Hybris Security

This section seeks to help Hybris partners and customers manage their application security during the SAP Hybris Commerce implementation and deployment phases.

The scope of this section is the security of SAP Hybris Commerce web applications and the security of the components of the architecture such as the security of web servers, application servers, and databases. This does not include other aspects of security that are not related to the specific application, such as the security of the network infrastructure that supports the applications and constitutes a valued asset whose security properties such as confidentiality, integrity, and availability need to be protected as well.

## Default Passwords

Default SAP Hybris Commerce configuration comes with preconfigured set of users/passwords. These passwords should be disabled.

**Note** Hybris recommends that you disable the preconfigured setting of users passwords on production systems.

You should disable default passwords in property files. To do so:

1. Search for the default.login and default.password keywords in the .properties file.
2. Disable the default passwords and users that can be found.

###############################################################

# Disable default user/password for login cockpit pages and hmc

###############################################################

productcockpit.**default**.login=

productcockpit.**default**.password=

cmscockpit.**default**.login=

cmscockpit.**default**.password=

cscockpit.**default**.login=

cscockpit.**default**.password=

hmc.**default**.login=

hmc.**default**.password=

Change the Default Passwords for Administrative Users

Administrative users are created during initialization via ImpEx. The default passwords are stored in plain-text inside ImpEx, therefore anybody having access to the files or the source control system can read it. These passwords should be manually changed with the HMC after the initialization. Additionally, passwords should be different for each environment. It reduces the risk of password leakage and minimizes the impact if such a leakage should occur.

**Note** Hybris recommends that you change administrative users passwords manually after initialization. You should use different passwords on each environment.

**Note**The vjdbcReportsUser user (with access to HMC) default password is "1234" and is not visible in the HMC.

## Password Autocomplete

The login and password autocomplete feature is disabled on a global level in SAP Hybris Commerce. However, on the cockpit level, it is turned on causing the global settings to be overwritten by the ones specified in the extension's project.properties file.

productcockpit.**default**.login=productmanager

productcockpit.**default**.password=1234

You can disable a login and password autocomplete function on the cockpit level by deleting the values for the following properties in your extension's project.properties file:

# Default login and password for logging into your cockpit:

<yourcockpitname>.**default**.login=

<yourcockpitname>.**default**.password=

**Note**

Hybris recommends that you disable the Password Autocomplete feature during implementation phase.

## Password Encoding

You should configure the default password encoding. For historical reasons, the default strategy is plain text. This strategy is not safe and you should change it. Currently, the SAP Hybris Commerceplatform is shipped with the following secure encoding strategies:

* SHA-256
* SHA-512
* PBKDF2 - strongest algorithm

You can change default encoding strategy through a single property change:

#

# The code of the password encoder to use as default.

# These are the options (from core-spring.xml -> 'core.passwordEncoderFactory')

#

# '\*' .. legacy 'default' (not changeable before 5.7)

# 'plain' - plain text

# 'sha-256' - SHA 256

# 'sha-512' - SHA 512

# 'md5' - MD5

# 'pbkdf2' - PBKDF2 ( strong, configurable -> see below )

#

**Note**Hybris recommends that you use a secure encoding strategy from an early stage.

Changing the default strategy this way preserves previously stored passwords (using the "default" strategy) so that the user can still log in. In cases where the previous strategy was considered to be unsafe, all of the previous passwords remain unsafe. You should run a flex search query to find users with vulnerable passwords and passwords in clear text:

$passwordEncoding=sha-256

$defaultPassword=697AkzWKEkgS8Aqj9z1g

$setPassword=@password[translator=de.hybris.platform.impex.jalo.translators.ConvertPlaintextToEncodedUserPasswordTranslator][**default**='$passwordEncoding:$defaultPassword']

INSERT\_UPDATE Employee;UID[unique=**true**];$setPassword;

;admin;

;cmsmanager;

;cmsmanager;

;csagent;

;productmanager;

..

Since **vjdbcReportsUser** is a user, you have to change impex too.

$passwordEncoding=sha-256

$defaultPassword=697AkzWKEkgS8Aqj9z1g

$setPassword=@password[translator=de.hybris.platform.impex.jalo.translators.ConvertPlaintextToEncodedUserPasswordTranslator][**default**='$passwordEncoding:$defaultPassword']

INSERT\_UPDATE User;UID[unique=**true**];$setPassword;

;vjdbcReportsUser;

More information about encoding strategies can be found at [Password Storage Strategies](https://help.hybris.com/6.0.0/hcd/8c256c54866910149de4d53a72e973d4.html).

## Password Policy

Hybris recommends that you implement password policy that meets minimal requirements, such as:

* the use of both upper- and lower-case letters (case sensitivity), inclusion of one or more numerical digits,
* inclusion of special characters,
* prohibition of words found in a dictionary or the user's personal information,
* prohibition of passwords that match the format of calendar dates, license plate numbers, or other common numbers.

**Note**Hybris recommends that you implement the password policy for customers and business users during the implementation phase.

# Authorization

To protect your SAP Hybris Commerce instance and data, you need to make sure that only authorized parties have access.

## Managing and Checking Access Rights

By using the permissions services framework you can manage and check the access rights for users and user groups. You can assign, modify or remove permissions related to your users like employees and customers. You can also manage and check access rights assigned for the user groups.

The permissions services described in [Managing and Checking Access Rights](https://help.hybris.com/6.0.0/hcd/8c0ae05c8669101481d3cb6fb6bdec16.html) can be used to implement access rights system on the Hybris Platform. The framework has separate services for managing the access rights and checking what kind of access rights are assigned.

**Note**Access Rights should correspond to your security requirements, data model, and organizational structure, and must be defined during design of your application, as a part of a security policy.**Tip**Hybris recommends that you define the access rights system properly during the application design phase.

## Creating Permissions

Take into account that, once a permission is defined and used, there are no means to delete it. Such an operation would require checking the entire database to ensure that the permission is not used anywhere. This could also lead to a hole in the security system. Since permissions are only distinguished by name, removing a permission and then adding another one with the same name makes the existing security audit logs useless. Therefore plan your access rights system upfront and validate it before implementation.

## Checking Permissions

The [permission checking service](https://help.hybris.com/6.0.0/hcd/8c0ae05c8669101481d3cb6fb6bdec16.html#loio8c0ae05c8669101481d3cb6fb6bdec16__managingandcheckingaccessrights-permissioncheckingservice) provides the answer about assigned permissions. These questions can be specified for:

* Given permission
* Principal
* Object
* Item
  + Item type
  + Attribute descriptor

The answer is calculated out of the existing permission assignments and provides information about**actual permission assignments** (i.e. the permissions that were directly assigned). Permissions assigned indirectly are calculated by **permission checking rules**, which take into account permission inheritance within principal group hierarchy and inheritance within type and attribute descriptor hierarchy.

actual permission assignments + permission checking rules = effective permission assignments

**Effective permission assignments** might not physically exist in the system, but are the effect of using checking rules. For example, if there is an actual permission assignment to a user group, then every member of the group has that permission effectively assigned. This happens because user group permissions are inherited by users who belong to the group.

## Permission Checking Rules

Permission checking rules prioritize different kind of permission assignments. For example, global assignments have the lowest priority.

The result of checking permission assignment can be one of the following constants defined by**PermissionCheckValue**:

* **ALLOWED**: Permission is granted for a principal
* **DENIED**: Permission is explicitly denied for a principal
* **NOT\_DEFINED**: No actual permission assignment was found for the given permission checking operation
* **CONFLICTING**: Situation when there are both DENIED and ALLOWED equal-priority assignments for a principal

**PermissionCheckValue** should be mapped to a single boolean **yes** or **no** answer. Default mapping can be changed by using different PermissionCheckValueMappingStrategy. By default this strategy assumes that:

* Only ALLOWED maps to **true**
* Other values map to **false**

Permissions are not automatically enforced. This means that you have to assign them properly using**PermissionCheckingService** and then act accordingly in order to have a permission-based security.

## Permissions Scope

Generally, you can assign permissions to types, items, and attributes. Additionally you can assign global permissions to the users.

* **Global Permissions**: Related to the user or user group and is meant to have the lowest priority. It means that the global permissions assigned to the particular user/group can be overwritten by assigning permissions to the specific item, type or attribute. The global permissions apply in case there are no other permissions defined. Global permission assignments should be used as a **fall-back** or **default** assignment.

**Note**The global permissions have the lowest priority and are always overriden by the item, type, or attribute permissions assignments.

* **Type Permissions**: Apply to the entire type's data. Access rights to the type can be granted in several stages, for example **Read**, **Change**, **Create**, and **Delete**. When you assign permissions to the types. Every item of that type will inherit this permission assignments. This works across type hierarchy.
* **Item Permissions**: Can be assigned to the particular instance of the previously defined type. Simply speaking, if you can create many items of some type, this could allow you to override type-related access rights and assign permissions for a certain user to the particular concrete item.
* **Attribute Permissions**: Can be granted or denied explicitly for the individual attributes holding the informational content of a type or an item. This can be used for even more refined access control. Attribute-wide access rights can be granted in a few stages, for example: **Read** and**Change**. You should use the attribute descriptor assignments for fine-grained control of the attribute values.

Permission assignments can be either positive (allowed or granted) or negative (disallowed or denied).

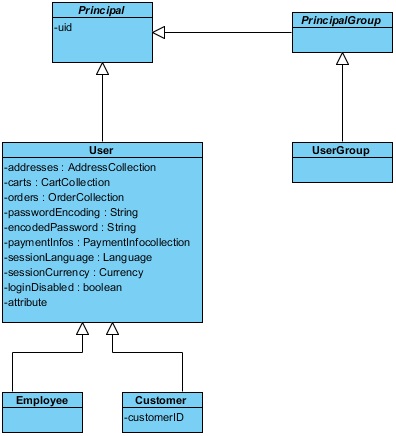
## Limitations of Permissions

* Effective permission assignments are calculated, and there is no easy way for the database server to perform such calculations when retrieving item rows from the database. This also means that permissions cannot be used for filtering rows in the data access layer.
* Permission assignments are currently not represented by models, you cannot directly import them using the ImpEx scripts. The only way is to use **PermissionManagementService** from ImpEx using scripting.

## Users in the SAP Hybris Platform

The Principal type is the main abstract class for user and usergroup types.

The following diagram gives an overview of Principal types.



The uid must be **globally unique** for all types that descend from the **Principal** type. A user and a group cannot have the same identifier, same applies for an employee and a customer.

### System Accounts

There are three special system user items, that are essential to the platform and cannot be modified nor deleted.

These are:

* **employee**: admin
* **customer**: anonymous
* **usergroup**: admingroup

Both the**anonymous** and **admin** users are protected from removal and against renaming.

You can add other users to the **admingroup**. Those users would have the same rights as the **admin**user.

**Caution**Be careful when using the **admin** user, because the Rule Framework is evaluated for all users, while it is not evaluated for the **admin** user, as the administrator has full access rights to everything within the system by default. There is no way to limit the access rights of the **admin**user.**Tip**Hybris recommends that you limit the access to the **admin** account and **admingroup** to members carrying out the admin role inside organization. Information concerning these accounts should be well protected.

## Users

You should use the **user** type to create more specific types that define accounts for your employees and customers. It is possible to create an item of a user type, but it should be avoided. Items of user type will have another permission hierarchy than items with type employee or customer and there will be no possibility to distinguish the role played by the items.

**Tip**Hybris recommends you to use the user type only in order to create more specific types that define accounts for employees, customers, or other more specific roles that need to be mapped in the organization hierarchy.

customer Type

Extended from the **user** type, the **customer** type is designed to be used for customers who visit your front-end application. Depending on your business context, customers can create their own accounts and have full or limited access to all their data.

Crucial customer data for ordering process, like address or payment information, are kept safe by being duplicated to the **Order** item. When a customer makes an order, the delivery address and payment information are copied to the **Order** item and are stored separately to secure that the shipment arrives at the correct address and is correctly paid. Even, if a customer changes payment information or delivery address after the order was made, the shipment is paid and delivered according to the information stored with the **Order** item.

**Caution**Customers are not allowed to manage SAP Hybris Commerce. It is technically possible to grant a customer account access to the SAP Hybris Commerce management tools. However this is not recommended and should not be allowed.**Note**By factory default, SAP Hybris Commerce has one customer: **anonymous**. Other customers may be created as the need emerges. You can let the customers create their own account through the front-end application, or you can do it yourself, manually or by using the ImpEx script to automate the process.

employee Type

Extended from the **user** type, the **employee** user can have access to the back office management tools and can perform several management actions as defined by the system of permissions and other regulations.

The **employee** user is a representation of a member of your company. This account is used to, depending on assigned permissions, manage objects in SAP Hybris Commerce.

A special kind of employee user is the **admin** user. This account cannot be removed, renamed, or restricted. Access rights framework checks if the user is **admin**, if yes, then no evaluation rule is checked to defined permissions for **admin**. User admin has access to everything within SAP Hybris Commerce. The **admin** user is special in that extent that it is member of the **admingroup** user group. This groups has no factory default limitations, nor restrictions.

You can create other employee users later as the need arises, manually or by using ImpEx scripts. By installing new cockpits, you also receive some factory default users that come together with particular cockpit extensions. There are several factory default employee accounts that come with theHybris system.

**Tip**You should change the credentials for all default accounts during the implementation phase, or these accounts will be disabled.

### Restrictions

Restrictions are rules obeyed by FlexibleSearch which allow you to limit search results depending on which type is searched and which user is currently logged in.

A [restriction](https://help.hybris.com/6.0.0/hcd/8c428f8286691014970ceee87aa01605.html) always applies to a specified type and a specified User or UserGroup. If the type has subtypes, then the subtypes are affected by the restriction as well. If the restriction applies to a UserGroup, then it affects all members of that UserGroup, including other UserGroups (whose members are affected by the restriction as well).

Restrictions work on FlexibleSearch queries only and do not affect the following use cases:

* External search engines like Lucene

Search results supplied by third-party search engines are not affected by restrictions. To get third-party search engine search results affected by restrictions, you will need to filter these search results by running a FlexibleSearch statement over them.

* Item.getProperty(), LocalizableItem.getLocalizedProperty() fetches item references directly via PK – any stored item will be returned no matter if it would have been filtered by a currently active restriction

## Restrictions and admin users

Restrictions do not apply to admin users (that is, users who are members of the **admingroup** user group). This means that a session assigned to a member of the **admingroup** user group is not affected by restrictions. This setting is in effect as long as the session is assigned to an admin user and applies to all restrictions.

By convention, restrictions do not apply to admin users (that is, users who are members of the**admingroup** user group). This means that a session assigned to a member of the admingroup user group is not affected by restrictions. This setting is in effect as long as the session is assigned to an admin user and applies to all restrictions.

## Restrictions for cronjobs

[Restrictions](https://help.hybris.com/6.0.0/hcd/8c428f8286691014970ceee87aa01605.html) can be used for [cronjobs](https://help.hybris.com/6.0.0/hcd/8b9ce4868669101499b2f0f25ef9395f.html" \o "The cronjob functionality is used for executing tasks, called cron jobs, regularly at a certain point of time. Typically cron jobs can be used for creating data for backups, updating catalog contents, or recalculating prices.). The effect is that you can limit the scope of items that the job can access. For example, you can explicitly exclude certain catalog versions from the scope of the SyncItemJob, with the effect that the SyncItemJob ignores these catalog versions.

The restriction attribute of the Job type can hold any number of JobSearchRestrictionModelobjects. Each object holds mainly the type, on which the restriction takes effect and the restricting query.

To allow access to attributes of the Job and CronJob, the SessionContext used within the Jobholds a reference to the Job and the CronJob via session.currentJob andsession.currentCronJob, respectively. For example, to ensure that a Job only uses the set catalog version, you could create a restriction using the following query:

{code} IS NULL AND {catalogversion}=?session.currentJob.targetversion

## Disabling Anonymous Login in Core

By default, the **anonymous** login cannot be used to log into the platform.

In the platform's advanced.properties by default there is:

#################### ANONYMOUS LOGIN SETTINGS #########################

# Fixedly disables the ability to log in using the 'anonymous' account.

# If you really need require using that account for login you must change

# this setting to 'false'!

#######################################################################

login.anonymous.always.disabled=**true**

This property ensures that the User.isLoginDisabled will always return true for the anonymous customer and User.checkPassword will always return false for the anonymous customer. Make sure that this property has not been overridden in the local.properties or project propertiesfiles.

## Secure Access to the Cockpits

Make sure that only Employees in the Employeegroup can access the cockpits and Customers in the customergroup can access the storefront.

The cockpit spring configuration enforces this through thedefaultCorePreAuthenticationChecks bean, defined in cockpit-spring-security.xml

<alias name="defaultCorePreAuthenticationChecks" alias="corePreAuthenticationChecks"/>

<bean id="defaultCorePreAuthenticationChecks" class="de.hybris.platform.spring.security.RejectUserPreAuthenticationChecks">

<property name="allowedUserGroups">

<list>

<value>Admingroup</value>

<value>Employeegroup</value>

</list>

</property>

<property name="allowedUserTypes">

<list>

<value>Employee</value>

</list>

</property>

</bean>

## Secure Access to Media

All media files, that can contain any personal data should be placed in secured media folders.

Placing media in a secured media folder doesn't make it secure without the proper configuration.

The persisted data of a media will be stored in an unprotected media folder when using the default settings.

Permissions work on several levels in the platform. Having read access to a media is granted if:

* a principal has a positive read permission on it (what you have set up), or
* a principal has a positive read permission on the Media type (which is mandatory to view medias in general in HMC or Backoffice), or
* a principal has a global positive read permission (without type)

The secure media folder does work correctly if permissions are set up correctly.

For scenarios where only a specific principal should have access to specific media, permissions granted on the type level have to be overwritten by:

* Assigning positive read permissions for the principal who should see it
* Assigning positive read permissions for all the principal groups which have positive read permissions on Media types but should not have access to specific media.

**Tip**You should secure media access properly during the implementation phase.

# Secure Configuration

Prior to deploying your SAP Hybris Commerce installation into production, you should make sure the configuration is secure. Here you can find some tips on how to do so.

## Review Installed Extensions

Review and disable unnecessary and risky extensions (e.g. virtualjdbc, ws410, testweb). Additionally verify all loaded web modules and remove any that are not needed. Adjust extensioninfo.xml to remove the web module entry for extensions that do not require a web extension.

## Tenant Settings

Make sure that junit tenant is not activated on the production environment.

**Tip**You should make sure that junit tenant is not activated on the production environment.

## Secure Backoffice Extensions

Secure backoffice extensions from internet access. Access to backoffice applications should be limited to appropriate group of users inside your organization. Backoffice applications should be deployed within a secured network and protected from unauthorized access.

**Tip**You should make sure that backoffice extensions are accessible only from the internal network.

## Application Server User

The SAP Hybris Commerce application server should not run as a root user. Check the RUN\_AS\_USER variable in hybris/bin/platform/tomcat/bin/wrapper.sh.

**Tip**You should make sure that the application server is not running as root.

## Database User Settings

Particular database users associated with the storefront should have limited permissions. A proper configuration of Hybris Commerce is defined as follows:

* Front-end, DB user should have SELECT, INSERT, UPDATE and DELETE rights.
* Back-end, DB user should additionally have CREATE and ALTER TABLE rights. These are needed for application deployments or updates.

**Note**You should configure the DB users during the implementation phase.

Database connection details and JNDI

When using local.properties for DB connection configuration, DB password and other informations are stored in plain text. You can avoid this situation using JDNI for DB configuration.

In order to use a JNDI datasource you have to:

1. Define the datasource in the Application Server you use. How you define the datasource depends on the Application Server you use. Check the instructions specified by the manufacturer of the Application Server.
   * [JNDI Resources HOW-TO for Tomcat 8](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/jndi-resources-howto.html)
   * [WebLogic JNDI](http://help.sap.com/disclaimer?site=https://docs.oracle.com/cd/E13222_01/wls/docs81/jndi/jndi.html)
2. Configure Hybris Commerce to use the JNDI datasource
3. Configure the name of the JNDI datasource using the config property in local.properties.

db.pool.fromJNDI=<JNDI datasource name>

1. Then call ant deploy and start the Application Server.

Parameters used during JNDI configuration are XMLENCODED, therefore you should encode all characters which are not allowed in XML but which are part of the configuration (passwords and URLs often contain characters which need to be escaped in XML).

Example: If a password contains a "&" you need to encode it as "&", otherwise there will be an XML parse exception.

## Platformwebservices

WebServices related to platformwebservices are considered a back office management tool. Therefore platformwebservices should be secured in the same way as backoffice applications.

**Tip**You should make sure that Platformwebservices are secured like a backoffice application.

Webservice User and Usergroup

Webservices will adopt the default Access Manager Security Strategy. In platformwebservices-web-spring.xml:

<bean id="abstractSecurityStrategy" class="de.hybris.platform.webservices.AbstractSecurityStrategy" abstract="true" scope="prototype">

<property name="modelService" ref="modelService"/>

</bean>

<bean id="securityStrategy" class="de.hybris.platform.webservices.AccessManagerSecurityStrategy" scope="prototype" parent="abstractSecurityStrategy"/>

This strategy authorizes a user only if they are in a new security group named webservicegroup. This is not set up by default, so be sure to create this group for your RESTful calls to be authorized. The name of this user group is configurable in the platformwebservices/project.properties file:

webservices.security.**group**=webservicegroup

For more information on WebServices security, see [WebService API - Security Architecture](https://help.hybris.com/6.0.0/hcd/8c85126186691014ae4bea5dee5d5901.html" \o "The security mechanism underlying the Hybris WebService API is based upon Spring's widely adopted and stable security framework. This ensures a consistent approach with the rest of Hybris security architecture, which is also based upon Spring.).

## Solr

It is not recommended to run the embedded Solr server in the production environment. Use solrfacetsearch and, when installing the Apache Solr, bear in mind the security considerations described in [SolrFacetSearch - Installation Guide](https://help.hybris.com/6.0.0/hcd/8c6035a086691014a42ae443f651eedc.html" \o "Information about the installation of solrfacetsearch extension). Details on hardening Apache Solr can be found in then official [Solr Security documentation](http://help.sap.com/disclaimer?site=http://wiki.apache.org/solr/SolrSecurity" \o "http://wiki.apache.org/solr/SolrSecurity" \t "_blank).

**Tip**You should make sure that the embedded Solr server is not running in the production environment. Solr hardening should take place during the implementation phase.

## FlexibleSearch

FlexibleSearch enables searching by types and items using an SQL-based syntax. The execution of a FlexibleSearch statement takes place in two phases: pre-parsing into an SQL-compliant statement and running that statement on the database. During the pre-parsing phase, the FlexibleSearch framework resolves the FlexibleSearch syntax into SQL-compliant syntax.

When using [FlexibleSearch](https://help.hybris.com/6.0.0/hcd/8bc399c186691014b8fce25e96614547.html" \o "SAP Hybris Commerce comes with a built-in query language of a SQL-based syntax, FlexibleSearch. FlexibleSearch enables searching over the items in SAP Hybris Commerce.), never build a query by concatenating the query code and the data.

To pass parameters, create a Map instance holding the parameters and pass the Map to the search( ... ) method, as in:

**final** Map<String, Object> **params** = **new** HashMap<String, Object>();

String query =

"SELECT {" + PriceRowModel.PK + "} FROM {" + PriceRowModel.\_TYPECODE "} "+

"WHERE {" + PriceRowModel.PRODUCT + "} = ?product AND "+

"{" + PriceRowModel.NET + "} = ?net AND "+

"{" + PriceRowModel.CURRENCY + "} = ?currency AND "+

"{" + PriceRowModel.UNIT + "} = ?unit AND "+

"{" + PriceRowModel.UNIT\_FACTOR + "} = ?unitfactor AND "+

"{" + PriceRowModel.UG + "} = ?userpricegroup AND "+

"{" + PriceRowModel.MIN\_QUANTITY + "} = ?minquantity AND "+

"{" + PriceRowModel.PRICE + "} = ?price ";

**params**.put("product", product);

**params**.put("net", priceCopy.isNet());

**params**.put("currency", priceCopy.getCurrency());

**params**.put("unit", priceCopy.getUnit());

**params**.put("unitfactor", priceCopy.getUnitFactor());

**params**.put("userpricegroup", priceCopy.getUserPriceGroup());

**params**.put("minquantity", priceCopy.getMinQuantity());

**params**.put("price", priceCopy.getPriceValue());

**final** SearchResult<PriceRowModel> searchResult = flexibleSearchService.search(query, **params**);

## Lock the System for Initialization

By default Hybris Commerce isn't locked for system initialization. System initialization destroys tables in the database and recreates them. Locking initialization disables the initialization button in Hybris Administration Console so that initialization cannot be triggered accidentally.

The following steps are necessary to lock the system initialization. These steps should be performed after every deployment.

* Lock the system for initialization and update:

Set the system.unlocking.disabled property to true (this disables the *Unlock* button of the system configuration tab in the Hybris Administration Console).

Tto unlock it before the deployment:

1. Set system.unlocking.disabled to false, without restarting Hybris.
2. Click on *Unlock*.

## Transparent Attribute Encryption (TAE)

TAE provides the option to encrypt sensitive data before persisting in the database with AES encryption. It's done by declaring a String-Attribute as encrypted by adding the modifier encrypted="true" in the items.xml of your extension.

<itemtype generate="true" code="MyProduct" jaloclass="org.training.jalo.MyProduct"

extends="Product" autocreate="true">

<attributes>

<attribute qualifier="myEncryptedField" type="java.lang.String" >

<modifiers read="true" write="true" optional="true" encrypted="true"/>

<persistence type="property" />

</attribute>

</attributes>

</itemtype>

This attribute will be automatically encrypted before persisting in the database and decrypted after loading. The corresponding database field will look like:

1:LFGjqOi4quc=K8p+AIJJBoIQTPBh7yaSojr97rHES7YaTkYO6SqBA/M=

The number before the colon ( : ) refers to the number of keys used in the encryption process, so that it can be used to decrypt the data. Since AES is a symmetric encryption, the same key is used to both encrypt and decrypt messages. Encrypted data is safe as long as the secret-key is protected and uncompromised. You can find an AES key generator for 128, 192 and 256 bits key in the Hybris Administration Console. More information on how to generate, change and migrate Encryption Keys can be found in [Transparent Attribute Encryption (TAE)](https://help.hybris.com/6.0.0/hcd/8b2c75c886691014bc12b8b532a96f58.html).

**Tip**You should make sure to rotate keys used for encryption at least once a year.

Configuration

By default, the Hybris Platform ships with a 128-bit key, which can be found in:PLATFORMHOME/core/resources/security

The default symmetric key should be replaced with a new one using a generator provided by Hybris Commerce.

**Tip**You should replace the default symmetric key during the implementation phase.

The key's configuration is stored in the project.properties file and should be overwritten in thelocal.properties file.

* symmetric.key.file.<KEY\_NUMBER> is the property that points to the key file. All keys should be in the config/security folder.
* symmetric.key.file.default property defines the currently used key.
* symmetric.key.master.password is the password used to protect the key files.

Sample configuration:

symmetric.key.file.1=Generated-128-Bit-AES-Key.hybris

symmetric.key.file.2=Generated-192-Bit-AES-Key.hybris

symmetric.key.file.**default**=2

symmetric.key.master.password=ShItN7yqJYMvNfOODmS26DcwB5OXnOa3etCzqxRjFMZ811UUYc

It is recommended to create a longer password and use bigger key sizes, such as 256-bit. For this, unlimited jurisdiction policy files have to be downloaded from the Oracle site and installed inJAVA\_HOME/lib/security.

Key Rotation

The keys should be changed at least once a year and then the existing values can be migrated to the new key. We suggest to migrate all existing values. If not migrated, they will be translated to the new key on the first reference. However, if a key has to be completely removed, then migration is necessary. At the beginning, all sensitive data will be encrypted with the key that is specified by the setting symmetric.key.file.default (key id = 1).

project.properties/local.properties

# all keys have to be stored under: ${platform\_config\_dir}/security/

symmetric.key.file.1=before\_migration\_key.hybris

symmetric.key.file.**default**=1

symmetric.key.master.password=ShItN7yqJYMvNfOODmS26DcwB5OXnOa3etCzqxRjFMZ811UUYc

To make this encryption key unique for a specific Hybris installation this one is protected by thesymmetric.key.master.password.

During key rotation a new key is added and set as the default as shown in the sample configuration below.

project.properties/local.properties

**Sample Code**

# all keys have to be stored under: ${platform\_config\_dir}/security/

symmetric.key.file.1=before\_migration\_key.hybris

symmetric.key.file.2=after\_migration\_key.hybris

symmetric.key.file.**default**=2

symmetric.key.master.password=ShItN7yqJYMvNfOODmS26DcwB5OXnOa3etCzqxRjFMZ811UUYc

We specify a new encryption key (id=2) and by setting symmetric.key.file.default=2 this key will be used for all new **encryption** operations.

Encrypted attributes will only be re-encrypted with the new key if the corresponding instance is loaded and restored (for example by opening the instance in the HMC). To encrypt all encrypted attributes with the new key, use the Hybris Administration Console. See also the Encryption Keys Migrationsection below.

Encryption Keys Migration

Encryption key migration importance and frequency depends on your security policy. It may also become necessary as an emergency measure in the case where existing keys have become unsecure or were compromised.

1. Open the Hybris Administration Console.
2. Go to the *Maintenance* tab and select the *Encryption Keys* option.
3. The *Encryption Keys* page in the *Generator* tab displays. If you did not configure your migration key in project.properties or local.properties file, then instructions on how to do it would display. Otherwise you can see the fields below:
   * *Encryption Keys*: List of the configured encryption keys
   * *Encrypted Attributes*: All types with their encrypted attributes are listed. You see the total count of encrypted instance (#15) and how often every key is used (id=1, #15). Every selected row is encrypted with the new default key after clicking the Migrate button.

Figure: Encryption Keys page in the Migration tab.

Tips and Pitfalls

* dontOptimize=true

Because you will not be able to search for encrypted values, it could be a good idea to add the modifier "dontOptimize=true" in your items.xml for the encrypted attribute.

* DB Persistence Type

Encrypting a string value increases its length, so be sure that you have chosen the right persistence type (VARCHAR, CLOB, TEXT) for storing your encrypted values.

* java.security.InvalidKeyException: Illegal key size

The Java Runtime Environment (JRE) must support the level of encryption required by the key length that you select. For example, you cannot use a 256-bit encryption key with a JRE that supports only 128-bit encryption. By default, the Platform is configured with restricted or limited strength ciphers. To use less restricted encryption algorithms, you must download and apply the unlimited jurisdiction policy files (local\_policy.jar and US\_export\_policy.jar).

* Cluster Configuration

For guaranteing the integrity of your data, you have to be sure that every node is using the same encryption key. A general default key is part of the release and is stored at$platformhome/bin/platform/ext/core/resources/security/default-128-bit-aes-key.hybris. But Hybris highly recommends, that you replace this weak and unsecure keyfile with your own one, which has to be placed in $platform\_config\_dir/security/.

Download for Java8: [http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html](http://help.sap.com/disclaimer?site=http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html)

## XSS Encoder

The security.core.server.csi-1.00.8.jar library encodes different kinds of content and helps prevent XSS attacks from malicious data being stored inside the Platform database. The library was added to the hybris ext/core/lib folder. The following methods are available to prevent XSS.

| **Context** | **Method** |
| --- | --- |
| CSS | out = XSSEncoder.encodeCSS( val ) |
| HTML/XML | out = XSSEncoder.encodeHTML( in ) andXSSEncoder.encodeXML( val ) |
| JavaScript | out = XSSEncoder.encodeJavaScript( val ) |
| URL | out = XSSEncoder.encodeURL( val ) |

## XSS Filter

XSS Filter is a generic cross-site scripting protection filter for the Hybris Platform.

The filter is part of the platform global classpath. Therefore there is no need to add any library to your platform web application.

Configuring XSS Filter

XSS is configured by specific properties included in platform/project.properties

########################### WEB-SECURITY SETTINGS ######################################

#

# Here web related security settings can be found.

#

########################################################################################

# enable globally

xss.filter.enabled=**true**

# define action on violation matching globally

# STRIP .. strips all text occurrences which match the patterns below but allow

# processing the request (default)

# REJECT.. if any pattern matches the whole request gets rejected with the BAD REQUEST

# error code

xss.filter.action=STRIP

# our default rules

xss.filter.rule.script\_fragments=(?i)<script>(.\*?)</script>

xss.filter.rule.src=(?ims)[\\s\r\n]+src[\\s\r\n]\*=[\\s\r\n]\*'(.\*?)'

xss.filter.rule.lonely\_script\_tags=(?i)</script>

xss.filter.rule.lonely\_script\_tags2=(?ims)<script(.\*?)>

xss.filter.rule.**eval**=(?ims)**eval**\\((.\*?)\\)

xss.filter.rule.expression=(?ims)expression\\((.\*?)\\)

xss.filter.rule.javascript=(?i)javascript:

xss.filter.rule.vbscript=(?i)vbscript:

xss.filter.rule.onload=(?ims)onload(.\*?)=

The filter can be disabled / enabled globally with xss.filter.enabled parameter. By default, the filter is enabled and Hybris recommends that it remains enabled.

xss.filter.enabled=**true**

The filter can be disabled if there is an application firewall in front of the Hybris Commerce instance or if there is another solution dealing with malicious input.

The filter can be disabled/enabled per extension by adding the extension name as a prefix to the parameter:

hac.xss.filter.enabled=**false**

hmc.xss.filter.enabled=**true**

Actions on Match

The default reaction on matching any rule is that each occurrence of these text fragments are stripped and the request proceeds.

xss.filter.action=STRIP

There is also possibility to change that setting to reject the request with the error code BAD REQUEST:

xss.filter.action=REJECT

Action can be set also for a specific extension by adding extension name as a prefix to the parameter:

hmc.xss.filter.action=REJECT

Rules

**Tip**You should use the default rules as a basic set and expand upon them within a specific project to match the security requirements defined for input validation.

Rules are globally defined as properties. A rule property must start with xss.filter.rule in its name, followed by a specific rule id, which is used for information purposes only.

xss.filter.rule.script\_fragments=(?i)<script>(.\*?)</script>

xss.filter.rule.src=(?ims)[\\s\r\n]+src[\\s\r\n]\*=[\\s\r\n]\*'(.\*?)'

xss.filter.rule.lonely\_script\_tags=(?i)</script>

xss.filter.rule.lonely\_script\_tags2=(?ims)<script(.\*?)>

xss.filter.rule.**eval**=(?ims)**eval**\\((.\*?)\\)

xss.filter.rule.expression=(?ims)expression\\((.\*?)\\)

xss.filter.rule.javascript=(?i)javascript:

xss.filter.rule.vbscript=(?i)vbscript:

xss.filter.rule.onload=(?ims)onload(.\*?)=

Rules can be overridden per extension. Make sure to match the exact rule id and add the **extension name** as a prefix to the parameter:

hac.xss.filter.rule.vbscript=

Configuration in web.xml

To enable the filter you need to change web.xml files for each web application. Make sure that the filter:

1. Matches all requests
2. Comes first in the overall filter chain

<filter>

<filter-name>XSSFilter</filter-name>

<filter-class>de.hybris.platform.servicelayer.web.XSSFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>XSSFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

Servlet 3.0

In Servlet 3.0 containers (from tomcat 7), there may be servlets making use of asynchronous processing. In this case, configure the filter to support it by adding <async-supported>true</async-supported>:

<filter>

<filter-name>XSSFilter</filter-name>

<filter-class>de.hybris.platform.servicelayer.web.XSSFilter</filter-class>

<async-supported>true</async-supported>

</filter>

## CSRF Protection

Hybris Commerce provides a CSRF protection mechanism, that is enabled by default.

This solution is based on Spring CSRF protection introduced in Spring 3.1.**Caution**

CSRF protection works properly as long as:

* There is no XSS Vulnerability in the application
* A POST HTTP Method is used for any non-readonly operation

**Tip**You should enable CSRF protection on all productive environments.

In order to confirm that CSRF protection is enabled check that:

* CSRFHandlerInterceptor bean is activated (in spring-mvc-config.xml, in the<mvc:interceptors> tag
* <!-- CSRF Handler Interceptor -->
* <bean name="csrfHandlerInterceptor" class="de.hybris.platform.yacceleratorstorefront.util.CSRFHandlerInterceptor">
* <property name="csrfAllowedUrlPatterns" ref="csrfAllowedUrlPatternsList"/>
* </bean>
* <mvc:interceptors>
* ...

<ref bean="csrfHandlerInterceptor" />...

* CSRFRequestDataValueProcessor alias to requestDataValueProcessor is not commented

<bean name="crfRequestDataValueProcessor" class="de.hybris.platform.yacceleratorstorefront.util.CSRFRequestDataValueProcessor"/>

More information about CSRF and CSRF mitigation can be found at:

* [OWASP Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet](http://help.sap.com/disclaimer?site=https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet)

## Clickjacking Protection

Hybris Commerce provides a Clickjacking protection mechanism, which is based on the XSS filter. This general feature will inject X-Frame-Options header to all requests of web application that are using the XSS filter. The possible header values are:

* DENY - will block any site (regardless of the domain) from framing the content.
* SAMEORIGIN - will block all sites from framing the content, except sites within the same domain.

The header value can be configured in platform/project.properties.

# ----------------------------------------------------------------

# inject static http response headers

# (again this can be done globally or per extension )

# ----------------------------------------------------------------

#

# setting 'X-Frame-Options=SAMEORIGIN' to prevent clickjacking attacks

xss.filter.header.X-Frame-Options=SAMEORIGIN

Same as for XSS filter, rule can be override per extension by adding extension name as a prefix to the parameter:

hac.xss.filter.header.X-Frame-Options=DENY

## JMX Settings

By default, the JMX connection to Hybris Commerce is not secured and not encrypted.

**Note**

For production systems, JMX is:

* Using [SSL client certificates](http://help.sap.com/disclaimer?site=http://docs.oracle.com/javase/8/docs/technotes/guides/management/agent.html#gdepp) for authentication instead of passwords, or
* The client obtains the remote connector object securely, for example through a secure LDAP server or a file in a shared secure file system.

See the official JMX documentation: [Using SSL](http://help.sap.com/disclaimer?site=http://docs.oracle.com/javase/8/docs/technotes/guides/management/agent.html#gdemv) and [Using LDAP Authentication](http://help.sap.com/disclaimer?site=http://docs.oracle.com/javase/8/docs/technotes/guides/management/agent.html#ldap) for more information.

Login and password configuration

To configure a login/password, edit the local.properties and add the following parameters to JVM startup arguments:

tomcat.generaloptions=...

-Dcom.sun.management.jmxremote.authenticate=**true**

-Dcom.sun.management.jmxremote.password.file=$CATALINA\_HOME/conf/jmxremote.password

-Dcom.sun.management.jmxremote.access.file=$CATALINA\_HOME/conf/jmxremote.access

Then add the following files in $CATALINA\_HOME/conf:

* jmxremote.access - the access control file defines the permitted access for each role.
* jmxremote.password - the password file.

**Password Files**

Only the owner should have read and write permissions on this file (it contains the passwords in clear text). For security reasons, the system checks that the file is only readable by the owner and exits with an error if it is not.

Property names are roles, and the associated value is the role's password.

**Sample Code**

monitorRole passwordForMonitorRole

adminRole passwordForAdminRole

On Solaris, Linux, or Mac OS X operating systems, you can set the file permissions for the password file by running the following command. chmod 600 jmxremote.password

For instructions on how to set file permissions on a Windows platforms, see [Java JMX: Additional Security Information For Microsoft Windows](http://help.sap.com/disclaimer?site=http://docs.oracle.com/javase/8/docs/technotes/guides/management/security-windows.html).

**Access Files**

The access file defines the roles and their access levels. Two primary roles are defined by default:

* monitorRole - grants read-only access
* controlRole - grants read-write access

An access control entry consists of a role name and an associated access level. The role name cannot contain spaces or tabs and must correspond to an entry in the password file. The access level can be either readonly or readwrite.

A role should have only one entry in the access file. A role without an entry, has no access. For a role with multiple entries, the last entry takes precedence.

**Sample Code**

monitorRole **readonly**

adminRole readwrite

SSL Client Authentication

SSL client authentication should be used on production environment. To enable it, set the Java VM system property:

com.sun.management.jmxremote.ssl.need.client.auth=**true**

Information on how to generate a certificate and configure SSL on the client system can be found in[Java JMX: Using SSL](http://help.sap.com/disclaimer?site=http://docs.oracle.com/javase/8/docs/technotes/guides/management/agent.html#gdemv).

**Note**By default, JMX communication is not encrypted. It can be encrypted by setting the following parameter to the JVM startup arguments: -Dcom.sun.management.jmxremote.ssl=true

Tomcat

The Tomcat instance should be hardened.

**Tip**You should harden the Tomcat instance (if Tomcat is used) during the implementation phase.

HTTP Header Security Filter

There are a number of HTTP headers that can be added to the response to improve the security of the connection. With Tomcat comes the HTTP Header Security Filter, which provides a mechanism for adding those headers. Security related headers with more complex requirements, like CORS, are implemented as separate Filters.

Hybris recommends that you set:

* blockContentTypeSniffingEnabled - this option will set X-Content-Type-Options header

More information can be found in the [Apache Tomcat 8 Configuration documentation.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

# [Security Features](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[Some security features are included in SAP Hybris Commerce, such as the ldap extension.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

## [LDAP](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[The Hybris ldap extension allows authentication of users and user groups via the LDAP protocol and the import and/or synchronisation of data which is stored on a LDAP server.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

* [Supported authentication modes are:](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)
* [Anonymous](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)
* [User and password](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)
* [Anonymous, SSL-encrypted](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)
* [User and password, SSL-encrypted](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)
* [SASL and keystore password, SSL-encrypted](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)
* [The Hybris ldap extension gives you the chance to implement a Single-Sign-On concept.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)
* [ImpEx-based import of LDIF files and search results (LDAP query language).](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[More information about ldap extension can be found in](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)[[ldap Extension](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)](https://help.hybris.com/6.0.0/hcd/8c03cf3f866910149eaa8b410d09cd58.html)[.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[The use of a central authentication system helps to manage password policy and account status.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

# [Unsecure Components](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[SAP Hybris Commerce contains unsecured components that you should secure or remove prior to moving into production.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

## [Test and Debug Pages](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[Test and debug pages are unsecured by definition and should be disabled in production.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[debugFooter.tag](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[debugFooter.tag that leads to XSS.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[<%-- Debug footer. Not](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank) **[for](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)** [production. Outputs](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank) **[in](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)** [an HTML comment --%>](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[<c:if test="${showStorefrontDebugInfo}">](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[<!-- TODO: Remove From Production](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[DEBUG INFO](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[...](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[This footer should be removed from production.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

**[Tip](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)**[You should not have any test or debug pages in production.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

## [Mocks](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[Mocks provided by Hybris Commerce are not considered as secure and therefore should not be used in production.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

**[Tip](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)**[You should not have any mocks in production.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[Accelerator Payment Mock](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[Accelerator Payment Mocks are unsecured and have known security vulnerabilities like reflected XSS.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[Accelerator Payment Service Provider Mock](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[Accelerator Payment Service Provider Mocks are unsecured and have known security vulnerabilities like open redirect and reflected XSS.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

## [Templates](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)

[Extensions generated using Hybris Commerce templates are not secured. These extensions should be validated and secured during the implementation phase.](http://help.sap.com/disclaimer?site=https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html" \l "HTTP_Header_Security_Filter" \o "https://tomcat.apache.org/tomcat-8.0-doc/config/filter.html#HTTP_Header_Security_Filter" \t "_blank)