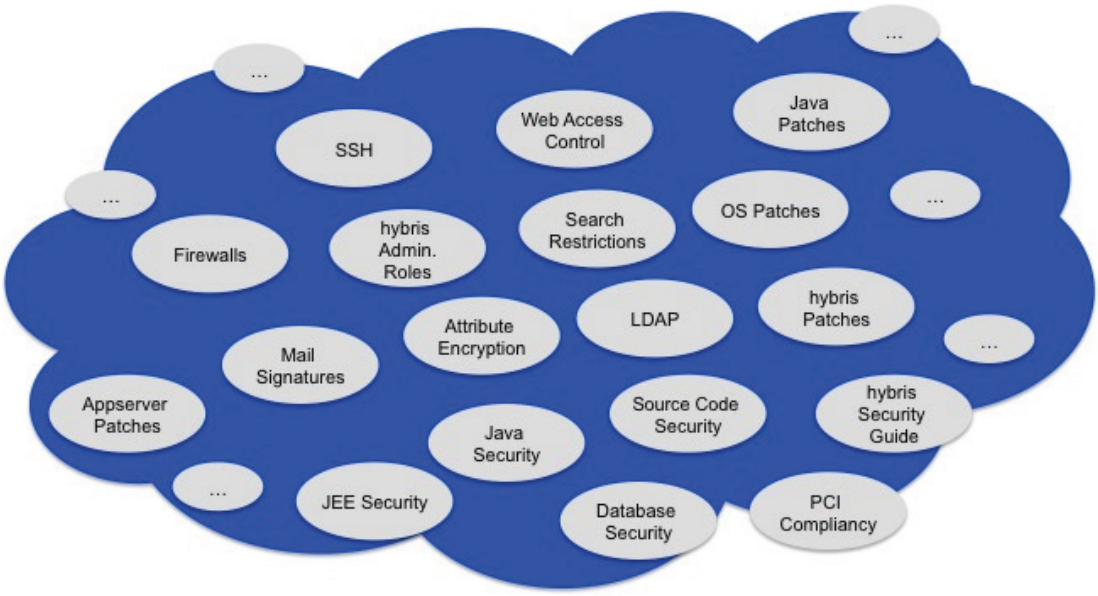


Figure 25: The results of the Oracle Exadata Proof of Concept Test

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. hybris places a huge emphasis on securing its software and addresses a number of areas including those illustrated below:

Figure 26:
Security Topics



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 hybris persistence model or a service in their ServiceLayer. Using the hybris Management Console (hMC), 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), hybris 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 hybris FlexibleSearch filters.

The FlexibleSearch service is a feature of 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 hybris supports the standard hashing and encryption practices to avoid this. But in addition, hybris can transparently encode certain highly-sensitive data before it even crosses the DAO layer to the database, with hybris Advanced Data Security Option. This contains a feature called hybris 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 hybris Management Console. 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 hybris 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

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

- the hybris Multichannel Suite 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), 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 requirements[17]. These are rather complex, time consuming and expensive, and consequently hybris recommends that you use a Payment Service Provider (PSP) that is

[17] http://www.pcicomplianceguide.org/

already compliant to this standard. Many hybris installations are already running with this kind of integration, which allows a comfortable one-click shopping experience. In this scenario, hybris stores a hash-code per user that is itself non-critical, yet provides unique identification with which hybris 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.

hybris software
THE AGILE RESPONSE

hybris AG
Binzstrasse 23
8045 Zürich
Switzerland

hybris AG Office France
168 avenue Charles
de Gaulle
92200 Neuilly sur Seine

hybris Germany
Nymphenburger Str. 86
D-80636 München

hybris Austria
Kirchengasse 48
1070 Wien

hybris UK
hybris UK Ltd.
5th Floor, 2 Copthall
Avenue
London, EC2R 7DA

hybris Benelux
Herengracht 574
1017 CJ Amsterdam

hybris (U.S.) Corp.
1 South Dearborn /
Suite 2100
Chicago, IL 60606

hybris Italia
Piazzale Biancamano, 8
20121 Milano Brera (MI),
Italia

hybris Canada
999 de Maisonneuve
Blvd West, 3rd Floor
Montréal, Québec,
Canada H3A 3L4

(y) Software AB
Fallhammargatan 8
72133 Västerås
Sweden