

OpenText™ DAMLink for SAP® Solutions Hybris Integration

Installation and Configuration Guide

This guide describes the installation and configuration of OpenText DAMLink for SAP Solutions Hybris Integration. The OpenText DAMLink for SAP Solutions Hybris Integration component integrates OpenText Digital Asset Management for SAP Solutions (DAM) in Hybris Product Content Management (PCM), in the Web Content Management Module (CMS Cockpit), and in Hybris Backoffice.

DAMSAP160201-IHI-EN-03

OpenText™ DAMLink for SAP® Solutions Hybris Integration Installation and Configuration Guide

DAMSAP160201-IHI-EN-03

Rev.: 2017-Dec-18

This documentation has been created for software version 16.2.1.

It is also valid for subsequent software versions as long as no new document version is shipped with the product or is published at <https://knowledge.opentext.com>.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111

Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440

Fax: +1-519-888-0677

Support: <https://support.opentext.com>

For more information, visit <https://www.opentext.com>

Copyright © 2017 Open Text. All Rights Reserved.

Trademarks owned by Open Text.

Disclaimer

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, Open Text Corporation and its affiliates accept no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Table of Contents

1	Overview	7
1.1	System landscape and components	8
1.2	Workflow	10
1.3	Further documentation sources	11
2	Installation and basic configuration	13
2.1	System Prerequisites	13
2.2	Installing the DAMLink OTMM Add-on	14
2.3	Multiple Hybris systems	14
2.4	Installing DAMLink Hybris Extensions	15
2.5	Installing Media Delivery Service	20
2.6	Installing language packs	21
2.7	Upgrading	22
2.7.1	Upgrading Media Delivery Service	22
2.7.2	Upgrading DAMLink Hybris Extensions	23
2.8	Testing the integration	26
2.9	Registering DAMLink Hybris in the SAP System Landscape Directory	30
3	Advanced configuration	33
3.1	Configuring assignment spec and asset delivery	33
3.1.1	Configuring an asset delivery	35
3.1.1.1	Media Delivery Service type	37
3.1.1.2	ContentByExport type	41
3.1.1.3	External URL type	41
3.1.1.3.1	URL template string placeholders	42
3.1.2	Configuring an assignment spec	43
3.2	Online property changes	45
3.3	Exporting and importing an assignment configuration	45
3.4	Using Media Delivery Service	46
3.4.1	Rendition production	47
3.4.2	Configuring properties	48
3.4.3	Sending requests	51
3.4.4	Using command line tools	52
3.4.5	Using cloud storage	56
3.4.6	Version info and logging	57
3.5	Configuring Hybris Product Cockpit editor	58
3.5.1	Using the default editor configuration	59
3.5.2	Defining a new editor configuration	60
3.6	Configuring Hybris WCMS cockpit editor	61
3.6.1	Registration of OpenText editors for CMS component types	62

3.7	Configuring Hybris Backoffice	63
3.7.1	Customizing Media Reference Editors in otmmaddonbackoffice configuration	65
3.8	Configuring smartedit	66
3.9	Setting up an individual Asset Assignment Dialog	67
3.10	Scheduling asset assignment with cron jobs	69
3.10.1	Check assignment status	70
3.10.2	Cronjob details	71
3.10.3	Scheduling cron jobs	73
3.10.4	OtmmExportServiceJob	74
3.10.5	OtmmAssetSyncJob	76
3.10.6	OtmmAssociationPurgeJob	77
3.11	Automated asset assignment with cronjob (AutoAssignJob)	78
3.11.1	Prepare your OTMM metadata	79
3.11.2	Configuring Hybris properties	82
3.11.3	Setting up the AutoAssignment cronjob (AutoAssignJob)	84
3.12	Metadata synchronization	86
3.13	Configuring automatic update to latest version	88
3.14	Configuring asset purge notifications	92
3.15	Configuring assignment of approved assets	92
3.16	Thumbnails for non-image assets	93
3.17	Product lookup	94
3.17.1	Example for product lookup with AutoAssignment	95
3.17.1.1	Configuration of the example	95
3.17.2	Configuring product lookup	97
4	Security	103
4.1	Configuring OTMM and FTP connection	103
4.2	SSL support	105
4.3	Single sign-on between Hybris and OTMM	107
4.4	Managing Media Delivery Service user permissions	109
4.5	Shared secret of Media Delivery Service and DAMLink Hybris	109
4.6	Thumbnail related security	110
4.7	HTTPS and CORS	111
4.8	Securing notifications	111
4.9	Restricting CORS requests to specific host	112
4.10	Log forging prevention	112
5	Operations	113
5.1	Administration tools	113
5.2	Identify the version and build number	113
5.3	High availability	114
5.4	Starting and stopping the system	114

5.5	Periodical tasks	114
5.6	Backup and recovery	114
5.7	Monitoring	114
5.8	Logging	115
5.8.1	Configure Hybris log	115
5.8.2	Configure SAP LogViewer compliant logging	117
5.8.3	Logging on OTMM	118
5.9	Data consistency	118
5.10	Software maintenance	119
5.11	Troubleshooting	119
5.11.1	Login to OTMM failed	120
5.12	Keyboard accessibility	120
6	Example: project.properties	121
7	Appendix	137
7.1	Content types and MIME types	137

Chapter 1

Overview

OpenText Digital Asset Management (DAM) solutions serve as centralized, secure and accessible repository to manage digital media, branding and video. Scalable for large enterprises it offers capabilities for advanced end-to-end workflows and features for creative production, review, approval, publishing and distribution of assets.

With **OpenText DAMLink for SAP Solutions** components (sold by SAP as part of **SAP Hybris Digital Asset Management by OpenText**), you are able to integrate your content supply chain with your business processes. Digital Asset Management for SAP Solutions enables you to find, manage, and distribute your digital assets, make your content supply chain transparent, and maintain control of your digital assets - all in context of your business processes.

The **OpenText DAMLink for SAP Solutions Hybris Integration** component integrates OpenText Digital Asset Management for SAP Solutions (DAM) in Hybris software.



Note: For brevity, **OpenText DAMLink for SAP Solutions Hybris Integration** is referred to as **DAMLink Hybris** throughout this guide.

*Involved
products*

DAMLink Hybris can be integrated into the following Hybris software components:

- **Hybris Product Content Management (PCM)**
Hybris, an SAP company, is a leading e-commerce software company, delivering enterprise software and on-demand solutions for e-commerce, multi-channel commerce, product catalog management, master data management and order management. PCM is the Hybris master data management product for commerce which delivers consolidation and centralized management of product information and attributes across all channels.
- **Hybris Web Content Management (WCMS)**
The Hybris Web Content Management is a marketing and publishing tool with Web content management capabilities, fully in the Product Content Management (PCM) and the multi-channel commerce functions are integrated.
- **Hybris smartedit**
Hybris smartedit is the successor component for Hybris WCMS and is fully supported by DAMLink Hybris. DAMLink Hybris adds cropping functionality to Hybris smartedit.
- **Hybris Backoffice Admin Area**
The Hybris Backoffice Admin Area is an administrative tool to help you run your business efficiently. It enhances the capabilities of the Backoffice UI, providing ready-to-use, reusable components that enable you to build custom business tools tailored to specific user needs, simplifying administrative tasks.

DAMLink Hybris uses the capabilities of **OpenText™ Media Management** (OTMM): OTMM is the central repository storing media assets, managing creative workflows and processing of images, video and other media assets.

Benefits With the integration of **OTMM** into **Hybris**, merchandising teams working in Hybris can now gain the benefits of leveraging a market leading DAM solution integrated directly into the Product Cockpit / CMS Cockpit of Hybris.

This allows business users to:

- Save time: Leverage search and browse capabilities in OTMM to quickly locate rich media content.
- Improve consistency: Access to server-side media transformations allows Hybris PCM users have pre-rendered content for omni-channel delivery.
- Reduce errors: OTMM metadata ensures Hybris business users have access to only approved content and exposure to usage rights
- Avoid duplication: By leveraging central single-source DAM for all related media assets.

1.1 System landscape and components

DAMLink Hybris is composed of program and data components that are distributed on the involved product platforms. The following picture illustrates the DAMLink Hybris components in the system landscape.

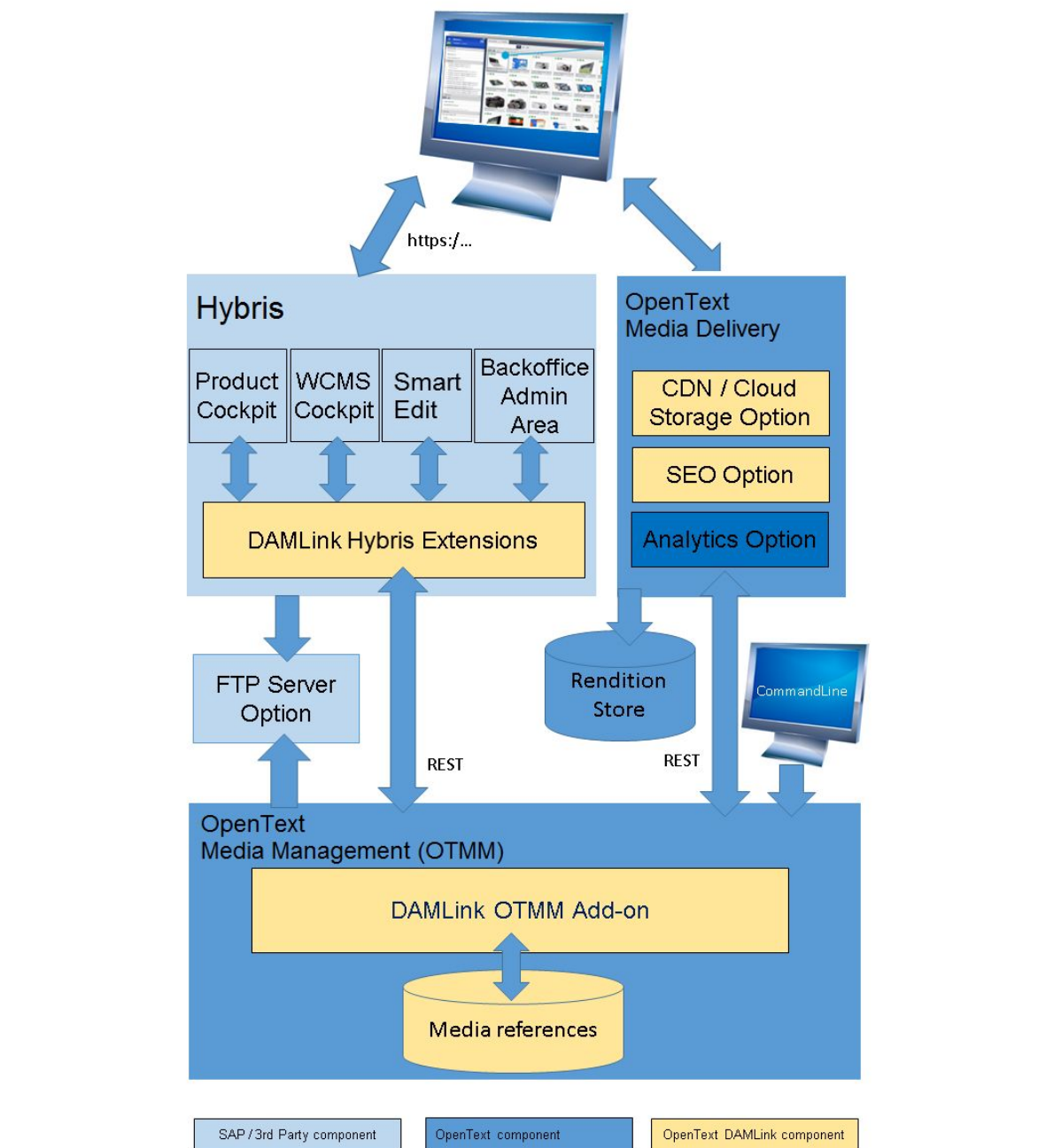


Figure 1-1: System landscape

The following components are provided:

- **DAMLink Hybris Extensions:**
Java extensions implement the integration into Hybris including UI and processing. In Hybris UIs like Product cockpit, asset assignment can be implemented.
- **DAMLink OTMM Add-on:**
This OTMM Application add-on provides the DAMLink Hybris interface on OTMM side.
- **Media reference:**
DAMLink Hybris uses OTMM data storage capabilities for the media data. In the OTMM database, the belonging metadata are organized. DAMLink Hybris introduces an own field group storing the references to Hybris items aside the OTMM asset data.
- **Media Delivery Service** is the core of a high-performance media provisioning. It is able to support a large number of end users of Hybris webstores efficiently.

Media Delivery Service, as one delivery option, provides SEO friendly asset names and also supports asset delivery via Amazon S3 cloud storage or Microsoft Azure Blob storage as alternative to a local asset delivery store. The servlet can be installed on an external Tomcat web server or in the Hybris system. Alternatively OTMM **Adaptive Media Delivery** with its analytics capabilities can be used.

1.2 Workflow

Integrated in Hybris **Product Cockpit (PCM)**, **DAMLink Hybris** provides the assignment of assets stored on **(OTMM side)**.

Asset assignment

Asset assignment is the assignment of an OTMM asset to a Hybris item attribute (for example a product attribute). Different asset deliveries of the asset can be created to reduce storage needs, the assets can be used multiple times. Thus, items or item attributes can share assets, if exactly the same asset and asset delivery already exists.

The assigned asset can be updated automatically with the latest asset version of the media in OTMM or is bound to the exact version of asset that was assigned.

Assignment techniques

DAMLink Hybris supports a variety of different assignment techniques:

- Asset deliveries of the assets are created on the fly, cached and managed and served by Media Delivery Service. Hybris only holds references (URLs) to Media Delivery Service or the asset deliveries posted to cloud storage like Amazon S3 cloud storage or Microsoft Azure Blob storage. Alternatively Adaptive Media Delivery with its Analytics capabilities can be used.
- Asset deliveries of an asset are created during assignment of the asset to a Hybris item by an OTMM process. OTMM submits exported or converted assets to the FTP server for further import into Hybris. The FTP server can be hosted on the same machine as Hybris or OTMM. Hybris holds the asset delivery files in its media store.

- Hybris Media items that hold URLs to streaming service for videos or to a cloud storage

For information about all available assignment techniques, see “[Configuring an asset delivery](#)” on page 35.

1.3 Further documentation sources

OpenText My Support (<https://knowledge.opentext.com>) offers software, patches, and documentation that are key to implementing OpenText products. It also offers self-serve support tools that help you get peer advice, quick guidance, and valuable troubleshooting options.

- For information about **OpenText Digital Asset Management for SAP Solutions**, see <https://knowledge.opentext.com/knowledge/lisapi.dll/Open/27729663> on My Support.
- For information about **OTMM**, see <https://knowledge.opentext.com/knowledge/lisapi.dll/Open/6290311> on My Support.
- For information about **Hybris Commerce Suite**, see the Hybris documentation, for example the Hybris wiki (<https://wiki.hybris.com>).

Chapter 2

Installation and basic configuration

The integration of Hybris and OTMM is provided as extensions for the Hybris platform. On OTMM side, the DAMLink OTMM Add-on must be installed. Additionally, as a preferred option, Media Delivery Service or OpenText Adaptive Media Delivery Service has to be deployed. This chapter describes the necessary installation steps.

To identify assets that can be assigned to items in Hybris (for example products), DAMLink Hybris extends the product editor of the Hybris UI with media reference editors. The extended functionality allows users to browse or search in OTMM folders. Found assets can be assigned to Hybris item attributes. Metadata of the Hybris item that the asset is assigned to is recorded at the asset in OTMM. There is also an option to upload an asset to OTMM and at the same time assign it to a Hybris item attribute. Additionally there are Hybris cronjobs provided that support batch assignment of assets to Hybris items. This chapter describes the basic configuration steps to set up the asset assignment.

2.1 System Prerequisites

The following products and components are prerequisites for a **DAMLink Hybris** scenario.



Note: For required version numbers, see the Release Notes.

- OTMM
- Hybris Commerce Suite or Hybris MDM
- Depending on how the assets should be assigned to the Hybris items you need:
 - Media Delivery Service (preferred option):
Asset deliveries of the assets are created on the fly, cached and managed and served by Media Delivery Service. Hybris only holds references (URLs) to Media Delivery Service.

Media Delivery Service also supports asset deliveries to be stored in Cloud Storage as well as descriptive filenames for SEO.
 - Adaptive Media Delivery Service as an alternative for asset delivery with Analytics support option.
 - FTP server:
Asset deliveries of an asset are created during assignment of the asset to a Hybris item by an OTMM process. OTMM will submit exported or converted assets to the FTP server for further import into Hybris. The FTP server can be hosted on the same machine as Hybris or OTMM. Hybris holds the asset delivery files in its media store.

- Others:
Environment for the desired assignment technique and storage medium like streaming services for videos.

For information about available assignment techniques, see “[Configuring an asset delivery](#)” on page 35.

- HTTPS and CORS related configuration is required. For more information, see “[HTTPS and CORS](#)” on page 111.

2.2 Installing the DAMLink OTMM Add-on

This section describes the necessary steps on **OTMM** side.

You need to install the DAMLink OTMM Add-on which is part of the OpenText product **DAMLink for SAP Solutions**. This is required for example for event notifications like on version updated or asset purge sent from OTMM to Hybris or for Single Sign On between Hybris and OTMM.

To install DAMLink OTMM Add-on on OTMM 16.0 or 16.2 server:

- Use the instructions in section 2.2 “Installing or upgrading the DAMLink OTMM Add-on” in *OpenText DAMLink for SAP Solutions - Installation and Configuration Guide (DAMSAP-IGD)* to install the add-on.

2.3 Multiple Hybris systems

DAMLink Hybris supports scenarios with multiple Hybris systems running against one OTMM instance. To make such scenarios work, you need to configure a unique Hybris system ID for each Hybris system. You need to make known all the Hybris IDs to the OTMM server. Then, the Hybris system ID of the Hybris system in which an item resides, is stored with the asset in OTMM automatically.

- For each **Hybris system**, you configure the Hybris system ID in the `local.properties` file. For instructions, see [Step 5](#).
- To **OTMM** you can make known one Hybris system ID as part of the installation or upgrade of the DAMLink OTMM Add-on. For instructions, see *OpenText DAMLink for SAP Solutions - Installation and Configuration Guide (DAMSAP-IGD)*.
- To make known additional Hybris system IDs to **OTMM**, you have to open the following URL:
`http://<otmmhost:port>/ot-damlink/api/config/addHybris?url=https://<hybrishost:port>&systemid=<Hybris system ID>`
<otmmhost:port>: Replace with the name and port of your OTMM server.
<hybrishost:port>: Replace with the name and port of your Hybris server.
<Hybris system ID>: Replace with the Hybris system ID that you have configured for this Hybris server.

2.4 Installing DAMLink Hybris Extensions

This section describes the installation of **DAMLink Hybris** Extensions into a running Hybris system.



Note: For information about upgrading from DAMLink Hybris version 1.5, 1.5.1, or 16.0, see [“Upgrading” on page 22](#).

To install DAMLink Hybris Extensions:

1. Unzip the damlink-hybris-integration-*<version>*.zip file into the Hybris Suite installation directory.
The path in the zip file starts with the Hybris sub directory below the Suite installation directory.

2. Register the **otmmaddon** extensions with this Hybris platform:

Open the hybris/config/localextension.xml file.

Add the following text to the <extension> section depending on your Hybris version:

- For Hybris version 6.0

```
<!-- Other hybris extensions -->
<extension name="otmmaddon" />
<extension name="otmmaddonui" />
<extension name="otmmaddonbackoffice" />
```

- For Hybris version 6.1, 6.2, or 6.3

```
<!-- Other hybris extensions -->
<extension name="otmmaddon" />
<extension name="otmmaddonui" />
<extension name="otmmaddonbackoffice" />
<extension name='otmmws' />
```

- For Hybris version 6.4 or later
For supported Hybris versions, see the Release Notes.

```
<!-- Other hybris extensions -->
<extension name="otmmaddon" />
<extension name="otmmaddonui" />
<extension name="otmmaddonbackoffice" />
<extension name='otmmws' />
<extension name='otmmsmartedit' />
```



Note: Add <extension name='otmmsmartedit' /> only if you want to use Hybris smartedit and have added the Hybris smartedit extension to your Hybris configuration.

3. In OTMM, create a user that has permissions to search, browse, export, and change metadata of the assets that are relevant for use in Hybris. This user will be used to connect from Hybris to OTMM in the following steps.

4. Update the `local.properties` file of your Hybris installation with the configuration settings necessary for the **otmmaddon** extensions.

Open the `hybris/bin/custom/otmmaddon/project.properties` file. There you find all the configuration keys with default configuration settings. Override these configuration values in `hybris/config/local.properties`.



Note: The `local.properties` file is not overwritten if the module is updated. Any settings in `local.properties` override settings for the same properties in `project.properties`.

5. As a minimum, add the following configuration settings to the `local.properties` file:
 - `otmm.server.host`: Replace the `<YourOTMMHost>` placeholder with the OTMM server host name.
 - `otmm.server.login` and `otmm.server.password`: Specify the credentials of the user that you created in the previous step.
 - `otmm.model.image.format.list`: Specify a metadata model, for example `OPENTEXT.PRODUCTS`.
 - `otmm.hybris.systemId`: Replace the `<SystemId>` placeholder with a value you define. The ID must be unique within the Hybris systems that share one OTMM server.
 - `installed.tenants`: This key must not have a value. Required for the “Auto update to latest version” and “Deletion on Purge” features.
 - `log4j2.config.xml`: For Hybris 6.2 and higher this key needs to be set to an empty value. This is required to get the `otmmaddon.log` file created and written.

For example:

```
#####
# OTMM Server related configuration                                     #
#####
otmm.server.scheme=http
# otmm.server.domain is an obsolete alias for otmm.server.host
otmm.server.host=<YourOTMMHost>
otmm.server.port=11090

# OTMM User Login
#otmm.server.login=
# OTMM User Password
#otmm.server.password=

# Comma separated list of OTMM Models which are configured
# according to documentation.
otmm.model.image.format.list=OPENTEXT.PRODUCTS

#####
# Hybris System ID                                                  #
```



```
#####
otmm.hybris.systemId=<SystemId>

# Disable slave tenants like junit.
# Required for "Automatic update to latest version" and
# "Deletion of OpenTextMedia items for purged assets"
installed.tenants=

log4j2.config.xml=
```



For demo and test only

If you are running Hybris with HSQLDB for demo and test purposes and you want to work with the cronjobs `otmmAssetSyncJob` and `otmmExportServiceJob` for asynchronous and batch assignment, you have to add the following configuration setting: `otmm.db.ignore.exclusive.lock=TRUE`

Do not use this configuration on productive environments!

6. For asset delivery, Media Delivery Service is recommended.

For installation instructions, see [“Installing Media Delivery Service” on page 20](#).

Configure Media Delivery Service properties in the `local.properties` file.

As a minimum, the connection information has to be configured:

```
# Media Delivery Service URL and password
otmm.media.servlet.url=http://<host>:<port>/ImConvServlet/imconv
otmm.media.servlet.pwd=<password>
```

7. **Only if an FTP server is to be used as alternative to Media Delivery Service:** Add the following configuration settings in the `local.properties` file:

Replace the variables `<otmmhost>`, `<ftphost>`, `<ftpuser>` and `<ftppassword>` with your settings.

For example:

```
#FTP Server hostname or IP
10012=<ftphost>
#FTP User
10013=<ftpuser>
#FTP User Password
10014=<ftpuserpass>

# Local path to FTP home folder of <ftpuser>, assumes local
# destination i.e FTP server runs on the same machine as Hybris
otmm.locationOfFiles=c:\\FTP
# Local or FTP depending on if the FTP server is locally running
# on Hybris instance or not
otmm.AssetSync.SourceRepository=Local
```



Important

The directory that you configured with the `otmm.locationOfFiles=c:\\FTP` property must be shared between FTP server and Hybris server. Alternatively use `otmm.AssetSync.SourceRepository=FTP`.

8. Stop the Hybris server if it is running.
9. Run the ant build:
 - Open a command prompt window or a terminal with a user that is permitted to change the Hybris configuration and code on file system level.
 - Change the directory to `hybris/bin/platform`.
 - Run the following commands and wait for the build to complete:
 - On Windows: `setantenv.bat`
 - On UNIX: `setantenv.sh`
 - `ant clean all`
10. Start the Hybris server. For example, from a command prompt window or a terminal with: `hybrisserver.bat`.
11. After the Hybris server has started, open **`http://<hybrishost:port>/platform/update`** in a web browser.

In the **Project data settings** list, select the check boxes for the **DAMLink Hybris Extensions**. Only the extensions that you specified in step 2 are available.

- **otmmaddon**
- **otmmaddonui**
- **otmmaddonbackoffice**
- **otmmws** (if available)
- **otmmsmartedit** (if available)

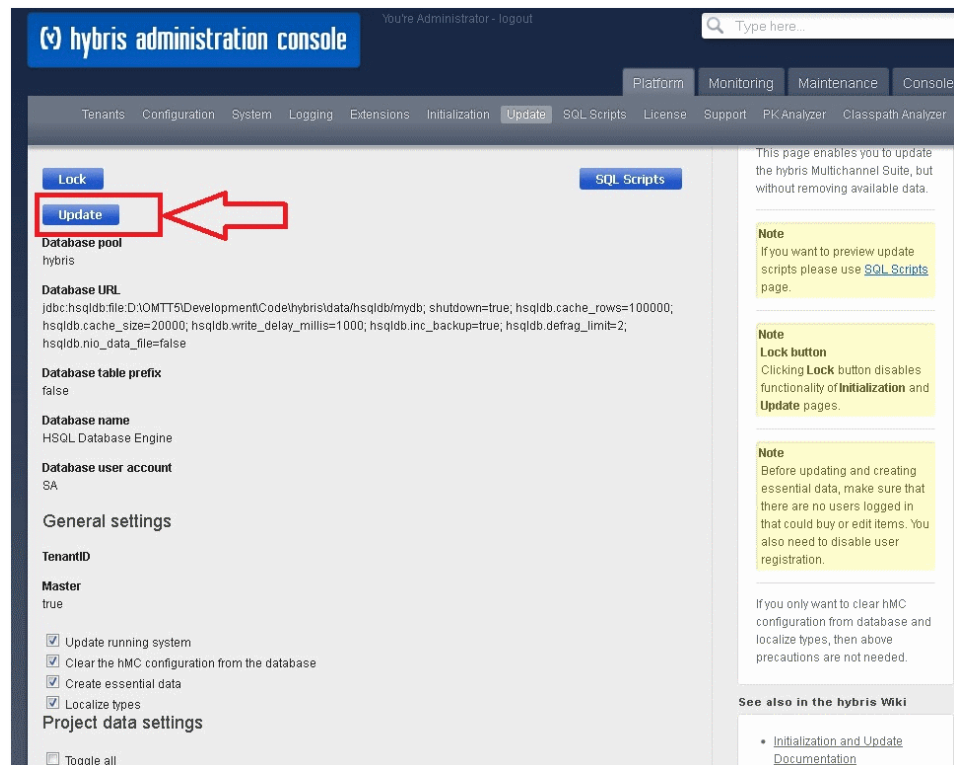
otmmaddon settings: Assignment Configuration

- **No:** With this option no predefined assignment configuration is installed. You need to configure the entire assignment configuration from scratch after the installation. To get the system running, configuration of at least one asset delivery is required.
- **Initialize with default configuration (if empty):** Installs a default example assignment configuration. If there is an assignment configuration already, this assignment configuration is kept unchanged.
- **Migrate from properties file (<= 16.0.0):** Migrates the assignment configuration from DAMLink Hybris 16.0. This option is relevant for upgrades only.
- **Restore default configuration:** Installs a default example assignment configuration, even if there is an assignment configuration already.

otmmaddonui settings:

- To enable the integration of the **Asset Assignment Dialog** in the Hybris **Product Cockpit**, set **Import Product Cockpit Data** to **Yes**. This setting is activated by default.
- To enable the **Asset Assignment Dialog** for the **CMS Cockpit** tool, set **Import CMS Cockpit Data** to **Yes**.

12. Click **Update**.



Cron jobs

Aside from the synchronous assignment of assets to Hybris items, the actual export and synchronization of converted media files from OTMM back to Hybris can also be done asynchronously. It is controlled by Hybris cron jobs. Several predefined cron jobs are installed with **DAMLink Hybris**.

Per default, the export and synchronization jobs are scheduled to run every 5 minutes to support automatic update to the latest version. The purge job is disabled.

For information about adjusting these settings, see **“Scheduling cron jobs” on page 73**.

2.5 Installing Media Delivery Service

This section describes the installation of Media Delivery Service provided by **DAMLink Hybris**. The servlet is introduced as core of a high-performance media provisioning. It is able to support a large number of Hybris webstore end users efficiently. If Media Delivery Service is to be used, follow the described installation procedure.



Note: The following prerequisites are valid for version 16.2.1 Build 610 or higher. Build 610 is identical with patch 1 (DAMLINK_MEDSRVR-160201-001).

OpenText recommends that you install version 16.2.1 Build 610 or higher.

Prerequisites

Media Delivery Service uses **ImageMagick** as the tool for graphic asset delivery functions. ImageMagick is not delivered with the product and you must download and install it manually.

OpenText recommends that you install ImageMagick 7.0.7 or higher, because of security issues with older versions.



Note: Different versions of ImageMagick store the executable file under a different file name and path. This also depends on the settings of the ImageMagick installation.

If `magick.exe` is not found, you can do one of the following after the installation of Media Delivery Service:

Adjust the path environment of the MediaServlet process

Adjust the `webapps/ImConvServlet/WEB-INF/resources/imconv.properties` configuration file. Add the absolute path to the `magick.exe` command with the `magickPath` parameter.

Example: `magickPath = C:\\Program Files\\ImageMagick-7.0.7-Q8\\magick.exe`

Package

Media Delivery Service comes in a package which consists of the servlet (`ImConvServlet.war`) to be deployed in a servlet container and a few batch scripts for UNIX and Windows. The batch scripts use the `ImConvServlet.war` as a command line tool.

The file `ImConvServlet.war` must be copied into the `webapps` directory of a servlet container, for example `tomcat`.

This may be a standalone Tomcat, or the Tomcat directory of a Hybris installation.

To install Media Delivery Service package:

1. On Windows, it may be necessary to install a Microsoft VC redistributable for VC10 from www.microsoft.com: Microsoft Visual C++ 2010 Redistributable Package (x64) (<https://www.microsoft.com/en-us/download/details.aspx?id=14632>).

2. To deploy into a **standalone Tomcat**:
Copy the `ImConvServlet.war` into the webapps directory.

To deploy into the embedded **Hybris Tomcat**:
If it does not exist, create the directory `hybris/bin/platform/tomcat/webapps`.

Unpack the `ImConvServlet.war` into the webapps directory (with zip or jar).

Alternatively in case of an embedded Hybris Tomcat:
Look for the `<hosts>` section in `hybris/config/tomcat/conf/server.xml`, and change **unpackWARs** to `true`, run an `ant clean all` to make the change effective and copy the war file into the webapps directory.
3. As soon as the war file is unpacked into the `webapps/ImConvServlet` directory, you must edit `webapps/ImConvServlet/WEB-INF/resources/imconv.properties`. There, change the OTMM URL, user and password, the root directory, where the asset deliveries shall be stored, and the asset delivery sets.

Whenever `imconv.properties` is changed and saved, the changes are immediately effective.

For information about all configuration properties of Media Delivery Service, see [“Configuring properties” on page 48](#).

Test To test the service, you can try some URL requests to get images from the service. Information about examples, see [“Sending requests” on page 51](#).

2.6 Installing language packs

DAMLink Hybris supports the user interface in core OTMM languages (DE, FR, IT, ES, PT, JA, ZH).

These languages are shipped in additional language packs. This section describes the necessary steps to install a language pack replacing the default English UI.

To install a language pack:

1. Copy the localization files:
Copy `langpack-damlink-hybris-<version>.zip` to `${HYBRIS_CONFIG_DIR}/languages` directory.
Make sure you have a sub folder named `customize` in `${HYBRIS_CONFIG_DIR}` folder. If not, create one.
2. Open a command interpreter window in Microsoft Windows (shell in Unix family systems) and navigate to the `${HYBRIS_BIN_DIR}/platform` directory.
Set the Ant environment by running the `setantenv.bat` file.

Call `ant customize` to start unpacking the language pack and copying localization files to the proper directory.
3. After unpacking, remove the language pack from `${HYBRIS_CONFIG_DIR}/languages` directory.
4. Restart the Hybris server.

5. To update the system: Open the **Hybris administration console**. Go to the **Platform > Update** tab.

Perform a system update with following options selected:

- **Create essential data**
- **Localize types**

After the system update, the DAMLink Hybris UI is shown in the selected language (chosen at login).

2.7 Upgrading

This section describes the necessary steps to upgrade DAMLink Hybris Extensions and Media Delivery Service.



Notes

- You need to upgrade the DAMLink OTMM Add-on which is part of the OpenText product **DAMLink for SAP Solutions** also.
For more information, see section 2.2 “Installing or upgrading the DAMLink OTMM Add-on” in *OpenText DAMLink for SAP Solutions - Installation and Configuration Guide (DAMSAP-IGD)*.
- OpenText recommends that you install ImageMagick 7.0.7 or higher, because of security issues with older versions. For more information, see [“Installing Media Delivery Service” on page 20](#).

2.7.1 Upgrading Media Delivery Service

This section describes the necessary steps for an upgrade of an existing Media Delivery Service from version 16.0 or 16.2.0 to version 16.2.1.

To upgrade Media Delivery Service:

1. If not part of the file already, add the following key to your `hybris/config/local.properties` file together with Media Delivery Service password:
`otmm.media.servlet.pwd=<password>`
2. Copy the configuration file of the existing Media Delivery Service `imconv.properties` from `webapps/ImConvServlet/WEB-INF/resources` folder to a folder of your choice outside of the `ImConvServlet` folder. In this way, the existing configuration is saved. This file can be reused after installation of the new Media Delivery Service.



Note: The `webapps` directory is either below `hybris/bin/platform/tomcat`, if you deployed Media Delivery Service in Hybris, or in a separate Tomcat installation.

3. Remove the old `ImConvServlet.war` and the `ImConvServlet` sub folder in the `webapps` folder.

4. Unpack the `ImConvServlet.war` into the `webapps` directory (with zip or jar).
5. Rename the new `imconv.properties` file in the new `webapps/ImConvServlet/WEB-INF/resources` folder, so that you can use it as a reference, and copy back your saved `imconv.properties` file.

2.7.2 Upgrading DAMLink Hybris Extensions

This section describes the necessary steps for upgrading an existing DAMLink Hybris of version 1.5, 1.5.1, or 16.0.

Prerequisites Upgrading Media Delivery Service first: If Media Delivery Service is used in your environment, make sure that you upgrade Media Delivery Service before upgrading DAMLink Hybris Extensions. For an instruction, see [“Upgrading Media Delivery Service” on page 22](#).

To upgrade DAMLink Hybris Extensions:

1. Shut down the Hybris server.
2. Move the existing DAMLink Hybris Extensions folders from the installation folder (`<hybris suite install directory>/hybris/bin/custom`) to a folder of your choice outside of the installation folder.

For DAMLink Hybris 1.5, this is only the folder for the extension **otmmaddon**, for 1.5.1 and 16.0 these are additionally the folders for the extensions **otmmaddonui** and **otmmaddonbackoffice** (or at least one of them).

In this way, the existing extensions are saved. Some customizing or configuration details, especially settings in the `project.properties` file, can be reused after installation of the new **DAMLink Hybris**.

3. For properties that are not defined in the `local.properties` file already: Copy the properties starting with the following prefixes from the `otmmaddon/project.properties` file to your `local.properties` file:
 - `otmm.rendering`
 - `otmm.renderings`
 - `otmm.otmmAssetSync.mapTo`
4. For Hybris 6.0: Add the property `otmm.thumbnail.protected = true` to your `local.properties` file.
5. Add the property `otmm.hybris.isDefault = true` to your `local.properties` file. You must keep this value after upgrade.
6. Install the new DAMLink Hybris Extensions 16.2.1.
 - a. Unzip the extensions. For information about extensions, see [“Installing DAMLink Hybris Extensions” on page 15](#).
 - b. If there have been any custom specific changes in the `project.properties` file of the previous **otmmaddon** extension, integrate these changes from the `project.properties` file that has been moved in [Step 2](#).



Note: Normally the `project.properties` should not be changed. The settings can be overwritten and adjusted to the customer's need in the `local.properties` file of the Hybris deployment.

- c. For an upgrade from version 1.5:
In case of a test or demo installation with `hsqldb` change the `db.ignore.exclusive.lock=TRUE` setting to `otmm.db.ignore.exclusive.lock=TRUE`. The name of this setting has changed with version 1.5.1.
 - d. If there was a Hook implemented as a jar file that was deployed into the `otmmaddon/lib` folder: Re-deploy the jar file:
Copy it from the previous `otmmaddon` that has been moved in [Step 2](#).
Paste it to the `lib` folder of the new `otmmaddon` deployment (`<hybris suite install directory>/hybris/bin/custom/otmmaddon/lib`).
 - e. In case of SSO the `security-sso.jar`, which contains the security digest, has to be copied back from the saved `otmmaddon/lib` folder to `otmmaddon/lib`.
7. Register the added extensions with your Hybris platform: Open the `hybris/config/localextension.xml` file. Add the DAMLink Hybris Extension names in the `<extension>` section:
 - a. For an upgrade from version 1.5:

```
<!-- Other Hybris extensions -->
<extension name="otmmaddon" />
<extension name="otmmaddonui" />
<extension name="otmmaddonbackoffice" />
```
 - b. For Hybris 6.1 or higher:

```
<extension name="otmmws" />
```
 - c. For Hybris 6.4 or higher together with smart edit:

```
<extension name="otmmsmartedit" />
```
 8. If you use Product Cockpit: If there have been any custom specific changes to your editor configurations, integrate these changes from the previous editor configuration file into the new one.
Reimport `productcockpit_config.impex` in order to make the changes effective.



Important

To ensure that the new editor configurations are applied, all Product Cockpit users must delete their personalized settings.

From the Hybris Product Cockpit menu, select **User settings > Reset Personalized Settings**. For more information, see [“Testing the integration” on page 26](#).

9. If there have been any custom changes of the `otmmaddonbackoffice-backoffice-config.xml` file, integrate these changes from the previous `otmmaddonbackoffice-backoffice-config.xml` file into the new one.

In order to make the changes effective do one of the following:

- restart your Hybris Server.
 - reset `cockpit-config.xml` at Hybris **Backoffice Application Orchestrator**. (See “Customizing Media Reference Editors in `otmmaddonbackoffice` configuration” on page 65).
10. Undo earlier changes to the cockpit configuration files. With earlier DAMLink Hybris versions, it was necessary to add spring bean definitions for the OpenText Cockpit Editors and the OpenText Asset Assignment Dialog. This was done in the `cmscockpit-editors.xml` and `cmscockpitwizards.xml` files in the following `yacceleratorcockpits` extension of your Hybris installation:

```
hybris\bin\ext-template\yacceleratorcockpits\resources\yacceleratorcockpits
\cmscockpit\string
```

Either restore the original XML files or manually remove the OpenText related entries in these files.

11. Run the ant build:
 - a. Open a command prompt or terminal at `hybris/bin/platform`. The user needs permissions to change the Hybris configuration and code on file system level.
 - b. Run the following commands and wait for each to complete:
 - On Windows: `setantenv.bat`
 - On Unix: `setantenv.sh`
 - `ant clean all`
12. Start the Hybris server.
13. After the Hybris server has started, open the Hybris administration console, **Platform > Update**, in a web browser: `http://<hybrishost:port>/platform/update`.
Media Delivery Service must run during the **Platform > Update** step. If Media Delivery Service does not run, the upgrade is unable to fetch the ImageMagick commands to upgrade your configuration. In this case, you would have to enter the ImageMagick commands manually in the Assignment Configuration in Hybris Backoffice.
 - a. In the **Project data settings** list, select the check boxes for the **DAMLink Hybris Extensions**:
 - **otmmaddon** – For an upgrade from 16.2.0 to 16.2.1 select **-NO**. Otherwise, select **Migrate from properties file (<= 16.0.0)** for the assignment configuration.

- **otmmaddonui** – Choose if you want to configure the Asset Assignment Dialog in the Hybris Product Cockpit and/or CMS Cockpit. Per default both are configured.
 - **otmmaddonbackoffice**
 - **otmmws (if available)**
 - **otmmsmartedit (if available)**
- b. Click the **Update** button at the top of the page.
 - c. Wait for the update to finish.
14. Check version/build number to verify that the new version 16.2.1 is active. For more information, see [“Identify the version and build number” on page 113.](#)
 15. After the upgrade, there possibly can be additional, superfluous assignment specifications. Review the list of assignment specs and delete any unwanted items. For information, see [“Configuring an assignment spec” on page 43.](#)

2.8 Testing the integration

This section describes how you can test the integration.

To test the integration:

1. Log into **Hybris Product Cockpit** as a product manager.

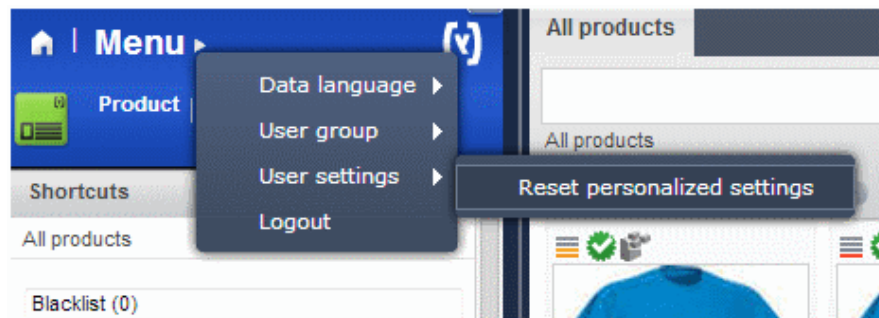


Example 2-1: Product Cockpit URL

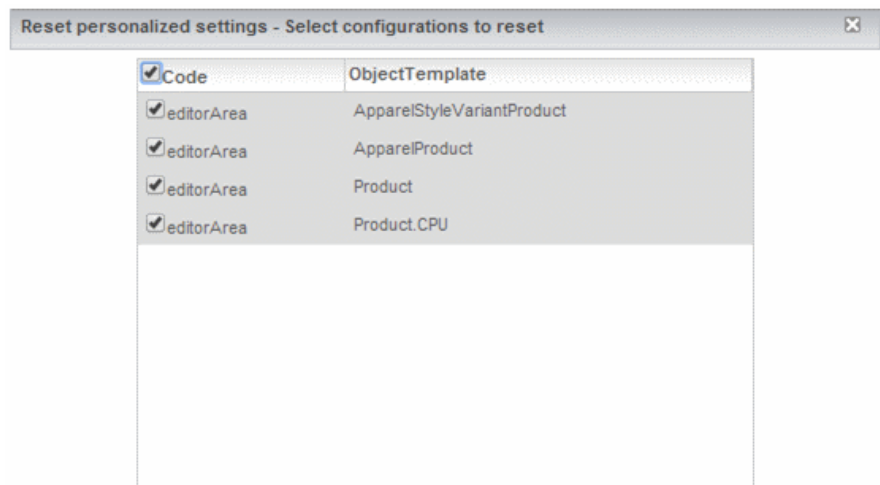
If Hybris is running on localhost: use **http://localhost:9001/productcockpit** as URL.



Note: To ensure that the new editor configurations are applied, users must delete their personalized settings. From the Hybris menu, select **User settings > Reset Personalized Settings**.



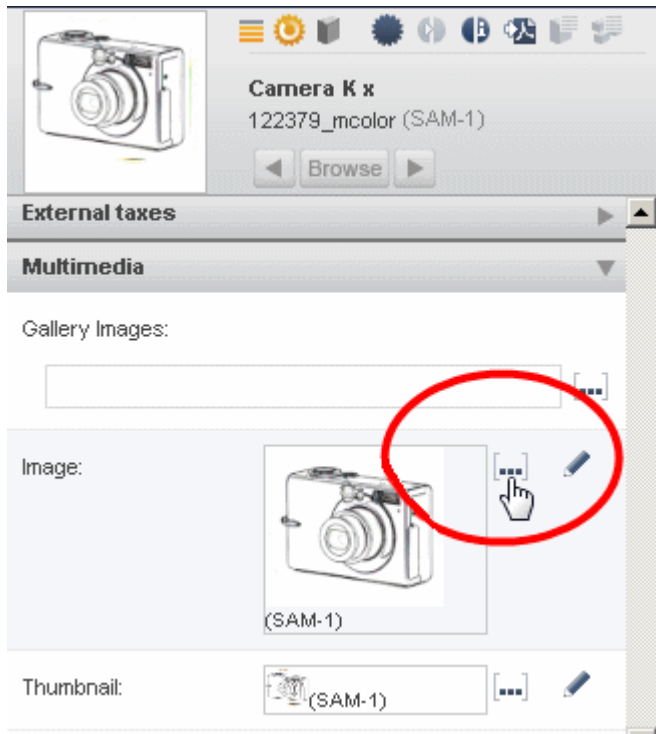
Reset the settings for all product editors from which you want to access media from OTMM.



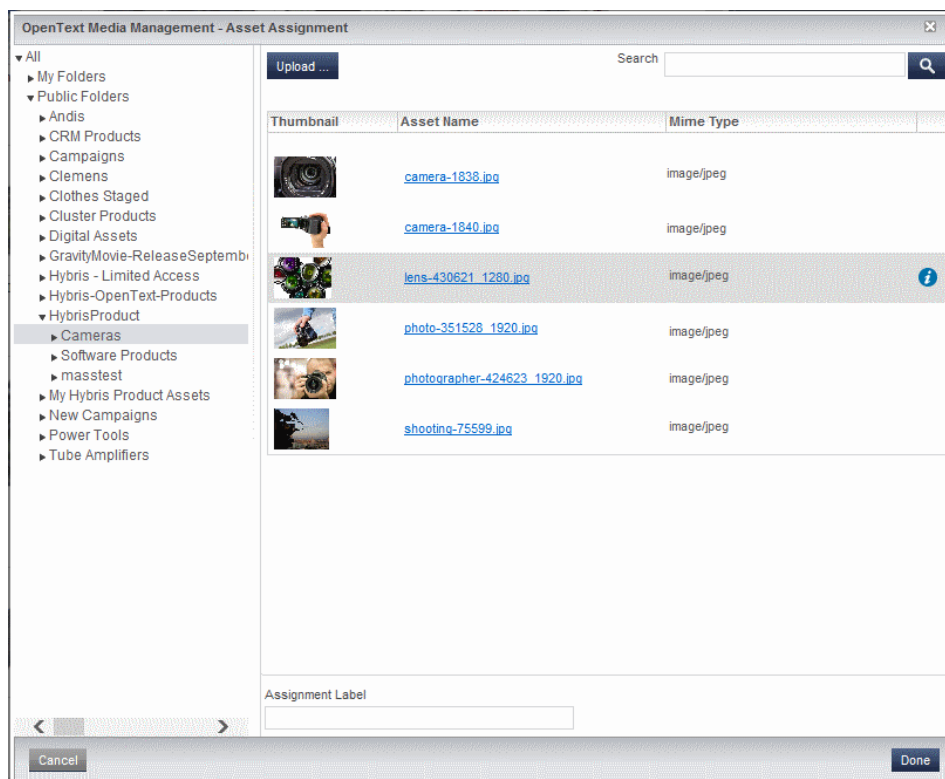
2. Open the product editor of a product:



Navigate to **Multimedia** attributes. Open the media reference wizard with click on [...].



3. The customized asset assignment dialog **OpenText Media Management – Asset Assignment** of the extension opens. The wizard allows to search or browse the OTMM repository.
Search for an asset keyword of which you know that it returns a result in OTMM.



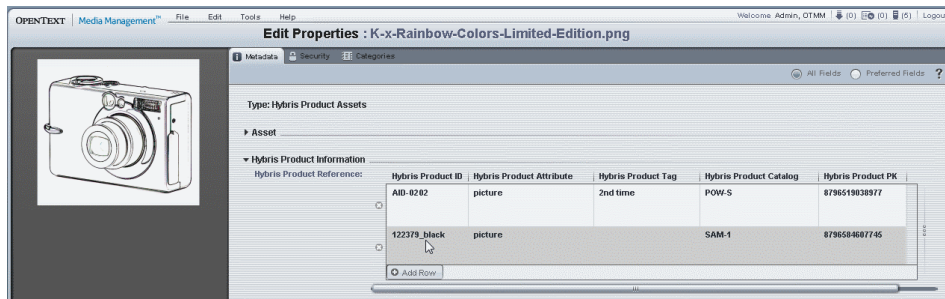
4. You can assign assets that are visible in folders or that are returned as search results to the current product:
Select the asset. Optionally, enter in the **Assignment Label** field a tag name that will be stored with the product ID in OTMM as asset metadata.



Note: Get more info about an asset: If you hover your mouse over an asset in the list or select it, a blue info icon is displayed. With a click on the icon, a new browser tab with the OTMM asset details is opened.

With a click on the thumbnail picture in the list, the asset details are shown in a popup window.

5. Click **Done**.
6. Once the asset has been assigned, log on to **OTMM**. To find the same asset, enter a search string in the search field top right. Validate that the Hybris Product ID and Tag was saved in the asset metadata model:



2.9 Registering DAMLink Hybris in the SAP System Landscape Directory

You can register DAMLink Hybris in the SAP *System Landscape Directory (SLD)*. This does not happen automatically, but you must send the registration files to the SLD server.

For registration at SLD, the servers provide XML files containing registering information. The registration can be executed with the `uploadSLDfile` command which adds the registering information in the SLD.

This section describes the steps necessary to register **DAMLink Hybris** at SAP System Landscape Directory (SLD).

Prerequisites You have a user assigned to a particular SLD security role.

To register DAMLink Hybris in the SAP SLD:

1. From My Support, download the following ZIP file: `../Digital Asset Management for SAP Solutions 16.2.1/DAMLink OTMM Add-ons/SLD_registration_templates.zip` (<https://knowledge.opentext.com/knowledge/lisapi.dll/Open/69431407>)
2. From `SLD_registration_templates.zip` extract the `ThirdPartySystem_Hybris.xml` file to `<TEAMS_HOME>/data/sap/sld`.
Overwrite the existing file that has the same file name.
3. Edit the XML file manually. Adjust the string `$HYBRIS_COMPUTER_SYSTEM$` to the name of the computer name, where **DAMLink Hybris** is deployed.
4. Send the XML file to the proper SLD server. To do this, you can use your own tool or use the scripts that are provided by OpenText in `<TEAMS_HOME>/data/sap/sld`.
 - `uploadSLDfile.bat` for Windows systems
 - `uploadSLDfile.sh` for Unix systems

Use the following parameters:

- `SLD_HOST_NAME`: name of the SLD server

- *SLD_HOST_PORT*: port number of the SLD server
- *SLD_USER_NAME*: name of a user with SLD security role
- *FILE_TO_UPLOAD*: Name and path to the XML configuration file in *<TEAMS_HOME>/data/sap/sld*.

You will be prompted for the password of the user.

Example:

```
uploadSLDfile.bat sld.mycompany.com 50000 sld_user  
ThirdPartySystem_HYBRIS.xml
```


Chapter 3

Advanced configuration

Configuration options for DAMLink Hybris are partly maintained within the `project.properties` or `local.properties` file and partly in Hybris Backoffice. The `project.properties` file contains documentation for each setting.

You can override the settings in the `project.properties` with setting the same property in the `hybris\config\local.properties` file of the main Hybris installation.



Note: It is recommended that you store your configuration changes in the `local.properties` file. This minimizes the effort for applying updates or patches to DAMLink Hybris, as changes to `project.properties` have to be maintained manually after the application of updates or patches.

The configuration options include:

- Connection parameters
These are required parameters that define the connection to the OTMM server and to Media Delivery Service or a FTP server.
For information about the connection configuration, see [“Installing DAMLink Hybris Extensions” on page 15](#).
- Used OTMM asset metadata models
- Assignment, asset delivery options, and conversion formats for images
Assignment for videos and other formats
Configured in Hybris Backoffice
- Parameters for cron jobs and logging



Important

After changing the `project.properties` file or a `local.properties` file, restarting Hybris is necessary to activate the changed properties.

3.1 Configuring assignment spec and asset delivery

Assignment spec and asset delivery are configured in Hybris Backoffice under the **DAM by Opentext > Assignment Configuration** node.

You always assign an OTMM asset to an attribute of a Hybris item, for example to the picture attribute of an item of type Product.

The assignment triggers the creation of a variant or variants of the OTMM asset. You define this in an **asset delivery** with the following settings:

- Properties of the image, for example a resolution with a width of 515 pixels and a height of 515 pixels

- Way to provide the image, for example with Media Delivery Service

You then have to specify in an assignment spec which asset delivery definition (for example the one with 515 x 515 pixels) is used for the assignment to the `picture` attribute of `Products`.

Additionally, you can specify an implicit assignment by referring to other assignment specs. In our example that means that you can not only create and assign a 515x515 rendition for the `picture` attribute of `Product`, but at the same time “implicitly” a smaller rendition of the same OTMM image to the `thumbnail` attribute of `Product`.

Another aspect is that some attributes of a product can not only hold one Media Item, but a whole set of Media Items. For example, if the attributes are of type `MediaCollection` or `MediaContainer`. In this case you can refer not only to one asset delivery, but to a whole set and thus say that you want to create several different renditions of the OTMM asset and assign them all for example to the `others` attribute of the `Product`.

You need to define the following items:

1. **Asset delivery**

The asset delivery defines how assets are transferred to Hybris. This includes the dimensions of graphics, the file format of graphics, and the method of transfer. There are three methods of transfer, also called asset delivery types: Media Delivery Service, Content by Export, and External URL.

2. **Assignment spec**

An assignment spec assigns one or more asset deliveries to a Hybris type and its attributes. As an example the attribute could be gallery images and the type the product. In this example you would probably want to have several asset deliveries with different dimensions assigned. This way you could support the image sizes you need for preview, display, and zoom in the gallery.

3. **Content types**

Content types are taken from OTMM during installation of DAMLink Hybris. Manual configuration is only necessary if content types are changed in OTMM after the DAMLink Hybris installation or if OTMM has not been available while DAMLink Hybris was installed, which is not recommendable. Content types are used to control the asset delivery depending on the content type, that is depending on MIME types. One content type is a collection of MIME types.

Hybris attribute types

The following Hybris attribute types are supported and appear in Hybris Backoffice:

- For a single asset, `Media`
- For a single asset per language (Locale), `localized:Media`
- For a list of assets:
 - `MediaCollection` (a list of `Media` items)
 - `MediaContainer` (a list of `Media` items with different asset deliveries of the same asset)
 - `MediaContainerList` (a list of `MediaContainer` items)

- `localized:MediaContainer`

Export For a set of asset deliveries defined by assignment spec the export from OTMM to the FTP server is performed in the following cases:

- The asset delivery mode `ContentByExport` is used for a asset delivery.
- The **URL Attribute** property is set and used as placeholder in the **Template String** property.
- The **MIME Type Attribute** property is set.

In other cases, no export is triggered for a set of asset deliveries.

3.1.1 Configuring an asset delivery

If a new asset is assigned, the selected image goes through a process that generates the required asset deliveries. There are different ways how the asset deliveries are created and how they are connected with the created Hybris media items.

To create an asset delivery:

1. In Hybris Backoffice /backoffice, click **DAM by Opentext > Assignment Configuration > Asset delivery**.
2. Click **+ Asset Delivery** to create a new asset delivery and select the asset delivery type from the list.
3. The properties of the new asset delivery are on the **Common** tab. For information on the meaning of the individual properties refer to the table below.

Properties The following properties exist for the asset delivery depending on the asset delivery type:

Property	Description	Used for asset delivery types
Name	Used as the name for the media asset delivery of the created media item. The name must be unique. With the syntax <code><pixel>Wx<pixel>H</code> , you can define the dimensions of the asset delivery.	Mandatory for all types
Type	Asset delivery type: Media Delivery Service, Content by Export, or External URL See the following sections for more information.	Mandatory for all types
Media Format Qualifier	By default the media format associated with the created media has the asset delivery name as qualifier. If you want a different qualifier, enter a value for this attribute.	Optional for all types

Property	Description	Used for asset delivery types
Media Folder	Name of assigned Hybris MediaFolder, for example <code>otmmRoot</code> For more information about MediaFolders, see Hybris wiki: MediaFolders (https://wiki.hybris.com/x/vZZzC).	Mandatory for all types
Display Name	Used in the asset assignment dialog if you configured an asset delivery selection. If left empty, the Name property is used instead. The Hybris default language is the default language for this property. You can enter separate values for multiple languages in this field. Click the globe right of Display name to do so.	Optional for all types
Is Default	Exactly one asset delivery must be defined as the default. This asset delivery is used if no appropriate assignment spec can be found for the assignment. The created Media item is assigned to the explicit attribute. Setting an asset delivery to default automatically marks the current default to not default. If you delete the default asset delivery, take care to specify another asset delivery as the default.	Mandatory for all types
Rendition Set	Specifies a rendition set. To support cloud storage with Media delivery service the behavior of the setting is as follows: If Rendition Set is empty just this rendition is pre-cached by Media Delivery Service. If Rendition Set is specified all renditions of the set are pre-cached by Media Delivery Service. For more information about rendition sets, see “Rendition production” on page 47 .	Optional for: Media Delivery Service
Command	Custom ImageMagick command. If you do not use the syntax described for the Name property, you must use an ImageMagick command to define the dimensions for the asset delivery. Example: <code>-resize 2000x2000</code>	Optional for: Media Delivery Service ContentByExport
Template String	The template is used to create the URLs to the assigned assets. For the syntax, see “URL template string placeholders” on page 42	Mandatory for: External URL

Property	Description	Used for asset delivery types
URL Attribute	Attribute path within the <code>assetProperties.xml</code> file that contains the URL or a part of the URL (see Template string). If this property is defined, an export to the FTP server is triggered. For information about available placeholders, see “URL template string placeholders” on page 42 .	Optional for: External URL
Fixed MIME Type	A fixed MIME type used for the asset delivery.	Optional for: Media Delivery Service External URL
MIME Type Attribute	Attribute path within the <code>assetProperties.xml</code> file that contains the MIME type of the rendering. If this property is defined, an export to the FTP server is triggered. Per default, no MIME type is set. Hybris then automatically sets application/octet-stream.	Optional for: Media Delivery Service External URL

3.1.1.1 Media Delivery Service type

When you use **Media Delivery Service** type, only the metadata of the media item is stored in Hybris. The media content, however, is delivered by a Java servlet. The servlet fetches the media item from OTMM, creates the requested asset delivery first and then all other configured asset deliveries of the rendition set, stores the asset deliveries for future requests, and delivers the requested asset delivery as soon as it is created. To generate the asset deliveries, the servlet calls ImageMagick via a Java JNI interface. The asset deliveries can either reside on a local hard disk or on a cloud storage like Microsoft Azure Storage or Amazon S3.

Media Delivery Service is the default asset delivery type.

This service is introduced as core of a high-performance media provisioning. In this service, an ImageMagick Java interface is implemented to generate asset deliveries. For more information, see [“Using Media Delivery Service” on page 46](#).

Alternative As an alternative to Media Delivery Service, you can use the OpenText Adaptive Media Delivery. See the configuration key `otmm.media.servlet.useAdaptiveMediaDelivery` below. For information on the OpenText Adaptive Media Delivery, see the OTMM documentation on My Support.

Media Delivery Service properties Media Delivery Service properties are saved in the configuration file `webapps/ImConvServlet/WEB-INF/resources/imconv.properties`.

The asset deliveries defined in Hybris Backoffice must match the configuration in the `imconv.properties` file of Media Delivery Service.

Settings in Hybris Backoffice	Entry in <code>imconv.properties</code> of the Media Delivery Service
Productpictures original	<code>rendering.productpictures.0.name=original</code>
Productpictures 96Wx96W	<code>rendering.productpictures.1.name=96Wx96W</code>
Productpictures 300Wx300W	<code>rendering.productpictures.2.name=300Wx300W</code>
Productpictures 512Wx512W	<code>rendering.productpictures.3.name=512Wx512W</code>
Productpictures 1200Wx1200W	<code>rendering.productpictures.4.name=1200Wx1200W</code>

Hybris generates an URL containing the name, and Media Delivery Service must have a rule for that name.

For more information, see [“Configuring properties” on page 48](#).

Hybris properties

In the `local.properties` configuration file, the following properties must be defined for Media Delivery Service:

- The URL strategy `otmmUrlStrategy` must be assigned for the media folder.
With the `otmm.media.servlet.url` property, the first part of the generated URL will be defined.

```
otmm.media.servlet.url=http://localhost:${tomcat.http.port}/
ImConvServlet/imconv
```

The generated URL is created by the following scheme: `<otmm.media.servlet.url>/<assetId>/<name>`

Media Delivery Service can either run within the Hybris Tomcat or within a separate Tomcat 8 deployment.

For Media Delivery Service, the URL typically looks like this:

```
http://<host>:<port>/ImConvServlet/imconv
```

For OpenText Adaptive Media Delivery Service the URL looks like:

```
http://<host>:<port>/adaptivemedia/rendition
```

- The `otmm.media.servlet.pwd` property specifies the Media Delivery Service password as defined in the `imconv.properties` file with the `IMCONVPWD` parameter or is empty, if the password is specified in the **Hybris Backoffice, DAM by OpenText > Logins**, with the identifier **Media Delivery Service**.
- The `otmm.media.servlet.prepopulate` property specifies the asset delivery creation behavior. During asset assignment, if Media Delivery Service is enabled, Hybris sends requests to Media Delivery Service to generate the asset deliveries early. You can disable this by setting the property to `FALSE`. Then the asset deliveries will be created when needed. This parameter must be `TRUE` if `useCloudUrl` is set to `TRUE`.

- The `otmm.media.servlet.useCloudURL` property specifies where the generated URLs should point to:
 - **FALSE:** The URL points to Media Delivery Service. The service checks if the wanted asset deliveries exist locally or in the cloud. If not, compute them, read them and return the contents to the caller.
 - **TRUE:** The URL points directly to the cloud, bypassing Media Delivery Service. Media Delivery Service gets no image requests any more, and thus the asset deliveries must have been created during asset assignment, with `populate=TRUE`. DAMLink Hybris issues a cache request to the servlet during an asset assignment, so the asset delivery should always exist. But if the wanted asset delivery is not in the cloud, an error results.
- The `otmm.media.servlet.cloudUrlTemplate` property specifies the pattern for cloud URLs. Hybris generates the cloud URLs according to this template. If it is not set, it obtains the template directly from Media Delivery Service, whose URL and password are specified above, and which must be configured to use the cloud.
 An Azure template has the form `http://XXX.core.windows.net/<container>/%P%/R%`.
 An AWS template has the form `http://XXX.amazonaws.com/<bucket>/%P%/R%`
 where `%P%` is replaced by a path and `%R%` by a file name.
 Once the servlet has written asset deliveries into the cloud, you can see the URLs in the Azure or AWS console.
- The `otmm.media.servlet.useAdaptiveMediaDelivery` property specifies if the URL provided by `otmm.media.servlet.url` points to the Adaptive Media Delivery servlet of the OTMM server.
 Set it to **FALSE** if you use Media Delivery Service.
- If using the Adaptive Media Delivery servlet, the `otmm.media.servlet.amd.clid` property can specify the client ID (`clid`) to be used in AMD requests. The default is `SAPDAM`. The client ID is used by the Analytic capabilities of the Adaptive Media Delivery.



Example 3-1: Media Delivery Service configuration

The following example defines an early 700Wx700H rendering with Media Delivery Service:

Asset definition in Hybris Backoffice:

The screenshot shows the configuration page for a media delivery service. The title bar indicates '300Wx300H (Media Delivery Service)'. There are tabs for 'Common' and 'Administration'. The 'Common' tab is active, showing fields for 'Name' (300Wx300H), 'Type' (MediaDeliveryService), 'Media Format Qualifier', 'Media Folder' (otmmRoot - otmmRoot), 'Display Name', 'Is Default' (radio buttons for True, False, N/A), 'Rendition Set' (productpictures), 'Command', 'Fixed MIME Type', and 'MIME Type Attribute'. There are 'Refresh' and 'Save' buttons at the top right.

Settings in local.properties:

```
# Define URL Strategy for folder "otmmMediaServlet"
media.folder.otmmMediaServlet.url.strategy=otmmRoot
otmm.media.servlet.url=https://myhybris:9632/ImConvServlet/imconv
```

Settings in imconv.properties

```
otmm.rendering.productpictures.1.name = 300Wx300H
```

Using this rendering configuration and the asset ID 123abc, the created URL is:

```
https://myhybris:9632/ImConvServlet/imconv/123abc/300Wx300H?
use=productpictures&assetDescr=camera-158259
```

In the OpenText Media Editor, the rendered assets, URL and folder are displayed:

The screenshot shows the 'Edit OpenTextMedia' dialog box. It contains fields for 'Mime type' (image/jpeg), 'URL' (with a camera image preview), 'Download URL' (https://myhybris:9632/ImConvServlet/imconv/af831dc0c8502c04736f160e8ac5c571a716afb6/300Wx300H?use=productpictures&assetDescr=camera-158259), 'Folder' (otmmRoot), and 'Media format' (300Wx300H). There are expand/collapse icons next to the Folder and Media format fields.



3.1.1.2 ContentByExport type

If a new asset is assigned, with ContentByExport, the selected image goes through an OTMM process that generates the configured asset deliveries. For this process, the included ImageMagick tool is used to generate the asset deliveries and exports the asset deliveries to an FTP server.

The asset deliveries can simply be defined by fixed size. But there is also the possibility to formulate graphic transformations provided by ImageMagick.

3.1.1.3 External URL type

When you use the **External URL** type, the assignment saves a URL to the asset delivery at the media item. Use this method for media items with very big content for example HD videos. You can just set an URL to the content for example as a URL to a video streaming server. OTMM supports several video streaming servers. See the OTMM documentation for details on the supported streaming servers.

With the **External URL** type, an export to the FTP server is triggered, if the **URL Attribute** field or the **MIME Type Attribute** field is specified.

If you use one of the following placeholders, you do not need to specify the **URL Attribute**:

- {2} Asset ID
- {3} Original asset ID
- {4} Asset name



Example 3-2: Video with external URL type

The following example assumes that IIS is configured as video streaming server for OTMM. Usually, there is a master and a low resolution video generated when you upload a video to OTMM, which is configured for video streaming. To use these video streams as URLs for Media items you need to configure the following asset deliveries:

Name	LOW_RES
Type	External URL
URL Attribute	UOIS/VIDEO_URLS/LO_RES_LOCATION
MIME Type Attribute	UOIS/VIDEO_URLS/LO_RES_MIME_TYPE
Template String	{proxy}

Name	ORIG_RES
Type	External URL
URL Attribute	UOIS/VIDEO_URLS/MASTER_LOCATION

MIME Type Attribute	UOIS/VIDEO_URLS/MASTER_OBJ_MIME_TYPE
Template String	{proxy}

Create an assignment spec with the following settings:

Type

Product

Attribute

Others

Content type

VIDEO

Asset delivery definitions

LOW_RES

ORIG_RES



3.1.1.3.1 URL template string placeholders

With the External URL asset delivery mode, you can use the following placeholders in the field for the URL template string:

Placeholder	Description
{0}	The OTMM server URL prefix It consists of the following properties: <otmm.server.scheme>: //<otmm.server.domain>: <otmm.server.port>
{1}	With URL attribute , you can specify an attribute within assetProperties.xml . The value of this attribute is used for this placeholder.
{2}	Asset ID
{3}	Original asset ID
{4}	Asset name
{proxy}	Depending if the URL defined by URL attribute is absolute or relative, it replaces protocol, host and port by a proxy or it appends the proxy to make the URL absolute. The value is defined by otmm.url.template.proxy property.
{proxy:<ProxyUrl>}	Like {proxy} but uses <ProxyUrl> instead of URL defined by otmm.url.template.proxy for example {proxy:http://www.opentext.com}

**Example 3-3:**

Example URL template string with placeholders:
 {0}/otmmapi/v3/renditions/{1}



Note: An `assetProperties.xml` file is generated by OTMM during export and saved in the export directory. It contains metadata about the exported asset and its asset deliveries. The contents can be adapted by changing the asset model(s) you defined. You can look up the content of this file by starting an export manually with the OTMM web UI.

You can manually start an export in OTMM to see how the `assetProperties.xml` will look like.

Attribute path

In Hybris Backoffice, **URL attribute** and **MIME type attribute** define a path to an XML attribute. The path is defined relative to the METADATA tag: `tag1/tag2/.../tagN/attribute`.

For example

```
UOIS/THUMB_NAIL_OBJ_ID
<METADATA>
  <UOIS THUMB_NAIL_OBJ_ID="706e49f64a81fb26c5c453a9317b2cd90ae74fe7" ...>
    ...
  </UOIS>
</METADATA>
```

3.1.2 Configuring an assignment spec

An assignment spec assigns one or more asset deliveries to a Hybris type, attribute, and content type.

To create an assignment spec:

1. In Hybris Backoffice /backoffice, click **DAM by Opentext > Assignment Configuration > Assignment Spec**.
2. Click **+ Assignment Spec** to create a new assignment spec.
3. The properties of the new asset delivery are on the **Common** tab. With **Type**, **Attribute**, and **Content Type** you specify the Hybris items to assign assets to. With **Asset delivery definitions**, you specify the properties the assets will have.

Content Type	Type	Attribute	Asset Delivery Definition(s)
VIDEO	Product [Product]	others	test
BITMAP	Product [Product]	galleryImages	1200Wx1200H, 515Wx515H, 300Wx300H, 96Wx96H, 65Wx65H, 30Wx30H
BITMAP	Product [Product]	others	1200Wx1200H, 515Wx515H, 300Wx300H, 96Wx96H, 65Wx65H, 30Wx30H

BITMAP.Product.galleryImages

Refresh Save

Spec Administration

Type ? Product [Product]

Attribute ? galleryImages

Content Type ? BITMAP

Asset Delivery Definition(s) ?

1200Wx1200H (Media Delivery Service)
515Wx515H (Media Delivery Service)
300Wx300H (Media Delivery Service)
96Wx96H (Media Delivery Service)
65Wx65H (Media Delivery Service)
30Wx30H (Media Delivery Service)

Trigger implicit assignment to ?

Implicit assignment

The **explicit attribute** is the item attribute for which a user triggers an assignment explicitly for example by starting the Asset Assignment Dialog in Product Cockpit within the attribute and choosing an image from OTMM. During this assignment, implicitly also other asset deliveries of the chosen asset can be assigned to other attributes. These attributes are called **implicit attributes**.

Configure implicit assignment to attributes with **Trigger implicit assignment to**.



Example 3-4: Images: Implicit attribute assignment

Let's assume, there is an assignment spec for the explicit attribute picture of type Product for the content type BITMAP.

Add these assignment specs to the **Trigger implicit assignment** field of the Product.picture assignment spec.

To create asset deliveries for the attributes normal and thumbnails, add normal and thumbnail to the **Trigger implicit assignment** field of the assignment spec.

If now a BITMAP asset is assigned to the explicit attribute picture, asset assignments are created for the attributes normal and thumbnails as well.



Note: The mapping to an existing MediaContainer is deprecated and equivalent configuration can be done with asset delivery.

3.2 Online property changes

It is possible to change or add properties online, that is in a running system, with the **Hybris administration console**. These property changes take effect immediately. However, they are not persisted in the properties file. You have to edit the properties additionally in the properties file otherwise they are lost after next Hybris restart.

Therefore, this approach is mainly useful for testing purposes.

To change a property or to add a new property in a running system:

1. Open the **Hybris administration console**. Under **Platform > Configuration**, all properties are available. Filter for the property you want to change and enter the value. Click the green check mark to save the value.
2. To add a new property, add it first with the **Add** button and change the value then.



Note: The online change **does not work** for the properties:

- `otmm.server.host`
- `otmm.server.scheme`
- `otmm.server.port`
- `otmm.server.contextpath`
- `otmm.server.notification.port`
- `otmm.server.login`
- `otmm.server.sso`
- All log configuration properties starting with `log4j2`

3.3 Exporting and importing an assignment configuration

Use the export import mechanism to transfer an assignment configuration from a staging or a test environment to a productive system.

The export or import of an assignment configuration is performed with the Hybris Backoffice ImpEx mechanism.

To export an assignment configuration:

1. In Hybris Backoffice, select **System > Tools > Export**.
2. At **ImpEx Media**, search for the `OtmAssetAssignmentExportScript.impex` script. Just enter the first letters of the name of the script.

3. On the **Results** page, download the exported ZIP file.

To import an assignment configuration:

1. In Hybris Backoffice, select **System > Tools > Import**.
2. Upload the ZIP file that you exported previously.

The import does the following:

- Create required Media Folders. If a Media Folder with the same qualifier exists, it remains untouched.
- Create asset deliveries. If an asset delivery with the same name exists, it is updated.
- Create assignment specifications. If an assignment specification with the same otmmItemType (Type), otmmExplicitAttribute (Attribute), and otmmContentType (Content Type) exists, it is updated.

3.4 Using Media Delivery Service

DAMLink Hybris provides Media Delivery Service. This service is introduced as core of a high-performance media provisioning. It is able to support a large number of Hybris webstores end users efficiently.

By default, the OTMM server is used as storage medium, but Media Delivery Service offers the possibility to use cloud storages also.

Features Media Delivery Service provides the following features:

- On-the-fly provisioning of media
- High performance front store user support
- Use of cloud storages
- Original, thumbnail and preview of an asset
- Support of any types of assets
- Different sets of asset deliveries for different usage contexts
- Robustness, efficiency and security
- Serving a huge number of clients in parallel by appropriate multi-threading support
- The URLs and HTTP responses support web browser and (reverse) proxy caching best and have always the same URL for one and the same asset delivery
- Requests can be called by command line also.

Package Media Delivery Service comes in a package which consists of the servlet (`ImConvServlet.war`) to be deployed in a servlet container, a jar-file containing code to be used as a command line tool, and a few batch scripts for Unix and Windows.

The installation of Media Delivery Service package is described in “[Installing Media Delivery Service](#)” on page 20.

3.4.1 Rendition production

The main purpose of the service is to produce and store renditions of images. For example, if OTMM has stored a high resolution image, the servlet can produce several smaller versions of this image for thumbnail, preview and so on.

To compute the renditions, the service uses the ImageMagick tool (<http://www.imagemagick.org>). Multiple renditions can be combined to a rendition set. This set is computed during one ImageMagick call (instead of calling the conversion for each rendition) to avoid performance loss due to program loading and starting. On some operating systems the ImageMagick code is called via JNI, again to avoid performance loss due to program loading and starting.

If ImageMagick can not be called via JNI, the service falls back to a mode where the convert tool of ImageMagick is called as a subprogram.

Renditions Normally, a rendition is just a resize with a given resolution, for example with the ImageMagick parameters `-resize 96x64`. But more complex ImageMagick commands can be called as well, for example `-resize 800x800 -pointsize 20 -draw "gravity south fill black text 0,24 'Copyright MyCompany' "` not only resizes the image, but draws a copyright at the bottom of the image. This is the reason to call it renditions, not resolutions. A rendition is defined as a visual representation or reproduction.

OTMM stores normally not only the image (asset), but also preview and thumbnail pictures. It does this also for assets that are not pictures, for example PDF files.

For these non-image assets, the servlet gets:

- the original, if `original` is configured for the rendition set
- the thumbnail
- the preview, if not video

For image assets, thumbnail and preview are obtained from OTMM if configured for the rendition set.

Rendition sets The renditions can be grouped into rendition sets. One asset delivery set contains one or more asset deliveries.

URL The servlet accepts an URL asking for a specific resolution of a specific rendition set:

```
http://<host>:<port>/ImConvServlet/imconv/<assetId>/96Wx64H?
use=gallery
```

This URL returns a 96x64 pixel rendition of the image `<assetId>` stored in OTMM with the given asset identifier, but it produces also all other renditions in the rendition set `gallery`, if they do not already exist.

Processing of a request

The given rendition is produced if it does not exist already. Missing renditions in the given rendition set will be computed asynchronously in a sub-thread.

If no rendition set is specified in the URL, the rendition set `default` is used. If the rendition set `default` is not configured, an error is returned. An error is also returned if the rendition set is specified in the URL, but not configured.

Rendition store

All renditions are stored under a configurable root directory. To avoid memory problems caused by too many files in a directory, subdirectories are used. Below the root directory, three levels of directories are used with names from 00 to 99.

Under the third level directory, the asset directories are placed, named by asset ID. In these directories, the original file (`<assetId>.<suffix>`) is stored and the computed renditions with the name `<assetdeliveryname>.jpg`.

**Example 3-5: Stored renditions**

For example, with the root directory `E:/imconvroot` the rendition `a453e3dd123da7f8315bda929954a8cf71e69dbb` is placed under the following path:

```
E:/imconvroot/05/85/15/  
a453e3dd123da7f8315bda929954a8cf71e69dbb
```

This directory contains the original file `a453e3dd123da7f8315bda929954a8cf71e69dbb.jpg` and the renditions `30Wx30H.jpg`, `700Wx700H.jpg`, `copyright.jpg`, `preview.jpg`.

**Command line tool**

The servlet accepts some more requests, used to get the configuration of the servlet, to delete asset folders or to write an original file to it (instead of getting it from OTMM). These requests are called by the command line tools `bulkImport` and `deleteAsset`. For more information, see [“Using command line tools” on page 52](#).

3.4.2 Configuring properties

The Media Delivery Service properties are saved in the configuration file `webapps/ImConvServlet/WEB-INF/resources/imconv.properties`.



Note: Changes in the configuration file will get effective without a restart of Media Delivery Service 1 minute after the changed configuration file was saved, by the latest.

If the `INFO` log level is enabled, in the `imconvservlet.log` log file you will see the line

```
rereading imconv.properties
```

The configuration file looks like this, after the war file is unpacked:

```
# The following password is used by bulkImport and deleteAsset,  
# edit bulk.*/deleteAsset.* accordingly  
IMCONVPWD=aPassword
```



```

# OTMM user name, e.g. tsuper
USER_ID=OTMM_User

# OTMM user password
PASSWORD=OTMM_Password

# URL of OTMM server
MEDIAMANAGER_URL=host:port # e.g. http://otmm.company.net:11090

# Set to true, if original file shall be deleted after conversion
delorig=false

# directory into which the renditions are written
rootdir=x:/some_directory

# List of mime types that are rendered (with ImageMagick)
# according to otmm.rendering entries below.
# E.g. files with suffix .jpg have mimetype image/jpeg and
# are rendered, thumbnails and previews are obtained from
# OTMM, if configured below. You may add to this list more
# of the many image formats that ImageMagick can handle.
# Normally, renditions are done to jpg, but with renderedTo you
# can specify an other output format.
# With IMsuffix, ImageMagick is instructed to convert layer 0 of a .psd file.
#
rendered.0.mimetype=image/jpeg
rendered.0.suffix=.jpg
rendered.1.mimetype=image/gif
rendered.1.suffix=.gif
rendered.2.mimetype=image/tiff
rendered.2.suffix=.tiff
rendered.3.mimetype=image/png
rendered.3.suffix=.png
rendered.3.renderedTo=.png
rendered.4.mimetype=application/photoshop
rendered.4.suffix=.psd
rendered.4.IMsuffix=[0]

# List of mimetypes that are not rendered, but preview's
# and thumbnail's are (always,
# implicitly) obtained from OTMM.
notrendered.0.mimetype=application/pdf
notrendered.0.suffix=.pdf
notrendered.1.mimetype=video/mp4
notrendered.1.suffix=.mp4

# Mime types not in this list cause an error!

# Rules for otmm.rendering properties:
# The "name" entries have the form
# otmm.rendering.<rendition set name>.number.name.
# All names must be unique.
# The names must have consecutive numbers. A gap in name numbers
# makes names with higher numbers invisible.
# If a gap exists, you can fill it with the name skip, e.g.
# otmm.rendering.default.20.name=skip.
# name may be original, preview, thumbnail, skip, a resolution
# (nWxmH), or a general name.
# command may be nonexistent, or one of original, preview, thumbnail,
# or an ImageMagick command.
# The name will be the file name, under which the rendition will
# be stored, and is used in the URL to access this rendition.
# The one exception is original, where the
# file name is asset id.suffix, not original.suffix

```

```
# (suffix = .jpg, .png, ...).
# If the name is original, this implies the command original,
# and command may be non-existent,
# or must also be original.
# The names preview and thumbnail imply the commands preview
# and thumbnail, if and only if
# the command is nonexistent.
# If the name is a resolution nWxmH, and if the command is
# empty, this implies the ImageMagick command -resize nxm.

# The commands original, preview and thumbnail load original,
# preview and/or thumb-nail from OTMM.

# If any ImageMagick command is implied or explicitly specified,
# and original is not specified, then original is implied
# (we need the original to compute the renditions).
# I.e. the original image is not fetched, if and only if only
# preview and/or thumb-nail are specified.

# examples:
# i.name=thumbnail1, i.command=thumbnail, j.name=thumbnail2,
# j.command=-resize 64x64
# means one thumbnail is obtained from OTMM,
# one is computed from original

# i.name=166Wx166H, no i.command means the same as
# i.name=166Wx166H, i.command=-resize 166x166


# i.name=preview, no i.command means the same as
# i.name=preview, i.command=preview

otmm.rendering.default.0.name=original
otmm.rendering.default.1.name=166Wx166H
otmm.rendering.default.2.name=copyright
otmm.rendering.default.2.command=-resize 800x800 -pointsize 20
    -draw "gravity south fill black text 0,24 'Copyright MyCompany'"
#otmm.rendering.default.3.name=thumbnail
#otmm.rendering.default.4.name=preview

otmm.rendering.gallery.0.name=original
otmm.rendering.gallery.1.name=332Wx332H
otmm.rendering.gallery.2.name=skip
otmm.rendering.gallery.3.name=700Wx700H
```

To configure the servlet:

1. Edit the configuration file according to your needs:
Define a new IMCONVPWD and edit `bulk.bat/deleteAsset.bat` accordingly.

 **Note:** It is recommended to use a strong password.
2. Choose an OTMM user/password. Currently, access to OTMM assets is only regulated by proper choice of this user. The servlet makes all assets that this user can access publicly available.
3. Define the root directory. The `rootdir` must point to a directory on a file system with enough space to store all asset deliveries that the servlet might ever compute.

For information about security aspects, see [“Managing Media Delivery Service user permissions” on page 109](#).

3.4.3 Sending requests

Requests to the servlet can be send in a browser via URL or using command line tools.

Get / Delete renditions

With a browser, you can get images from Media Delivery Service with the following URL:

```
http://host:port/ImConvServlet/imconv/<assetId>/name?use=renditionset
```

where name must be one of the names in `imconv.properties`.

In the root directory, you should see how more and more files are created there.

A first access to an asset ID may be slow, each further access to the same asset ID should be fast.

To delete the files, the `deleteAsset.bat` (resp. `deleteAsset.sh`) script can be used. The files `bulk.*`, `deleteAsset.*`, `imconvtool.jar` can be stored in any directory. To delete a whole asset directory from the root directory, you can, for example, call `deleteAsset.bat -imconv host:port assetid`. Repeating the above browser request should bring it back.

If you delete some renditions or the whole directory manually with a file explorer, the next browser call brings them back. Existing asset deliveries are not written again.

The rendition that the URL asked for is always computed first and returned to the caller immediately, while the other renditions are computed asynchronously.

Cache renditions

You may want to precompute the certain renditions, so that the first “customer” access to a rendition does not need to trigger the creation of the rendition. One way to do this is just to access the renditions with HTTP Get requests as described above. If you choose the special rendition name “cache”, then the renditions are computed synchronously, and no content is transferred. For example the following URL causes the servlet to compute all asset deliveries of the specified rendition set, and returns just success or failure:

```
http://host:port/ImConvServlet/imconv/<assetId>/cache?use=renditionset
```

This caching can be done with the `bulkImport` command tool also. It causes the `ImConvServlet` to preload assets from the specified OTMM folders. The tool is called via the scripts `bulk.bat` or `bulk.sh`, respectively. For more information, see [“Using command line tools” on page 52](#).

3.4.4 Using command line tools

The Media Delivery Service accepts the `bulkImport` and `deleteAsset` commands to write a file or delete an asset folder. The commands are used to get the configuration of the Media Delivery Service instead of getting it from OTMM. The commands can be called with the `bulk.bat/bulk.sh` and `deleteAsset.bat/deleteAsset.sh` scripts. These scripts are delivered with the Media Delivery Service package.



Important

The requests are password protected. The scripts used to invoke the tools must be edited before first use to set the same password that the servlet uses.

bulkImport The usage of `bulk` is the following:

```
bulk.bat -par <integer> -imconv <host:port> -rdset <name>
        [ -root <directory> ][-impex <propfile>]
```

The available parameters are:

-b	Run OTMM bulkimport job first, then like -d
-folder <folderId>	Pre-existing folder to add the assets into (-b), or get them from (-d)
-impex <propfile>	property file with rules for ImpEx file
-count <integer>	Number of import files processed per database transaction
-d	Store image/assetId tuples to ImConvServlet
-dir <directory>	Directory containing the import files
-f	Store one or more folders to ImConvServlet
-h, --help	print this message
-imconv <host:port>	host/port of ImConvServlet
-par <integer>	Number of parallel threads
-password <OTMM user password>	Media Management user password (if missing, obtain from servlet)
-profile <id name>	Metadata profile ID or Name to apply to the new assets (optional)
-rdset <name>	Rendition set name (name after otmm.rendering in imconv.properties)
-root <directory>	Path to rootdir of ImConvServlet containing the computed renditions
-user <OTMM user>	Media Management user to create the import job (if missing, obtain from servlet)
-v	Version info

Operation modes The following **modes** of operation are available:

- **-b**: Run an OTMM bulkimport first, then continue as with -d. Use the following parameter set:

```
-b -dir <directory> -folder <folderId> [-profile id|name] [-user <user>] [-password <password>] [-count <integer>]
```

Using the **-b** parameter, the `bulkimport` command is called with fix parameters `-mode files`, `-nodelete`. For example:

```
$TEAMS_HOME/bin/bulkimport.bat -dir <directory>
    -folder <folderId>
    -mode files [-profile <profile>]
    -user <user> -password <password>
    [-count <count>] -nodelete
```

If that does not suit, you may as well call the OTMM bulkimport before and then use option -d.

- **-d:** List all assets in the OTMM folder. If an image in the directory matches the asset, then image and assetId are sent to the servlet, to compute and store renditions there.

Use the following parameter set:

```
-d -dir <directory> -folder <folderId>
```

- **-f:** List all assets in the folder(s). For each asset, the servlet will load the image from OTMM, then compute and store the renditions of the image, only as needed.

Use the following parameter set:

```
-f folderId ...
```

- **-i:** Prepare just an ImpEx file from the OTMM folder. The -impex and -folder options are mandatory.

Use the following parameter set:

```
-i -folder <folderId>
```

ImpEx files produced by bulkImport are used to bulk assign OTMM assets to Hybris. An ImpEx file consists of a database insert statement, for example

```
INSERT OtmmAssignmentStatus; otmmProductCode; otmmCatId; otmmCatVersion;
otmmAttributeId; otmmAssetId; otmmOriginalAssetId; otmmMimeType;
otmmModelType; otmmAssetName; otmmContentType; otmmStatus(code)
```

followed by one line of values for each asset. The ImpEx property file passed with -impex determines how the values are computed. The idea is, that there are separate folders for example for product, shop and banner images, and that for each folder a separate ImpEx property file exists, for example product.properties, shop.properties and banner.properties. Each property file assigns different attributes to the asset, and specifies different rendition sets.

It is more efficient to send the image to the servlet (-d) than to let the servlet get the image from OTMM (-f).

If parameter -root is given, then this tool computes all asset deliveries itself in this directory. The root directory can be a remote path to the servlet's root directory, or a local directory, then you must later copy it over yourself.



Note: The bulk import tool is not idempotent. If you call it with the -b parameter twice, then the images will be loaded twice! The -d/-f parameters ensure that the bulk import can be called multiple times without changing the result.



Example 3-6: Bulk import

You have a directory with product images in e:/productImages, and an OTMM folder f123.

Then you may call:

```
bulk.bat -b -dir e:/productImages -folder f123 -par 5 -imconv host:port -rdset default
```

to import the images into OTMM, then send the images to Media Delivery Service and compute the renditions of the default asset delivery set. Or, if the OTMM folder contains images from the directory, you may call

```
bulk.bat -d -dir e:/productImages -folder f123 -par 5  
-imconv host:port -rdset default
```

Or, you may compute all renditions in the folders f123 and f234 into a local directory E:/renditions like this:

```
bulk.bat -f -root e:/renditions f123 f234
```

To prepare an ImpEx file:

```
bulk.bat -i -folder f123
```

An example ImpEx file for product assets:

```
# With impex.stem, we map file suffixes of assets without versions  
# (i.e. the "main" asset, like hammer.jpg) to an attribute (to be stored  
# as  
# otmmProductCode). The numbers must be sequential, to fill any gaps,  
# you may use impex.stem.i.suffix=skip  
impex.stem.0.suffix=.jpg  
impex.stem.0.attribute=picture  
impex.stem.1.suffix=.png  
impex.stem.1.attribute=picture  
impex.stem.2.suffix=.psd  
impex.stem.2.attribute=picture  
impex.stem.3.suffix=.pdf  
impex.stem.3.attribute=data_sheet  
impex.stem.4.suffix=.mp4  
impex.stem.4.attribute=detail  
  
# with impex.versioned, we map file suffixes of assets with versions  
# (like ham-mer_2.jpg)  
# to an attribute (to be stored as otmmProductCode).  
impex.versioned.0.suffix=.jpg  
impex.versioned.0.attribute=galleryImages  
  
# this statement is written as first line of the impex file  
#INSERT OtmmAssignmentStatus;  
otmmProductCode;otmmCatId;otmmCatVersion;otmmAttributeId;otmmAssetId;otmmOriginalAssetId;otmmMimeType;otmmModelType;otmmAssetName;otmmContentType;otmmStatus(code)  
  
# In this section we map fields to values. The values are literals,  
# if not starting with impex.  
# impex.stem means here the "stem" of the filename,  
# i.e. "hammer" for both,  
# hammer.jpg and hammer_2.jpg.  
# impex.attribute is the attribute as derived from the impex.stem  
# and impex.versioned  
# rules above, dependent on the file suffix, and if the file  
# is versioned or not.  
# The other impex. values are taken from the OTMM asset description.
```

```

otmmProductCode=impex.stem
otmmCatId=hwcatalog
otmmCatVersion=Staged
otmmAttributeId=impex.attribute
otmmAssetId=impex.assetId
otmmOriginalAssetId=impex.originalAssetId
otmmMimeType=impex.mimetype
otmmModelType=OPENTEXT.PRODUCTS
otmmAssetName=impex.fileName
otmmContentType=impex.contentType
otmmStatus(code)=waitingForSync

# If you specify a rendition set here, you must not specify the
# rendition set with the -rdset parameter.
renditionSet=default

# The default for the char separating the version from the stem
# is _, e.g. ham-mer_2.jpg.
# With versionDelim you can specify a different char,
# e.g. - for hammer-2.jpg.
versionDelim=-

# With this specification, an OTMM folder with the files
# prodA.jpg, prodA_2.jpg, prodA_3.jpg, prodA.pdf, prodA.mp4
# produces these lines:
;prodA;hwcatalog;Staged;picture;a1;a1;image/
jpeg;OPENTEXT.PRODUCTS;prodA.jpg;BITMAP;waitingForSync
;prodA;hwcatalog;Staged;galleryImages;a2;a2;image/
jpeg;OPENTEXT.PRODUCTS;prodA_2.jpg;BITMAP;waitingForSync
;prodA;hwcatalog;Staged;galleryImages;a3;a3;image/
jpeg;OPENTEXT.PRODUCTS;prodA_3.jpg;BITMAP;waitingForSync
;prodA;hwcatalog;Staged;data_sheet;a4;a4;application/
pdf;OPENTEXT.PRODUCTS;prodA.pdf;ACROBAT;waitingForSync
;prodA;hwcatalog;Staged;detail;a5;a5;video/
mp4;OPENTEXT.PRODUCTS;prodA.mp4;VIDEO;waitingForSync

```

In this example, prodA.jpg is the main picture, with an attribute picture, prodA_2/_3.jpg are gallery images, prodA.pdf is a data sheet, and prodA.mp4 exhibits details. They all belong to the product with code prodA. If the rules do not cover everything, then the resulting ImpEx file (called folder[_i].impex) contains values with the letters XXX. These values can then be edited manually.

If the ImpEx property file contains a value for renditionSet, and the -impex parameter is specified, then the -rdset parameter is not necessary.



Delete assets

The URL to delete assets from the servlet is password protected. You may use the deleteAsset.bat or deleteAsset.sh script to delete assets from Media Delivery Service. Deleting an asset means deleting the asset directory.

The usage of deleteAsset.bat is the following:

```
deleteAsset.bat -imconv <host:port> [ -folder <folderId> ] [ assetId ... ]
```

Either all asset IDs retrieved from the OTMM folder, or the asset IDs specified as parameter(s) are deleted.

The parameters are:

-folder <folderId>	Folder whose assets are to be deleted from the servlet
-h, --help	print this message
-imconv <host:port>	host/port of ImConvServlet
-password <OTMM user password>	Media Management user password (if missing, obtain from servlet)
-user <OTMM user>	Media Management user to create the import job (if missing, obtain from servlet)



Example 3-7: Delete assets

Delete assets a123 and a234:

```
deleteAsset.bat -imconv host:port a123 a234
```

Delete all assets contained in OTMM folder f123 from servlet (not from OTMM!):

```
deleteAsset.bat -imconv host:port -folder f123
```



3.4.5 Using cloud storage

Media Delivery Service can be configured to support custom cloud storage. When a cloud is enabled, all newly computed renditions will be copied into the cloud. A local hard disk is only used temporarily for asset delivery creation.

Also the bulk utility supports the creation of renditions on cloud storages.

Out of the box supported cloud storages are **Microsoft Azure Blobs (Azure)** and **Amazon Web Services (S3)**.



Note: Support of other cloud storages:

There is an API available to support other cloud storage providers by custom implementations. For more information, see *OpenText DAMLink for SAP Solutions Hybris Integration - Customization and Programming Guide (DAMSAP-PHI)*.

To enable cloud support:

1. **Configuration** - Set the **cloud** parameter in `imconv.properties` configuration file:

For **S3**, the following parameters are expected:

```
cloud=s3
S3Bucket=<bucket name>
S3Region=<region, optional, e.g. eu-central-1>
```

For **S3**, the credentials are passed in `$HOME/.aws/credentials`, for **Azure** in `$HOME/.azure.properties`.
`$HOME` defines the the home directory of the user that runs Media Delivery Service.

For **Azure**, the following parameters are expected:

```
cloud=azure
AzureContainer=<container name>
```

For Azure, the credentials are passed in `$HOME/.azure.properties`.

2. When the cloud is enabled, all newly computed renditions are copied into the cloud. You can specify if you want to keep the files in the root directory, or remove them, after they are copied, with the **dellocal** parameter in `imconv.properties`.

You can enforce to copy all files from root directory into the cloud by calling the **syncAll** tool. Only the files that are not yet in the cloud are copied, so when you call **syncAll** a second time, nothing is copied.

3. Configure the necessary properties for Media Delivery Service on Hybris side in the `local.properties` configuration file. For more information, see [“Media Delivery Service type” on page 37](#).

*Cloud
registering*

Also, when you register with the cloud, you get credentials, which are or must be stored somewhere, normally in a user specific directory. This is normally done automatically, but you will also see appropriate error messages in `imconvservlet.log`, when the credentials are missing or other problems occur. If the user which was used when the cloud was registered is not the same or was on another machine, then it may be necessary to copy the credentials over.

3.4.6 Version info and logging

Media Delivery Service and the command line tool have their own software version and build number. Recorded logs are stored in a log file. You can configure the location of the log file.

To get the version info:

1. To get the version of Media Delivery Service, enter this URL:
`http://<host>:<port>/ImConvServlet/imconv/version`.
Media Delivery Service writes the version info to the log file.
2. To get the version of the command line tool use the `-v` option of the command:
`bulkImport -v`

Logging

The log configuration file for Tomcat deployments is located in `<TOMCAT_ROOT>/webapps/ImConvServlet/WEB-INF/classes/log4j.properties`.
By default, the log file `ImConvServlet.log` is located in `<TOMCAT_ROOT>/logs`.

If you use Tomcat within Hybris, the log configuration file is located in `<HYBRIS_ROOT>/bin/platform/tomcat/webapps/ImConvServlet/WEB-INF/classes/log4j.properties`.
By default, the log file `ImConvServlet.log` is located in `<HYBRIS_ROOT>/bin/platform/tomcat/logs`.



Tip: To change the log directory to the standard Hybris log directory, change the following configuration setting
`log4j.appender.A0.File=${catalina.base}/logs/ImConvServlet.log` to
`log4j.appender.A0.File=${catalina.base}/../../../../log/ImConvServlet.log`

3.5 Configuring Hybris Product Cockpit editor

Core components of **DAMLink Hybris** are the two cockpit editors, the **OpenText Media Reference Editor** and the **OpenText MediaCollection Reference Editor**.

These Hybris cockpit editors can be configured to be used for any Media related attribute of any Hybris item type like Product or CMS component types using the standard Hybris editor configuration mechanism.

For information about importing cockpit configuration, see the **Hybris Commerce Suite** product documentation: Hybris wiki (<https://wiki.hybris.com>).

Default Editor Configuration

After the installation of **DAMLink Hybris** and running a Hybris platform update as the final installation step, the editors are registered by default in the Product Cockpit for the following product attributes:

- **Media Reference Editor** for picture and thumbnail attributes
- **MediaCollection Reference Editor** for galleryimages, detail, logo, normal, others and thumbnails attributes

In addition, **OpenText MediaCollection Reference Editor** will be registered for attribute media of item type MediaContainer. This is required because galleryImages is a list of MediaContainers. And with this configuration you are able to add certain assets to one of these existing MediaContainers.

The **OpenText Media Reference Editor** and **OpenText MediaCollection Reference Editor** are then launching the **Asset Assignment Dialog** to assign OTMM assets as Media Items to the respective item's attribute.

Automatic Configuration Import

The out of the box configuration is provided by **DAMLink Hybris** through the files:

```
resources/otmmaddonui/productcockpit/xml/  
Editor_Product_ProductManager.xml  
resources/otmmaddonui/productcockpit/xml/  
Editor_MediaContainer_ProductManager.xml
```

This cockpit editor configuration files are loaded automatically during a Hybris platform initialization or update through the `productcockpit_config.impex` file of the **otmmaddonui** extension:

```
resources\impex\productcockpit_config.impex
```



Note: For more information about the Hybris ImpEx files and customization, see the Hybris documentation: *“Further documentation sources”* on page 11.

The import creates:

- a CockpitUIScriptConfigMedia item containing the Editor_Product_ProductManager.xml file
- a CockpitUIScriptConfigMedia item containing the Editor_MediaContainer_ProductManager.xml file

The CockpitUIScriptConfigMedia item for item type Product is then used in various CockpitUIComponentConfiguration items to configure the editors for the item type Product as well as the following sub types of product:

- VariantProduct
- ApparelStyleVariantProduct
- ApparelSizeVariantProduct
- ApparelProduct
- MensWear
- Product.Hardware

As MediaContainer has no subtypes, the CockpitUIScriptConfigMedia item for item type MediaContainer will only be used for MediaContainer.



Note: If you changed the configuration for existing item types and attributes, don't forget to reset personalized settings. For more information, see [“Testing the integration” on page 26](#).

*Supporting
custom sub
types*

To customize the editors for supporting custom sub types of the Product item type, additional configuration steps are necessary.

If the custom sub type does not contain additional attributes for Medias, the already existing standard editor configuration (CockpitUIScriptConfigMedia) for product based on the Editor_Product_ProductManager.xml file can be used. This procedure is described in [“Using the default editor configuration” on page 59](#). If the custom sub type contains additional attributes of Media type, then also a new editor configuration has to be created. This procedure is described using an example, see [“Defining a new editor configuration” on page 60](#).

3.5.1 Using the default editor configuration

If the custom sub type of product does not contain additional attributes for medias, the already existing standard editor configuration (CockpitUIScriptConfigMedia) for product based on the Editor_Product_ProductManager.xml file can be used.

To use the default editor configuration:

1. In this case just an additional line has to be added to the INSERT_UPDATE CockpitUIComponentConfiguration section of the essentialdata_otmmaddon.impex file for the custom sub type (for example MyProduct) referring to the existing CockpitUIScriptConfigMedia item:

```
;editorArea;editorConfigurationFactory;MyProduct;productmanagergroup;  
editor_product_productmanagergroup_ui_config_otmm
```

2. To import the changed configuration into Hybris:
Either run a Hybris platform update (`http://<hybrishost>:9001/platform/update`) or import the `productcockpit_config.impex` file through console import:
 - a. Open the following URL in a browser: `http://<hybrishost>:9001/console/impex/import`
 - b. Select **Import Script** and choose the `productcockpit_config.impex` file.
 - c. Click **Import** to finish the import of the custom configuration.



Note: The import of the editor configuration files with the ImpEx Web does not require a Hybris restart. They can be re-imported for example with custom specific changes and will take effect immediately after the user has executed **Reset personalized settings** in the respective cockpit under **Menu > User Settings > Reset personalized settings**.

3.5.2 Defining a new editor configuration

The editor configuration can be adjusted to specific customer needs. If the MyProduct item type contains additional attributes, for example called myMedia of type Media, then also a new editor configuration has to be created.

This section illustrates with the help of an example how the editor configuration can be adjusted.

Example The example works with a custom sub type of the standard Hybris Product item type called MyProduct.

To configure the **Media Reference Editor** for the myMedia attribute and use the **Asset Assignment Dialog** for assigning Media Items based on OTMM assets, follow these steps:

1. In `resources/otmmaddonui/productcockpit/xml` folder, create a copy of the `Editor_Product_ProductManager.xml` file in the same folder called `Editor_MyProduct_ProductManager.xml`.
2. Add the following entry for the myMedia attribute to the MultiMedia group:

```
<property qualifier="myproduct.myMedia"
editor="openTextMediaReferenceSelector">
  <parameter>
    <name>mimeTypees</name>
    <value>image/jpeg;image/gif;image/png</value>
  </parameter>
  <parameter>
    <name>allowCreate</name>
    <value>true</value>
  </parameter>
  <parameter>
    <name>imageHeight</name>
    <value>80px</value>
  </parameter>
</property>
```

```

</parameter>
<parameter>
  <name>restrictToCreateTypes</name>
  <value>Media,OpenTextMedia+</value>
</parameter>
</property>

```

For information about all supported parameters, see [“Setting up an individual Asset Assignment Dialog” on page 67](#).

3. Extend the `productcockpit_config.impex` file with the following entries:
 - a. To the `INSERT_UPDATE CockpitUIScriptConfigMedia` section, add the line:


```

;editor_myproduct_productmanagergroup_ui_config_otmm;true;
text/xml;Editor_MyProduct_ProductManager.xml;
$jarResourceProductCockpit/xml/Editor_MyProduct_ProductManager.xml

```
 - b. To the `INSERT_UPDATE CockpitUIComponentConfiguration` section, add the line:


```

;editorArea;editorConfigurationFactory;MyProduct;productmanager-group;
editor_myproduct_productmanagergroup_ui_config_otmm

```
4. To import the changed configuration into Hybris:

Either run a Hybris platform update (`http://<hybrishost>:9001/platform/update`) or import the `productcockpit_config.impex` file through console import:

 - a. Open the following URL in a browser: `http://<hybrishost>:9001/console/impex/import`
 - b. Select **Import Script** and choose the `productcockpit_config.impex` file
 - c. Click **Import** to finish the import of the custom configuration.



Note: The import of the editor configuration files with the ImpEx Web does not require a Hybris restart. They can be re-imported for example with custom specific changes and will take effect immediately after the user has executed **Reset personalized settings** in the respective cockpit under **Menu > User Settings > Reset personalized settings**.

3.6 Configuring Hybris WCMS cockpit editor

The two editors **OpenText Media Reference Editor** and the **OpenText MediaCollection Reference Editor** provided by **DAMLink Hybris** can also be used in the **CMS cockpit** of the Hybris Web Content Management Module (WCMS). This section describes the necessary configuration to make use of the OTMM integration.

The configuration of **OpenText Media Reference Editor** in CMS cockpit requires the following:

- Registration of the OpenText editors for Media attributes of various CMS component item types. See [“Registration of OpenText editors for CMS component types” on page 62](#).

Configuration files

DAMLink Hybris comes with the following configuration files:

- Two spring bean definitions files `cmscockpit-editors.xml` and `cmscockpit-wizards.xml` located in `otmmaddonui\resources\otmmaddonui\cmscockpit\spring` to make the Asset Assignment Dialog known in the CMS cockpit spring context. These files can be used for the manual configuration steps.
- A set of CMS cockpit editor configuration files located in `otmmaddonui\resources\otmmaddonui\cmscockpit\xml`. Additionally there is Hybris ImpEx file `cmscockpit_config.impex` in `otmmaddonui\resources\impex` to load these configuration files into the Hybris system. These files are required for the registration of the OpenText editors in the second step.

3.6.1 Registration of OpenText editors for CMS component types

CMS cockpit editor configuration

By default, the **Media Reference Editor** is registered for the media attribute for the following CMS component item types:

`BannerComponent`, `CMSImageComponent`, `DynamicBannerComponent`, `ImageMapComponent`, `RotatingImageComponent`, `SimpleBannerComponent` and `SimpleResponsiveBannerComponent`.

Additionally the **Media Reference Editor** is registered for attribute `previewImage` of item type `ContentPage`.

In addition, **MediaCollection Reference Editor** will be registered for attribute `media` of item type `MediaContainer`. This is required because `SimpleResponsiveBannerComponent` contains a localized map of `MediaContainers`. And with this configuration you are able to add certain assets to one of these `MediaContainers`.

To configure the Hybris CMS cockpit to make use of the OTMM integration:

- Load the editor configuration files for the CMS Cockpit into your Hybris system. Therefore import the `cmscockpit_config.impex` ImpEx file located in `otmmaddonui\resources\impex`.

There are two options:

- Run a Hybris platform update and select the extension **otmmaddonui** for **Project Data Settings** update and switch **Import CMS Cockpit Data** to yes.
- Using the ImpEx Web UI provided by the **Hybris administration console**

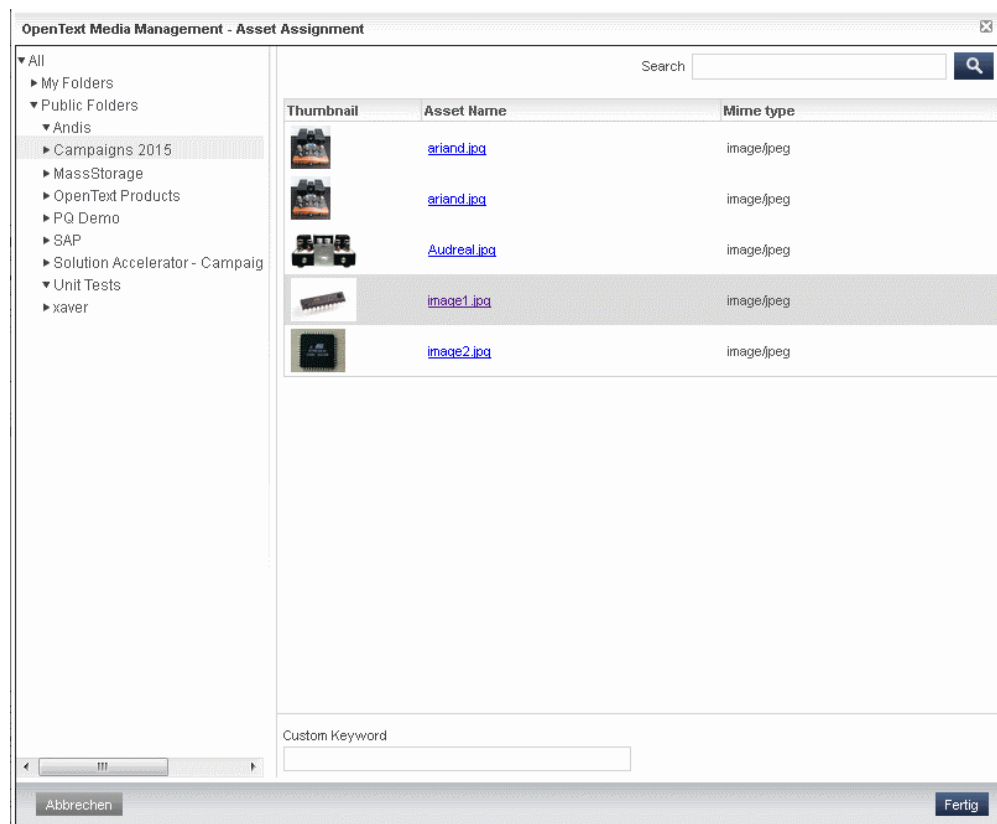


Note: The import of the editor configuration files with the ImpEx Web UI does not require a Hybris restart. They can be re-imported for example with custom specific changes and will take effect immediately after the user has executed **Reset personalized settings** in the respective cockpit under **Menu > User Settings > Reset personalized settings**.

As a result, the Asset Assignment Dialog is registered as editor for the Media attributes of various CMS component objects types.

If you want to adjust the configuration of the individual OpenText editor registrations or change/extend the registration, see [“Defining a new editor configuration” on page 60](#).

Test To test the integration, edit a web page in the CMS cockpit. Insert a new picture ([...]). The **OpenText Media Management Asset Assignment** dialog opens:



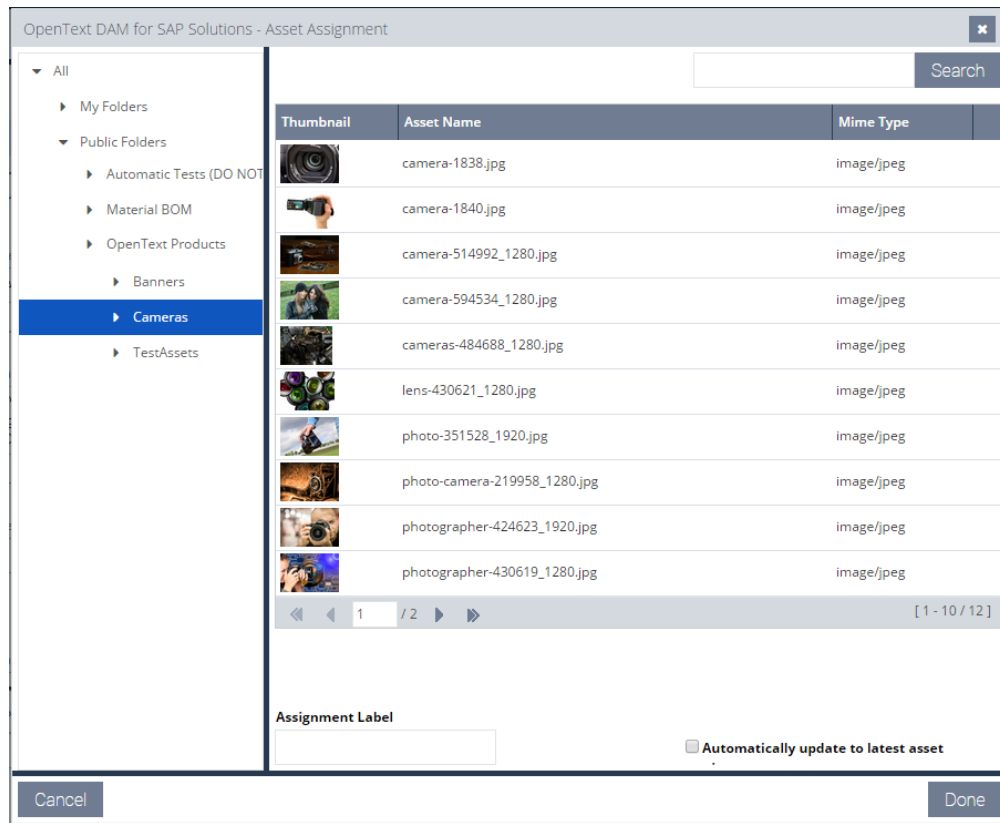
3.7 Configuring Hybris Backoffice

For asset assignment, two editors can be used in Hybris Backoffice:

- With the **OTMM Media Reference Editor**
`com.opentext.hybris.otmmconnector.backoffice.
editors.mediareferenceeditor`
(standard OpenText assignment dialog), a media item can be selected. The Backoffice user selects an Hybris item, opens the editor and selects a media item which should be assigned to the Hybris item.
- The **OTMM Multi Media Reference Editor**
`com.opentext.hybris.otmmconnector.backoffice.
editors.multimediareferenceeditor`

provides the same UI and function, but allows the selection of multiple media items. This is useful for Hybris item attributes containing a list of media items.

After installation of the **otmmaddonbackoffice** extension and Hybris server restart, the asset assignment for the Product item is available, using the **OTMM Media Reference Editor**.



Configuration Asset Assignment Dialog adjustments like a different OTMM root folder or an OTMM property template to enable the upload functionality can be configured in configuration files.

The editors are defined in two delivered configuration files: The socket configuration can be found at `otmmaddonbackoffice\resources\otmmaddonbackoffice-backoffice-widgets.xml`. The Media Reference Editor configuration can be found at `otmmaddonbackoffice\resources\otmmaddonbackoffice-backoffice-config.xml`. These configuration files are loaded automatically when you start the Hybris server.

Change these files with a text editor. To activate changes in these files, they must be reloaded. For this, you have to enter the Application Orchestrator mode and execute **Reset to Defaults** for the `widgets.xml` and/or `cockpit-config.xml` file. For more information, see [“Customizing Media Reference Editors in otmmaddonbackoffice configuration” on page 65](#) for a description of this procedure.



Note: For more information about **Application Orchestrator**, see the Application Orchestrator - End User Guide in Hybris wiki (<https://wiki.hybris.com/display/release5/Application+Orchestrator+-+End+User+Guide>).

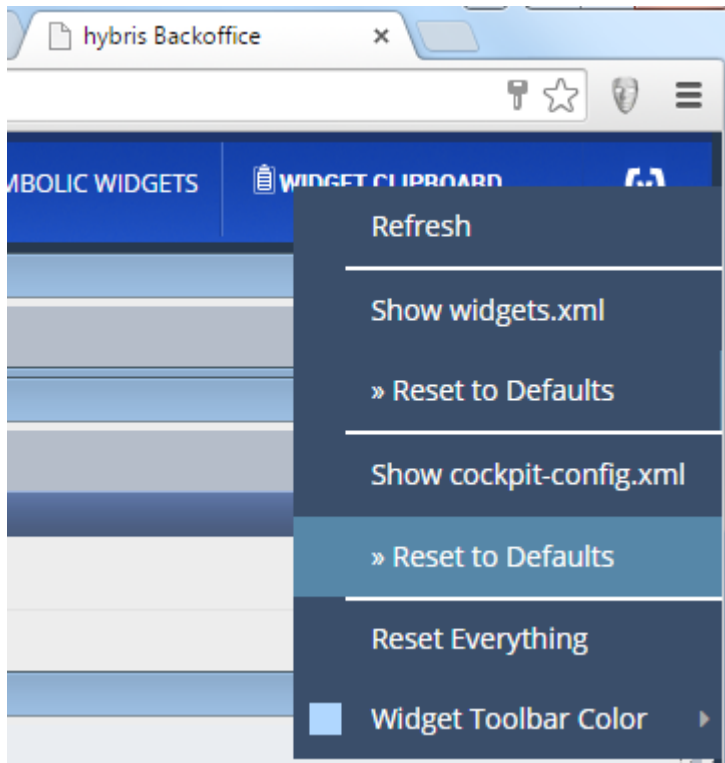
3.7.1 Customizing Media Reference Editors in otmmaddonbackoffice configuration


To adjust the OTMM Media Reference Editors, you can change the socket configuration in `otmmaddonbackoffice\resources\otmmaddonbackoffice-backoffice-widgets.xml` file or the Media Reference Editor configuration in `otmmaddonbackoffice\resources\otmmaddonbackoffice-backoffice-config.xml` file.

For example if you want to configure an OTMM property template to activate the upload functionality, you have to adjust the configuration in `otmmaddonbackoffice-backoffice-config.xml`.

To configure OTMM Media Reference Editors:

1. Change the configuration file `otmmaddonbackoffice-backoffice-config.xml` with a text editor.
2. To activate changes in the configuration file, it must be reloaded. For this, enter the Application Orchestrator mode and execute `Reset to Defaults` for the `cockpit-config.xml` file.



 **Note:** A reset removes all manual changes in cockpit-config.xml.

3.8 Configuring smartedit

You can configure Hybris smartedit to use the cropping feature that DAMLink Hybris adds. This configuration is part of the OTMM administration. To use the cropping feature Media Delivery Service is required.

For asset assignment in smartedit, OTMM assets must have a metadata model that contains the ARTESIA.CATEGORY.IMAGE.ATTRIBUTES field group. This is required to determine the width and height of the asset.

To access the configuraton:

1. Sign in to the OTMM administration.
2. Click **Settings > DAMLink_UI > Edit components**.
Descriptions for the configuration options are available on screen.

3.9 Setting up an individual Asset Assignment Dialog

The **Asset Assignment Dialog** can be enabled and configured individually per Hybris item attribute and per user group with the cockpit configuration capabilities.

The **Asset Assignment Dialog** provides the following configuration options for each instantiation:

- Root folder: Only the folder and assets below the configured root folder in OTMM are visible in the **Asset Assignment Dialog**.
- OTMM property templates for upload: In the **Asset Assignment Dialog** an **Upload** button can be configured.
To upload an asset, an **OTMM Property Template** is used. The newly created asset gets the OTMM model and the security policies from this template.
Multiple property templates can be configured. In the **Asset Assignment Dialog**, the configured list of OTMM property templates is displayed for selection. If there is no OTMM property template configured, no upload capabilities are offered in the dialog.
- The `mediaContainerDisplayFormat` property (only used by **OpenText MediaCollection Reference Editor**):
It determines the asset delivery of the Media item that is displayed by the editor to display a MediaContainer.
The editor looks for a Media item within the MediaContainer with the configured asset delivery. If the MediaContainer does not contain such a Media item, the editor looks for a Media item with asset delivery 300Wx300H. If also such a Media item does not exist, the editor just takes the first Media item in the MediaContainer.
- In the dialog, the visibility of the Asset Inspector launch button can be controlled with the `isAssetInspectorLaunchButtonVisible` property. In case the cockpit users should not directly access OTMM, set it to `False`. Per default (empty value for this property) the global `otmm.is.asset.inspector.launch.button.visible` property is evaluated instead.
- The properties to configure the automatic update to latest version.

Example: The following configuration shows an example configuration for the product attribute `galleryImages`.

```
<property qualifier="product.galleryImages"
  editor="openTextMediaReferenceCollectionSelector">
...
  <!-- The Asset Assignment Dialog will show this OTMM folder and
    all subfolders.
    - In case of an empty value all OTMM folders are visible
    - In case of 'Public Folders' only 'Public Folders' and
      subfolders are shown.
    Otherwise the value must be a valid ID of an OTMM Container,
```

```
e.g. '64bb9c93c0941f7aa25fabb38e539a396b9fd887' -->
<parameter>
  <name>otmmRootFolderId</name>
  <value></value>
</parameter>

<!-- If someone wants to upload an asset the Asset Assignment
      Dialog will offer a selection of OTMM Property Templates.
      - Property Templates IDs have to be specified.
      - Multiple Property Template IDs have to be separated with
        semicolons.
      - Non existing Property Template IDs will be ignored.
      - In case of an empty value no upload capabilities will be
        displayed. -->
<parameter>
  <name>otmmUploadPropertyTemplateIds</name>
  <value></value>
</parameter>

<!-- For usage with MediaContainerLists,
      e.g. for galleryImages attribute
      This is the MediaFormat (qualifier) of the image to be
      displayed. Default is 300Wx300H.
      If no image with the MediaFormat as configured here or with
      default format exists, the first image in the MediaContainer
      is taken. -->
<parameter>
  <name>mediaContainerDisplayFormat</name>
  <value>300Wx300H</value>
</parameter>

<!-- Option to make the checkbox in the Asset Assignment dialog
      for update to latest version visible or invisible. -->
<parameter>
  <name>isAutomaticallyUpdateToLatestAssetVersionVisible</name>
  <value>true</value>
</parameter>

<!-- Default setting for automatic update to latest version:
      If the checkbox is enabled, then the default setting
      controls whether the checkbox initially is marked or not.
      If the checkbox is disabled, then the default setting
      controls whether the assignments are always created
      for update to latest version or not (without user control).
      -->
<parameter>
  <name>automaticallyUpdateToLatestAssetVersionDefault</name>
  <value>false</value>
</parameter>
<!--
      Option to make the button to view a certain asset in OTMM visible or invisible.
      Possible values are 'true' or 'false'.
      In case of an empty value the global property
      'otmm.is.asset.inspector.launch.button.visible' ist taken instead.
      -->
<parameter>
  <name>isAssetInspectorLaunchButtonVisible</name>
  <value></value>
</parameter></property>
```

3.10 Scheduling asset assignment with cron jobs

Aside the synchronous assignment of assets to Hybris items, the actual export and synchronization of converted media files from OTMM to Hybris can also be done asynchronously. This is controlled by Hybris cron jobs, which can be triggered or scheduled with the **Hybris Backoffice**. Several predefined cron jobs are installed with **DAMLink Hybris**.

As any cronjob in Hybris, the provided cron jobs can either be triggered manually or scheduled to run at specific intervals.

The following cron jobs are delivered with **DAMLink Hybris**:

Cronjob name	Purpose
OtmExportServiceJob	Used to trigger the OTMM asset export service by Hybris scheduler. The job considers all asset assignments that have a status of export: <code>waitingForExport</code> . They have not yet been exported since the association was made. After successful execution of Export Service , the job sets the status to <code>waitingForSync</code> .
OtmAssetSyncJob	Used to assign previously exported assets along with the asset metadata descriptor file. The file contains references between assets and Hybris items. The assets are assigned to Hybris item attributes from the shared staging location, for example FTP or local disk. The file contains a subset of the asset metadata, for example content type and the relative file paths of the transformed images defined by asset deliveries. The job considers all asset-product associations that have been exported but not yet assigned.
OtmMetadataSynchronizationJob	Used to synchronize the manual assignment of existing OpentextMedia items with the Hybris Item Reference table in OTMM. Metadata synchronization is enabled by setting the property <code>otmm.metadata.synchronization.enabled</code> to <code>true</code> . If you don't use metadata synchronization you can disable the cronjob.
OtmAutoAssignJob	Used to automatically assign OTMM assets to Hybris products based on matching product ID or product EAN. Refer to <i>“Automated asset assignment with cronjob (AutoAssignJob)”</i> on page 78.

Cronjob name	Purpose
OtmAssociationPurgeJob	Used to purge the data from the OtmAssignmentStatus table for combinations of assets and Hybris media items that have been successfully assigned, that is exported as well as synched. If you want to keep the assignment entries as a kind of trace in the internal Hybris table then this job does not need to be scheduled.

3.10.1 Check assignment status

The current status of an assigned asset and more information about responsible jobs can be checked in the **Hybris Backoffice** (/backoffice).

To check the assignment status:

1. In the **Hybris Backoffice** select **DAM by OpenText > Asset Assignment Status**.
2. If you are not in search mode already, click **Switch search mode**.
3. Insert search values and click **Search**.
4. Select an entry in the result list.
5. The status of the selected entry is displayed.

6.png --> produkt1000 [produkt1000] - Apparel Product Catalog : Staged (SYNCHRONIZED)

Asset Assignment Status Administration

Essential

Item and Attribute

Asset

Asset ID 1765f8b2e4fbe843a9aad3f9059cd6e1fa7 Original Asset ID b178a5f8e2ff7ce5ba13682a99035423fac Asset Name 6.png

Model Type OPENTEXT.PRODUCTS Content Type BITMAP Mime Type image/png

Size 104105

Export and Assignment

Export Job ID Synchronization Directory ExportData_baa6031f2d9847fa8d1c8fce8 Job Status synchronized

Use Latest Version True False N/A Version Update True False N/A Selected Asset Delivery

Additional Parameters

For more information about status values, see to “Cronjob details” on page 71.



Important

Don't change the values in the table. Changes might impact the process export/synchronization process.

3.10.2 Cronjob details

During the system update all cron jobs are loaded in the system as defined in the `otmmaddon/resources/impex/essentialdata_otmmaddon.impex` import file.

DAMLink Hybris creates the database table `OtmAssignmentStatus` during installation. This table tracks assignments of assets to Hybris items and if the asset content was exported, and finally imported into Hybris. The cron jobs only processes entries that have pending statuses, thus any assets that have already been synchronized will not be considered. Such entries will instead be processed by the **OtmAssociationPurgeJob** if the `project.properties` configuration entry `otmm.otmmAssetSync.purgingRequired` was set to `TRUE`.

The `OtmmAssignmentStatus` table is generated from the custom model **OtmmAssignmentStatus**, that keeps among others reference of Asset ID, Asset Name, Product ID, OTMM Model ID as well as the current status of the assignment.

The status of the association synchronization is recorded in the `otmmStatus` property with the following default values:

- `waitingForExport`: Used to trigger the export service by Hybris scheduler.
- `processingExport`: Indicates that the **OtmmExportServiceJob** is currently processing this entry and starts an export process in OTMM. No other **OtmmExportServiceJob** will pick up this entry for processing.
- `processingDirectExport`: Indicates that the export service triggered by the Assignment Editor UI is currently processing this entry and starts an export process in OTMM.
- `waitingForSync`: After triggering the export service successfully, the system updates the `otmmStatus` property to `waitingForSync`, which then will ensure that the **OtmmAssetSyncJob** will process the entry for importing the asset into Hybris.
- `processingSync`: Indicates that an assignment of the asset is ongoing. No other **OtmmAssetSyncJob** will pick up this entry for processing.
- `processingDirectSync`: Indicates that an assignment of an asset is ongoing triggered by the Assignment Editor UI.
- `synchronized`: The **OtmmAssetSyncJob** set the `otmmStatus` property to `synchronized`, when the assignment of the asset to an attribute of a Hybris item was successfully processed.
- `failed`: If the export or the asset synchronization runs into an error.

The sync process is divided into two cron jobs in OTMM Hybris connector:

- **OtmmExportServiceJob** triggers the OTMM export service for assets that are ready to be synchronized. The export service then will do the necessary image transformation, and upload the export content to a shared FTP path.
- **OtmmAssetSyncJob** will then process these exported assets for associations for which the export was completed.

For more information, see [“Configuring assignment spec and asset delivery” on page 33](#).

3.10.3 Scheduling cron jobs

As any cronjob in Hybris, the DAMLink Hybris cron jobs can either be triggered manually or scheduled to run at specific intervals. This section describes the necessary steps to schedule a cronjob depending on your needs.



Note: Per default, the **OtmExportServiceJob** and **OtmAssetSyncJob** are scheduled to run every 5 minutes to support automatic update to the latest version. The **OtmAssociationPurgeJob** is disabled.

To adjust a cronjob schedule:

1. In **Hybris Backoffice** (/backoffice), select **System > Background Processes > CronJobs**.
2. If you are not in search mode already, click **Switch search mode**.
3. Search for cron jobs of which Code starts with **otmm**.

Attribute	Comparator	Value	Sort Order
Code	starts with	otmm	
Job definition	equals		
Current status	equals		
Last result	equals		
Alternative Data Source ID	equals		

Code	Job definition	Current status	Last result	Timetable
otmmAssetSyncJob	otmmAssetSyncJob	ABORTED	ERROR	Interval gap: 5 minutes
otmmExportServiceJob	otmmExportServiceJob	ABORTED	ERROR	Interval gap: 5 minutes
otmmAssociationPurgeJob	otmmAssociationPurgeJob	NEW	N/A	No next execution time
otmmMetaDataSynchronizationJob	otmmMetaDataSynchronizationJob	FINISHED	SUCCESS	Interval gap: 5 minutes

4. In the result list click the cronjob you want to schedule.
5. Click the **Time Schedule** tab.
6. In the **Schedule** area, change an existing trigger or add a new trigger.

7. Click **Save** to save the changed time schedule.
8. Click **Save** to save the cronjob changes. The changed schedule is activated at the defined start time.

3.10.4 OtmmExportServiceJob

The Hybris export service fetches entries from the Hybris database table `OtmmAssignmentStatus` with the `otmmStatus` property set to `waitingForExport`. For those assets, the standard OTMM export process will be called with transformation options configured in the in asset deliveries in Hybris Backoffice.

If no export is required, the job only sets the `otmmStatus` property to `waitingForSync`.

While invoking **OTMM WebService** to export data, the Hybris service layer provides the asset ID for each assignment and details of the FTP server (server URL, user name, password, port and so on) as request parameters. In return, OTMM sends an acknowledgment that the request has been received. OTMM later exports the corresponding assets (media files) to an configured and accessible FTP server.

The figure shows the **OtmmExportServiceJob** in **Hybris Backoffice** (/backoffice). You can start the cronjob by clicking the **Run CronJob** icon or by setting a **Time Schedule**.

The screenshot displays the Administration console interface. On the left, the 'EXPLORER' sidebar shows a tree view with 'CronJobs' selected under 'Background Processes'. The main content area shows a search for 'CronJob' with a 'Global Operator' set to 'and' and 'Include subtypes' checked. A table lists three cron jobs:

Code	Job definition	Current status	Last result	Timetable
otmmAssetSyncJob	otmmAssetSyncJob	ABORTED	ERROR	Interval gap: 5 minute
otmmExportServiceJob	otmmExportServiceJob	ABORTED	ERROR	Interval gap: 5 minute
otmmAssociationPurgeJob	otmmAssociationPurgeJob	NEW	N/A	No next execution tim

Below the table, the details for 'otmmExportServiceJob' are shown, indicating it is 'FINISHED - SUCCESS'. The 'Timetable' section shows 'Interval gap: 5 minutes' and 'Last start time' as 'Feb 21, 2017 12:37:02'. The 'Enabled' checkbox is checked.

Failed jobs If the processing of a job fails (for example due to errors reported by OTMM WebService call), the cronjob continues and sets the status to **failed** for the job entry.

Only when a maximum number of subsequent failed jobs is reached, the **OtmExportServiceJob** cronjob stops the processing. This limit can be defined with the `otmm.exportServiceJob.maxFailedSize` property in `project.properties` file:

```
otmm.exportServiceJob.maxFailedSize=5
```

The cronjob of course also stops if the maximum number of jobs to process in one run is reached. This number is given by the configuration property:

```
otmm.exportServiceJob.batchSize=100
```

After a successful export service call, OTMM shares media files for associated assets through an ftp server. OTMM also exports an XML file to the same location. This XML file contains detailed information about all exported assets (media files) and their metadata.

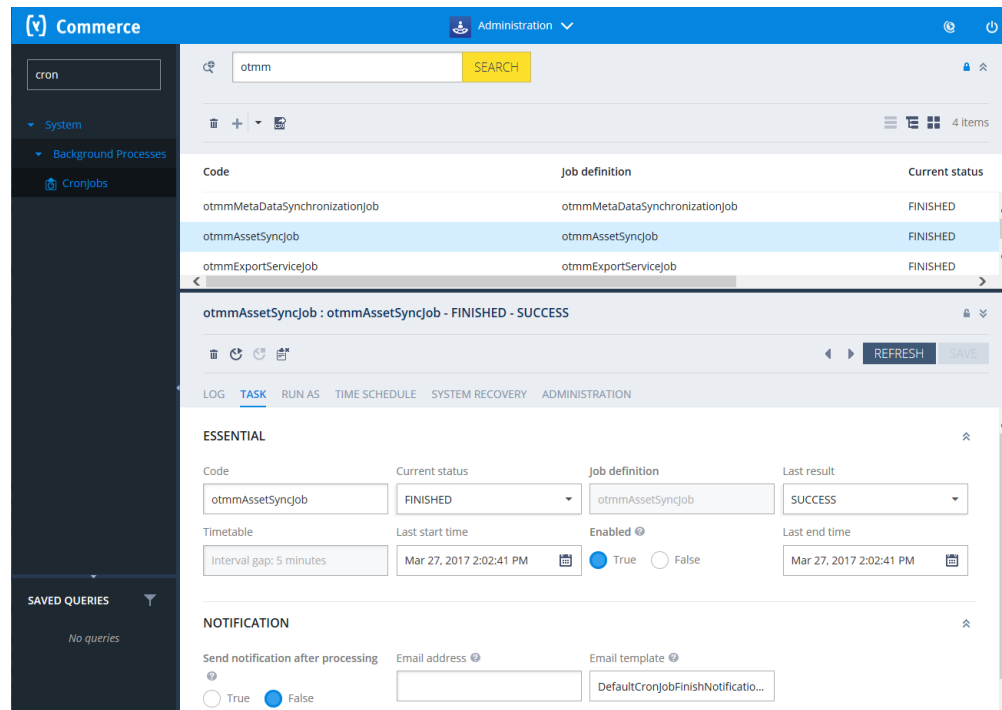


Note: The export service is not involved in every assignment type, especially not with **Media Delivery Service** assignment type.

3.10.5 OtmmAssetSyncJob

For all the `OtmmAssignmentStatus` entries with `otmmStatus=waitingForSync`, media items are created in Hybris according to the configured asset deliveries and assigned to Hybris items, for example products.

The figure shows the **OtmmAssetSyncJob** in **Hybris Backoffice** (`/backoffice`). You can start the cronjob by clicking the **Run CronJob** icon or by setting a **Time Schedule**.



Failed jobs

If the processing of a job fails (for example due to errors reported by OTMM WebService call), the cronjob continues and sets the status to **failed** for the Asset Assignment Status entry.

Only when a maximum number of subsequent failures is reached, the **OtmmAssetSyncJob** cronjob stops processing. This limit can be defined with the `otmm.assetSyncJob.maxFailedSize` property in `project.properties` file:

```
otmm.assetSyncJob.maxFailedSize=5
```

The cronjob also stops if the maximum number of jobs to process in one run is reached. This number is given by the configuration property:

```
otmm.assetSyncJob.batchSize=100
```

3.10.6 OtmAssociationPurgeJob

The purge job first picks up the entries from the `OtmAssignmentStatus` transaction table with the status `synchronized`, which means that the respective assignment was successfully processed and deletes these entries.

The figure shows the **OtmAssociationPurgeJob** in **Hybris Backoffice** (/backoffice).

You can start the cronjob by clicking the **Run CronJob** icon or by setting a **Time Schedule**.

The screenshot displays the Hybris Backoffice interface for managing cron jobs. The left sidebar shows the navigation menu with 'CronJobs' selected. The main area shows a list of cron jobs, with 'otmmAssociationPurgeJob' highlighted. Below the list, the configuration details for 'otmmAssociationPurgeJob' are shown, including the 'Time Schedule' tab.

Attribute	Comparator	Value	Sort Order
Code	starts with	otmm	
Job definition	equals		
Current status	equals		
Last result	equals		
Alternative Data Source ID	equals		

Code	Job definition	Current status	Last result	Timetable
otmmAssetSynJob	otmmAssetSynJob	ABORTED	ERROR	Interval
otmmExportServiceJob	otmmExportServiceJob	ABORTED	ERROR	Interval
otmmAssociationPurgeJob	otmmAssociationPurgeJob	NEW	N/A	No next

otmmAssociationPurgeJob : otmmAssociationPurgeJob - UNKNOWN - UNKNOWN

Log Task Run as Time Schedule System Recovery Administration

Code: otmmAssociationPurgeJob

Current status: NEW

Job definition: otmmAssociationPurgeJob

Last result: N/A

Timetable: No next execution time

Last start time:

Enabled: ☐ True ☒ False

Last end time:

3.11 Automated asset assignment with cronjob (AutoAssignJob)

The **Automated asset assignment** feature offers the possibility to automate the process of asset assignment completely. The manual assignment via GUI can be replaced by automated procedures. With appropriate configuration, the provision of assets on OTMM side suffices to perform the assignment to Hybris product items.

AutoAssignJob This section describes the configuration of an automated asset assignment cronjob (AutoAssignJob). This cronjob searches in defined time intervals for unassigned OTMM assets. Found new assets will be assigned automatically. The job parameters specify the used Hybris catalog and the catalog version. The asset metadata defines the product item(s) to which the asset should be assigned.

When the AutoAssignJob finds an unassigned OTMM asset that does not match an existing product in the defined catalog version, then the asset is skipped.

OTMM metadata The asset metadata must define the product item(s) to which the asset should be assigned. It has to match either a product ID or a product EAN on Hybris side.

To which item attribute the assignment is made (picture, galleryImages, detail, ...) can be defined with another metadata field or can be derived from the asset file name based on configurable rules.



Example 3-8: Map the attribute from asset name

hammer.jpg as **BITMAP** attribute,
hammer-2.jpg as **galleryImages** attribute,
hammer.pdf as **detail** attribute.



One asset can be assigned to one or more Hybris items. This affects the OTMM metadata: In case the asset should be assigned to more than one item, the assignment data must be stored in tabular fields.

The product ID / EAN field can be provided as follows:

- With the asset. The product ID field is one of the asset fields.
- In the folder (**AutoFolder**) containing the asset, and must then be inherited to the asset.
Here all assets in the folder share the same product ID and will be assigned to the same product.

Example This description shows the configuration steps by way of examples configuring the assignment of assets to different attributes of Hybris product items.

3.11.1 Prepare your OTMM metadata

This section describes the necessary OTMM metadata for the automated asset assignment.

To prepare OTMM metadata for automated assignment:


1. **Check your existing metadata** whether it fits for automated assignment.



Note: For multiple assignments, the fields must be tabular.

If the asset has only one field containing the product ID, it can be assigned only to one product. A tabular field is contained in rows, and each row contains a different product ID, so that ultimately the asset gets assigned to all these products.

The metadata for assets must contain the following fields:

- **For use of individual assets (no AutoFolders): the product ID**
The automated asset assignment requires a field that matches either a product ID or a product EAN on the Hybris side.
If AutoFolders are used, this field is not necessary. The product ID will be inherited from the folder in which the asset is stored.
-  **Note:** The Hybris catalog ID and catalog version which are also necessary to uniquely match a product, are parameters of the AutoAssignJob.
- **Assignment status field**
The automated assignment requires one additional field, normally alongside the product ID field. You must change your model to include this additional field. It must be a numeric field, and it must be empty, if the asset has not been assigned yet. It is set to 1 during the asset assignment.
- **Optional: Attribute ID field**
It defines to which Hybris attribute the asset should be assigned, for example picture or galleryImages.
The other possibility is to derive the attribute from the asset file name, see [“Configuring Hybris properties” on page 82](#).



Note: The attribute ID field can also be mapped with custom code. Per default the name is not mapped. The custom code must be in a bean with name `mapOtmAttributeIdField`, implementing the `com.opentext.hybris.otmmconnector.cronjobs.MapOtmAttributeIdField` interface.

Example: A function named `List<String> map2AttributeId(String attributeIdFieldValue)` can be written to convert the String `gallery+pict` to the attribute list `{"galleryImages", "picture"}`.



Example 3-9: Metadata for automated assignment

This example shows asset metadata for automated assignment in the OTMM GUI with product ID, the assignment status, and attribute ID (scalar and tabular).

▼ Auto Ass. Metadata

Auto Ass. scalar isAssigned indicator

Auto Ass. scalar attribute id

Auto Ass. scalar product id

Auto Ass. tabular group

Auto Ass. tabular product id	Auto Ass. tabular isAssigned indicator	Auto Ass. tabular attribute id
------------------------------	--	--------------------------------

▼ hybris Item Reference

hybris Item Reference

hybris Item ID	hybris Assignment Label	hybris Attribute Name	hybris Catalog Mnemonic	hybris Item PK
----------------	-------------------------	-----------------------	-------------------------	----------------



2. For AutoFolders: Define folder metadata model

In case the assets for each product should be stored in a respective OTMM folder, the folder type must be defined.

The product ID field must be a scalar attribute of the folder.

It makes sense to store all product related folders under a common ancestor.

The folder ID of this ancestor can be passed to the `AutoAssignJob`, so that the search is restricted to the common ancestor and all its descendants (recursively).




Note: The common ancestor must be a folder under `Public Folders`.
Folders below `My Folders` are not found by searches!



Example 3-10: Auto folder model and type definition

The example shows a folder model:



Engellnh

[Metadata: Muh Folder Model](#)
[Security Policies](#)

View
[Close all groups](#)
☐ Preferred fields only
 ☐ Hide empty fields

▼ **Basic Folder Information**

ID *	c4147b65f6cbe2d2d778389047b8660d51e5d089
Name	Engellnh
Description	
Container type	Muh Folder

▼ **Auto Ass. Folder Metadata**

Auto Ass. folder product id	Engel
Auto Ass. folder attribute id	

The folder attributes will be inherited to the assets in the folder. In the screenshot below (/teams GUI, **Template > /Folder Types**), the fields `AUTO.FOLDER.ATTRIBUTE_ID` and `AUTO.FOLDER.PRODUCT_ID` are in the **All Level Inheritance** list.

The (AUTO.FOLDER.*) fields must be selected to make them inheritable.

Folder Type : MUH.FOLDER

Properties

* ID: MUH.FOLDER

* Name: Muh Folder

Description: Folder with product id

* Model Identifier: MUH.FOLDER.MODEL

All Level Inheritance: ARTESIA.FIELD.ASSET NAME, ARTESIA.FIELD.ASSET DESCRIPTION

One Level Inheritance: ARTESIA.FIELD.ASSET NAME, ARTESIA.FIELD.ASSET DESCRIPTION

3.11.2 Configuring Hybris properties

Parameters which are identical for all AutoAssignJobs are stored in the local.properties configuration file.

To define the AutoAssignJob properties:

1. Add the following otmm.auto.* properties:
 - otmm.auto.productid.field defines the field containing the product ID or EAN field.
It can be a scalar or a tabular metadata field, if not inherited. If inherited from folder, it must be a scalar field.
 - otmm.auto.productid.field.isinherited is true or false. For use of AutoFolders, set it to true.
 - otmm.auto.assigned.field defines the ASSIGNED field in metadata.
If not specified, it is taken to be the product ID field name, with the last name replaced by ASSIGNED. It must only be specified if isinherited is true, and then it must be scalar.
 - otmm.auto.attributeid.field defines the field in metadata containing the attribute ID, if it exists. It must be a scalar, if the product ID field is scalar, or tabular, if the product ID field is tabular.
If tabular, both fields should be in the same table. But if they are contained in different tables, than at least the row numbers must match.
If no attribute ID field is defined, the attribute will be derived from the asset file name.

- `otmm.auto.useEan` defines if the `productid.field` contains a Hybris product EAN (true) or product ID (false)



Example 3-11: AutoAssignJob properties in local.properties

```
otmm.auto.productid.field = AUTO.SCALAR.PRODUCT ID
otmm.auto.attributeid.field = AUTO.SCALAR.ATTRIBUTE ID
```

If the product ID field is inherited from a folder, use

```
otmm.auto.productid.field=AUTO.FOLDER.PRODUCT ID
otmm.auto.productid.field.isinherited=true
otmm.auto.assigned.field = AUTO.SCALAR.ASSIGNED
```

Properties for tabular fields:

```
otmm.auto.productid.field = AUTO.TABULAR.GROUP /
AUTO.TABULAR.PRODUCT ID
otmm.auto.attributeid.field = AUTO.TABULAR.GROUP /
AUTO.TABULAR.ATTRIBUTE
```



2. Define the rules if the attribute ID is derived from asset file name:
If `otmm.auto.attributeid.field` is not specified, then the attribute ID is derived from the OTMM asset file name with which the asset was originally stored.
If the name before the file suffix terminates with a hyphen or underscore, followed by a number, as in `hammer-2.jpg`, the number is interpreted as version number and the asset is called "versioned".



Example 3-12: Asset file name rules in local.properties

These example rules specify, that the delimiter is a minus sign (hyphen), that versioned jpg-files get the attribute ID `galleryImages`, that (non-versioned) jpg-, png-, or psd-files get the attribute ID `picture`, that pdf-files get `data_sheet`, and mp4-files `detail`.

```
# With impex.stem, we map file suffixes of assets without
versions (i.e. the "main" asset, like hammer.jpg)
# to an attribute (to be stored as otmmAttributeId).
# The numbers must be sequential, to fill any gaps, you
may use impex.stem.i.suffix=skip
otmm.impex.stem.0.suffix=.jpg
otmm.impex.stem.0.attribute=picture
otmm.impex.stem.1.suffix=.png
otmm.impex.stem.1.attribute=picture
otmm.impex.stem.2.suffix=.psd
otmm.impex.stem.2.attribute=picture
otmm.impex.stem.3.suffix=.pdf
otmm.impex.stem.3.attribute=data_sheet
otmm.impex.stem.4.suffix=.mp4
otmm.impex.stem.4.attribute=detail
```

```
# with impex.versioned, we map file suffixes of assets
```

```

with versions (like hammer_2.jpg)
# to an attribute (to be stored as otmmAttributeId).
otmm.impex.versioned.0.suffix=.jpg
otmm.impex.versioned.0.attribute=galleryImages

# The default for the char separating the version from the
stem is _, e.g. hammer_2.jpg.
# With versionDelim you can specify a different char, e.g.
- for hammer-2.jpg.
otmm.impex.versionDelim=-

```




3. You must restart Hybris after changing the `local.properties` file.

3.11.3 Setting up the AutoAssignment cronjob (AutoAssignJob)

This section describes the configuration of the cronjob for automated asset assignment.

To set up an AutoAssignJob:

1. In the **Hybris Backoffice** (`/backoffice`), click **System > Background Processes > CronJobs**.
2. Click the arrow right of **+ CronJob** and select **Automatic Asset Assignment Job**.
3. Specify the following parameters:

Field	Description
Code (mandatory)	Define a job name.
Catalogversion (mandatory)	Specify the catalog and version.
Job definition (mandatory)	Define the job type to otmmAutoAssignJob .  Note: If you cannot find the otmmAutoAssignJob , an update can be necessary. Go to Hybris administration console > Platform > Update . Select the otmmaddon and the Create Essential Data option.

4. Click **Done**.
5. In the **Task** tab of cronjob specify the **Job Parameters** as required:

Field	Description
Max Assets	Specify the maximum count of assets per job.
Custom Keyword	The Custom Keyword is stored as Hybris Assignment Label in the Hybris item reference metadata.

Field	Description
Use Latest Version	Select Yes to automatically reflect later OTMM asset updates in Hybris.
Model ID	Specify a model ID to restrict the asset search to this model.
Folder ID	Specify a folder ID to restrict the asset search to this folder.
Age In Days	Specify to search only for assets created at most these number of days ago. With 0, the search is limited to all assets newer than the last startdate of the job minus 1 day, or unlimited, if the job was not yet started. Thus, you can eliminate assets from the search that never found a Hybris product counterpart. Of course, you can then setup a second job, running less frequently, that searches with a higher limit.
Number of Threads	Specify the number of threads and hence the degree of parallelism of the job.

6. You can check your new AutoAssignJob:

Edit the asset metadata manually, add product ID's and/or attribute ID's either in the scalar or the tabular rows, then run the AutoAssignJob, and you will see that the ASSIGNED field (the isAssigned indicator) is set to 1, and that there is a new row in the Hybris Item Reference. The Hybris Item ID is the product ID. The Hybris Assignment Label is a parameter of the AutoAssignJob. The Hybris Attribute Name is or is derived from the attribute ID or the OTMM name. The Hybris Catalog Mnemonic is the mnemonic name of the catalog specified as AutoAssignJob parameter. The Hybris Item PK is the product key of the product.



Example 3-13: Metadata content after automated assignment

The following image shows the asset metadata after an assignment took place. Here, the scalar product ID Engel was specified, and no attribute ID. The attribute ID detail was derived from the suffix of the asset name Engel.pdf. The ASSIGNED field is set to 1.

▼ Auto Ass. Metadata

Auto Ass. scalar isAssigned indicator 1

Auto Ass. scalar attribute id

Auto Ass. scalar product id Engel

Auto Ass. tabular group

Auto Ass. tabular product id	Auto Ass. tabular isAssigned indicator	Auto Ass. tabular attribute id
------------------------------	--	--------------------------------

▼ hybris Item Reference

hybris Item Reference

hybris Item ID	hybris Assignment Label	hybris Attribute Name	hybris Catalog Mnemonic	hybris Item PK
Engel	Auto1	detail	APP-O	8796387966977



3.12 Metadata synchronization

With the assets, **DAMLink Hybris** stores metadata. These metadata contain information about the linked Hybris item. This information is used by the technical integration process that synchronizes assigned assets from OTMM back to Hybris. It provides the benefit that OTMM end users can search and find out which OTMM assets are used in which item (for example product) by way of normal metadata search capabilities.

To make this possible, any OTMM asset model that should be available to be assigned to Hybris items needs to be adapted for specific Hybris metadata. The additional metadata constitute a tabular field group that captures the actual product references.

Synchronization modes

To keep the stored metadata up-to-date on OTMM side, an automatic synchronization mechanism is necessary. For this, two synchronization modes can be maintained:

- **Basic synchronization** (Default mode): When an asset is assigned with the **Asset Assignment Dialog**, the metadata of the assignment are saved at the asset.
- **Enhanced synchronization**: In this mode, DAMLink Hybris reacts on all changes on Hybris side regarding asset assignments. This affects the creation of assignments as well as deletions and removal of assignments.

As an example, if a product manager removes an assignment of an OpenTextMedia item to a product attribute like **picture**, this is traced and the respective Item Reference entry at the asset on OTMM side will be removed.

All changes are collected and a cronjob retraces these changes on OTMM side in the metadata. The tabular field group **Hybris Item Reference** of the corresponding asset will be updated. The essential attribute is the **Hybris Item PK**.

Configuring the cronjob, time and frequency of the updating can be controlled.

The adjustment of this synchronization mode is described in this section.

To configure the enhanced synchronization mode:

1. In DAMLink Hybris, enable the complete synchronization mode: Edit the `local.properties` file of your Hybris installation: Set the `otmm.metadata.synchronization.enabled` property to `true`. This setting activates the use of Hybris interceptors to track the changes.

For more information about interceptors, refer to Hybris wiki: Interceptors (<https://wiki.hybris.com/display/release5/Interceptors>).
2. Configure a whitelist of catalogs which should be synchronized with the `otmm.metadata.synchronization.catalogs` property. It expects a list of catalogs / catalog versions. Only the listed catalogs / catalog versions will be tracked and synchronized.

Use the following syntax:

- Catalog ID and catalog version part must be separated by a slash “/”.
- Several catalogs (patterns) can be separated by comma.
- Asterisk “*” is the wildcard for any character.

Examples:

`*/*` : all catalogs / all catalog versions

`*/Staged`: all catalog versions with version named Staged

`*/Staged,*electronics*/*`: all catalog versions with version named Staged and all catalogs with electronics in their name.

If this property is not configured or has an empty value, the default value `*/Staged` is used.

3. The metadata changes are written by a scheduled Hybris cronjob named **otmmMetaDataSetSynchronizationJob**.

Activate and configure the **otmmMetaDataSetSynchronizationJob** cronjob for your needs.

You can configure the schedule as usual but also the batch size. The batch size defines how many entries (collected Hybris assignment changes) should be processed.

For more information about cronjob administration, see [“Scheduling asset assignment with cron jobs” on page 69](#).

4. The metadata synchronization can be configured for certain item types. By default, it is configured for product items and its sub types. In the `otmmaddon\resources\otmmaddon-spring.xml` configuration file, an `InterceptorMapping` bean named `otmmTrackingMappingForProduct` is defined.

To configure other types: Provide an additional `InterceptorMapping` for each type. Configure for your `InterceptorMapping` the same order as for `otmmTrackingMappingForProduct`.

Monitoring DAMLink Hybris offers monitoring functions in the Hybris Backoffice. The tracked assignment changes made in Hybris can be listed.

The following changes can be tracked:

- The **Metadata Changes** list shows the last assignment changes in Hybris. The listed changes will be processed with the next **otmmMetaDataSetSynchronizationJob** job.
- The **MediaContainer Usage** list tracks which item uses which MediaContainer. This is necessary to synchronize the metadata correctly if a MediaContainer was deleted.

To monitor metadata changes pending for synchronization:

1. Open the **Hybris Backoffice** /backoffice and sign in as administrator.
2. To monitor metadata changes, select **DAM by OpenText > Metadata Synchronization > Metadata Changes** in the navigation. Switch to **Search mode** and define your criteria. Start the search.

3. To monitor media container changes, select **DAM by OpenText > Metadata Synchronization > MediaContainer Usage** in the navigation. Switch to **Search mode** and define your criteria. Start the search.

Events The result list entries show the triggering Hybris events with the following data:

- **Item PK:** Key of the product
- **Media PK:** Key of the media item (OpenTextMedia key)
- **Asset ID:** Original ID of the referenced asset
- **Attribute ID:** Attribute to which the asset was assigned
- **Event:** Event type

The available event types:

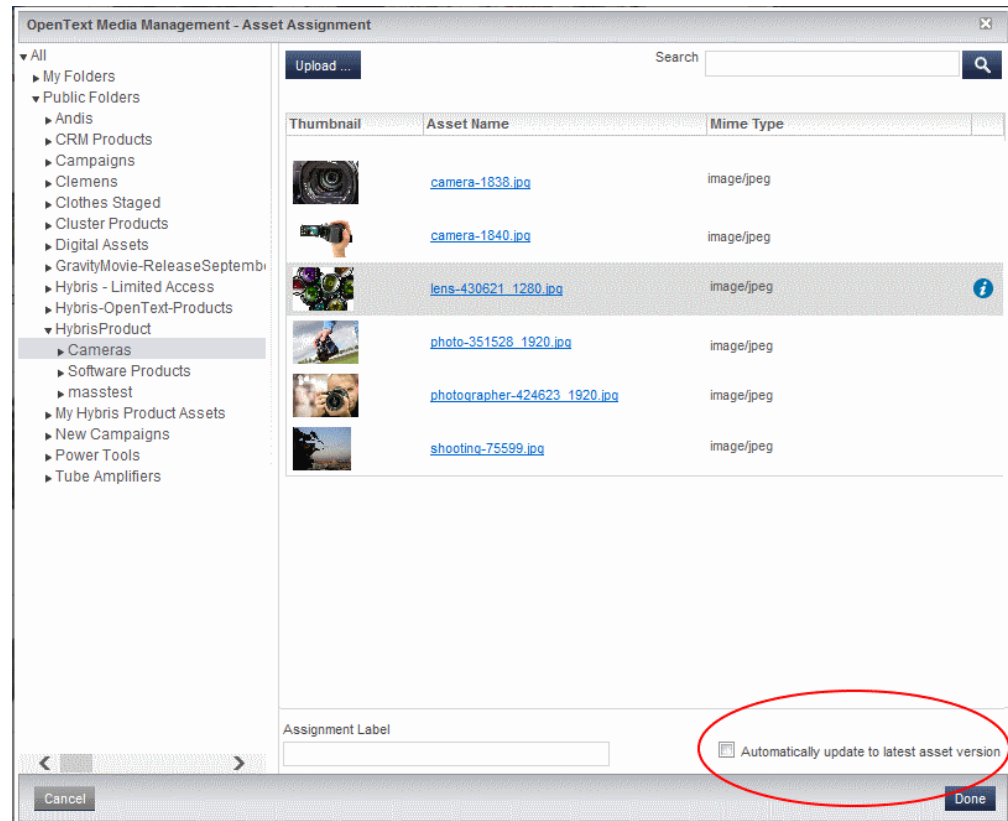
- **deleteMedia:** An OpenTextMedia item was deleted.
- **removeMedia:** An OpenTextMedia item was removed from a Media, localized:Media or MediaCollection attribute (but not deleted).
- **addMedia:** An OpenTextMedia item was added to a Media, localized:Media or MediaCollection attribute.
- **deleteItem:** An Item was deleted.
- **deleteContainer:** A MediaContainer was deleted.
- **removeMediaFromContainer:** An OpenTextMedia item was removed from a MediaContainer.
- **addContainer:** A new MediaContainer was added to a MediaContainer or a MediaContainerList attribute.
- **addMediaToContainer:** A new MediaContainer was added to a MediaContainer or a MediaContainerList attribute.
- **removeMediaFromContainerImplicit:** When an OpenTextMedia item is already assigned to a MediaContainer and then it is assigned to a different MediaContainer, the Media item is implicitly removed from the first MediaContainer.

3.13 Configuring automatic update to latest version

OTMM allows versioning of stored assets. With DAMLink Hybris, assigned media items can be automatically updated on Hybris side to the latest OTMM version. This section describes the underlying process and the configuration options for the automatic update function.

With the **Automatically update to latest asset version** checkbox in the Asset Assignment dialog, the product manager can decide:

- The assigned asset will be updated always with the latest asset version in OTMM (checked)
- Or the media is bound to the exact version of the asset that was assigned (not checked)



Processing The creation of a new asset version in OTMM triggers a notification for Hybris. For each notification, a Hybris task is started that updates all Media Items for the affected assets that have the update flag set to the new asset version.

To configure Hybris for automatic update:

- With update to latest version, existing Media items are adjusted to the new asset version. To consider these changes in Hybris version synchronization, the respective Media item attributes have to be registered in the respective synchronization service:
 - the product item synchronization for **Hybris Product Cockpit** :
Add the Media item attributes to the `relatedReferencesTypesMap` property in `... \hybris \bin \ext -template \yacceleratorcockpits \resources \yacceleratorcockpits \productcockpit \spring \productcockpit -services.xml`.
 - the CMSComponent item synchronization for **CMS Cockpit** :
The `... \hybris \bin \ext -content \cmscockpit \resources \cmscockpit \cmscockpit -spring -service.xml` file has to be adjusted.



Example 3-14: Configure synchronization for Hybris Product Cockpit

The following configuration shows added attributes for the Product and MediaContainer Media items to the relatedReferencesTypesMap property in productcockpit-services.xml:

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd"
       default-autowire="byName">

    <alias alias="synchronizationService"
           name="accSynchronizationService" />

    <bean id="accSynchronizationService"
          class="de.hybris.platform.cockpit.services.sync.impl.SynchronizationServiceImpl"
          parent="defaultSynchronizationService">

        <property name="relatedReferencesTypesMap">
            <map merge="true">
                <entry key="Product">
                    <list>
                        <value>Product.productImages</value>
                        <value>Product.galleryImages</value>
                        <value>Product.picture</value>
                        <value>Product.thumbnail</value>
                        <value>Product.data_sheet</value>
                        <value>Product.detail</value>
                        <value>Product.logo</value>
                        <value>Product.normal</value>
                        <value>Product.others</value>
                        <value>Product.thumbnails</value>
                    </list>
                </entry>
                <entry key="MediaContainer">
                    <list>
                        <value>MediaContainer.medias</value>
                    </list>
                </entry>
            </map>
        </property>

    </bean>
</beans>
```



Example 3-15: Configure synchronization for CMS Cockpit

For CMS Cockpit, the <value>AbstractBannerComponent.media</value> entry is added to the relatedReferencesTypesMap property for AbstractPage key.

The following configuration shows adjusted configuration in ..\hybris\bin\ext-content\cmscockpit\resources\cmscockpit\cmscockpit-spring-service.xml:

```
<bean id="defaultCMSynchronizationService"
```

```

class="de.hybris.platform.cmscockpit.sync.CMSSynchronizationService"
scope="tenant" autowire="byName">
<property name="relatedReferencesTypesMap">
<map>
<entry key="AbstractPage">
<list>
<value>AbstractPage.restrictions</value>
<value>AbstractPage.contentSlots</value>
<value>ContentSlotForPage.contentSlot</value>
<value>ContentSlot.cmsComponents</value>

<value>AbstractCMSComponentContainer.simpleCMSComponents</value>

<value>AbstractCMSComponentContainer.currentCMSComponents</value>
<value>RotatingImagesComponent.banners</value>
<value>AbstractCMSComponent.restrictions</value>
<value>abstractMediaContainerComponent.media</value>
<value>AbstractBannerComponent.media</value>
</list>
</entry>

```



Hybris configuration

You find the important configuration entries in your `local.properties` or `project.properties` file:

- To make the **Automatically update to latest asset version** check box visible in the **Asset Assignment** dialog at Hybris Product Cockpit or in Hybris CMS cockpit and to define a default value (checked or not checked) the following two cockpit configuration parameters are used:.

```

<!-- Option to make the checkbox in the Asset Assignment dialog
for update to latest version visible or invisible. -->
<parameter>
<name>isAutomaticallyUpdateToLatestAssetVersionVisible</name>
<value>true</value>
</parameter>

<!-- Default setting for automatic update to latest version:
If the checkbox is enabled, then the default setting controls
whether the checkbox initially is marked or not.
If the checkbox is disabled, then the default setting controls
whether the assignments are always created for update to latest
version or not (without user control). -->
<parameter>
<name>automaticallyUpdateToLatestAssetVersionDefault</name>
<value>false</value>
</parameter>

```

For information about the **Opentext Media Reference Editor** configuration in Product Cockpit or CMS cockpit, see [“Defining a new editor configuration” on page 60](#) and [“Setting up an individual Asset Assignment Dialog” on page 67](#).

- To make the **Automatically update to latest asset version** check box visible in the **Asset Assignment** dialog at Hybris Backoffice, the following parameters in the `otmmaddonbackoffice\resources\otmmaddonbackoffice-backoffice-config.xml` configuration file must be set:

```

<editorArea:editor-parameter>
<editorArea:name>isAutomaticallyUpdateToLatestAssetVersionVisible
</editorArea:name>
<editorArea:value>true</editorArea:value>

```

```
</editorArea:editor-parameter>

<editorArea:editor-parameter>
  <editorArea:name>automaticallyUpdateToLatestAssetVersionDefault
</editorArea:name>
  <editorArea:value>false</editorArea:value>
</editorArea:editor-parameter>
```

Activate this changes with a Hybris restart or a reset of the cockpit-config.xml.

With this configuration, all assets, where the check box was activated during assignment, always will be updated automatically to the latest version.

3.14 Configuring asset purge notifications

Asset Purge notifications are sent from OTMM to Hybris. For each notification, a Hybris task is started that deletes all OpenText media items that are related to the purged asset. For OpenText media items that make use of Media Delivery Service, also Media Delivery Service is informed to remove all asset deliveries related to the purged asset from its asset delivery store.

Administrators can search for tasks started by asset purge notification using **Hybris Backoffice**.

To administer asset purge notifications:

1. Login as administrator to **Hybris Backoffice (/backoffice)** and select **DAM by OpenText > OTMM Event Tasks**.
2. Search for tasks started by asset purge notification:
Use search condition **ActionStarts with: otmm**.

The search result shows all purge tasks (otmmDeleteAssetTaskRunner) that are still in progress or have failed. It is possible to invoke a task once more, after the cause of the failure is eliminated.

3.15 Configuring assignment of approved assets

OTMM offers capabilities to manage an asset lifecycle including the possibility to maintain an approval state for assets. With DAMLink Hybris, this approval state can be evaluated: assets can only be assigned in Hybris, if they have been approved.

This section describes the configuration of this function for specific metadata models.

After configuration, not finally approved asset are not visible in the Asset Assignment Dialog and thus cannot be used for manual assignment. They are also not considered and assigned by the AutoAssignment cronjob.

Prerequisite The field ARTESIA.FIELD.LIFECYCLE APPROVAL STATE must be contained and maintained in the OTMM metadata model.

To configure use of approved assets only:

1. In `local.properties` configuration file, add the following entry:
`otmm.approved.model.list = <model1>, <model2>, ...`
The list should contain the metadata models that require approval (comma separated).
Example:

```
otmm.approved.models.list = OPENTEXT.PRODUCTS
```

With this configuration, only approved assets with metadata model `OPENTEXT.PRODUCTS` can be assigned to Product items in Hybris. For assets with other metadata model, this restriction does not apply.
In detail this means that the attribute `ARTESIA.FIELD.LIFECYCLE APPROVAL STATE` must have the value `APPROVED`.

2. Restart Hybris.

3.16 Thumbnails for non-image assets

You can supply thumbnail images for assets that are no images. For example a PDF icon for all PDF assets or some kind of icon to identify all videos. These thumbnails are used in Hybris Backoffice, in product cockpit, and in WCMS cockpit.

The configuration of the usage of thumbnails for certain MIME-types is done in the `local.properties` file. For a description of the properties, refer to the `project.properties` file.

DAMLink Hybris uses the thumbnails that are defined in OTMM.

You can implement the public API interface `OtmmThumbnailService` to create your own version of the thumbnail support or to create thumbnail support for your customer facing web front end. For more information, see *OpenText DAMLink for SAP Solutions Hybris Integration - Customization and Programming Guide (DAMSAP-PHI)*.

Excerpt from the `project.properties` file with the thumbnail related configuration:

```
#####
#####
# Thumbnail
configuration                                     #
#####
#####
#
# This property configures a pattern whitelist of mime types.
# If the mime type of a Media item matches a pattern, their OTMM
thumbnail is displayed.
# If the property is not configured or has an empty value no
thumbnail is used.
#
```

```
# Syntax:
# - "*" is wildcard for any character
# - several Mime Type (patterns) can be separated by comma
#
# Examples:
# */* - thumbnails are used for all mime types
# video/* - thumbnails are only used for videos
# application/pdf - thumbnails are only used for PDF files
otmm.thumbnail.mimes=\
video/*,application/pdf,\
application/msword,application/vnd.ms-word,\
application/vnd.ms-excel,application/excel,\
application/vnd.ms-powerpoint,\
application/powerpoint

# This mime type is returned by the default implementation of
# OtmmThumbnailService.retrieveThumbnailMimeType.
otmm.thumbnail.fixed.mime=image/jpeg

# Set this to true if you enabled authentication
# for thumbnail URLs in the file
# /otmmws/web/webroot/WEB-INF/config/security-spring.xml
# See the comment there for further details
# If set to true the images are not cached
# by the browser after a session ends.
# that implies that for each user session the URLs are different.
otmm.thumbnail.protected = false
```

3.17 Product lookup

The goal of the product lookup feature is to copy metadata from a product in Hybris to an asset in OTMM. Typically, you do not only want to copy the metadata, but also to assign the Hybris product to the OTMM asset. To achieve this, you need to configure AutoAssignment: If you configure the product lookup such that the OTMM fields necessary for AutoAssignment are filled, then, after the values are saved, an assignment of the asset to the product is performed by a suitably configured AutoAssignment cronjob.

The product lookup works as follows:

1. While an OTMM asset is open for edit, click “...” to open the **DAMLink Lookup** dialog.
2. The search returns a list of (the metadata of) one or more Hybris products, like product name or product catalogversion, or more specific metadata.
3. Select one or more products from the list. You can only select more than one product if you copy to a tabular field group.
4. The metadata in the UI are filled with the values from the list. Click **Save** to store the metadata with the OTMM asset.
5. Optional: If you copied the fields necessary for an AutoAssignment (that is product ID or EAN, and product attribute, perhaps also systemid and tagname),

and an AutoAssignment cronjob exists for the catalogversion of the product, then, when the cronjob runs, an automatical assignment takes place.

The search within Hybris, display of the hit list, and transfer of metadata are highly customizable.

The following sections contain an example that is delivered with the product including required configuration steps and a general description of the configuration options.

3.17.1 Example for product lookup with AutoAssignment

An example for a product lookup together with AutoAssignment is delivered with the product.

The example works using the following files:

- `metadataview.xml` file

The file is located in `<TEAMS_HOME>/ear/artesia.ear/otmmux.war/ux-html/ot_damlink_bo_lookup/metadataview.xml`

- `damlink_config.xml` file

The file is located in `<TEAMS_HOME>/data/sap/damlink_config.xml`

- `AutoAssignmentExample.json`

The `AutoAssignmentExample.json` file needs to be located here:

`<TEAMS_HOME>/data/sap/otmmsetup/AutoAssignmentExample.json`. No changes are required.

3.17.1.1 Configuration of the example

To get the example running, some configuration is required.

metadataview.xml

In the `<TEAMS_HOME>/ear/artesia.ear/otmmux.war/ux-html/ot_damlink_bo_lookup/metadataview.xml` file you configure the fields or tabular field groups that show the “...” button to open the **DAMLink lookup** dialog. You need to adjust or in some cases create the file. For the example, you need the following configuration of the file which configures the “...” button for the `AUTO.SCALAR.PRODUCT ID` field.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MetadataView>
<MetadataView>

<FieldLookup id="AUTO.SCALAR.PRODUCT ID"/>

</MetadataView>
```

damlink_config.xml

The <TEAMS_HOME>/data/sap/damlink_config.xml file contains the connection settings and settings for search and results. For this example, the settings for search and results are preconfigured.

Adjust the <connection> tag with id="Hybris_Search":

- Set active="true".

- Adjust the URL according to your environment: Protocol (HTTP or HTTPS), port, and hybrisserver.

```
<connection active="true" id="Hybris_Search" url="http://hybrisserver:9001/otmmaddon/OTDC_DAM_INTERFACE?wsdl">
```

For the example, you should have a line in damlink_config.xml with <connection active="false" id="Hybris_Search" ... />, and a section <searchconfig connectionid="Hybris_Search" ... />. If not, locate the damlink_config.xml in the installation media, and copy the lines from there into your damlink_config.xml.

AutoAssignment Metadata

The following configuration is required:

1. Make sure OTMM is running.
2. In a web-browser open this URL:
`http://<otmmhost>:11090/ot-damlink/monitor?checkAndCreateMetadata&schema=AutoAssignmentExample.json`
An OTMM field group AUTO.FIELD.GROUP is added to your metadata field groups.
You can verify this at: `http://<otmmhost>:11090/teams/Admin.do?action=display-meta-editor,field group`.
3. A metadata model with at least the Associated Field Groups ARTESIA.CATEGORY.ASSET, AUTO.FIELD.GROUP, and HYBRIS.PRODUCT.INFO is required. If you don't have a Property Template for this model, you must create it.
Assets for this model now display a section with Auto Ass. Metadata and a section with Hybris Item Reference.
If you edit such assets there now are 3 dots (...) near the Auto Ass. scalar product ID

Hybris configuration

On Hybris side the following configuration is required.

The installation of DAMLink Hybris and the otmmaddon update with the Hybris admin console is a prerequisite and makes an otmmAutoAssign cronjob available.

To configure the otmmAutoAssign cronjob

1. Open Hybris Backoffice and click **System > Background Processes > cronjobs**.
2. Search for cronjobs starting with otmm.
3. Select the otmmAutoAssign cronjob.

4. Set the **Catalog version** for your products.
5. Set **Enabled** to true.

otmmAutoAssignJob : otmmAutoAssign - FINISHED - SUCCESS

LOG TASK RUN AS TIME SCHEDULE SYSTEM RECOVERY ADMINISTRATION

ESSENTIAL

Code otmmAutoAssign	Current status FINISHED	Job definition otmmAutoAssignJob	Last result SUCCESS
Timetable Several execution times	Last start time Aug 17, 2017 3:08:01 PM	Enabled <input checked="" type="radio"/> True <input type="radio"/> False	Last end time Aug 17, 2017 3:08:01 PM

JOB PARAMETERS

Catalog version Default-Catalog : Staged	MaxAssets 1000	Custom Keyword	Use Latest Version <input checked="" type="radio"/> True <input type="radio"/> False <input type="radio"/> N/A
Model Id	Folder Id	Age In Days 10000	Number of Threads 1

3.17.2 Configuring product lookup

The following configuration of the product lookup feature is required or possible, respectively.

- Field or tabular field group for which the **DAMLink lookup** dialog is available.
- Connection to your Hybris server.
- Settings for the search:
 - Fields that are available for the search
 - Available search methods, for example that a value begins with the characters that are entered in the search.
 - Predefined search values
 - Fields that are used for search with predefined values but are not visible to users.
 - Type ahead search
- Settings for the result list:
 - Set of metadata that is copied to OTMM.
 - Fields that are displayed in the result list; one column for each field.
 - Fields that are displayed in the result list but are not copied.
 - Fields that are not displayed in the result list but are copied nevertheless.

The settings are configured in the following files:

metadataview.xml

In the `<TEAMS_HOME>/ear/artesia.ear/otmmux.war/ux-html/ot_damlink_bo_lookup/metadataview.xml` file you configure the fields or tabular field groups that show the ... button to open the **DAMLink lookup** dialog. The field or tabular field group must be part of your metadata model. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MetadataView>
<MetadataView>

<FieldLookup id="AUTO.SCALAR.PRODUCT ID"/>

</MetadataView>
```

damlink_config.xml

The `<TEAMS_HOME>/data/sap/damlink_config.xml` file contains the connection settings and settings for search and results.

DAMLink Lookup

Product ID	begins with		
Product Name	begins with		X
Product Description	begins with		X
Product Catalog	begins with		X
Product CatalogVersion	equals		X
Product Category	begins with		X

☐ Show All Languages

Product ID	Product Attribute ID	System ID	Product Name	Catalog Mnemonic
No Items				

Figure 3-1: DAMLink Lookup dialog

Connection

In the `<otdcservices>` section add a section:

```
<connection active="true" id="Hybris_Search_<hostname>"
url="https://<hostname>:9002/otmmaddon/OTDC_DAM_INTERFACE?wsdl" >
```

```
<events>0</events>
</connection>
```

- Set `active="true"`.
- Use an ID that is unique, for example `Hybris_Search_<hostname>`.
- Adjust the URL according to your environment: Protocol (HTTP or HTTPS), port, and `hybrisserver`.
- Set `<events>0</events>` to 0.

Search configuration

Each `<searchconfig>` element defines basic search options:

- **id** - ID of this search configuration. Used for internal mapping only. The ID needs to be unique within the `damlink_config.xml` configuration file.
- **damfieldid** - ID of the field where the lookup button is displayed in the OTMM UI, for example `OPENTEXT.TABLE.PRODUCT`. It must match one of the FieldLookup IDs in `metadataview.xml`.
- **objecttype** - The `objecttype` is either `Product` or the name of a subtype of `Product`, for example `ApparelProduct`.
- **connectionid** - ID of connection as defined in the `<connection>` node.
- **istabular** - Set to `true`, if the `damfieldid` is the name of a tabular field group. Set to `false`, if `damfieldid` is the name of a scalar field.

Type ahead (optional)

The `<typeahead>` tag makes search values available from existing values while a user starts typing a value in the search field.

- **max-number-of-suggestions** - Maximum number of suggested search values that are displayed in the list.
Default is 7.
- **min-length** - Number of characters a user needs to type, until the list of search values is populated.
Default is 2.

Search fields

The search fields define which searches can be performed on Hybris. Each search field represents one search expression, with which one Hybris product property is compared to one value. For example, you can search for products where the catalog version is "Staged".

The search fields are shown as rows in the **DAMLink Lookup** dialog. If you enter a value into the value field, the search is performed. If you enter values in more than one row, the searches are combined implicitly with "AND". For example, you can search for a Hybris product with a name starting with "Canon" AND with a catalog version "Staged".

You can enter multiple values to the field. These values are combined with an implicit OR.

A search field is defined:

```
<searchfield id="<unique id>" sapid="<name of a searchable
Hybris product property>" name="<displayed name>"
datatype="string">
  <operators>
    ...
  </operators>
  <value />
</searchfield>
```

- **ID (mandatory)** - Identifies a search field internally and must be unique within a <searchfields> node.
- **sapid (mandatory)** - The sapid name must be a name as used by the Hybris Flexible Search. For the Product base type, the following names are supported
 - code
 - name
 - description
 - catalog
 - catalogversion
 - categoryFor subtypes of Product, you can use the names of attributes as defined in the respective <itemtype> which is defined in <extensionname>-items.xml.
- **name (mandatory)** - Specifies the title displayed in the first column of the UI.
- **datatype (mandatory)** - You can only use the data type string.
- **visible (optional)** - visible="false" hides the searchfield from the UI. This can be useful to include a predefined value to the search which cannot be changed by the user and in this way generally restricts the search.

Operators

Each search field can support a different set of <operator> elements. The operators are displayed in the second column of each search condition.

The following operators are supported:

```
<operator id="BG" name="begins with" />
<operator id="CP" name="contains pattern" />
<operator id="EQ" name="equals" />
<operator id="EN" name="ends with" />
<operator id="<id>" name="typeahead" />
```

<operator id="<id>" name="typeahead" /> makes the type ahead functionality available for the respective searchfield. If used: If a user starts typing a value in a searchfield, then after a few characters a popup list with suggestions appears. The suggestions are taken from existing values for the respective attribute. For information about configuring the type ahead functionality, see [Type ahead \(optional\) on page 99](#).

Value

With the `value` element you can predefine values for a specific searchfield. Users can overwrite the predefined values. Predefined values are shown in light grey in the UI, unless the `<visible>` field is set to `false`.

Example: If only search for “Staged” products is required, you can set the value for the catalog version to Staged and the `<visible>` field to `false`.

You can define a list of comma-separated values. These are combined with an implicit OR.

Result fields

The `<resultfields>` elements represent the columns in the result list and control which metadata is copied.

A result field is defined:

```
<resultfield visible="<true|false>" id="unique id" sapid="Name
of Hybris Product attribute or Spring SpELExpression"
      damid="OTMM field name" name="displayed name"
datatype="string"
/>
```

The combination of `sapid` and `damid` matches Hybris Product attributes to OTMM asset fields.

- **visible (optional)** - If set to `true`, the column is displayed.
Invisible columns are useful for including data that you want to store in the asset without showing them in the result list of the lookup.
The default is `false`.
- **ID (mandatory)** - Identifies a result field internally and must be unique within a `<resultfields>` node.
- **sapid (mandatory)** - The `sapid` determines which metadata value is retrieved from the Hybris Product.
Special names are:
 - `attrName` that evaluates to the string “picture”
 - `tagName` that evaluates to “TAG”
 - `systemId` that evaluates to the name of the Hybris system ID, as it is configured by `otmm.hybris.systemId`

If the `sapid` is just a string `xxx`, then there must exist a string valued attribute with the name `xxx`, for example `xxx=code`, `pk`, or `ean`.

The product name, however, is “multilingual” in OTMM terminology, or “localized” in Hybris terminology. When you just use `name`, the default language of OTMM is chosen, if you assign to a non-multilingual OTMM field. You can specify `name(name["en"])` as the `sapid`, to get only the English name.

The attribute `catalogVersion` in Hybris is not evaluated into a simple string. To get a simple string though, you can use Spring SpELExpressions:

 - `catalogMnemonic(catalogVersion.mnemonic)`

- `catalogId(catalogVersion.catalog.id)`
- `catalogVersion(catalogVersion.version)`

For SpelExpressions, the ID of the resultfield must be the first part of the sapid before "(".

- **damid (mandatory)** - The damid must be the name of an OTMM field. Use the value unused to display the column in the result list but to not copy the data.
- **name (mandatory)** - Specifies the column title displayed in the UI
- **datatype (mandatory)** - You can use the data type string.
- **ismultilanguage (optional)** - Only needed for fields that have values in multiple languages. Set this value to `true` to assign "localized" Hybris attributes to "multilingual" OTMM-fields.

If `ismultilanguage="true"`, then add for each supported language code an element `<resultlanguage name="display name" damid="<same damid as the resultfield>" code="<language code>" />`.

Chapter 4

Security

DAMLink Hybris uses user management and authentication mechanisms of OTMM and Hybris. For information, refer to the corresponding documentation (see [“Further documentation sources” on page 11](#)). This chapter provides references to additional information and gives some special comments.

4.1 Configuring OTMM and FTP connection

In the `local.properties` file, configuration data for OTMM and FTP connections must be entered. This includes passwords that only can be entered in clear text in the `local.properties` file. As an alternative, **DAMLink Hybris** allows you to configure OTMM and FTP user names together with passwords in an encrypted manner in Hybris Backoffice. The Hybris “Transparent Attribute Encryption” feature is used.



Note: Use of encrypted passwords is recommended. But it is still possible to insert unencrypted passwords.

This section describes the configuration of a user with encrypted password for the FTP connection to OTMM with and without using Single Sign-on (SSO).

Prerequisites The following prerequisites must be met:

- The following properties from `local.properties` or `project.properties` file are commented out. (This is default configuration.):

```
#otmm.server.login=<otmmuser>
#otmm.server.password=<pw>
...
#10013=FTPUser
#10014=Ot$temp123
```

If you have changed the `local.properties` or `project.properties` file, restart the Hybris server.

- A technical OTMM user account exists that can be used for the connection.

To configure the OTMM connection:

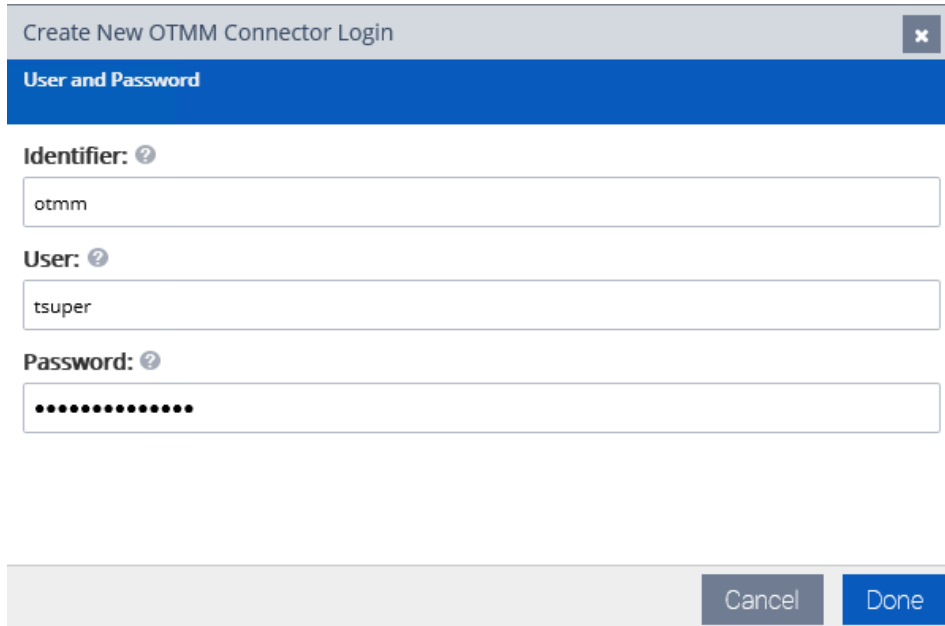
1. **Only for SSO:** Enable Single Sign-on between Hybris and OTMM. For information, see [“Single sign-on between Hybris and OTMM” on page 107](#). Restart the Hybris server after enabling SSO.
2. Sign in to **Hybris Backoffice** (`/backoffice`) and navigate to **DAM by OpenText > Logins**.
3. Click **+ OTMM Connector Login**.

4. Enter the following values for the OTMM connection:

Identifier: otmm

User: <OTMM user name>

Password: Without SSO, enter the <password>. With SSO, leave this value blank.



Create New OTMM Connector Login

User and Password

Identifier: ?
otmm

User: ?
tsuper

Password: ?
.....

Cancel Done

5. Click **Done**. A new entry is inserted into the **OTMM Connector Login** list.
6. To test the new connection:
Login to the **Hybris Product Cockpit** as a product manager. Open the product editor of a product and navigate to Multimedia attributes.
Click the **Media Reference Wizard** next to the product's main image attribute.
The **OpenText Media Management – Asset Assignment** dialog should be displayed.
Since the OTMM connection user name or password is invalid, an appropriate error message is displayed.
7. Repeat steps 3 to 5 with the following values for the FTP connection:
- Identifier: ftp
 - User: <FTP user name>
 - Password: <FTP password>
8. To test the connection:
Login to the **Hybris Product Cockpit** as a product manager. Open the product editor of a product and navigate to Multimedia attributes.
Click the **Media Reference Wizard** next to the product's main image attribute.
The **OpenText Media Management – Asset Assignment** dialog should be displayed. Choose an image and click **done**. If you have set `otmm.direct.sync`

to FALSE you additionally have to start the cronjob `OtmExportServiceJobExport` and afterwards `OtmAssetSyncJob`. If the assignment is successful your FTP connection works.



Note: If automatic login should be performed even if SSO is disabled, set the following property to true. With this setting, the login is done with the technical user defined with `otmm.server.login` property.

```
otmm.server.for.productcockpit.users.isPreAuthenticationForTechnicalUserEnabled=true
```

More
information

For more information about the Hybris encryption feature, see Hybris wiki: Transparent Attribute Encryption (TAE) (<https://wiki.hybris.com/pages/viewpage.action?pageId=141793006>).

4.2 SSL support

The OTMM extension supports the communication to OTMM based on Web Services with SSL protection. This section describes the necessary configuration steps to configure OTMM Web Services with SSL:

- Creation and import of the certificate
- Configuration of OTMM web services
- Configuring Hybris for HTTPS



Adapt to your environment

The following procedure describes the SSL configuration steps for a given system configuration. Consult the current documentation of your OTMM and tool versions. For more information, see *“Further documentation sources” on page 11*.

For information about SSL configuration on OTMM, see section 3.9 *“Configuring SSL”* in *OpenText Media Management - Administration Guide (MEDMGT-AGD)*.

For more information about the `%JAVA_HOME%\bin\keytool` utility, see the Java documentation: <http://docs.oracle.com/javase/8/docs/technotes/tools/windows/keytool.html>.

To create the certificate on OTMM:

1. Login as Administrator to OTMM server.
2. Create a certificate. Use the command line tool `keytool` provided as part of a java installation for creation of a SSL key and certificate.

For a description of the necessary steps, refer to the OTMM documentation: see section 3.9.1 *“Configuring OTMM for SSL”* in *OpenText Media Management - Administration Guide (MEDMGT-AGD)*.

To add the OTMM certificate as trusted SS certificate in Hybris:

1. The created SSL certificate file (.cer) is stored in the keystore used by OTMM JBoss server of your OTMM server. Export it to the Hybris server using the following command:

```
keytool -export -alias <alias> -file <CertificateFilePath>
      -keystore <KeyStoreFilePath> -storepass <KeystorePassword>
```

The <CertificateFilePath> parameter defines the target path. The <alias>, <KeyStoreFilePath> and <KeystorePassword> parameters refer to the created OTMM certificate. For example:

```
keytool -export -alias otmm -file otmm.cer
      -keystore C:\OT_Install_Sources\JBoss\jboss-eap-5.2.0\jboss-
eap-5.2\jboss-as\server\teams\conf\otmm.keystore
      -storepass opentext
```

2. Import the certificate to a new keystore:

```
keytool -import -alias <alias> -file <CertificateFilePath>
      -keystore <KeyStoreFilePath> -storepass <KeystorePassword>
```

For example:

```
keytool -import -alias otmm
      -file C:\Users\Administrator\Desktop\SSL_CERTS\otmm.cer
      -keystore C:\Users\Administrator\Desktop\SSL_CERTS\otmmint.keystore
      -storepass opentext
```

3. Open the wrapper.conf file from the <HYBRIS_PLATFORM_HOME>\tomcat\conf location and add the following entries to the existing configuration (with new counter number):

```
wrapper.java.additional.<counter+1>=-
Djavax.net.ssl.keyStore=<KeyStoreFilePath>
```

```
wrapper.java.additional.<counter+2>=-
Djavax.net.ssl.keyStorePassword=<KeystorePassword>
```

for example if last existing entry is wrapper.java.additional.21:

```
wrapper.java.additional.22=-Djavax.net.ssl.trustStore=C:\Users\Administrator
\Desktop\SSL_CERTS\otmmint.keystore
```

```
wrapper.java.additional.23=-Djavax.net.ssl.trustStorePassword=opentext
```

If you execute ant clean all , you have to add the properties again.

To enable SSL connection from Hybris to OTMM:

1. Login to Hybris server as administrator.
2. Change configuration of OTMM server from **http** to **https** in local.properties or project.properties file. The local.properties overwrites the project.properties values.
Use your technical OTMM user account for the OTMM connection.

**Example 4-1: OTMM connection with default user**

```
# OTMM Server configuration
otmm.server.scheme=https
```

```
otmm.server.host=otmm.opentext.com
otmm.server.port=11090
otmm.server.login=<otmmuser>
otmm.server.password=opentext
```



3. To activate your changes, restart the Hybris server.



Note: To avoid the so called “Poodle security bug” in SSLv3, you can limit SSL protocols to Transport Layer Security protocol (TLS) support. This can be done by starting the respective Java Virtual Machine with the following parameter:
-Dhttps.protocols= TLSv1, TLSv1.1, TLSv1.2.

4.3 Single sign-on between Hybris and OTMM

This section describes the configuration steps to enable Single Sign On (SSO) for Hybris and OTMM.

You can use the same login for OTMM and for Hybris. This login is also used in the Asset Assignment Editor for searching and browsing assets in OTMM and for the upload of assets into OTMM.

To enable Single Sign On:

1. Set the configuration entry: `otmm.server.sso = TRUE` in `local.properties` or `project.properties` file.
2. Deploy a jar file with the shared secret between Hybris and OTMM:

- a. If there is already a `security-sso.jar` in the following location:
<OTMM installation directory>/MediaManagement/plugins/damlink
then just copy this file to the otmmaddon extension’s lib folder (<hybris suite install directory>/hybris/bin/custom/otmmaddon/lib).

- b. Otherwise: In the `security.properties` template file located in
<hybris suite install directory>/otmmaddon-templates
(formerly extracted from `damlink-hybris-integration-version.zip`),
you find the
`com.vignette.shared.security.shared_key` key.
To adjust the key, write some arbitrary characters into the key.

Store the adjusted `security.properties` file in a jar file in these locations:

Hybris: <hybris suite install directory>/hybris/bin/custom/
otmmaddon/lib/security-sso.jar

OTMM based on JBOSS: <OTMM installation directory>/
MediaManagement/plugins/damlink/security-sso.jar

OTMM based on TomEE: <OTMM installation directory>/ear/
artesia/lib/security-sso.jar



Note: A restart of the MediaManager Service is required in case of 2b.



Note: Single sign-on for launching the OTMM Asset Inspector is done with Javascript inside the browser. Therefore, the cross-origin resource sharing (CORS) configuration for the REST API in OTMM has to be extended with the Hybris hostname.

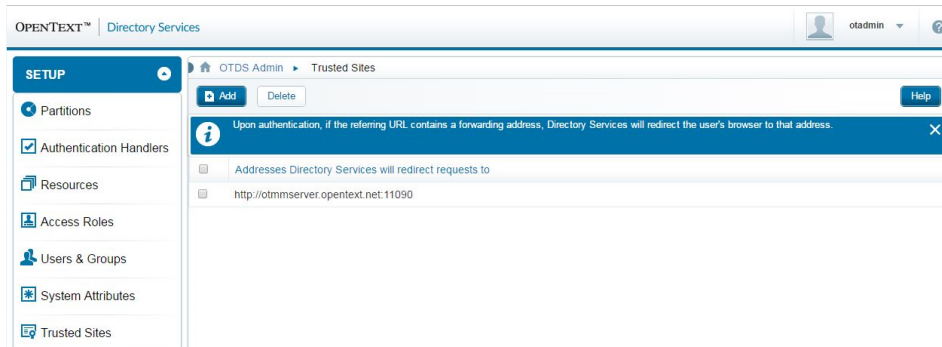
You can edit the respective configuration setting in the administration UI in **Settings** under Configuration Group **API Configuration**.

OTDS login page enabled

In case the OpenText™ Directory Services (OTDS) Login Page (Redirect to OTDS) is enabled, the **OTMM web access** must be configured as a trusted website to avoid an OTDS login for each Asset Inspector launch.

To configure OTMM web access as a trusted site:

1. Check if the OTDS login page is enabled:
Login to OTMM as administrator (/teams).
Select **Settings** menu. Search for the OTDS_LOGIN_PAGE_ENABLED property. If this property is set to TRUE, the OTDS login page is enabled.
2. Find the URL of OTMM web access:
Select **Settings** menu, WEB_APPLICATIONS\CONFIG component configuration. The **OTMM web access** parameter is defined by WEB_APPS_BASE_URL.
3. Set OTMM web access as a trusted site in OTDS:
Login to OTDS as administrator to **Trusted Sites** (/otds-admin#trustedsites).
Add the OTMM web access URL as a trusted site.



4.4 Managing Media Delivery Service user permissions

Media Delivery Service is designed to be used in the Hybris webstore. Depending on your scenario, it might be exposed to the external internet. It is recommended to use an OTMM user with limited permissions.

Media Delivery Service user needs the following permissions:

- View
- View Summary
- Preview
- Export



Note: It is recommended to use a strong password for IMCONVPWD property. (For Media Delivery Service configuration, refer to [“Configuring properties” on page 48.](#))



Important

If you need to remove the access to certain assets from external internet, for example because you have no longer the license for it, it is not sufficient to delete or revoke the permission from OTMM. You have to delete it also from the asset delivery store of Media Delivery Service using the `deleteAsset` command line tool. For a detailed description, refer to [“Using command line tools” on page 52.](#)

4.5 Shared secret of Media Delivery Service and DAMLink Hybris

The properties `otmm.media.servlet.pwd` in `local.properties` file and `IMCONVPWD` in `imconv.properties` file must have the same value. This value is used as a shared secret by the communication between Media Delivery Service and the DAMLink Hybris.

It is recommended to use a password generator to create this secret, especially as this value need not to be entered or remembered by an user.

For more information on configuring of the `pwd` property, see [“Media Delivery Service type” on page 37.](#)

4.6 Thumbnail related security

- Thumbnails displayed in the Asset assignment dialog are not protected by authentication by default. This implies that anyone who has the URL of the thumbnail is able to access the thumbnail of the respective asset. This is also valid for the preview image of the asset.

You can configure the system to use authentication. However, this excludes thumbnails from caching and their display might be slow.

To enable authentication:

1. In the `/otmmws/web/webroot/WEB-INF/config/security-spring.xml` file, change the `access` setting for the rendition URLs from `permittAll` to the following:

```
<intercept-url pattern="/otmmProxy/otmmapi/v*/renditions/*"
  access="isAuthenticated() and
  (hasRole('ROLE_CMSMANAGERGROUP') or
    hasRole('ROLE_PRODUCTMANAGERGROUP') or
    hasRole('ROLE_ADMINGROUP'))"
  requires-
  channel="#{configurationService.configuration.getString
    ('webservicesscommons.required.channel', 'https')}"
/>
```

2. Enable `appendOAuth2BearerTokenToThumbnailUrl`. This is done in the OTMM administration, **Settings > DAMLink_UI > Edit components**.
 3. In the `local.properties` file, set the following property: `otmm.thumbnail.protected = true`
- Thumbnails for OpenText Media items of type Media Delivery Service or External URL as described in *“Thumbnails for non-image assets” on page 93* are not protected by authentication. The thumbnail images are retrieved using Media Delivery Service.
 - Thumbnails for OpenText Media items of type Content By Export as described in *“Thumbnails for non-image assets” on page 93* are protected by Hybris. For more information, see the Secure Media Access chapter in the Hybris documentation: <https://help.hybris.com/6.3.0/hcd/8c586b0686691014ad288714ae4e3924.html>.

4.7 HTTPS and CORS

For DAMLink Hybris the HTTPS protocol and cross origin requests (CORS) are a requirement.

HTTPS The following components must be accessible via HTTPS and must have a valid certificate:

- DAMLink Hybris extensions, into which the OpenText Asset Assignment Dialog is integrated
- OTMM
- Media Delivery Service in scenarios that make use of Media Delivery Service

CORS The following components must permit cross-origin requests to the Hybris server.

- OTMM
- Media Delivery Service in case the cropping feature within smartedit is used

4.8 Securing notifications

Hybris receives notifications from OTMM on updated or deleted assets. This could be exploited maliciously.

There are two possible countermeasures:

- Secure the communication using SSL with client certificates.
This measure is more effective.
- Allow Hybris to receive notifications only from specific IP addresses.
This measure is less effective but easier to implement.

To restrict to specific IP addresses:

1. Open the *<hybris installation directory>/bin/custom/otmmaddon/web/webroot/WEB-INF/web.xml* file.
2. In the file, search for RemoteAddrFilter.
By default, all IP addresses can access: `<param-value>.*</param-value>`
3. Replace `.*` with IP addresses to which you want to grant access. Separate multiple IP addresses with a vertical bar: `|`. `|` means "or".
You can use regular expressions to specify the IP addresses.



Example 4-2:

```
<filter-class>
    org.apache.catalina.filters.RemoteAddrFilter
</filter-class>
<init-param>
<param-name>allow</param-name>
<!-- the param-value must be a java.util.regex.Pattern
-->
```

```
<!-- next line allows every address: -->
<param-value>.*</param-value>
<!-- Example for localhost: param-value -->
<!-- 127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:1</param-
value> -->
<!-- Example for two addresses: param-value -->
<!-- 123.123.123.123|123.123.123.124</param-value> --
>
</init-param>
```



4.9 Restricting CORS requests to specific host

There is a CORS filter in `webapps/ImConvServlet/web.xml` that allows CORS requests from any host:

```
<param-name>cors.allowed.origins</param-name>
<param-value>*</param-value>
```

To restrict the access, you can set a more specific `cors.allowed.origins` parameter.

4.10 Log forging prevention

Attackers might forge log files by adding line feeds to content they expect to be logged. For more information, see https://www.owasp.org/index.php/Log_Injection. To prevent such attacks for the `otmmaddon.log` file, you can configure in the `local.properties` the following `log4j2` layout:

```
log4j2.appender.otmm.layout.pattern = \
%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%t] [%c] - \
%replace{%m}{(\r\n|[\r\n])}{$1NEWLINE:    }%n
```

This configuration adds the following string in front of line feeds that are part of a log file entry:
"NEWLINE:" plus 4 blanks.

Chapter 5

Operations

This chapter provides a summary of important administration topics for the DAMLink Hybris. For general operations and support topics for SAP customers, there is also a specific section on “Operations and Support Notes” maintained in the SAP Digital Asset Management Release Notes.

5.1 Administration tools

DAMLink Hybris does not provide any administration or configuration tools of its own because it is run as an add-on to hosting applications. Administrative tasks make use of the following:

- **Hybris Backoffice** (/backoffice) This UI supports administration and configuration.
Start **Hybris Backoffice** with the following URL: `http://<hybris server>:<port>/backoffice`.
You find the DAMLink Hybris administration UI in the navigation under the **DAM by OpenText** node.
- The tools supplied by SAP or OTMM. For information about OTMM administration, see *OpenText Media Management - Administration Guide (MEDMGT-AGD)*.

5.2 Identify the version and build number



Note: This section describes the version identification for **DAMLink Hybris Extensions**. For information about Media Delivery Service, see “[Version info and logging](#)” on page 57.

To identify the version and build number:

1. Open **Hybris Backoffice**: `http://<hybris server>:<port>/backoffice`
2. In the menu, click **DAM by Opentext > Support Information**.

5.3 High availability

DAMLink Hybris is deployed as an add-on to Hybris Commerce Suite and OTMM. Hence, high availability aspects are covered by those two host environments.

For more information on high availability and scalability aspects of the solution, see:

- *OpenText Media Management - Installation Guide (MEDMGT-IGD).*
- *OpenText Media Management - Architecture Overview Guide (MEDMGT-ARC).*

5.4 Starting and stopping the system

Because of the add-on nature of DAMLink Hybris, it cannot be started or stopped separately from its host environments.

DAMLink Hybris, deployed in Hybris Commerce Suite, is started and stopped as Hybris Commerce Suite is started and stopped.

5.5 Periodical tasks

DAMLink Hybris does not require any periodical tasks.

The current status of an assigned asset and more information about responsible jobs can be checked in **Hybris Backoffice** (/backoffice). For more information, see [“Check assignment status” on page 70](#).

5.6 Backup and recovery

To store its data, DAMLink Hybris leverages the technical environments of its two host systems Hybris Commerce Suite and OTMM. Any backup or recovery strategy needs to involve the infrastructure provided by those two systems.

Because DAMLink Hybris itself does not store any business data, the only back-up topic relevant for DAMLink Hybris is the application configuration and the software itself.

5.7 Monitoring

You can check the current status of an assigned asset and more information about responsible jobs in the **Hybris Backoffice** (/backoffice). For more information, see [“Check assignment status” on page 70](#).

For more information, see Section 10.6 “Monitoring” in *OpenText DAMLink for SAP Solutions - Installation and Configuration Guide (DAMSAP-IGD)*.

5.8 Logging

To record **DAMLink Hybris** events, log functions on Hybris and on OTMM side can be used. This section describes the setup and use of event log files.



Note: This section describes the logging for **DAMLink Hybris** Extensions. For information on Media Delivery Service logging, see “[Version info and logging](#)” on page 57.

5.8.1 Configure Hybris log

The **DAMLink Hybris** Extensions write their messages in a separate log file. A default for logging pre-configured. For customization, this section describes the necessary settings. Regular Java **log4j2** syntax is used. For information about log4j2, see [apache.org \(http://logging.apache.org/log4j/2.x/manual/configuration.html\)](http://logging.apache.org/log4j/2.x/manual/configuration.html).



Important

To activate changes of the logging configuration, a restart of Hybris is necessary.

<i>Configuration files</i>	You can control log level settings with logging properties in the <code>bin\custom\otmmaddon\project.properties</code> file. These settings can be overridden with the <code>config\local.properties</code> file (file paths below Hybris main installation directory). The logging settings in the <code>project.properties</code> file, are commented out by default. See also the Logging part in “ Example: project.properties ” on page 121.
<i>Log file</i>	The log file is defined in the <code>project.properties</code> setting <code>OTMM_LOG_DIR</code> . By default, the log file is stored in the <code>log/otmm_connector</code> directory in the Hybris main installation directory.
<i>Log levels</i>	<p>The available log levels are (ordered by descending severity):</p> <ul style="list-style-type: none"> • fatal: Fatal errors that prevent the Hybris system from working • error: Error messages (exception is caught) • warn: Warning about a behavior that is not necessarily an error but requires awareness of an administrator • info: General information (program start and stop, connections info ...) • debug: General debugging (finding bugs) • trace: Detailed debugging
<i>Log settings</i>	The <code>loglevel</code> for the DAMLink Hybris logger can be set with the <code>log4j2.logger.otmmconnector.level</code> property. By default, the log level <code>warn</code> is set.



Example 5-1: Set log levels

You can set the log level to `debug` with the following configuration. It is possible to adjust the log level granular for each class in the java package `com.opentext.hybris.otmmconnector.service`. Add the properties to the `local.properties` file as you require:

```
# General log level
log4j2.logger.otmmconnector.name = com.opentext.hybris.otmmconnector
log4j2.logger.otmmconnector.additivity = false
log4j2.logger.otmmconnector.level = debug
log4j2.logger.otmmconnector.appenderRefs = otmm
log4j2.logger.otmmconnector.appenderRef.otmm.ref = OTMM_LOG

# The log level can be specialized for packages or even classes
log4j2.logger.otmmconnectorrest.name =
    com.opentext.hybris.otmmconnector.rest
log4j2.logger.otmmconnectorrest.additivity = false
log4j2.logger.otmmconnectorrest.level = error
log4j2.logger.otmmconnectorrest.appenderRefs = otmm
log4j2.logger.otmmconnectorrest.appenderRef.otmm.ref = OTMM_LOG

log4j2.logger.otmmconnectorservice.name =
    com.opentext.hybris.otmmconnector.service
log4j2.logger.otmmconnectorservice.additivity = false
log4j2.logger.otmmconnectorservice.level = warn
log4j2.logger.otmmconnectorservice.appenderRefs = otmm
log4j2.logger.otmmconnectorservice.appenderRef.otmm.ref = OTMM_LOG

log4j2.logger.otmmconnectorui.name =
    com.opentext.hybris.otmmconnector.ui
log4j2.logger.otmmconnectorui.additivity = false
log4j2.logger.otmmconnectorui.level = info
log4j2.logger.otmmconnectorui.appenderRefs = otmm
log4j2.logger.otmmconnectorui.appenderRef.otmm.ref = OTMM_LOG

log4j2.logger.otmmconnectorsync.name =
    com.opentext.hybris.otmmconnector.sync
log4j2.logger.otmmconnectorsync.additivity = false
log4j2.logger.otmmconnectorsync.level = debug
log4j2.logger.otmmconnectorsync.appenderRefs = otmm
log4j2.logger.otmmconnectorsync.appenderRef.otmm.ref = OTMM_LOG
```



Log API Additionally there is a logger for the Public API. The log level can be set with the `log4j2.logger.otmmconnectorapi.level` property:
`log4j2.logger.otmmconnectorapi.level=warn`

This logger also logs to the default log4j2 appender `otmm`.



Example 5-2: Debug Public API

To set the log level to debug, use following configuration:
`log4j2.logger.otmmconnectorapi.level=debug`



Performance analysis

You can define dedicated logging for performance analysis into a separate file. For Hybris 5.x, this is done in the `otmmaddon\project.properties` file. The configuration is deactivated in the condition as delivered. To activate, remove the comment character “#” from the `log4j.appender.otmmperf` settings.



Example 5-3: Log settings for performance analysis

```
#log4j2.appender.otmmperf.type = RollingFile
#log4j2.appender.otmmperf.name = OTMM_PERF_LOG
```

```
#log4j2.appender.otmmperf.fileName = ${OTMM_LOG_DIR}/otmm_perf.csv
#log4j2.appender.otmmperf.filePattern = \
#${OTMM_LOG_DIR}/otmm_perf.%d{yyyy-MM-dd}.csv
#log4j2.appender.otmmperf.layout.type = PatternLayout
#log4j2.appender.otmmperf.layout.pattern = \
#%d{yyyy-MM-dd HH:mm:ss,SSS};%t;%c;%m%n
#log4j2.appender.otmmperf.policies.type = Policies
#log4j2.appender.otmmperf.policies.time.type = TimeBasedTriggeringPolicy
#log4j2.appender.otmmperf.policies.time.interval = 1
#log4j2.appender.otmmperf.policies.time.modulate = true
#log4j2.appender.otmmperf.strategy.type = DefaultRolloverStrategy
#log4j2.appender.otmmperf.strategy.max = 10
#
#log4j2.logger.otmmperformance.name = \
#com.opentext.hybris.otmmconnector.Performance
#log4j2.logger.otmmperformance.additivity = false
#log4j2.logger.otmmperformance.level=trace
#log4j2.logger.otmmperformance.appenderRefs = otmmperf
#log4j2.logger.otmmperformance.appenderRef.otmmperf.ref = OTMM_PERF_LOG
```



5.8.2 Configure SAP LogViewer compliant logging

The log format can be adjusted to an SAP LogViewer compliant message format.

For this, you can configure an appender in the `local.properties` configuration file.



Important

Prerequisite is the installed **SAP LogViewer patch** of DAMLink Hybris.

Use the product area in My Support to find the download of the patch:
Digital Asset Management for SAP Solutions (<https://knowledge.opentext.com/knowledge/llisapi.dll/open/19514765>). To install it, follow the patch documentation.

The Log4j log levels are mapped to the SAP log levels as follows:

Log4j log level	SAP log level
fatal	Fatal
error	Error
warn	Warning
info	Info
debug	Debug
trace	Debug

To configure the appender for SAP compliant logging:

1. Add the following properties to the `local.properties` configuration file:
Define a new `otmmsap` appender:

```
# SAP logger
# The log files are written into a format
```

```
# read by the LogViewer application of NetWeaver.
#log4j2.appender.otmmsap.type = RollingFile
#log4j2.appender.otmmsap.name = OTMM_SAP_LOG
#log4j2.appender.otmmsap.fileName = ${OTMM_LOG_DIR}/otmmaddon_sap.log
#log4j2.appender.otmmsap.filePattern = ${OTMM_LOG_DIR}/otmmaddon_sap-%i.log
#log4j2.appender.otmmsap.layout.type = SapOTPatternLayout
#log4j2.appender.otmmsap.policies.type = Policies
#log4j2.appender.otmmsap.policies.size.type = SizeBasedTriggeringPolicy
#log4j2.appender.otmmsap.policies.size.size = \
#${log4j2.appender.otmm.policies.size.size}
#log4j2.appender.otmmsap.strategy.type = DefaultRolloverStrategy
#log4j2.appender.otmmsap.strategy.max = ${log4j2.appender.otmm.strategy.max}
```

2. Add the new appender to the list of used appenders.

Example:

```
#log4j2.logger.otmmconnectorapi.appenderRefs = otmm, otmmsap
#log4j2.logger.otmmconnectorapi.appenderRef.otmm.ref = OTMM_LOG
#log4j2.logger.otmmconnectorapi.appenderRef.otmmsap.ref = OTMM_SAP_LOG
#
#log4j2.logger.otmmconnector.appenderRefs = otmm, otmmsap
#log4j2.logger.otmmconnector.appenderRef.otmm.ref = OTMM_LOG
#log4j2.logger.otmmconnector.appenderRef.otmmsap.ref = OTMM_SAP_LOG
```

3. To activate it, restart the Hybris server.
The `otmm-sap.log` log file will be created in the `log/otmm_connector` directory.

5.8.3 Logging on OTMM

For a detailed description of DAMLink Hybris logging functions and configuration instructions on OTMM side, see Section 10.7 “Logging on OTMM” in *OpenText DAMLink for SAP Solutions - Installation and Configuration Guide (DAMSAP-IGD)*.

5.9 Data consistency

Data synchronization is a key feature of DAMLink for SAP Solutions. DAMLink Hybris provides data consistency through notifications sent by OTMM to Hybris through Web Services.

Depending on the customer’s scenario, metadata can be aligned implementing a notification hook using the DAMLink Hybris Public API. For more information about the API, see Section 1 “Overview” in *OpenText DAMLink for SAP Solutions Hybris Integration - Customization and Programming Guide (DAMSAP-PHI)*.

5.10 Software maintenance

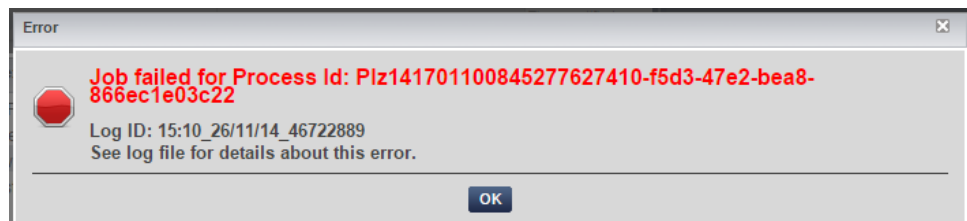
Patches and documentation updates for DAMLink Hybris are available on OpenText My Support (<https://knowledge.opentext.com/>) (<https://knowledge.opentext.com/>), under the product Digital Asset Management for SAP Solutions. OpenText customers can obtain new releases of the software directly from My Support. SAP customers can download new releases of the software from SAP Service Marketplace, once released by SAP.

Reporting problems to Support - SAP customers should report any software issues by going to SAP Service Marketplace and selecting component XX-PART-OPT-DAM. Any messages related to this component are routed to SAP Digital Asset Management Support. OpenText customers should contact OpenText Support directly, see <http://www.opentext.com/support>.

5.11 Troubleshooting

Troubleshooting hints:

- Look into the logfiles for errors and warnings.
- The UI provides a LOG ID in the error message dialogs. Search for this LOG ID in the logfiles.
- Increase the log level to INFO to get more information. If even further information is required, increase to DEBUG.
- Additionally, have a look into the OTMM logfiles, especially for information about the import and export process. The error message dialog or the OTMM Connector log files often contain the OTMM process ID. These IDs can be used for searches in the OTMM logfiles.



5.11.1 Login to OTMM failed

Symptom A “Login to OTMM failed” UI message is displayed.

Solution Look in the log file for “OTMM Connector configuration errors”.
Correct the mentioned invalid configurations. Verify that the login credentials and SSO are configured correctly.

5.12 Keyboard accessibility

You can control the Asset Assignment Editor UI with the following keyboard shortcuts:

- **ENTER**: Trigger search, if you are inside the search text box.
- **TAB**: Switch from one element to another.
- **SHIFT + TAB**: Switch from one element to another in reverse order
- **SPACE-BAR**: Trigger the selected button
- **S**: If **Multiselect** is enabled, you can select or deselect assets inside an asset list.



Note: Keyboard shortcuts are not working with Internet Explorer in standard product cockpit. This is because Product Cockpit is forcing IE to run in a IE7 compatibility mode. In order to activate keyboard shortcuts for IE, you can remove the according line from `index.zu1` configuration file of Product Cockpit.

Chapter 6

Example: project.properties

Example The following listing shows a `project.properties` configuration example:

```
#####  
## Adapt the following settings to your environment ##  
#####  
  
#####  
# OTMM Server related properties #  
#####  
otmm.server.scheme=http  
# otmm.server.domain is an obsolete alias for otmm.server.host  
# set otmm.server.host=<YourOTMMHost>  
otmm.server.host=  
otmm.server.port=11090  
  
# Set the OTMM login details in Hybris Backoffice.  
# If you do not require password encryption, you can also use the configuration  
# for login and password below.  
  
# Login of OTMM technical user  
#otmm.server.login=  
# OTMM User Password  
#otmm.server.password=  
  
# Set this property to true to use SSO.  
# For details refer to the DAMLink for SAP Solutions hybris Integration:  
# Installation and Configuration Guide  
#otmm.server.sso = FALSE  
  
# The port configured on Tomcat to receive OTMM notifications.  
# If this property is set to 0 (default value), notifications are accepted on  
# all Tomcat HTTP and HTTPS listener ports.  
# For a productive environment, this port should be set  
# to the SSL port of a "Trusted SSL" connector configuration  
# having "authClient=true". See the Tomcat SSL documentation.  
#otmm.server.notification.port=0  
  
# Timeout for waiting for the import request to finish (in seconds)  
#otmm.import.request.timeout=600  
  
# Polling interval to check the import status (in seconds)  
#otmm.polling.import.request=5  
  
# Timeout for waiting of the export request to finish (in seconds)  
#otmm.export.request.timeout=120  
  
# Polling interval to check the export status (in seconds)  
#otmm.polling.export.request=2  
  
# HTTP client communication related timeouts (in seconds)  
#otmm.server.http.connection.timeout=30  
#otmm.server.http.request.timeout=60  
  
#####  
# FTP configuration #  
#####
```

```
# FTP credentials needed for Content by Export content delivery.
#
# While invoking the OTMM export service, Hybris reads this server value and
# appends it to its export service request.

# FTP Server
10012=<YourFTPHost>

# Set the FTP login details in Hybris Backoffice.
# If you do not require password encryption, you can use the configuration
# for the FTP login and password below.

# FTP User for authentication
#10013=
# Password for FTP User
#10014=

# Destination folder of FTP export used by Content By Export Asset Deliver.
# The value below is put in front of the date format and results
# in the folder name: "<Property value>_yyyymmdd_HH:mm:ss"
# OTMM will store the result of the export in a FTP sub folder named
# after this calculated name.
#
# Later, the Hybris sync job uses the same path to read the export content.
10015=ExportData

# FTP port
10016=21

# Use FTP or FTPS (default is FTP)
#otmm.AssetSync.remote.protocol=FTP
# Use FTPS protocol (SSL/TLS)
otmm.AssetSync.ftps.protocol=SSL
# FTPS port (default 21)
#otmm.AssetSync.ftps.port=21

# Per default the character set used by the FTP server is detected automatically.
# Example UTF-8
# If this doesn't work you can set the encoding explicitly.
#otmm.AssetSync.ftp.encoding=

#####
# Synchronization directory #
#####
# Location where exported files on your local machine are located
# Only evaluated if otmm.AssetSync.SourceRepository is set to "Local"
# Example
#otmm.locationOfFiles=D:\\Images
otmm.locationOfFiles=<LocalDirectoryPath>

#####
# Hybris System ID #
#####
# To support more than one hybris system per OTMM server a system ID is now
# required. The system ID must be unique within the set of
# hybris systems sharing one OTMM server.
otmm.hybris.systemId=<SystemId>

# Set this to true for upgraded installations with version <16.2.
otmm.hybris.isDefault=false

#####
##
```

```

## Optional settings - adapt if the default that not fits to your needs    ##
##                                                                    ##
#####

# Used as proxy if the attributes urlTemplate of UrlDelivery contains {proxy}
# for example otmm.url.template.proxy=http://myhost:8080/proxy
otmm.url.template.proxy=

# For Media Delivery Service the OtmmUrlStrategy value is required to display
# the renditions correctly in Product Cockpit.
media.folder.otmmRoot.url.strategy=otmmUrlStrategy

# Especially with MediaServlet no export process from OTMM is required.
# Therefore otmm.export.isMandatory should be FALSE.
# See Export configuration section below.

# When using the Media Delivery Service, you must set
# the base URL to the Media Delivery Service.
# The Media Delivery Service can either run within the Hybris Tomcat or within
# a separate Tomcat deployment.
# The URL may also be the URL of the Adaptive Media Delivery servlet
# of the OTMM server.
# In this case you must set useAdaptiveMediaDelivery=TRUE.
# Example for Adaptive Media Delivery URL:
# http://<amd_host>:8080/adaptivemedia/rendition
#otmm.media.servlet.url=\
http://<mediaservlet_host>:<tomcat_http_port>/ImConvServlet/imconv

# The above URL is used inside the web pages produced by Hybris and displayed
# in a web browser from a machine that is normally not in the same subnet as the
# Hybris system. Thus it is called the "external" URL. You may also specify an
# "internal" URL, that Hybris uses itself to talk to the MediaServlet.
# If not configured, the internal URL is the same as the external URL.
#otmm.media.servlet.internal.url=\
http://localhost:${tomcat.http.port}/ImConvServlet/imconv

# The Media Delivery Service has a password defined in imconv.properties.
# Make this password known to Hybris either here,
# or better in Backoffice,
# DAM by OpenText > Logins, click + OTMM Connector Login,
# create an entry with the Identifier MediaServlet and User admin
otmm.media.servlet.pwd=

# If the Media Delivery Service is enabled, Hybris sends requests to the
# Media Delivery Service during asset assignment to generate
# the renditions early. You can disable this by setting this value to FALSE.
# When set to FALSE, the renditions are created when needed.
# If useCloudUrl is set to TRUE, you must set this parameter to TRUE as well.
# This implies that with useCloudUrl=TRUE there is only early generation
# of renditions.
otmm.media.servlet.prepopulate=TRUE

# With otmm.media.servlet.useCloudUrl=TRUE
# Media Delivery Service writes renditions into the cloud.
# Media Delivery Service is bypassed when renditions are requested.
# When images are stored in the cloud, you can generate URLs pointing directly
# into the cloud.
# With otmm.media.servlet.useCloudUrl=TRUE, the Media Delivery Service
# does not get image requests any more.
# Therefore, you must create the renditions during asset assignment,
# with prepopulate=TRUE.
otmm.media.servlet.useCloudUrl=FALSE

```

```
# Hybris generates the cloud URLs according to the following template.
# If it is not specified,
# Hybris obtains the template directly from the Media Delivery Service,
# whose URL and password
# are specified above, and which must be configured to use the cloud.
# The Azure template has the form
# http://XXX.core.windows.net/<container>/%P%/R%
# The AWS template has the form
# http://XXX.amazonaws.com/<bucket>/%P%/R%
# %P% is replaced by a path, %R% by a file name
# Once the servlet has written renditions into the cloud,
# you can see the URLs in the Azure or AWS console.
otmm.media.servlet.cloudUrlTemplate=

# Set to TRUE if the URL provided by otmm.media.servlet.url points to
# the Adaptive Media Delivery servlet of the OTMM server.
# Set to FALSE if you use the Media Delivery Service (ImConvServlet)
# of DAMLink Hybris
otmm.media.servlet.useAdaptiveMediaDelivery = FALSE

# If using the Adaptive Media Delivery servlet, you can set the client ID (clid)
# to be used in AMD requests. The default is SAPDAM.
otmm.media.servlet.amd.clid=SAPDAM

# If you configured the Media Delivery Service to require URLs with secKey,
# then you may configure the time in hours after which the URL expires.
# Values must be between 1 and 100. Default is 20.
# To configure no expiration date, leave the value empty.
otmm.media.servlet.expiration=20

# Determines which catalog is used if all other ways
# to determine a catalog fail.
# This catalog is used for the created Media item and MediaContainer
# This is the order how the catalog is determined
# 1. The catalogVersion of a Product or CMSItem
# 2. The catalogVersion attribute of a custom type
# 3. The catalog specified by otmmCatId and otmmCatVersion
# of the OtmmAssignmentStatus entry
# 4. The catalog specified by the properties below
otmm.assignment.defaultCatalog=Default
otmm.assignment.defaultCatalogVersion=Staged

#####
# MediaFolder configuration                                     #
#####
#
# With otmm.media.folder.<ModelType> you can configure the MediaFolder used
# during assignment per OTMM model type
# With "otmm.media.folder.default", you can configure a default folder
# that is used if no model type specific folder is specified.
# Leaving the value empty means that no MediaFolder is set
# for the created Media item. That means Hybris itself decides
# where to store the content.
# The value must be the MediaFolder qualifier. If there is no folder
# with that qualifier a folder with that qualifier is created
# and the same name is used for the path that is a relative path
# to the Hybris data directory.
otmm.media.folder.default=
otmm.media.folder.OPENTEXT.PRODUCTS=
```

```
#####
# Thumbnail configuration
#####
#
# This property configures a pattern whitelist of mime types.
# If the mime type of a Media item matches a pattern, their OTMM thumbnail is
# displayed.
# If the property is not configured or has an empty value no thumbnail is used.
#
# Syntax:
# - "*" is wildcard for any character
# - several Mime Type (patterns) can be separated by comma
#
# Examples:
# */* - thumbnails are used for all mime types
# video/* - thumbnails are only used for videos
# application/pdf - thumbnails are only used for PDF files
otmm.thumbnail.mimes=\
video/*,application/pdf,\
application/msword,application/vnd.ms-word,\
application/vnd.ms-excel,application/excel,\
application/vnd.ms-powerpoint,\
application/powerpoint

# This mime type is returned by the default implementation of
# OtmmThumbnailService.retrieveThumbnailMimeType.
otmm.thumbnail.fixed.mime=image/jpeg

# Set this to true if you enabled authentication
# for thumbnail URLs in the file
# /otmmws/web/webroot/WEB-INF/config/security-spring.xml
# See the comment there for further details
# If set to true the images are not cached
# by the browser after a session ends.
# that implies that for each user session the URLs are different.
otmm.thumbnail.protected = false

#####
# OTMM Model configuration and exported Mime types
#####
# Specify multiple model references separated by comma.
#
# This required property restricts the returned assets
# in the assignment dialog to the specified OTMM Model types:
# Also the format needs to be defined in which the transformation
# is needed as per the business requirement in the key
# "otmm.model.image.format.xxxxxxxxxxx".
# otmm.model.image.format.list = OPENTEXT.PRODUCTS,CUSTOM_PHOTO,CUSTOM_LOGO
# otmm.model.image.format.OPENTEXT.PRODUCTS=format\\#jpg
# otmm.model.image.format.CUSTOM_PHOTO=format\\#jpg
# otmm.model.image.format.CUSTOM_LOGO=format\\#jpg

# Comma separated list of OTMM Models which are configured according to
# the OpenText DAMLink for SAP Solutions Hybris Integration:
# Installation and Configuration Guide.
otmm.model.image.format.list =OPENTEXT.PRODUCTS

# Possible values are "format\\#<ImageFileExtension>"
# where ImageFileExtension is the extension of an image format
# which the ImageMagick version that is installed with OTMM supports.
otmm.model.image.format.default=format\\#jpg
otmm.model.image.format.OPENTEXT.PRODUCTS=format\\#jpg
```

```
# For each entry of otmm.model.image.format.<model> provide
# an entry otmm.model.image.mimetype.<model>
# The mime type must be consistent with the format above.
otmm.model.image.mimetype.default=image/jpeg
otmm.model.image.mimetype.OPENTEXT.PRODUCTS=image/jpeg

#####
# Export configuration
#####

# Determines if an export is always triggered no matter
# if an export is needed by the configured renditions.
# That means that if the property is set to FALSE and all renditions are
# of type MediaServlet no export job is started.
otmm.export.isMandatory = FALSE

#####
# Assignment configuration
#####
# Determines if the files exported by Content By export are read
# from a local directory or via FTP.
#otmm.AssetSync.SourceRepository=FTP
otmm.AssetSync.SourceRepository=Local

# Updates an existing OpenTextMedia instance or leaves it as it is.
# The OTMM Connector reuses existing OpenTextMedia items
# if they match the same asset, asset delivery name, and catalog version.
# If otmm.AssetSync.updateExisting is set to TRUE
# the content of the OpenTextMedia item is updated with the new image.
# Otherwise it is only created but never updated.
# Be aware that all products using this OpenTextMedia item
# will also update their images.
# However, the update will not happen automatically
# after changing the rendering format
# but it is done if someone assigns the same asset
# with the same asset delivery and within the same catalog version.
otmm.AssetSync.updateExisting = FALSE

#####
# List of models for which assets must be approved before assignment
#####
# If you use Lifecycle Management and
# if you want that only approved assets are available for assignment,
# then add a comma separated list of the models
# for which assets require approval.
# Assets belonging to these models are only available for assignment
# if their field
# "ARTESIA.FIELD.LIFECYCLE APPROVAL STATE" has the value APPROVED.
otmm.approved.models.list=

#####
# CronJob configuration for DAMLink Hybris extensions
#####

# Properties for OTMM Association Purge Job
# All OtmmAssignmentStatus (Asset Assignment Status)
# which creation time is older than the provided time and
# are synchronized or still in processing state are deleted.
# To delete the entries created two days ago configure
# timeToSubtract=2 and durationOfTime=days;
# Possible values for durationOfTime are "days", "hours", and "minutes".
# If nothing is provided then the default value is taken which is 1 day.
otmm.purgeJob.OTMMAssetProduct.timeToSubtract=1
otmm.purgeJob.OTMMAssetProduct.durationOfTime=hours
```

```

# Sets how many entries of OtmAssignmentStatus (Asset Assignment Status)
# are processed per run of the otmmExportServiceJob CronJob (default = 100)
otmm.exportServiceJob.batchSize=100

# Sets after how many successive failed entries of OtmAssignmentStatus
# the otmmExportServiceJob CronJob is aborted (default = 5).
otmm.exportServiceJob.maxFailedSize=5

# Sets how many entries of OtmAssignmentStatus are processed per run
# of the otmmAssetSyncJob CronJob (default = 100)
otmm.assetSyncJob.batchSize=100

# Sets after how many successive failed entries of OtmAssignmentStatus
# the otmmAssetSyncJob CronJob is aborted (default = 5).
otmm.assetSyncJob.maxFailedSize=5

#####
# Localization
#####

# Language used for the Media Container name attribute. Default is the Hybris
# default language.
#otmm.language.mediacontainer=

#####
# Logging
#####

## Project specific Logging
OTMM_LOG_DIR=${platformhome}/../../log/otmm_connector

## Issue for Hybris 6.0: The RollingFile appender
## wrongly creates empty files as "rolled files"!
## The issue is related to the log4j2 version used by Hybris 6.0.
# otmm appender configuration:
log4j2.appender.otmm.type = RollingFile
log4j2.appender.otmm.name = OTMM_LOG
log4j2.appender.otmm.fileName = ${OTMM_LOG_DIR}/otmmaddon.log
log4j2.appender.otmm.filePattern = ${OTMM_LOG_DIR}/otmmaddon-%i.log
log4j2.appender.otmm.layout.type = PatternLayout
log4j2.appender.otmm.layout.pattern = \
%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%t] [%c] - %m%n
log4j2.appender.otmm.policies.type = Policies
log4j2.appender.otmm.policies.size.type = SizeBasedTriggeringPolicy
log4j2.appender.otmm.policies.size.size = 10MB
log4j2.appender.otmm.strategy.type = DefaultRolloverStrategy
log4j2.appender.otmm.strategy.max = 10

# Configuration of loggers which use the otmm appender:
# otmmconnector logger
log4j2.logger.otmmconnector.name = com.opentext.hybris.otmmconnector
log4j2.logger.otmmconnector.additivity = false
log4j2.logger.otmmconnector.level=warn
log4j2.logger.otmmconnector.appenderRefs = otmm
log4j2.logger.otmmconnector.appenderRef.otmm.ref = OTMM_LOG
# otmmconnectorapi logger used by the public API
log4j2.logger.otmmconnectorapi.name=OTMMConnectorPublicAPI
log4j2.logger.otmmconnectorapi.additivity = false
log4j2.logger.otmmconnectorapi.level=warn
log4j2.logger.otmmconnectorapi.appenderRefs = otmm
log4j2.logger.otmmconnectorapi.appenderRef.otmm.ref = OTMM_LOG

#log4j2.logger.otmm.service.name = \

```

```
#com.opentext.hybris.otmmconnector.service
#log4j2.logger.otmm.service.additivity = false
#log4j2.logger.otmm.service.level=debug
#log4j2.logger.otmm.service.appenderRefs = otmm
#log4j2.logger.otmm.service.appenderRef.otmm.ref = OTMM_LOG

# SAP logger
# The log files are written into a format
# read by the LogViewer application of NetWeaver.
#log4j2.appender.otmmsap.type = RollingFile
#log4j2.appender.otmmsap.name = OTMM_SAP_LOG
#log4j2.appender.otmmsap.fileName = ${OTMM_LOG_DIR}/otmmaddon_sap.log
#log4j2.appender.otmmsap.filePattern = ${OTMM_LOG_DIR}/otmmaddon_sap-%i.log
#log4j2.appender.otmmsap.layout.type = SapOTPatternLayout
#log4j2.appender.otmmsap.policies.type = Policies
#log4j2.appender.otmmsap.policies.size.type = SizeBasedTriggeringPolicy
#log4j2.appender.otmmsap.policies.size.size = \
#${log4j2.appender.otmm.policies.size.size}
#log4j2.appender.otmmsap.strategy.type = DefaultRolloverStrategy
#log4j2.appender.otmmsap.strategy.max = ${log4j2.appender.otmm.strategy.max}

## Sample configuration which adds the "otmmsap" appender
# at the end of the appenders list:
#log4j2.logger.otmmconnectorapi.appenderRefs = otmm, otmmsap
#log4j2.logger.otmmconnectorapi.appenderRef.otmm.ref = OTMM_LOG
#log4j2.logger.otmmconnectorapi.appenderRef.otmmsap.ref = OTMM_SAP_LOG
#
#log4j2.logger.otmmconnector.appenderRefs = otmm, otmmsap
#log4j2.logger.otmmconnector.appenderRef.otmm.ref = OTMM_LOG
#log4j2.logger.otmmconnector.appenderRef.otmmsap.ref = OTMM_SAP_LOG

## Performance measurement logging
## The data is logged in CSV format.
#
#log4j2.appender.otmmperf.type = RollingFile
#log4j2.appender.otmmperf.name = OTMM_PERF_LOG
#log4j2.appender.otmmperf.fileName = ${OTMM_LOG_DIR}/otmm_perf.csv
#log4j2.appender.otmmperf.filePattern = \
#${OTMM_LOG_DIR}/otmm_perf.%d{yyyy-MM-dd}.csv
#log4j2.appender.otmmperf.layout.type = PatternLayout
#log4j2.appender.otmmperf.layout.pattern = \
#%d{yyyy-MMM-dd HH:mm:ss,SSS};%t;%c;%m%n
#log4j2.appender.otmmperf.policies.type = Policies
#log4j2.appender.otmmperf.policies.time.type = TimeBasedTriggeringPolicy
#log4j2.appender.otmmperf.policies.time.interval = 1
#log4j2.appender.otmmperf.policies.time.modulate = true
#log4j2.appender.otmmperf.strategy.type = DefaultRolloverStrategy
#log4j2.appender.otmmperf.strategy.max = 10
#
#log4j2.logger.otmmperformance.name = \
#com.opentext.hybris.otmmconnector.Performance
#log4j2.logger.otmmperformance.additivity = false
#log4j2.logger.otmmperformance.level=trace
#log4j2.logger.otmmperformance.appenderRefs = otmmperf
#log4j2.logger.otmmperformance.appenderRef.otmmperf.ref = OTMM_PERF_LOG

# Former hybris.log configuration converted to new log4j2 syntax
#log4j2.appender.hybris.type = RollingFile
#log4j2.appender.hybris.name = HYBRIS_LOG
#log4j2.appender.hybris.fileName = ${HYBRIS_LOG_DIR}/hybris.log
#log4j2.appender.hybris.filePattern = ${HYBRIS_LOG_DIR}/hybris-%i.log
#log4j2.appender.hybris.layout.type = PatternLayout
#log4j2.appender.hybris.layout.pattern = \
#%d{yyyy-MMM-dd HH:mm:ss,SSS} %-5p [%t] [%c] - %m%n
```

```

#log4j2.appender.hybris.policies.type = Policies
#log4j2.appender.hybris.policies.size.type = SizeBasedTriggeringPolicy
#log4j2.appender.hybris.policies.size.size = 10MB
#log4j2.appender.hybris.strategy.type = DefaultRolloverStrategy
#log4j2.appender.hybris.strategy.max = 10
#
#log4j2.rootLogger.level = info
#log4j2.rootLogger.additivity = false
#log4j2.rootLogger.appenderRefs = stdout, hybris
#log4j2.rootLogger.appenderRef.stdout.ref = STDOUT
#log4j2.rootLogger.appenderRef.hybris.ref = HYBRIS_LOG

#####
# Configuration how a Product Cockpit user can reach OTMM #
#####

## The values HTTP or HTTPS are possible.
## Default is the value of the otmm.server.scheme property
#otmm.server.for.productcockpit.users.protocol=http

## Fully-qualified OTMM host name.
## Default is the value of the otmm.server.host property
#otmm.server.for.productcockpit.users.host=localhost

## OTMM port.
## Default is the value of the otmm.server.port property
#otmm.server.for.productcockpit.users.port=11090

## Base-URL how a Product Cockpit user can access OTMM
otmm.server.for.productcockpit.users.baseurl=/otmm/ux-html/index.html

## Base-URL for SSO login into OTMM from Product Cockpit.
otmm.server.for.productcockpit.users.baseurl=/otmmapi/v3/sessions

## If set to true automatic login is performed even if SSO is disabled
## (otmm.server.sso=false).
## In case of true automatic login is done with the technical user specified
## in otmm.server.login.
## Default is false.
otmm.server.for.productcockpit.users.isPreAuthenticationForTechnicalUserEnabled=false

#####
# Asset upload configuration #
#####

## The maximal upload size of the component.
## A value of -1 means that there is no limitation.
## Specify in KB.
otmm.upload.max.upload.size.in.kb=-1

## For details please find http://www.w3schools.com/tags/att\_input\_accept.asp
## Examples:
## The value 'image/jpeg, image/png' accepts JPEG and PNG files.
## The value 'image/*' accepts images.
## The value 'video/mpeg' accepts MPEG videos.
#otmm.upload.html.accept.attribute=

## Only affects UploadButtonWidget
## The Asset Assignment Dialog and DropFileWidget aren't affected.
otmm.is.multi.upload.enabled=true

## Maximum number of uploads running in parallel per upload process

```

```
otmm.max.number.of.parallel.uploads.per.upload.process=5

## Time in milliseconds to wait until the next check after an upload
## has been completed
otmm.time.intervall.to.check.running.uploads=3000

#####
# Asset assignment configuration                                     #
#####

## Maximum number of asset assignments running in parallel
otmm.max.number.of.parallel.assetassignments=5

## Time in milliseconds to wait until the next check after an
# asset assignment has been completed
otmm.time.intervall.to.check.running.assetassignment=3000

#####
# Cockpit editor configuration                                     #
#####

## The Asset Assignment Dialog shows this OTMM folder and all subfolders.
## - In case of an empty value all OTMM folders are visible.
## - In case of 'Public Folders' only public folders and
#   subfolders are visible.
## Otherwise the value must be a valid ID of a OTMM Container,
## for example '64bb9c93c0941f7aa25fabb38e539a396b9fd887'
##
## This value is a default which can be overwritten
# in the editor configuration using the editor parameter 'otmmRootFolderId'.
otmm.directory.root.name=

## If someone wants to upload an asset the Asset Assignment Dialog
## offers a selection of OTMM property templates.
## - Property template IDs must be specified.
## - Multiple property template IDs must be separated with semicolons.
## - Non existing property template IDs are ignored.
##
## This value is a default which can be overwritten
## in the editor configuration using the editor parameter
## 'otmmUploadPropertyTemplateIds'.
otmm.upload.property.template.ids=

## For usage with MediaContainerLists, for example
## with the galleryImages attribute.
## This is the MediaFormat (qualifier) of the image to be displayed.
## Default is 300Wx300H.
## If no image with the MediaFormat as configured here or
## with default format exists, the first image in the MediaContainer is taken.
##
## This value is a default which can be overwritten
## in the editor configuration using the editor parameter
## 'mediaContainerDisplayFormat'.
otmm.media.container.display.format=300Wx300H

## Option to make the checkbox for update to latest version
## visible or invisible in the Asset assignment dialog.
##
## This value is a default which can be overwritten
## in the editor configuration using the editor parameter
## 'isAutomaticallyUpdateToLatestAssetVersionVisible'.
otmm.is.automatically.update.to.latest.asset.version.visible=true
```

```

## Default setting for automatic update to latest version:
## If the checkbox is enabled, then the default setting controls
## whether the checkbox is initially marked or not.
## If the checkbox is disabled, then the default setting controls
# whether the assignments are always created for update
## to latest version or not (without user control).
##
## This value is a default which can be overwritten
## in the editor configuration using the editor parameter
## 'automaticallyUpdateToLatestAssetVersionDefault'.
otmm.automatically.update.to.latest.asset.version.default=false

## Display the button to view an asset in OTMM .
##
## This value is a default which can be overwritten
## in the editor configuration using the editor parameter
## 'isAssetInspectorLaunchButtonVisible'.
otmm.is.asset.inspector.launch.button.visible=true

## This parameter configures the rendition that can be selected.
## - The single values must be the name of the wanted OtmmAssetDelivery.
## - Multiple Asset Delivery definitions have to be separated with semicolons.
## - Non existing Asset Delivery definitions or invalid values are ignored.
##
## for example
## otmm.selectable.assetDeliveries=<Name1>;<Name2>;<Name3>
##
## This value is a default which can be overwritten
## in the editor configuration using the editor parameter
## 'selectableAssetDeliveries'.
## If both settings have empty values,
## no selection is provided in the assignment dialog.
otmm.selectable.assetDeliveries=

## Backoffice only -
## Shall the Plus Button be shown inside the Media Reference Editor
otmm.is.media.reference.editor.show.upload.widget.button.visible=false

## Backoffice only -
## Shall the Plus Button be shown inside the Multi Media Reference Editor
otmm.is.multi.media.reference.editor.show.upload.widget.button.visible=false

# Custom column configuration
# This feature is only available in cockpit UIs
## (for example Product Cockpit, CMS Cockpit) but not in Backoffice
# You can configure additional columns in the Asset Assignment dialog
## for any scalar field of an asset.
# The additional columns are added after the Mime Type column.
# It is also possible to configure custom columns
# in the editor configuration using the editor parameter "customAssetColumns".
#
# Syntax:
# <FieldId>,<Width>,<Header>;<FieldId>,<Width>,<Header>;...
#
# Multiple columns are separated by ";"
# and the different configuration parts for a column by ",".
# <FieldId> - the ID of the scalar field
#           for example ARTESIA.FIELD.DATE IMPORTED
# <Width> - Either a fixed width in pixels
#           for example "100px" or "_" for automatic width.
# <Header> - the non-localized column label
#           for example "Import Date"
#
# Example:

```

```
# otmm.custom.asset.columns=ARTESIA.FIELD.DATE_IMPORTED,100px,\
# Import Date;ARTESIA.FIELD.CONTENT_TYPE,_,Content Type
#
otmm.custom.asset.columns=

# Configures if the last text search expression is run again
# when the assignment dialog is opened.
# If it is true the last search is executed again
# otherwise it starts with empty search.
# It is also possible to configure this in the editor configuration
# using the editor parameter "recoverLastSearchCondition".
otmm.recover.last.search.condition=false

# Only assets which have the specified OTMM Content Types
# are displayed in the Asset Assignment dialog.
# Separate multiple Content Types by comma.
# for example
# otmm.content.types.filter=VIDEO,BITMAP
otmm.content.type.filter=

#####
# Metadata synchronization                                     #
#####

# Enables the metadata synchronization between Hybris and OTMM if set to true.
# Refer to the DAMLink for SAP Solutions hybris Integration:
# Installation and Configuration Guide
# and the release notes before you enable this feature.
otmm.metadata.synchronization.enabled = false

# This property configures a pattern whitelist of catalogs
# for which items are tracked by Metadata Synchronization.
# If the property is not configured
# or has an empty value the default "*/Staged" is used.
# If the changed or deleted item is a Product CMSItem,
# or an Item that contains an attribute named "catalogVersion"
# of the CatalogVersion type then the Catalog Version is matched against
# the configured catalog list.
# If it matches the catalog the change is tracked.
#
# Syntax:
# - "*" is wildcard for any character
# - Catalog ID and Catalog Version part must be separated by "/"
#   e.g. "apparelProductCatalog/Staged"
# - several Catalogs (Patterns) can be separated by comma
#
# Examples:
# */* - all catalog versions are synchronized
# */Staged - all catalog versions with version named "Staged" are tracked.
# */Staged,*electronics*/* - all catalog versions with version named
#                               "Staged" and all catalogs with "electronics"
#                               in their name are tracked.
#
otmm.metadata.synchronization.catalogs=*/Staged

#####
# AutoAssetJob (Automatically assign assets to products)      #
#####

# Example where the product ID is stored in a tabular field
#otmm.auto.productid.field = AUTO.TABULAR.GROUP/AUTO.TABULAR.PRODUCT ID

# Example where the product ID is stored in a scalar field
#otmm.auto.productid.field = AUTO.SCALAR.PRODUCT ID
```

```

# Example where the product ID is stored in a scalar field
# in the enclosing folder
#otmm.auto.productid.field = AUTO.FOLDER.PRODUCT ID

# Set to true if productid.field is inherited from enclosing folder
#otmm.auto.productid.field.isinherited=false

# assigned.field normally == productid.field with last component
# replaced with "ASSIGNED",
# but it can also be specified explicitly,
# especially if productid.field is inherited.
#otmm.auto.assigned.field = AUTO.SCALAR.ASSIGNED

# The Hybris attribute is either determined from the asset name,
# or from an attributeid.field.
# Example where the attribute ID is stored in a tabular field
#otmm.auto.attributeid.field = AUTO.TABULAR.GROUP/AUTO.TABULAR.ATTRIBUTE ID

# Example where the attribute ID is stored in a scalar field
#otmm.auto.attributeid.field = AUTO.SCALAR.ATTRIBUTE ID

# When a systemid field is specified, then the AutoAssignJob
# will only search for assets
# where the systemid stored in the field matches
# the system ID of Hybris (see otmm.hybris.systemId above)
# Example where the system ID is stored in a tabular field
#otmm.auto.systemid.field = AUTO.TABULAR.GROUP/AUTO.TABULAR.SYSTEM ID

# Example where the system ID is stored in a scalar field
#otmm.auto.systemid.field = AUTO.SCALAR.SYSTEM ID

# Example where the system ID is stored in a scalar field in the enclosing folder
#otmm.auto.systemid.field = AUTO.FOLDER.SYSTEM ID

# Set to true if systemid.field is inherited from enclosing folder
#otmm.auto.systemid.field.isinherited=false

# With useAssetName=true,
# the product ID is derived from the stem of the asset name.
# With useAssetName=true, the attribute ID is derived
# from the suffix of the asset name, unless an attributeid.field is specified.
#otmm.auto.useAssetName=false

# If true, whatever was derived as OTMM product ID is matched
# against the Hybris product EAN instead of product ID
#otmm.auto.useEan=false

# Determines after how many successive failed assignments
# the CronJob otmmAutoAssignJob is aborted (default = 5).
otmm.autoAssignJob.maxFailedSize=5

# The following otmm.impex rules are taken, if useAssetName == true
# The stem of the asset name (for example hammer for hammer.jpg)
# is taken as the product ID.
# With impex.stem, file suffixes of asset names are mapped without versions
# (that is the "main" asset, like hammer.jpg) to an attribute ID.
# The numbers must be sequential, to fill any gaps,
# you may use impex.stem.i.suffix=skip
#otmm.impex.stem.0.suffix=.jpg
#otmm.impex.stem.0.attribute=picture
#otmm.impex.stem.1.suffix=.png
#otmm.impex.stem.1.attribute=picture
#otmm.impex.stem.2.suffix=.psd
#otmm.impex.stem.2.attribute=picture
#otmm.impex.stem.3.suffix=.pdf

```

```
#otmm.impex.stem.3.attribute=data_sheet
#otmm.impex.stem.4.suffix=.mp4
#otmm.impex.stem.4.attribute=detail

# With impex.versioned, file suffixes of asset names
# are mapped with versions (like hammer_2.jpg)
# to an attribute ID. Here, all versioned assets
# are stored with the attribute galleryImages
#otmm.impex.versioned.0.suffix=.jpg
#otmm.impex.versioned.0.attribute=galleryImages

# The default for the character separating the version from the stem is "_", for
# example hammer_2.jpg.
# With versionDelim you can specify a different character, for example "-" for
# hammer-2.jpg.
#otmm.impex.versionDelim=-

#####
# HTML5 Asset Assignment Dialog                                     #
#####

otmm.backoffice.is.zk.based.asset.assignment.dialog.enabled=true

#####
# Backoffice OAuth2 Settings - necessary for HTML5 Dialog         #
#####

otmm.backoffice.oauth2.client.id=otmmbackoffice

# Comma separated scopes
otmm.backoffice.oauth2.scopes=basic

# Comma separated resource IDs
otmm.backoffice.oauth2.resource.ids=otmmws

# Comma separated list of granted authorities
otmm.backoffice.oauth2.granted.authorities=

#####
##                                                                ##
##                                Advanced Settings                ##
##                                                                ##
#####

# Be careful with any changes in this section!
# Inappropriate configuration settings may result in data inconsistency!

# This option is for performance optimization in very specific scenarios.
# If set to true, the assetSync CronJob assumes that all the OTMM exports
# are finished and were successful.
# Please do not use this option, unless you are sure
# that all the OTMM exports are processed and were successful!
# Note: A successful ExportService CronJob run does not guarantee
#       that all the OTMM exports were successful,
#       as the CronJob only initiated the asynchronous OTMM export!
#otmm.internal.assetSyncJob.skipCheckJobResult=false

#
# Change the following parameter to "TRUE" for a development machine,
# if the embedded DB (HSQLDB) is used.
# This DB doesn't support "EXCLUSIVE LOCK",
```

```
# needed for running in a Hybris production environment.
# The default value for this parameter is "FALSE".
#
#otmm.db.ignore.exclusive.lock=FALSE

# Specifies the location of the spring context file
# put automatically to the global platform application context.
otmm.connector.extensions.context=otmm-extensions-spring.xml
otmmaddon.application-context=otmmaddon-spring.xml,\
${otmm.connector.extensions.context}
```


Chapter 7

Appendix

7.1 Content types and MIME types

The following table lists all content types (UOI_CONTENT_TYPES) that are known to OTMM by default.

Code	Name
ACROBAT	Acrobat PDF
AUDIO	Audio
BITMAP	Image
HTML	HTML
LAYOUT	Layout
MSOFFICE	MS Office
NONE	none
OTHER	Other
PROJECT	Project
XML	XML
SHORTCUT	Shortcut
TXT	Text
VIDEO	Video
XML	XML

This table shows MIME types with the associated content type.

MIME type	Content type code
NAME	CONTENT_TYPE
application/acad	BITMAP
application/applefile	OTHER
application/autocad_dwg	BITMAP
application/dca-rft	TXT
application/font	OTHER
application/futuresplash	OTHER
application/illustrator	LAYOUT

MIME type	Content type code
application/macbinary	OTHER
application/mac-binhex40	OTHER
application/msword	MSOFFICE
application/multimate	TXT
application/octet-stream	OTHER
application/octet-stream-fpx	OTHER
application/pdf	ACROBAT
application/photoshop	BITMAP
application/postscript	BITMAP
application/ProWrite	TXT
application/ps-font	OTHER
application/quarkxpress	LAYOUT
application/quattroPro	TXT
application/sit	OTHER
application/text-various	TXT
application/toolbook	OTHER
application/unknown	OTHER
application/visio	BITMAP
application/vnd.adobe-framemaker	OTHER
application/vnd.adobe-illustrator	LAYOUT
application/vnd.adobe-indesign	LAYOUT
application/vnd.adobe-photoshop	BITMAP
application/vnd.autocad-image	OTHER
application/vnd.borland-dBASE	TXT
application/vnd.corel-paradox	TXT
application/vnd.DEC-WPS	TXT
application/vnd.framemaker-mif	TXT
application/vnd.harvard-graphics	BITMAP
application/vnd.hp-HPGL	BITMAP
application/vnd.hp-PCL	BITMAP
application/vnd.hp-RTL	BITMAP
application/vnd.ibm-WritingAssist	TXT
application/vnd.ichitaro	TXT

MIME type	Content type code
application/vnd.interleaf	TXT
application/vnd.lotus1-2-3	TXT
application/vnd.lotus-freelance	TXT
application/vnd.Lotus-Manuscript	TXT
application/vnd.mm-Director	OTHER
application/vnd.ms-access	MSOFFICE
application/vnd.ms-excel	MSOFFICE
application/vnd.ms-excel.addin.macroEnabled.12	MSOFFICE
application/vnd.ms-excel.sheet.binary.macroEnabled.12	MSOFFICE
application/vnd.ms-excel.sheet.macroEnabled.12	MSOFFICE
application/vnd.ms-excel.template.macroEnabled.12	MSOFFICE
application/vnd.ms-powerpoint	MSOFFICE
application/vnd.ms-powerpoint.addin.macroEnabled.12	MSOFFICE
application/vnd.ms-powerpoint.presentation.macroEnabled.12	MSOFFICE
application/vnd.ms-powerpoint.slide.macroEnabled.12	MSOFFICE
application/vnd.ms-powerpoint.slideshow.macroEnabled.12	MSOFFICE
application/vnd.ms-powerpoint.template.macroEnabled.12	MSOFFICE
application/vnd.ms-word	MSOFFICE
application/vnd.ms-word.document.macroEnabled.12	MSOFFICE
application/vnd.ms-word.template.macroEnabled.12	MSOFFICE
application/vnd.ms-works	MSOFFICE
application/vnd.ms-write	MSOFFICE
application/vnd.NASA-PDS	BITMAP
application/vnd.openxmlformats-officedocument.presentationml.slide	MSOFFICE
application/vnd.openxmlformats-officedocument.presentationml.slideshow	MSOFFICE

MIME type	Content type code
application/vnd.openxmlformats-officedocument.presentationml.template	MSOFFICE
application/vnd.openxmlformats-officedocument.spreadsheetml.template	MSOFFICE
application/vnd.openxmlformats-officedocument.wordprocessingml.template	MSOFFICE
application/vnd.quark	LAYOUT
application/vnd.rn-realmedia	VIDEO
application/vnd.samna	TXT
application/vnd.visio	BITMAP
application/wordperfect	TXT
application/x-acad	BITMAP
application/x-amipro	TXT
application/x-autoCAD-dxf	OTHER
application/x-database	TXT
application/x-director	OTHER
application/x-DisplayWrite-txt	TXT
application/x-dvi	OTHER
application/x-envoy	OTHER
application/x-erdas-lan	BITMAP
application/x-FirstChoice	TXT
application/x-gtar	OTHER
application/x-incopy	TXT
application/x-indesign	LAYOUT
application/x-javascript	OTHER
application/x-latex	OTHER
application/x-msaccess	MSOFFICE
application/x-mswrite	MSOFFICE
application/x-Navy-DIF	TXT
application/x-pcpaint	BITMAP
application/x-pfsWrite	TXT
application/x-shockwave-flash	OTHER
application/x-SmartDraw	OTHER
application/x-spreadsheet	TXT

MIME type	Content type code
application/x-sprite	OTHER
application/x-tar	OTHER
application/x-tex	OTHER
application/x-totalWORD	TXT
application/x-visio	BITMAP
application/x-volkswriter	TXT
application/x-WordMARC	TXT
application/x-wordperfect-graphic	BITMAP
application/x-wordstar	TXT
application/x-xif	TXT
application/zip	OTHER
audio/aiff	AUDIO
audio/basic	AUDIO
audio/default	AUDIO
audio/midi	AUDIO
audio/mp4	AUDIO
audio/mpeg	AUDIO
audio/mpeg3	AUDIO
audio/quicktime	AUDIO
audio/vnd.realaudio	AUDIO
audio/x-au	AUDIO
audio/x-mpeg	AUDIO
audio/x-ms-wma	AUDIO
audio/x-pn-realaudio	VIDEO
audio/x-wav	AUDIO
flv-application/octet-stream	VIDEO
image/bmp	BITMAP
image/cgm	OTHER
image/citex	BITMAP
image/default	BITMAP
image/dng	BITMAP
image/eps	BITMAP
image/gif	BITMAP

MIME type	Content type code
image/ief	OTHER
image/jpeg	BITMAP
image/pdf	BITMAP
image/png	BITMAP
image/tiff	BITMAP
image/vnd.corel-draw	BITMAP
image/vnd.qrt-raw	BITMAP
image/x-adex	BITMAP
image/x-alias-pix	BITMAP
image/x-alpha-bmp	BITMAP
image/x-autologic-gm	BITMAP
image/x-avhrr	BITMAP
image/x-calcomp-ccrf	BITMAP
image/x-cals	BITMAP
image/x-canon-cr2	BITMAP
image/x-ccitt4	BITMAP
image/x-core-idc	BITMAP
image/x-eps-interchange	BITMAP
image/x-first-publisher-art	BITMAP
image/x-fractal	OTHER
image/x-gem-vdi	BITMAP
image/x-goes	BITMAP
image/x-hitachi-raster	BITMAP
image/x-hp-graphic-obj	BITMAP
image/x-ibm-picture-mkr	BITMAP
image/x-iff-ilbm	BITMAP
image/x-imaging-tech	BITMAP
image/x-IMG-software-set	BITMAP
image/x-imq-pds	BITMAP
image/x-iris-ct	BITMAP
image/x-jovian-vi	BITMAP
image/x-lumena-cel	BITMAP
image/x-macintosh-pict	BITMAP

MIME type	Content type code
image/x-msmetafile	BITMAP
image/x-mtv-ray	BITMAP
image/x-navb	OTHER
image/x-nikon-nef	BITMAP
image/x-os2-bmp	BITMAP
image/x-os2-icon	BITMAP
image/x-pc-paint	BITMAP
image/x-pcx	BITMAP
image/x-photo-cd	BITMAP
image/x-portable-anymap	BITMAP
image/x-portable-bitmap	BITMAP
image/x-portable-graymap	BITMAP
image/x-portable-pixmap	BITMAP
image/x-puzzle	BITMAP
image/x-qdv	BITMAP
image/x-raster-graphics	BITMAP
image/x-rgb	BITMAP
image/x-rix	BITMAP
image/x-rlc	BITMAP
image/x-scitex-ct	BITMAP
image/x-scodl	OTHER
image/x-sgi-image	BITMAP
image/x-sharp-gpb	BITMAP
image/x-sun-icon	BITMAP
image/x-sun-raster	BITMAP
image/x-targa	BITMAP
image/x-us-patent	BITMAP
image/x-utah-raster	BITMAP
image/x-verity-image	BITMAP
image/x-vitec	BITMAP
image/x-vivid	BITMAP
image/x-WMF	BITMAP
image/x-xbitmap	BITMAP

MIME type	Content type code
image/x-xbm	BITMAP
image/x-xpixmap	BITMAP
image/x-xwindowdump	BITMAP
model/iges	OTHER
text/css	TXT
text/default	TXT
text/html	HTML
text/miff	TXT
text/plain	TXT
text/richtext	TXT
text/rtf	TXT
text/sgml	OTHER
text/tab-separated-values	OTHER
text/x-db	TXT
text/xml	XML
text/x-setextetx	OTHER
video/3gpp	VIDEO
video/3gpp2	VIDEO
video/avi	VIDEO
video/default	VIDEO
video/dvd	VIDEO
video/gxf	VIDEO
video/mp4	VIDEO
video/mpeg	VIDEO
video/mpeg2	VIDEO
video/mpg	VIDEO
video/mxf	VIDEO
video/quicktime	VIDEO
video/smil	VIDEO
video/vnd.realvideo	VIDEO
video/x-avi	VIDEO
video/x-dv	VIDEO
video/x-flv	VIDEO

MIME type	Content type code
video/x-ms-asf	VIDEO
video/x-ms-asx	VIDEO
video/x-msvideo	VIDEO
video/x-ms-wmv	VIDEO
video/x-pn-realvideo	VIDEO
video/x-sgi-movie	VIDEO
x-world/x-3dmf	OTHER
x-world/x-vrml	OTHER

