ORACLE | Partner

XPLM®

Installation & Administration Guide

Dassault Systèmes SOLIDWORKS integration
for
Oracle Agile PLM

Valid for product version: 3.6.4

Published: 24.08.2023 | Build: 330 | Revision: 11079

XPLM Solution GmbH

Altmarkt-Galerie

Altmarkt 25

01067 Dresden

Office: +49 351 82658-0

Fax: +49 351 82658-88

Web: www.xplm.com

Legal information

Copyrights and trademarks

Disclaimer

XPLM Solution GmbH is not liable for errors in this documentation. This documentation may contain links to web sites of other companies or organizations that XPLM Solution GmbH does not own or control. XPLM Solution GmbH neither checks the accessibility of these websites nor makes any representations. The information in this document is based on a standard installation. The actual appearance of a specific installation may vary depending on the local environment and configuration realized for the individual customer requirements.

Support

For Oracle Agile Engineering Collaboration support, contact the Oracle Global Customer Support (GCS) at www.oracle.com/support, or My Oracle Support at https://support.oracle.com.

Contents

Glossary

Application Programming Interface (API)

Defines a set of routines, communication protocols and tools for building software. In general, they are clearly defined methods for communication between different components.

Bill of Materials (BOM)

Defines a list of assemblies, sub-assemblies, items and their quantities needed to produce a final product.

BOM position

Defines a position in the BOM with unique identification, name, quantity and other characteristics.

Component Object Model (COM)

Defines a binary-interface standard for software components introduced by Microsoft.

Connector

Defines an central interface component of each XPLM integration. The integration uses connectors for each participating application to exchange data via their API.

Datamodel

Defines objects and their relationships in a PLM system that are managed by the integration to store data from an authoring application.

Dynamic Link Library (DLL)

Defines a file with a library of functions and other information that can be accessed by a Windows program.

Java Runtime Environment (JRE)

Defines a runtime environment for Java technology. It is used to execute Java applications largely independently of the underlying operating system.

Payload

Defines the data contained within an API request. The description is borrowed from the transportation industry, where a truck carries its cargo (its payload) to a location. The truck, as with the API request, is always the same, but the payload changes with each request.

Product Lifecycle Management (PLM)

Defines systems and processes for managing data during the development of a product from creation through manufacturing to maintenance and disposal.

Revision

Defines a released object state in Agile PLM that cannot be modified.

Script engine

Defines the central component in each integration. It contains the integration logic for processing and forwarding the information and data coming from the connectors.

User Interface (UI)

Defines a (usually) graphical interface through which a user interacts with the computer.

Version

Defines an incremental counter of each object modification in Agile PLM on check-in.

x86/x64

Defines the processor architecture in a computer and thus also the performance of applications. x86 corresponds to 32-bit and x64 corresponds to 64-bit.

1 Introduction

1.1 About this document

You are reading the Installation and Administration Guide of the Dassault Systèmes SOLIDWORKS integration for Oracle Agile PLM by XPLM.

Purpose

This document describes how to install and set up the integration.

Target audience

This document is intended for system administrators in the company who install, adapt and deploy the integration. In addition, system administrators should also read the User Guide to understand the different scenarios this integration supports.

How to read this document

This document is structured chronologically and you should read it in the order of the chapters described. If you skip chapters, you will miss important information.

Where appropriate, cross-references to other chapters are listed. To quickly return to where you came from after clicking such a link, click the **Back** button in your PDF viewer. Try it right now with this link!

Notes used


This note highlights additional information about the current content.


This note highlights important instructions.


This note highlights from which integration version a feature was introduced or updated.

PDF recommendations

You can read this document in all supported PDF viewers, including PDF browser plugins. Check the following recommendations about copying code and searching for text strings.

Copying code

```
<XPlmConnector>
  <Properties>
    ...
  </Properties>
</XPlmConnector>
```

Not all PDF viewers support proper copying of code. If you open this document in Firefox, Acrobat Reader, or Acrobat and copy the above code, the lines will be pasted without indents and even in one line only, depending on the version of the PDF viewer.

Browsers that copy code correctly are Chrome, Edge and Brave. Keep your browsers and Acrobat tools always up to date.

Searching for text strings

Long text strings, for example variables, are sometimes manually wrapped in this document to fit in a table cell. When you search for such terms in this document, Acrobat Reader and Acrobat cannot interpret a line break, which results in the entire term not being found. Therefore, always open the PDF in one of the above browsers and search there.

1.2 Naming conventions

In addition to the terms already listed in the glossary, the following naming conventions are used in this document. Familiarize yourself with these names. You will find them throughout the document.

Form Used	Generally refers to
Agile PLM	Oracle Agile PLM
SOLIDWORKS	Dassault Systèmes SOLIDWORKS
Item	Item in Agile PLM
Attribute	Object property in Agile PLM
Toolbox Part	Standard Part

1.3 Overview of the integration

The integration is the interface between SOLIDWORKS and Agile PLM and is designed for seamless and efficient data exchange between both applications.

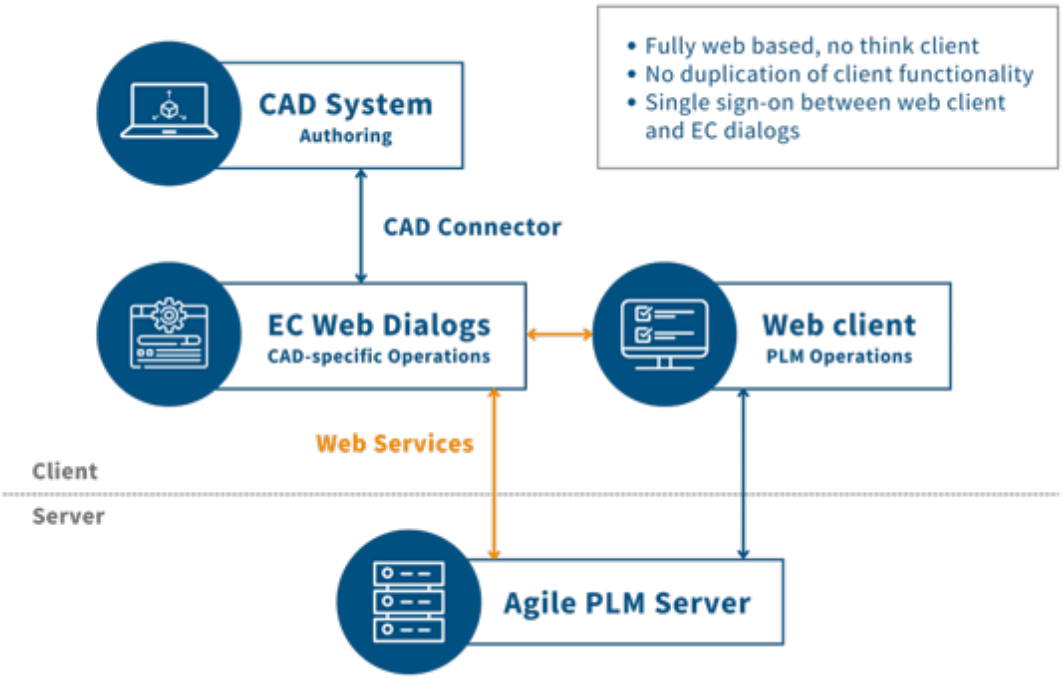
The integration is focused on a SOLIDWORKS-centric workflow that enables the SOLIDWORKS user to work with Agile PLM-based data managed directly from the SOLIDWORKS menu or ribbon bar.

The integration uses the API from both applications. It is designed to extend existing SOLIDWORKS and Agile PLM functionality. The handling of SOLIDWORKS structure and objects is done in SOLIDWORKS itself. Agile PLM manages SOLIDWORKS objects and structures that are important in the design and approves a possible workflow for individual design objects.

The main functions of the integration consist of saving, loading and item operations. Each of these operations has a certain effect on Agile PLM structures.

The integration also supports the Agile PLM system's Check Out mechanism to avoid data loss due to unresolved access conflicts. Concurrent engineering is supported through the Check Out of individual objects and structures in Agile PLM.

Functionalities of the integration are available via additional toolbars or extended menus in SOLIDWORKS.



1.4 Supported file types

The SOLIDWORKS connector supports the following file types:

File type	File extension
Part	.sldprt
Assembly	.sldasm
Drawing	.slddrw

2 System requirements

Before you install the integration, make sure the following system requirements for the integration are met.

Client-side requirements

Make sure the client computer running the integration has all the listed software installed:

- Windows 10/11 x64 with local administrator rights

  The integration runs on Windows 11, but applications to be integrated may not be compatible. Check the application on the manufacturer's website for compatibility with Windows 11.
- SOLIDWORKS in a supported version
- Java Runtime Environment 8 (x86 & x64)
- Microsoft .NET Framework 4.5.2

Hardware requirements for the client computer depend on SOLIDWORKS.

Server-side requirements

The following software is required:

Agile PLM in a supported version.

Supported versions and special requirements

Name	Version	Special requirements
Oracle Agile PLM	9.3.6 RUP 15 & 23	An administrator account is required during installation and setup of the integration. The Agile PLM environment must be configured for SOLIDWORKS. A personal login for Agile PLM is available to the end-user.
Dassault Systèmes SOLIDWORKS	2020-23	The user has been granted any necessary privileges to use the application.

Required installation media

Make sure you have the following installation media ready:

- Integration installer `*.7z.exe`
- Integration license `*.lic`

Where to get the installation media:

Log in to the *Oracle Software Delivery Cloud* at https://edelivery.oracle.com or *My Oracle Support* at https://support.oracle.com and download the installation media. Contact the *Oracle Global Customer Support (GCS)* via www.oracle.com/support for access if you don't have an account yet.

In *Oracle Software Delivery Cloud*, the Zip archive mentioned above is named using a unique numerical identifier that does not reflect the package type mentioned above. Please refer to the information given in *Software Delivery Cloud* to identify the packages.

2.1 Required integration licenses

The integration functions and connectors require corresponding licenses. Make sure you have them available.

License overview

License	Scope
Agile PLM connector	Access the API of Agile PLM to use functions and exchange data.
SOLIDWORKS connector	Access the API of SOLIDWORKS to use functions and exchange data.
Others	Depending on other components used in the integration, additional licenses may be required.

Organize the required licenses directly via XPLM Solution GmbH (https://support.oracle.com).

2.2 Installing and upgrading Java Runtime Environments

Since integration versions 3.4.3.0, 3.5.0.11 and 3.6.0.0, the installer no longer includes the Java Runtime Environments (JREs) necessary for running the integration and therefore, you must manually install them.

Installation

Only Java 8 is supported. Install the JREs for 32-bit and 64-bit JREs before installing the integration. You can download the required JREs from https://java.com.

The installed JREs should then be automatically detected by the installation wizard. However, you can manually define the installation directories of the corresponding JREs during the installation process if required or if the installation wizard cannot detect them automatically.

Upgrade

The JRE installation directories are version dependent. After updating the JREs, it is necessary to also update the following environment variables that are used by the integration to locate the JREs:

- `XPLMJavaX64`
- `XPLMJavaX86`

To update the variables, execute the script `%xPlmRootDir%\accessories\Set_Java_Paths.bat` with administrator rights.

After the update, run the command `set x` in a Windows command prompt and check that the variables show the correct path to the JRE. Set it manually if not correct.

2.3 Legacy Workspaces Requirements

Workspace folders created with MCAD 3.3 or earlier versions are not supported by MCAD 3.4 or later.

The file caching mechanism was changed in MCAD 3.4 (individual .CLB files instead of the workspace-specific `cache.xml` file). You, therefore, need to check in workspaces created from MCAD 3.3 (or earlier) before migrating to MCAD 3.4 (or later) and then check them out again after the migration.

3 Installation

3.1 Installing integration

This chapter explains in detail how the default installation procedure works.

Before you start

Close all related programs.

Procedure

1. Double-click the archive `<prefix>_<version>.7z.exe` and extract it to a directory with full write access.
2. Go to the directory with the extracted files and start `setup.exe` with administrative rights.
3. Install any Visual C++ runtimes that you are prompted for.
   ➔ After the runtimes are installed, the installation wizard appears.
4. On the welcome screen, click **Next** to start the installation.
   ➔ The step *License agreement for end-users* appears.
5. Read and accept the license agreement and click **Next**.
   ➔ The step *Installation path* appears.
6. Define the installation path for the integration and click **Next**. This path will later become available as environment variable `%xPlmRootDir%`. If overwriting of customized files is required, select the check box **Apply custom files after installation** and provide the path to the custom files. See Using overlay packages (p. 123) for more information..
7. Define Java path and click **Next**.

   The button Local Java environment **Update** is used to fill fields according to registry values for local JavaEnvironment. This can also be performed after installation without installation wizard, simply execute `%xPlmRootDir%\accessories\Set_Java_Paths.bat`.
8. In the step *MCAD components*, select the authoring tool in use and, if required, a corresponding version. Click **Next**.
9. In the step *Tool components*, select additional XPLM tools and click **Next**.
   - Batch Queue Admin
   - Batch Server
   - Encryption Helper
   - Toolbar Image Creator for Dassault Systèmes SOLIDWORKS: With this tool, the integration ribbon can be modified.

   The usage of tools is described in the related *Tools Guide*.
10. Click **Install** to start the installation.
11. After completion, click **Finish** to close the wizard and reboot the system.

Result

The integration is installed.

After pressing **Finish**, the system asks for a reboot.

It is recommended to reboot the system after installation. It may occur that Windows operation system does not update environment variables written by installer. This seems to be a Windows issue.

3.2 Installing/Upgrading the license file

The Agile PLM integration requires a license file to run.

Before you start

Make sure you have the license file available.

About this task

You need administrative rights on the workstation to install the license file.

Procedure

1. Close the SOLIDWORKS system if running.
2. Unzip the license zip archive
3. **Optional:** If you do not know the current MCAD Connector installation path, do the following:
   a) Open a CMD window and type `set xplmr`
   b) Press Enter on your keyboard
      ➔ The environment variable `xPlmRootDir` points to the installation directory.
4. **New Installation:** Copy the unzipped license file `*.lic` to the directory `%xPlmRootDir%\xml`
5. **Existing Installation:** To upgrade the license, copy the unzipped `*.lic` file to the `%xPlmRootDir%\xml` folder replacing the existing `*.lic`.
6. Restart SOLIDWORKS and Agile PLM

Result

The license file is installed/upgraded.

When using an integration function, the license file is checked. If no license is found, an error message appears and the integration menu is not loaded. Refer to License error codes (p. 73) for a list of license error codes.

4 Initial setup

4.1 Dassault Systèmes SOLIDWORKS

4.1.1 Connector Configuration Settings

The configuration of the connector is done in XML files, which are in the `components\xml` subdirectory of the integration.

The following files are important for the connector:

- `XPlmSolidWorksConnector.xml` - base configuration of the SOLIDWORKS connector
- `XPlmSolidWorksA9Connector.xml` - contains mostly Agile PLM related settings
- `XPlmSWA9Add-in.xml` - contains the Agile menu definition and Add-in registration. **This file should not be changed**.
- `XPlmAgile9SolidWorksTransaction.xml` - contains the configured transactions for the Agile load and save processes. **This file should not be changed**.
- `CAXConfig.xml` - controls general and numbering options for load and save. This file is located in the `components\ini` subdirectory.
- `extensions.xml` - lists all supported file extensions. This file is located in the `components\ini` subdirectory.

Related links
Settings in XPlmSolidWorksConnector.xml (p. 103)
Settings in XPlmSolidWorksA9Connector.xml (p. 107)
CAXConfig.xml Settings (p. 84)

4.1.2 Customize the Addin Menu

You can customize the integration menu according to the needs of your organization by enabling or disabling menu functions in the integration.

Buttons and ribbon groups

Configuration file: `XPlmSolidEdgeA9Addin.xml`

In this file, buttons and the ribbon groups are defined. The buttons are defined in the section `Buttondefinition`:

```
...
  <Buttondefinition>
    <Button id="..." caption="..." tooltip="..." statusBarText="..."
    callbackMethod="..." bitmap="..."  <optional settings>
  </Buttondefinition>
...
```

- `id`: Defines the unique button ID.
- `caption`, `tooltip`, `statusBarText`: Defines references to resource keys in the file `XPlmA9AddinResource.xml`.
- `callbackMethod`: Defines a function to execute when clicking the button.
- `bitmap`: Defines the location of the icon in the directory `%xPlmRootDir%\graphics`.

- `<optional settings>`: Defines additional button configuration, for example calling a different script engine, passing a parameter, executing pre- or post-actions or a macro.

The ribbon menu shown in SOLIDWORKS when opening a part, an assembly, a drawing, or if nothing is opened, is defined in the section `Ribbondefinition`:

```
...
  <Ribbondefinition>
    <Ribbon environment="..." tag="...">
      <Group id="..." name="...">
        <Button id="..." imageSize="..." />
      </Group>
    </Ribbon>
  </Ribbondefinition>
...
```

Each group also has a unique ID and references a button defined in section `Buttondefinition`.

To remove functions, you can comment single `Button` definitions or whole `Group` definitions. To create a new group, the group needs its own ID, a resource string and button definitions.

Button texts

Configuration file: `XPlmA9AddinResource.xml`

This file contains the language-specific definitions for ribbon groups, button captions, tool tips and status bar texts.

4.1.3 Setup the Root Workspace

The integration sets the root workspace automatically.

Configuration file: `%xPlmRootDir%\com\start_acw.bat`

Configure the root workspace by setting these values:

- `set cax_temp=C:\AgileEC\wspaces\Default`
- `set CAX_WORKSPACE_ROOT=C:\AgileEC\wspaces`

The default workspace is `C:\AgileEC\wspaces`.

Each user must have full read and write access to the workspace directory and all its sub-directories on the local workstation.

4.1.4 Starting the integration

No additional configuration is necessary.

Start the Dassault Systèmes SOLIDWORKS connector for Agile PLM by starting SOLIDWORKS and check whether the integration menu is available.

Open the Workspace Manager or the Save Preview dialog, log in to Agile PLM, if prompted. Verify that the version number displayed at the bottom bar of the dialog window matches the MCAD version installed.

4.2 Oracle Agile PLM

4.2.1 Required Server Configuration

This section provides a complete summary of configuration options required for the MCAD connector.

The MCAD connector requires an essential minimum set of fields enabled to work properly. Make sure all the fields listed in the following chapters are visible and enabled on your Agile PLM server and for all users that should work with the connector. You may need to enable additional fields according to the desired property mapping.

In most cases, the attribute name is predefined, although it may be disabled by default.

Do not change API names of Heading attributes.

Make sure the **User Roles** and **Modify Privileges** are setup for the login user to **discover**, **modify** and **read** all the fields mentioned in the following chapters.

Agile PLM may not explicitly support **Modify** privileges for all the attributes given in the following chapters. For such attributes, **Read** and **Discovery** privileges are sufficient.

Large text attributes are currently not supported

4.2.1.1 Enabling Class Objects Attributes

The MCAD connector requires an essential minimum set of fields enabled to work properly.

About this task

This use case describes how to enable Class Objects Attributes on the Java Client.

Procedure

1. On the Java Client, navigate to the **Admin** tab and expand the **Data Settings** options by clicking on the plus (+) sign besides the folder name.
2. Double click on **Classes**

   ➔ A dialog is opened displaying all the available Class Objects in Oracle Agile PLM.
3. Navigate to the desires Class object and double click on it.

   ➔ Another dialog is opened with the selected Class object as the header. The dialog has several tabs that provide more information about the Class Object. These may include **General Information**, **User Interface Tabs**, **Lifecycle Phases**, **Process Extensions**, **Event Subscribers** and **History**
4. Navigate to the **User Interface Tabs**.

   ➔ A list of available tabs is displayed.
5. Double click on the desired tab (for example Title Block) from the list displayed.

   ➔ Another dialog is opened displaying two tabs; **General Information** and **Attributes**

   More tabs maybe be available.
6. Go to the **Attributes** tab

   ➔ A list of available attributes is displayed in tabular form.

7. Double click on the desired attribute

   ➔ Another dialog is opened displaying all the option settings available for the selected attribute.

   Use the **Base ID** of the attribute to identify the desired attribute as the **Name** of the attribute may have been changed.

8. Set the option setting **Visible** to **Yes**
9. **Optional:** Set other options as required.
10. Click **Save**

Result

The attribute has been enabled in Oracle Agile PLM.

4.2.1.2 Required Designs Class Configurations

Ensure the following attributes are enabled;

Designs - Page Two

- Local Flag (ID: 1301)
- Part Number (ID: 1302)
- Model Type (ID: 1332)
- Model Reference (ID: 1333)
- Link Type (ID: 1334)
- Link Reference (ID: 1335)
- Design System (ID: 2007)
- Design System Identifier (ID: 2008)
- Filetype (ID: 2009)
- Subtype (ID: 2010)
- Family (ID: 2011)
- Variant (ID: 2012)
- Drawing Name (ID: 2013)
- Frame ID (ID: 2014)
- Name Format (ID: 2015)
- Project Name (ID: 2016)
- CAD Filename (ID: 2017)
- File Path (ID: 2019)

Designs - Files

- **File Category** (ID: 2000008509) - with a list of available values containing the **Source** and **Viewable** values.

Designs- Structure

- Quantity (ID: 2000008325)
- Component Type (ID: 2000008330)
- Model Name (ID: 2000008376)
- Identifier (ID: 2000008377)
- Component (ID: 2000008378)

- Reference (ID: 2000008379)
- Configuration (ID: 2000008380)

Designs - Where Used - Design

- Version (ID: 2000008501)
- Component Type (ID: 2000008508)
- Model Name (ID: 2000009311)
- Identifier (ID: 2000009312)
- Component (ID: 2000009313)
- Reference (ID: 2000009314)
- Configuration (ID: 2000009315)

Designs - Relationships

- Link Type (ID: 5846)
- Published Change (ID: 5847)
- CAD Model (ID: 5861)
- CAD Parent Model (ID: 5862)
- Relationship Type (ID: 2000007912)
- Number (ID: 2000007927)
- Version (ID: 2000008523)

4.2.1.3 Required Parts Class Configurations

Ensure the following attributes are enabled and configured as follows;

Parts - Title Block

- Number (ID: 1001)

We recommend that you set the attributes **Include Characters** to `All` and **MaxLength** to `75`. However, you can define the allowable characters and **MaxLength** as desired in your organisation.

Parts - BOM

- Item Number (ID: 1011)
- BOM Quantity (ID: 1035)
- CAD Filename (ID: 1341)
- Identifier (ID:2175)
- Component (ID: 2176)
- Reference (ID:2177)

We recommend that you keep the attributes **Maxlength** and **Max System Length** to their default settings or make sure they allow long enough values to avoid BOM publishing errors (for example Maxlength should have at least 150 characters for CAD Filename (ID: 1341)).

Parts - Relationships

- Link Type (ID: 5846)
- Published Change (ID: 5847)
- Number (ID: 2000007927)

Parts - Changes - Pending Changes

- Change Category (ID: 2156)
- Lifecycle Phase (ID: 2000009590)
- Workflow (ID: 2000009595)

4.2.1.4 Required Documents Class Configurations

Ensure the following attributes are enabled when using Documents classes as Items for Drawings:

Documents - Relationship

- Link Type (ID: 5846)
- Published Change (ID: 5847)
- Number (ID: 2000007927)

Documents - Pending Changes

- Change Category (ID: 2156)
- Lifecycle Phase (ID: 2000009590)
- Workflow (ID: 2000009595)

Documents - BOM Attributes

Since BOM are rather uncommon for Documents classes, usually no configuration of the BOM tab is required. However, if BOM should be used, the following configuration needs to be implemented.

- BOM Quantity (ID: 1035)
- CAD Filename (ID: 1341)
- Identifier (ID:2175)
- Component (ID: 2176)
- Reference (ID:2177)

4.2.1.5 Control Upload File Types and File Association Table

The MCAD connector does not verify the **Control Upload File Types** setting and the related **File Association** table in Oracle Agile PLM. Thus, you must ensure that the file types, which can be uploaded to Oracle Agile PLM, match the file types of the CAD system(s) in use.

About this task

This task describes how to set up **Control Upload File Types** and optionally **File Association** table Oracle Agile PLM Java Client.

Procedure

1. On the Java Client, navigate to the **Admin** tab and expand the **System Settings** options by clicking on the plus (+) sign besides the folder name.
2. Double click on **Viewers and Files**

   ➔ A dialog named **Viewers and Files** is opened displaying the **General Information** tab.
3. Set **Control Upload File Types** to **No** on the General Information's tab

   This is the recommended setting.
4. **Optional:** Set **Control Upload File Types** to **Yes**on the General Information's tab

5. **Optional:** Go to the **File Association** tab and verify that all the file types supported by the CAD system is use are listed and checked.

Result

Control Upload File Types and related **File Association** are set.

4.2.1.6 Allowing MCAD Engineers to See Checkout Users

Per default, Agile PLM is configured to not allow non-administrator users to see the content of the **Checkout User** attribute within the Designs class object.

About this task

This use case describes how to activate read-level access to the Checkout User attribute (Design.Title Block).

It is recommended to allow all MCAD users to **read** and **discover** this attribute to avoid conflicting access to Design objects.

Procedure

1. On the Java Client, navigate to the **Admin** tab and expand the **User Settings** options by clicking on the plus (+) sign besides the folder name.
2. Double click on **Roles**

   ➔ A dialog named **Roles** is opened.
3. Set the filter criteria **Match If** to **Show All** and Click **Apply** on the Roles dialog.

   ➔ A list of all available Roles is displayed in tabular form.
4. From the list, double click on **Design Engineer**

   ➔ Another dialog giving more information about the Design Engineer Role is displayed.

   Your organization might have this role renamed.
5. Go to **Privileges** tab
6. Click the **Add Privileges** icon (first icon on the left)

   ➔ A pop-up dialog **Select Privileges** is opened and may display already enabled privileges on the right hand section of the dialog called **Selected**. The **Choices** section is empty.
7. Set the filter criteria **Match If** to **Contains** and manually enter *Users* to the **Value** text field and then Click **Apply**.

   ➔ A list of all matching Privileges matching the query criteria is displayed in tabular form in the **Choices** section.
8. Select **Discover Users - Discovery** and click the arrow pointing right in the middle of the two sections.

   ➔ The **Discover Users - Discovery** privilege is moved to the **Selected** section.
9. Select **Read Users - Read** and click the arrow pointing right in the middle of the two sections.

   ➔ The **Read Users - Read** privilege is moved to the **Selected** section.
10. Click **OK**

Result

The Privileges **Discover Users - Discovery** and **Read Users - Read** have been added to the Design Engineer Role and are displayed in the Privileges tab. MCAD users can now see which other user is currently working on a checked out Design object.

4.2.2 Login Access Administration in Agile PLM

This section provides detailed information about all necessary steps to enable login restriction for defined client versions that connect to the Agile PLM server.

About this task

All steps need to be executed as an administrator user that has rights to modify roles and privileges.

Procedure

1. Create a new **FileFolder** subclass, for example *MCAD-Access*. This subclass should not have an AutoNumber assigned as the numbers need to be entered manually. This could also be a Designs subclass, but for separation purposes and easier configuration, a FileFolder subclass is recommended.
2. Find any existing **Discovery** privileges for File Folders (for example **Discover All File Folders**) and modify the criteria to exclude the new File Folder class created in step 1. This prevents any non-admin user to modify the access setup. It might be required to create a new criteria because the existing one could not be modified.
3. Create a new object in subclass **MCAD-Access**, for example *MCAD-3.2.0.0.130701*, for each version you want the user(s) to grant access to. The new object number needs to be prefixed with `MCAD-` and the version number is the same as displayed in the Web Components status line in the bottom of the dialogs.
4. Create a new privilege for **Discovery**, for example *Connect with MCAD 3.2.0.0* with criteria *Discover MCAD 3.2.0.0* on object type **MCAD-Access** and the condition *MCAD-Access Title Block.Number Equal To MCAD-3.2.0.0.130701*.
5. Add this privilege to the user's CAD specific role, for example **CAD Designer**.
6. Remove the privilege **Discover All File Folders** from that same role. This is vital as otherwise this privilege overwrites any restrictions created in step #4.
7. More specific privileges on certain file folders could be added to the user's role if needed.
8. To secure the access to the **MCAD-Access** subclass, any write privileges for non-admin users should be revoked on any object of that subclass.

   To be backwards compatible or to allow customers to use MCAD without any **MCAD-Access** setup, the client is connected successfully if the **MCAD-Access** subclass or the version-specific FileFolder does not exist.

   The latter requires having the appropriate **MCAD-Access** FileFolder created before an MCAD version, other than the ones already set up in the system, is passed to the user. For any MCAD version for which the access should be rejected, a corresponding FileFolder object needs to be created, but no discover privilege should be assigned to the user's role for this object.

4.2.3 Enabling HTTPS for Engineering Collaboration Clients

If the MCAD connectors or custom client applications utilizing the Agile PLM Core and EC Services need to use the HTTPS protocol instead of the HTTP protocol, some additional work and configuration needs to be done.

4.2.3.1 Using self signed server certificate

This description assumes that the server side (either the application server itself or an HTTPS proxy server) is already configured to run with HTTPS. This should be validated with the help of the Agile PLM Web Client.

To make the client-side work properly with the HTTPS server, you must import the server's Certificate Authority (CA) certificate into a Java keystore (.JKS) for the clients and a client certificate needs to be generated with the help of this CA certificate. Please consult the corresponding documentation for more information.

Configuring the MCAD Connectors for HTTPS

Configuration file: `acx.bat`

To make use of the keystore client file, certain Java startup parameters need to be added to the MCAD's JVM startup call (defined in `acx.bat`). This enables the Java application to validate the client certificate with the one from the server and then establish an HTTPS trusted and secure connection.

The Client's Java keystore file should be stored in the `%xPlmRootDir%\ini` directory.

Modify the `acx.bat` startup script by adding the following line:

```
set HTTPS_OPTS=-Djavax.net.ssl.trustStore=%CAX_ROOT%\ini\plm-client.jks
-Djavax.net.ssl.trustStorePassword=Agile123
-Djavax.net.ssl.trustStoreType=jks
```

where

- `plm-client.jks` is the Java Keystore file for the client
- `Agile123` is the keystore password

The code above shows how to use the required parameters on a Windows system only, as the CAD connectors are only supported on Windows systems.

The following line should be modified to use the `HTTPS_OPTS` definition. The variable `%HTTPS_OPTS%` could be included in the command line at any location, but it should be after the variable `%JAVA_HEAP_SIZE%`.

```
start /b "%JAVA_HOME%\bin\javaw.exe" %JAVA_HEAP_SIZE% %HTTPS_OPTS%
-Dcom.xplm.agile.Language=EN …
```

Configuring custom client applications for HTTPS

To make use of the certificates in the keystore and the client file, certain Java system properties needs to be added into the custom client code. This enables to Java application to validate the client certificate with the one from the server and then establish an HTTPS trusted and secure connection.

The files Java keystore file (*plm-client.jks*) should be copied into the client application's directory, but could essentially reside anywhere on the local file system.

The code below shows how to use the required properties for the Java programming language, but this should work similarly on any other programming language, for example C#.

```
System.setProperty("javax.net.ssl.trustStore", "plm-client.jks");
System.setProperty("javax.net.ssl.trustStorePassword", "Agile123");
System.setProperty("javax.net.ssl.trustStoreType", "jks");
```

In Java, the value for the property `javax.net.ssl.trustStoreType` does not need to be set as it is `jks` by default. But it might be necessary in other programming languages.

This also works with other programming languages that could be used to generate the client stubs for the web services. Please consult the corresponding documentation for more details.

4.2.4 Enabling WSS for Engineering Collaboration Clients

Since Oracle Agile PLM 9.3.4, Web Service Security (WSS) is supported. This is a core feature of the WebLogic application server to allow setting policies for web service clients to access web services with certain authentication methods only. For further information, please see the corresponding documentation for your WebLogic version on http://docs.oracle.com.

Configuring the MCAD Connectors for WSS

The MCAD connectors automatically detect if WSS is enabled for the Core Services and EC Services and use the required client-side policy. As WSS requires HTTPS to be used, you must first enable HTTPS for MCAD.

For WSS, it is usually only required to use One-Way Authentication and not Two-Way Authentication (called Mutual Authentication). With One-Way Authentication the client would only need to import the server certificate if the server certificate is self-signed. If the server certificate had been issued by a Certificate Authority (CA) so that it is considered as trusted, this is not needed.

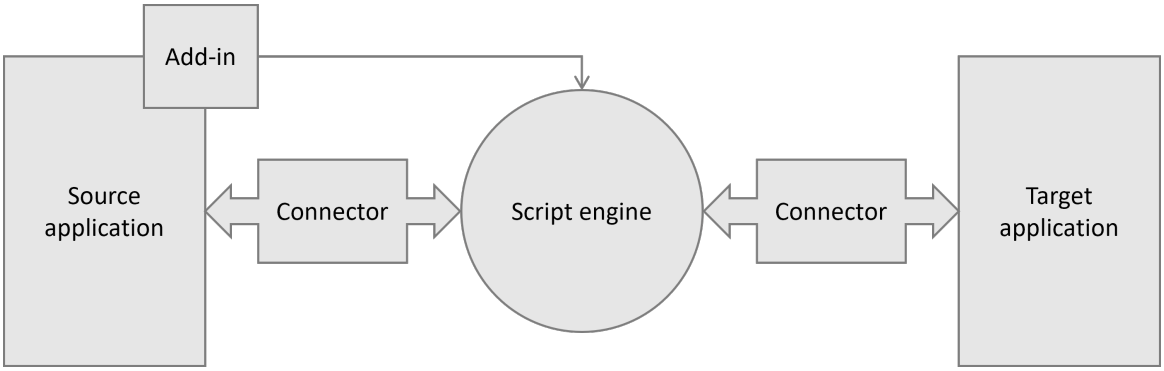
For any web services client with enabled web service security, only the policy `oracle/wss_username_token_over_ssl_client_policy` is supported. As this policy requires SSL, the WebLogic application server must be configured to serve HTTPS requests.

5 Configuration

5.1 General integration components and related files

The integration consists of several software components that interact with each other. Read this chapter to familiarize yourself with these topics.

The following components represent a classic integration setup. The setup can be more complex or use additional components for specific integrations. However, most of the components described below are always part of the setup in one way or another.



- The **connector** is a software library that provides access to an application through its API. It implements various functions based on the application API and defines a standardized interface to the script engine. Typical connector functions include for example *getActiveDocument*, *getOpenDocuments*, *Open*, *Save*, or *Close*.
- The **script engine** is a software library that provides the integration logic between applications and connectors. It does not directly access related applications, but communicates with them through the connectors. This architecture ensures that the script engine can support different application versions without needing to change the integration logic. Main functions of the script engine include for example *Connect*, *Save*, *Checkin/Checkout*, or *Load*.
- The **add-in** provides the integration functions in the source application. The add-in communicates with the script engine, which in turn communicates with the connector.
- The integration consists in general of the following configuration files:
  - : Contains global configuration options and lists other dependent files, for example connector and resource files.
  - Transaction files: Contain the data mapping between applications. Data mapping is essential for exchanging data using the integration.
  - Connector files: Contain options for the connectors and the script engine.
  - Dialog files: Contain the configuration of the integration dialogs.
  - Resource files: Contain the language-specific text strings used in dialogs and add-in.
  - Add-in file: Contains the configuration of the add-in.
  - License file: Contains the required product keys for the individual integration components. Without the license file the integration does not work.

5.1.1 Data mapping and transaction files

A key concept of the data exchange between applications is the data mapping. Read this chapter to familiarize yourself with these topics.

Data mapping

In each integration, object information is exchanged between connected applications. In the most simple form, a property or the content of a database field is transferred from the source application to the target application. The technical mapping and its process within the integration are commonly referred to as data mapping.

Data mapping is based on transaction files written in the XML syntax. For more complex structure transformations, also XSLT stylesheets can be used.

In both cases, the process flow is the same:

- The source data is read from the source application and provided internally in an *XPlmDocument* container.
- This source data is translated either via a transaction or a stylesheet into an object that can be understood by the target application. The translation is done by the script engine, but also by connectors.

Transaction files

Transactions are independent, unified data containers. As already mentioned, the data mapping is defined in transaction files. Each file can contain multiple transactions. A transaction is identified by the element `<Aliasname>`. It further consists of the element structures `<FieldCollection>`, `<StructureCollection>` and `<TableCollection>`. These structures contain values that are important for the underlying definition of the data mapping.

```
<Transaction>
  <Aliasname>MyTransaction</Aliasname>
  <Import>
    <Parameter>
      <FieldCollection>...</FieldCollection>
      <StructureCollection>...</StructureCollection>
      <TableCollection>...</TableCollection>
    </Parameter>
  </Import>
</Transaction>
```

There are a few points to keep in mind when working with transactions. Within the transaction itself, all information is treated as strings, which means that unless specific functions are used, there is no parsing or validation of the content.

Overwriting transactions

You can overwrite and customize the XML configuration files of the integration. In fact, you can overwrite any XML file that contains transaction structures.

To overwrite a transaction, create a new file with the same name as the original file, but with the additional prefix `Customer_`. This so called customer file may or may not contain a transaction from the original file. When executing transactions, the integration checks whether a corresponding customer file exists. If it exists, any transaction listed in that file is executed instead of the transaction defined in the original file. If no transaction is defined in the customer file, the transaction from the original file is used.

Never modify the original transaction files of the integration. Include all required modifications in customer files, as this makes it easier to update and troubleshoot the integration. Always overwrite complete transaction structures. You cannot overwrite a subset of a transaction, for example a single *FieldCollection* element, because these XML structures cannot be identified without the corresponding transaction.

5.1.2 XPLM data types

The data types listed here are used in transaction files, but also in connector files. They represent the internal data model of the integration on which the entire data exchange is based.

Field

Defines the smallest configuration unit. It describes a key/value pair and has its origin in a field of the source application, for example a database field or a property. The key is defined in the *Name* element, the value in the *Value* element .

```
<Field>
  <Name>MyField</Name>
  <Value>Value</Value>
</Field>
```

FieldCollection

Groups several *Field* elements. Within a *FieldCollection* element, field names should be unique. There can be only one *FieldCollection* element in an XML file.

```
<FieldCollection>
  <Field>
    <Name>Field1</Name>
    <Value>Value1</Value>
  </Field>
  <Field>
    <Name>Field2</Name>
    <Value>Value2</Value>
  </Field>
</FieldCollection>
```

Structure

Defines a named *FieldCollection* element. It is identified by the *Name* element and contains a *FieldCollection* element. Thus, it represents a collection of properties that are not combined in the simple FieldCollection element for technical reasons or for better readability of the source data. In addition, *Field* elements with the same name can be used in different structures.

```
<Structure>
  <Name>Structure1</Name>
  <FieldCollection>
    <Field>
      <Name>Field1</Name>
      <Value>Value1</Value>
    </Field>
    <Field>
      <Name>Field2</Name>
      <Value>Value2</Value>
    </Field>
  </FieldCollection>
</Structure>
```

StructureCollection

Groups several *Structure* elements but can also contain only one structure or be empty. There can be only one *StructureCollection* element in a transaction.

```
<StructureCollection>
  <Structure>
    <Name>Structure1</Name>
    <FieldCollection>...</FieldCollection>
  <Structure>
  <Structure>
    <Name>Structure2</Name>
    <FieldCollection>...</FieldCollection>
  <Structure>
</StructureCollection>
```

FieldCollectionTable

Contains one or more *FieldCollection* elements. It is used exclusively within the *Table* element, which provides a table-like mapping of data. Unlike a table definition in a relational database, the columns, in this case the *Field* elements, can be different in each row, although this will rarely be the case in practice. In this container, each *FieldCollection* element corresponds to a table row.

```
<FieldCollectionTable>
  <FieldCollection>...</FieldCollection>
  <FieldCollection>...</FieldCollection>
</FieldCollectionTable>
```

Table

Groups several *FieldCollectionTable* elements and is identified by the element *Name*.

```
<Table>
  <Name>Table1</Name>
  <FieldCollectionTable>...</FieldCollectionTable>
  <FieldCollectionTable>...</FieldCollectionTable>
</Table>
```

TableCollection

Groups several *Table* elements. There can be only one *TableCollection* element in a transaction.

```
<TableCollection>
  <Table>
    <Name>Table1</Name>
    <FieldCollectionTable>...</FieldCollectionTable>
   </Table>
   <Table>
      <Name>Table2</Name>
      <FieldCollectionTable>...</FieldCollectionTable>
   </Table>
</TableCollection>
```

5.1.3 Data referencing

The data types describe the source data format, whereby the target data format is semantically identical.

Basic rules for data referencing

The referencing of the individual fields is done with a special notation. This type of description is subject to certain restrictions and follows some basic rules:

- Referencing is always done at the field level, which means that the source of the data can be very broad, but the target is always a *Field* element.
- Referencing data in a table can only be made with a specific row number or with the help of a search expression. A dynamic reference is not possible.
- Partially qualified references are not supported, which means the name of an element must always be specified, case sensitive.
- If the referenced *Field*, *Structure*, or *Table* element does not exist, an empty string is used as the result.

Reference to fixed value

In the simplest form, there is no reference to the source data, but a fixed value is specified.

```
<Field>
  <Name>LANGUAGE</Name>
  <Value>DE</Value>
</Field>
```

Reference to field

To reference the value from a field in a *FieldCollection*, three parameters are specified instead of a fixed value:

- *Type*: Contains *XPlmDocument* or *ParameterList* and refers to the default object used in the integration source code.
- *Subtype*: Contains the name of the parent element, in this case *FieldCollection*.
- *Attribut*: Contains the name of the *Field* element from which the value is to be read.

```
<Field>
  <Name>DESCRIPTION</Name>
  <Type>XPlmDocument</Type>
  <Subtype>FieldCollection</Subtype>
  <Attribut>T_MASTER_DAT.PART_NAME</Attribut>
</Field>
```

Reference to field in StructureCollection

To reference the value from a field in a *StructureCollection*, four parameters are specified:

- *Type*: Contains *XPlmDocument* or *ParameterList* and refers to the default object used in the integration source code.
- *Subtype*: Contains the name of the parent element, in this case *StructureCollection*.
- *Structurename*: Contains the name of the structure.
- *Attribut*: Contains the name of the *Field* element from which the value is to be read.

```
<Field>
  <Name>DOCUMENTNUMBER</Name>
  <Type>XPlmDocument</Type>
  <Subtype>StructureCollection</Subtype>
  <Structurename>EDB-ART-DOC-RLI</Structurename>
  <Attribut>T_DOC_DAT.DOCUMENT_ID</Attribut>
</Field>
```

Reference to field in TableCollection

To reference the value from a field in a table and a defined row in a *TableCollection*, five parameters are specified:

- *Type*: Contains *XPlmDocument* or *ParameterList* and refers to the default object used in the integration source code.
- *Subtype*: Contains the name of the parent element, in this case *TableCollection*.
- *Table name*: Contains the name of the table.
- *Row*: Contains the number of the row from which the field is to be read.

- *Attribut*: Contains the name of the *Field* element from which the value is to be read.

```
<Field>
  <Name>URL</Name>
  <Type>XPlmDocument</Type>
  <Subtype>TableCollection</Subtype>
  <Tablename>EDB-ART-SLI</Tablename>
  <Row>5</Row>
  <Attribut>T_MASTER_DAT.UNIT</Attribut>
</Field>
```

5.1.4 Field mapping

The transfer of values alone is not sufficient in practice, since there is usually no conversion of different types in the source and target connectors. This means that all values in a data structure are not typed and must therefore be treated as simple strings. It can be assumed that the data types of the applications involved are usually not identical and treat types such as dates, logical values or numbers differently. For this reason and certain properties it is necessary to convert the content before further use.

For this purpose there is a possibility to convert the content with the help of certain functions, which is called field mapping. The basic principle is that in each field reference in the data mapping an additional element *Function* can be used, in which one or more sequentially acting functions for the conversion can be entered. The input of the function always consists of the referenced value and the specific function parameters, the result of the function call is then taken over for the data mapping.

Simple function call

The function name and its comma-separated parameters are written to the *Function* element. In this example, the referenced value has a prefix added.

```
<Field>
  <Name>DESCRIPTION</Name>
  <Type>XPlmDocument</Type>
  <Subtype>FieldCollection</Subtype>
  <Attribut>T_MASTER_DAT.PART_NAME</Attribut>
  <Function>Prefix,PRE-</Function>
</Field>
```

In another example, one string is replaced by another.

Spaces after the comma are considered part of the parameter!

```
<Field>
  <Name>DESCRIPTION</Name>
  <Type>XPlmDocument</Type>
  <Subtype>FieldCollection</Subtype>
  <Attribut>T_MASTER_DAT.PART_NAME</Attribut>
  <Function>Replace,-,/</Function>
</Field>
```

Multiple function call

Multiple functions can also be used:

- The functions are executed one after the other.
- The result of each function is used as input for the following function (sequential execution).
- If the execution of a partial function runs into an error, it is skipped and the result of the last successfully executed partial function is used as input for the subsequent function.

```
<Field>
  <Name>DESCRIPTION</Name>
  <Type>XPlmDocument</Type>
  <Subtype>FieldCollection</Subtype>
  <Attribut>T_MASTER_DAT.PART_NAME_GER</Attribut>
  <Function>Replace,#,-|Postfix,-POST</Function>
</Field>
```

5.2 Dassault Systèmes SOLIDWORKS

5.2.1 Setup the Attribute Mapping

Use the **Mapping Editor** to define the Attribute Mapping

See Mapping (p. 47) for more details.

5.2.1.1 Mapping SOLIDWORKS Properties

SOLIDWORKS has different property pages which are treated transparently by the integration.

SOLIDWORKS file attributes in **Summary** tab of the **SummaryInformation** properties dialog cannot be mapped to Agile PLM attributes.

Related links
Mapping (p. 47)

5.2.2 Handling of Standard Parts

The SOLIDWORKS connector provides additional functionality to manage Standard Parts.

This feature was introduced in release 3.6 of the SOLIDWORKS connector.

5.2.2.1 Configuring the Standard Parts Directory

To use the Standards Parts feature, you must configure the directory in which the Standard Parts are stored.

- **Configuration file:** `XPlmSolidWorksConnector.xml`
- **Parameter:** `SolidWorksStandardPartDir`

We recommend using the same directory used by the SOLIDWORKS system to store Standard Parts by default. Using this directory would be especially preferable when using the default Standard parts supplied with the SOLIDWORKS system. The default directory for SOLIDWORKS is usually C:\SOLIDWORKS Data

However, the SOLIDWORKS connector can support any other directory, thus the configuration can be adapted to support Standard part directories in any other place on the file system if the SOLIDWORKS system supports it.

Related links
Settings in XPlmSolidWorksConnector.xml (p. 103)

5.2.2.2 Saving Standard Parts

You can save Standard Parts to Agile PLM with normal save functions.

The Standard Parts are saved to Design objects with unique Design Number in Agile PLM.

Standard Parts are flagged on the Agile PLM Web Client with the value *TOOLBOX* in the **Design.PageTwo.Name Format** attribute.

Once saved to Agile PLM and thus Agile PLM-known, Standard Parts are not affected by the MCAD connector's **Save As** function. This means, that no new Design number is assigned to them in case that they are selected and the **Save As** button is clicked.

5.2.2.3 Loading Standard Parts

Standard parts are loaded the same way as any other SOLIDWORKS files.

Standard Parts loaded to the active workspace are displayed in the Workspace manager and have .CLB files assigned to them. They do not cause any conflicts related to the separate Toolbox Library outside of the workspace that might exist, since both SOLIDWORKS files (the standard part file in the active workspace and the standard part file in the Toolbox Library) link to the same Design object in Agile PLM.

Standard Parts are set as read only if downloaded into the active workspace directory.

5.2.2.4 Renaming Standard Parts

Standard parts never get renamed by the SOLIDWORKS connector.

The corresponding `Renaming on Save` and `Renaming on Load` functions have no effect on Standard parts.

5.2.2.5 Viewable Creation for Standard Parts

Viewables are not created for Standard parts. Regardless of the viewable related option settings selected in the Preferences dialog.

5.2.2.6 Standard Parts Without Collaboration Files

The SOLIDWORKS connector supports concurrent use of Standard Parts.

When saving Standard parts to Agile PLM, the SOLIDWORKS connector queries the Agile PLM system for the file name of the Standard part. If the query returns a result, the MCAD connector re-assigns the corresponding Design object to the Standard Part in question. This way, Standard Part that had been saved to Agile PLM before can be linked to their Design objects even without having a .CLB file assigned.

This functionality also allows several users that utilize the same set of Standard Parts (but not on a shared location, like a network drive) to work with Standard Parts without having to worry about conflicts regarding the concurrent usage of those files.

5.2.3 Strict CAD Modification Workflow

Since release 3.6, the MCAD connectors supports a strict CAD modification workflow based on Agile PLM checkout status. If enabled, users are restricted from modifying Agile PLM -known SOLIDWORKS objects unless they have reserved/checked out the SOLIDWORKS objects in Agile.

5.2.3.1 Option Settings

The strict CAD modification workflow feature is controlled by several options settings.

Configuration file: `XPlmSolidWorksA9Connector.xml`

Option Setting	Function
SolidWorks_ReserveBeforeModify_EnableUseCase	Global switch to activate or deactivate the strict CAD modification workflow feature. **Possible values: 0 (deactivated) and 1 (activated)**
SolidWorks_ReserveBeforeModify_NotifyUser	Enables or disables pop-up dialogs related to the strict CAD modification workflow feature. The actual dialog displayed also depends on the `SolidWorks_ReserveBeforeModify_AutoReserve` option setting. **Possible values: 0 (deactivated) and 1 (activated)**
SolidWorks_ReserveBeforeModify_AutoReserve	Enables or disables automatic check out attempts related to the strict CAD modification workflow feature. **Possible values: 0 (deactivated) and 1 (activated)**

Depending on the values of the option settings given above, the behaviour of the feature changes as described below

Option Setting	Behaviour				
	A	B	C	D	
SolidWorks_ReserveBeforeModify_EnableUseCase	1	1	1	1	0
SolidWorks_ReserveBeforeModify_NotifyUser	1	0	1	0	0 or 1
SolidWorks_ReserveBeforeModify_AutoReserve	1	1	0	0	0 or 1

Explanation

- **Behavior A** - The MCAD connector displays pop-up dialog for all file(s) that are not checked out. In the pop-up dialog, users can opt to check out the file(s) in question. This removes the read-only flag.
- **Behavior B** - The MCAD connector implicitly tries to check out the file(s) in question. If successful, the read-only flag is removed.
- **Behavior C** - The MCAD connector checks if the file(s) in question are checked out by the user. If so, the read only flag is removed, if not, an information pop-up is displayed.
- **Behavior D** - The strict CAD modification workflow feature is deactivated.

The SOLIDWORKS events `SolidWorksEvent_Part/Assembly/DrawingModifyNotify` must also be enabled in `XPlmSolidWorksConnector.xml` for the strict CAD modification workflow feature.

Related links
Activating the Feature (p. 36)
Settings in XPlmSolidWorksA9Connector.xml (p. 107)

5.2.3.2 Activating the Feature

The strict CAD modification workflow feature is deactivated per default.

It can be activated by changing the option settings as follows:

In `%xPlmRootDir%\xml` directory, modify the settings in `XPlmSolidWorksA9Connector.xml` as given;

```
SolidWorks_ReserveBeforeModify_EnableUseCase = 1
SolidWorks_ReserveBeforeModify_NotifyUser = 0 or 1 (*)
SolidWorks_ReserveBeforeModify_AutoReserve = 0 or 1 (*)
```

Refer to the Option Settings (p. 35) for details on which combination settings are valid and their behavior.

In `%xPlmRootDir%\xml` directory, modify the settings in `XPlmSolidWorksConnector.xml` as follows;

```
SolidWorksEvent_PartModifyNotify = true
SolidWorksEvent_AssemblyModifyNotify = true
SolidWorksEvent_DrawingModifyNotify = true
```

Related links
Settings in XPlmSolidWorksA9Connector.xml (p. 107)
Settings in XPlmSolidWorksConnector.xml (p. 103)

5.2.3.3 Changes to the SOLIDWORKS Workflow

If activated, the Strict CAD modification workflow feature introduces several changes to how users work with the SOLIDWORKS connector.

The feature models the **Check Out** status of an object in Agile PLM to the corresponding SOLIDWORKS object in the SOLIDWORKS system and prevents users from making or saving modifications to non-checked out objects.

To prevent users from making or saving modifications to non-checked out objects, the connector activates the SOLIDWORKS system's read-only mode implicitly whenever users open or load an Agile PLM managed SOLIDWORKS object that is not checked out for them. See User Guide for more information.

5.2.4 Transfer BOM Find Number Functionality

The SOLIDWORKS connector is capable of transferring BOM find numbers to Agile PLM.

5.2.4.1 Configure Transfer BOM FindNumbers

Configuration file: `XPlmAgile9SolidWorksTransaction.xml`

Transaction: `createBOM_Entry`

Configure the `createBOM_Entry` transaction to match the BOM table layout used for the drawing BOM tables.

The following field definitions need to be given in that transaction:

Field Name	Required Configuration for the Attribute [sic!]
FIND_NO	String given in the column header of the BOM table column containing the find numbers.
QUANTITY	String given in the column header of the BOM table column containing the quantities.
DESCRIPTION	String given in the column header of the BOM table column containing the item descriptions.

The attribute value given must match the existing column header names in the SOLIDWORKS BOM table.

Any number of field definitions for the same column may be added to the transaction, as shown in the example below:

```
<Transaction>
    <Aliasname>createBOM_Entry</Aliasname>
    <Import>
      <Parameter>
        <FieldCollection>
          <Field>
            <Name>FIND_NO</Name>
            <Type>ParameterList</Type>
            <Subtype>FieldCollection</Subtype>
            <Attribut>ITEM NO.</Attribut>
          </Field>
          <Field>
            <Name>FIND_NO</Name>
            <Type>ParameterList</Type>
            <Subtype>FieldCollection</Subtype>
            <Attribut>POS-NR.</Attribut>
          </Field>
          <Field>
            <Name>NUMBER</Name>
            <Type>ParameterList</Type>
            <Subtype>FieldCollection</Subtype>
            <Attribut>PART NUMBER</Attribut>
          </Field>
          <Field>
            <Name>NUMBER</Name>
            <Type>ParameterList</Type>
            <Subtype>FieldCollection</Subtype>
            <Attribut>BENENNUNG</Attribut>
          </Field>
          <Field>
            <Name>QUANTITY</Name>
            <Type>ParameterList</Type>
            <Subtype>FieldCollection</Subtype>
            <Attribut>QTY.</Attribut>
          </Field>
          <Field>
            <Name>QUANTITY</Name>
            <Type>ParameterList</Type>
            <Subtype>FieldCollection</Subtype>
            <Attribut>MENGE</Attribut>
          </Field>
          <Field>
            <Name>1002</Name>
            <Type>ParameterList</Type>
            <Subtype>FieldCollection</Subtype>
            <Attribut>DESCRIPTION</Attribut>
          </Field>
          <Field>
            <Name>DESCRIPTION</Name>
            <Type>ParameterList</Type>
            <Subtype>FieldCollection</Subtype>
            <Attribut>DESCRIPTION</Attribut>
```

Mapping of Agile PLM fields twice is not supported.

To send the white space string (length > 0) to server, the following entry has to be add to the `CAXConfig.xml`:

```
<Structure>
<Name>ITEM_WHITE_SPACE_LIST</Name>
<FieldCollection>
<Field><Name>1035</Name><Value>1</Value></Field>
</FieldCollection>
</Structure>
```

1035 attribute must not be read-only in this case.

However, empty values for BOM are not allowed, therefore the server reports the error: *"Error updating rows of BOM table: empty String"*

5.3 Oracle Agile PLM

5.3.1 EC Web Connector Administration

This section provides a complete summary of configuration options available for the EC Web connector.

5.3.1.1 Oracle PD Cloud Functionality

Since release 3.6.1, the MCAD connector comes with a special mode of operation for Oracle PD Cloud.

The option setting is available for MCAD administrators **only** in the **PLM Mode** drop-down list in the Preferences dialog. See User Guide for more information.

Option	Description
Oracle PLM	Normal connector behavior for Agile PLM. This is the default functionality for Agile PLM. Item and Item Change creation works normally.
CAD4Cloud	Alternative connector behavior when using Oracle PD Cloud. This mode of operation hides most Item related controls in the MCAD connector GUI and is intended to be used when using the MCAD connector to connect to PD Cloud via an Agile PLM server. Item creation in PD Cloud is then taken over by Agile PLM. When activated, this mode prevents the MCAD connector from creating and processing Items and Item Changes. Item related information are not displayed and also not transferred to or from Agile PLM.
Oracle PLM Hybrid Cloud	Works like **Oracle PLM** mode but uses PD Cloud hyperlinks for the Item related controls of the MCAD connector's GUI instead of Agile PLM hyperlinks.

The MCAD connector needs to be restarted after changing the **PLM Mode** value

The Fusion URL text field in the Preferences dialog is used to define the URL that points to the PD Cloud service. The MCAD connector uses the URL given in this text field to navigate to Item objects stored in PD Cloud in *CAD4Cloud* and *Hybrid Cloud* modes.

5.3.1.2 Filtering SubClasses and AutoNumber Generators Displayed by the MCAD GUI

The MCAD connector allows you to filter out Design, Item and Change subclasses as well as AutoNumber generators which should not be displayed in the GUI.

Configuration file: `CaxConfig.xml`

You can define filters in the MCAD connector to display only desired Design, Item or Change subclasses or AutoNumber generators. This way, certain Agile PLM object subclasses and AutoNumbers can be hidden from normal users to prevent unintended use. Additionally, you can also filter by file type (file ending). For any CAD file ending a separate set of filter criteria can be defined, if required.

These settings affect Create Object dialog, the Preferences dialog, the Save Preview window and the Save As override dialog. The subclass and number generator filtering is configured based on a `name`-and-`value` XML syntax:

- **Name:** A filter identifier optionally concatenated with the CAD file ending for the CAD files in which to apply the filter setting, for example `DisplayedDesignClassesSLDPRT` (file type specific filtering setting) or `DisplayedDesignClasses` (not file type specific setting).
- **Value:** Use `All` to not applying any filtering. Use a comma-separated list of the classes or number generators (display names) that should be displayed, for example: `Design,Construction,Prototype`

The Preferences dialog is not affected by file type specific filtering settings, except for the **Drawing Item** Subclass combo box.

File type specific filtering settings should be added to the `OverrideConfiguration` structure of the `CaxConfig.xml` . Refer to the example given below for a template on how a file type specific filtering setting might be configured.

```
<Structure>
    <Name>OverrideConfiguration</Name>
    <FieldCollection>
      <Field><Name>DisplayedDesignClasses</Name><Value>All</Value></Field>
      <Field><Name>DisplayedItemClasses</Name><Value>All</Value></Field>
      <Field><Name>DisplayedChangeClasses</Name><Value>All</Value></Field>
      <Field><Name>DisplayedDesignAutoNumbers</Name><Value>All</Value></Field>
      <Field><Name>DisplayedItemAutoNumbers</Name><Value>All</Value></Field>
      <Field><Name>DisplayedChangeAutoNumbers</Name><Value>All</Value></Field>
    </FieldCollection>
</Structure>
```

Filter Identifier	Description
DisplayedDesignClasses	List of Design classes that display in UI **Possible values:** Comma separated list of design classes (display names) or `All`.

Filter Identifier	Description
DisplayedItemClasses	List of Item classes that display in UI **Possible values:** Comma separated list of item classes (display names) or `All`.
DisplayedChangeClasses	List of Change classes that display in UI **Possible values:** Comma separated list of change classes (display names) or `All`.
DisplayedDesignAutoNumber	List of Design AutoNumbers that display in UI **Possible values:** Comma separated list of Autonumber generators (display names) or `All`.
DisplayedItemAutoNumber	List of Item AutoNumbers that display in UI **Possible values:** Comma separated list of autonumber generators (display names) or `All`.
DisplayedChangeAutoNumber	List of Change AutoNumbers that display in UI **Possible values:** Comma separated list of autonumber generators (display names) or `All`.

Example of File Ending Specific OverrideConfiguration Settings

```
<Structure>
  <Name>OverrideConfiguration</Name>
  <FieldCollection>
    <Field><Name>DisplayedDesignClasses</Name><Value>All</Value></Field>
    <Field><Name>DisplayedItemClasses</Name><Value>All</Value></Field>
    <Field><Name>DisplayedChangeClasses</Name><Value>All</Value></Field>
    <Field><Name>DisplayedDesignAutoNumbers</Name><Value>All</Value></Field>
    <Field><Name>DisplayedItemAutoNumbers</Name><Value>All</Value></Field>
    <Field><Name>DisplayedChangeAutoNumbers</Name><Value>All</Value></Field>

    <Field><Name>DisplayedDesignClassesSLDPRT</Name><Value>DesignClass1</Value></Field>
    <Field><Name>DisplayedDesignClassesSLDASM</Name><Value>DesignClass1</Value></Field>
    <Field><Name>DisplayedDesignClassesSLDDRW</Name>
      <Value> DesignClass1,DesignClass2</Value></Field>
    <Field><Name>DisplayedItemClassesSLDPRT</Name><Value>ItemClass1</Value></Field>
    <Field><Name>DisplayedItemClassesSLDASM</Name><Value>ItemClass2</Value></Field>
    <Field><Name>DisplayedItemClassesSLDDRW</Name>
      <Value>ItemClass1,ItemClass2</Value></Field>
  </FieldCollection>
</Structure>
```

5.3.1.3 Preferences Settings

The preferences are stored in Agile PLM in a Design object called **MCAD-CONFIG-SOLIDWORKS**

Only users with administrative rights and roles can update the template `Preferences.xml`.

The `Preferences.xml` is stored locally first and only an administrator can change the template and upload it to Agile PLM. This is done using the **Save** button in the Preferences dialog. You can also reset the template manually by checking out the MCAD-CONFIG-SOLIDWORKS Design object in Web Client, adding your local `Preferences.xml` to the files tab, and checking the object back in after upload. The next time a normal user logs in, the new template is downloaded. In case the system cannot generate the MCAD-CONFIG-SOLIDWORKS Design object automatically, create it with this name in the Agile PLM system.

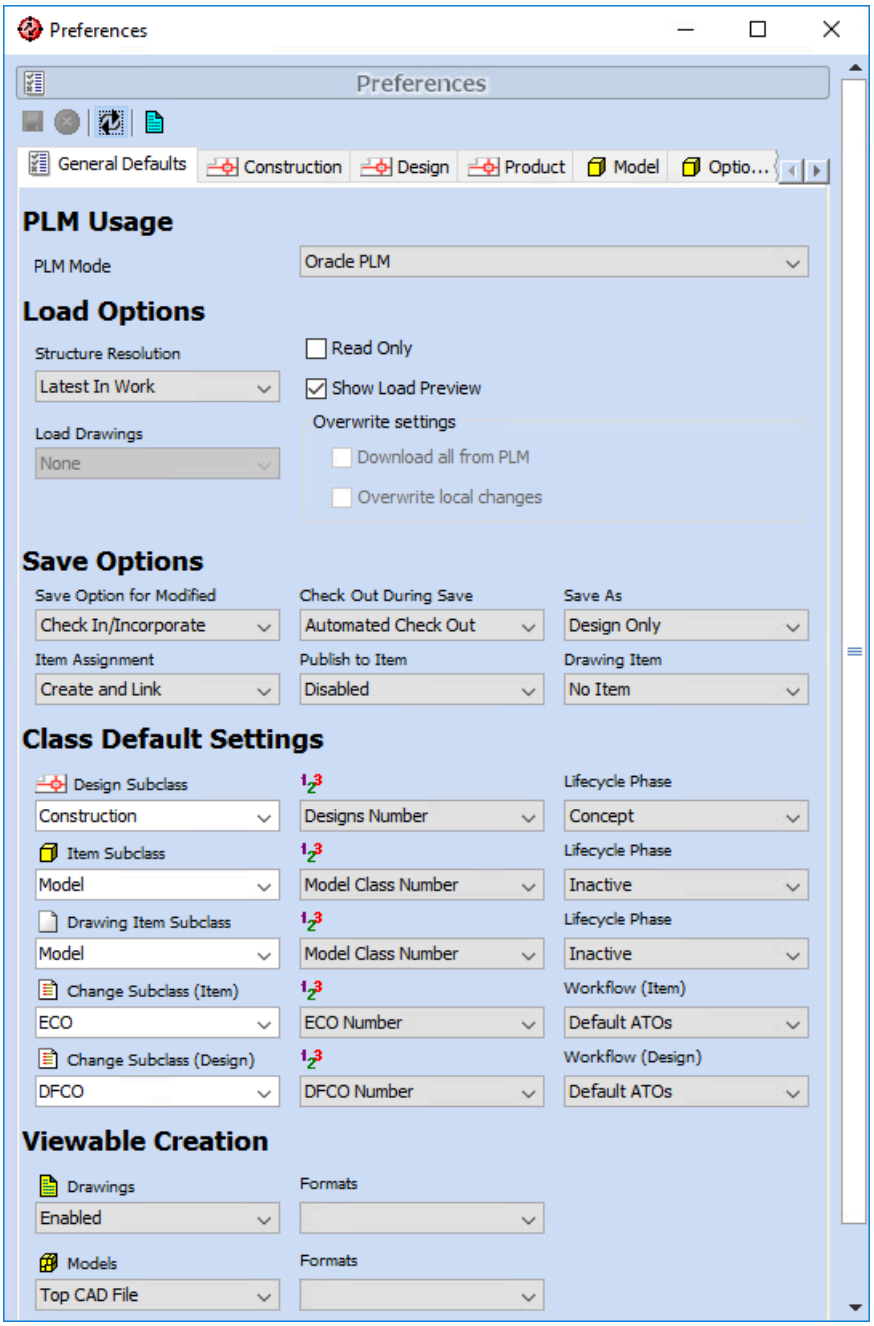
To lock an entry from user modification, you need to edit the `Preferences.xml`. The template in Agile PLM must also be replaced manually. Search for a section called `GeneralDefaults`, which contains several `FieldCollections`. Each collection describes one default. There are 3 fields with `name` / `value` pairs for each default. See proceeding table for optional settings:

Name	Value
CAX_NAME	Internal setting name (for instance, DesignClass)
Default	The default setting as a string
Editable	Determines whether the preference is editable or not **Possible values: true/false** If set to `false`, even the administrator gets a setting, that is not editable anymore.

Preferences Dialog

The Preferences dialog can be accessed through

- the integration menu
- the toolbar menu button on Save and Load previews
- expanding the left side panel of the save and load preview
- expanding the Preferences pane of the Create Object dialog



The Preferences dialog has the following sections;

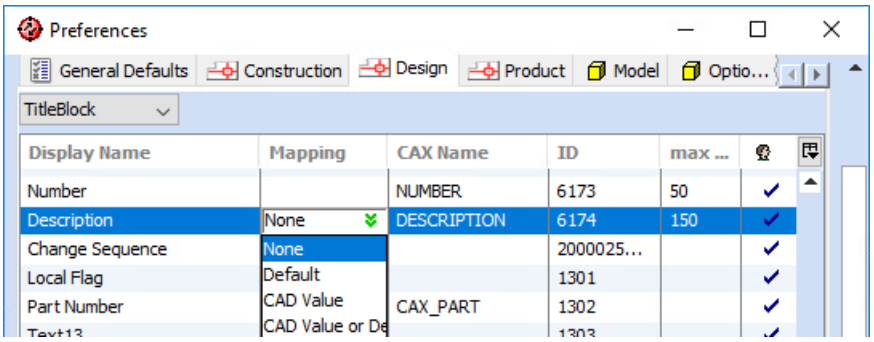
- **PLM Usage** - The PLM Mode drop-down list defines the mode of operation for the MCAD connector.
- **Load Options** - define the behaviour of the connector during load operations.
- **Save Options** - define the behavior during save operations.
- **Class Default Settings** - defines the default sub-classes and default AutoNumber sources for all Items, Designs and Change orders created by SOLIDWORKS integration
- **Viewable Creation** - defines the types of viewable files that are automatically created and attached in Agile PLM along with the native file. See Viewable Files Management (p. 60)
- **Property Value Preferences** - define the properties that are mapped between SOLIDWORKS and Agile PLM, as part of the save process. See Property Value Preferences (p. 44)

See User Guide for more information.

5.3.1.4 Property Value Preferences

Each Design and Item class is represented in the preferences to configure the mapping of symbolic CAX properties to fields in Agile PLM. You can set up the mapping interactively.

The preferences are saved into a **MCAD-CONFIG-SOLIDWORKS** File Folder object in Agile PLM if the current user is a member of the admin group. The values must be set in each subclass independently.



Additionally, each field may get a value default mapping.

The **Property Value Preferences** section of the Preferences dialog allows you to pre-define the properties that are mapped between SOLIDWORKS and Agile PLM, as part of the save process. By setting these preferences appropriately, you can reduce the use of the interactive save dialog and speed up the save process. The four mapping options are:

- **None** – No value is to be set for this property.
- **Default** – Use the value in the Default column.
- **CAD Value** – Use the value defined in the SOLIDWORKS properties, based upon the mapping defined by your administrator.
- **CAD Value or Default** – Use the value defined in the SOLIDWORKS properties, but if no value exists then use the default value in the Default column.

5.3.1.5 Customizing Quick View Attributes

The MCAD connector supports customizing the attributes, which are displayed in the Quick View dialog window.

Configuration file: `CAXConfig.xml`

To change the appearance, add the following `QuickViewDisplay` structure to `CaxConfig.xml`:

```
<Structure>
 <Name>QuickViewDisplay</Name>
 <FieldCollection>
  <Field><Name>CAX_FIL_NAME</Name><Value>1</Value></Field>
  <Field><Name>COMPONENTTYPE</Name><Value>1</Value></Field>
  <Field><Name>NUMBER</Name><Value>1</Value></Field>
  <Field><Name>REV</Name><Value>1</Value></Field>
  <Field><Name>REVISION</Name><Value>1</Value></Field>
  <Field><Name>DESCRIPTION</Name><Value>1</Value></Field>
  <Field><Name>LABEL</Name><Value>1</Value></Field>
  <Field><Name>LIFECYCLEPHASE</Name><Value>1</Value></Field>
  <Field><Name>CHECKOUTUSER</Name><Value>1</Value></Field>
  <Field><Name>Item.NUMBER</Name><Value>1</Value></Field>
  <Field><Name>Item.REV</Name><Value>1</Value></Field>
  <Field><Name>Item.ECO</Name><Value>1</Value></Field>
  <Field><Name>Item.LIFECYCLEPHASE</Name><Value>1</Value></Field>
  <Field><Name>Item.DESCRIPTION</Name><Value>1</Value></Field>
  <Field><Name>CAX_MODEL_TYPE</Name><Value>1</Value></Field>
  <Field><Name>CAX_MODEL_REF</Name><Value>1</Value></Field>
  <Field><Name>CAX_LINK_TYPE</Name><Value>1</Value></Field>
  <Field><Name>CAX_LINK_REF</Name><Value>1</Value></Field>
  <Field><Name>CAX_TYPE</Name><Value>1</Value></Field>
 </FieldCollection>
</Structure>
```

Additional attributes can be added to the structure as required. Make sure to assign the value `1` to each XML `Field`.

The Quick View dialog only displays attributes, which are set to visible in the class configuration of the Preferences dialog.

5.3.1.6 Re-reading Design Attributes

The MCAD connector supports re-reading a predefined list of design attributes during check out.

Attributes that have been updated in Agile PLM during or before check out are thus correctly displayed.

Only Design attributes (**Title Block and Page 2**) are supported.

If used in conjunction with a PX: The event subscriber used needs to have the following values selected:

- Trigger Type = Pre Execution
- Mode = Synchronous

The attributes need to be listed in the `CheckOutAttributes` Transaction in `%xPlmRootDir%\OraclePLM\jar\agile9\a934ws_proxy.jar..\com\xplm\agile93\ws\cad\xml\Agile93Templates.xml`, see example below.

```
<Transaction>
  <!-- Please extend the list only, do not remove the provided fields -->
 <Aliasname>CheckOutAttributes</Aliasname>
 <Import>
  <Parameter>
   <FieldCollection>
                              <!-- snip -->
    <Field>
     <Name>6388</Name>
     <Value></Value>
    </Field>
    <!--Description-->
    <Field>
     <Name>6174</Name>
     <Value></Value>
    </Field>
    <!--Field>
     <Name>6178</Name>
     <Value></Value>
    </Field-->
   </FieldCollection>
  </Parameter>
 </Import>
</Transaction>
```

5.3.2 Language and Localization Administration

This section provides information about how to setup the GUI languages for the integration.

Languages are set up on three different components:

- The Agile PLM user language
- EC Web Components
- The CAD connector

All are independent from each other

5.3.2.1 Agile PLM User and Data Language

The setup is done in the user preferences settings in Agile PLM.

The preferred user language controlled the data values which are displayed in EC dialogs and transferred between SOLIDWORKS and Agile PLM.

5.3.2.2 EC Web Components

The EC Web Component dialogs are localized.

- **Configuration file:** `acx.bat`
- **Valid Values:** `EN, FR, DE`

The desired language on runtime is defined using a switch in `(components)\com\acx.bat`.

```
:DEFAULT_AGILE_START

set CAX_JAVA_OPTIONS=%JAVA_HEAP_SIZE% -Dcom.xplm.agile.Language=EN -Dlog4j.config.....
```

Use the disconnect session command or restart the SOLIDWORKS Tool after changing the setting.

5.3.2.3 CAD Connector Components

The language of the Add-In in SOLIDWORKS can be configured in the following way:

- **Configuration file:** `XPLMConnector.xml`
- **Parameter:** `Language`
- **Valid Values:** `EN, FR, DE, CN`

Example:

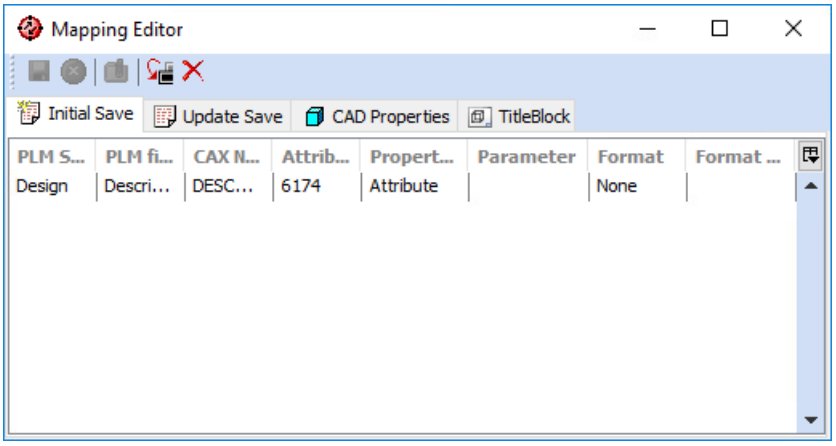
```
<Language>EN</Language>
```

5.3.3 Mapping

5.3.3.1 Mapping Editor

This section provides a complete summary of the options that are available in the Mapping Editor.

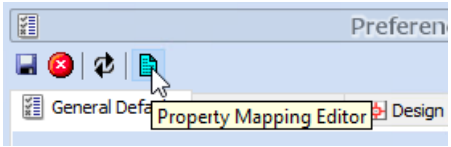
The Mapping Editor is used to define mappings of SOLIDWORKS properties to Agile PLM fields during save. It is used for mapping of Agile PLM values to SOLIDWORKS properties or the drawing title block as well.



Using the Mapping Editor

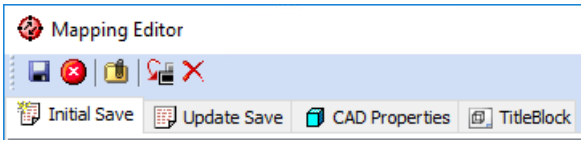
The Mapping editor is opened by pressing the **Property Mapping Editor** icon (blue page button) on the Preferences dialog.

The button is only visible to user with Administrative roles and privileges in Agile PLM.



Mapping Editor Toolbar

The toolbar of the Mapping Editor has the following functionality:



Command	Description
Save	Saves the mapping definitions The local save doesn't make the mapping available to all users and is lost if you restart the integration.
Upload to PLM	Save and upload the mapping definition to Agile PLM and make it available to all users. All other client machines must re-login using the **Disconnect Session** command to download the updated mapping.
Cancel	Cancel all changes to the mappings and reread the latest saved mapping.
Insert Mapping	Add a row into the current active mapping tab
Delete Mapping	Remove a selected row in the current active mapping tab

Mapping SOLIDWORKS properties to Agile PLM fields

The **Initial Save** and **Update Save** tabs define the mappings of SOLIDWORKS properties to and from Agile PLM fields.

For SOLIDWORKS objects that are not known in Agile PLM, the mapping in the **Initial Save** tab is used. The **Update Save** tab is used for SOLIDWORKS objects that already have an assigned Agile PLM object.

Both sections are configured the same way but may contain different settings. For instance, on **Initial Save** the predefinition of the Design number or the assigned Item number is important. On **Update Save** tab, there is only the need to map attributes like dimensions or descriptions.

The following columns are available:

Column	Description
PLM SuperClass	Switches between the target Agile PLM superclass objects; Design or Item
PLM Field	Depending on the selected PLM superclass object, the available Agile PLM fields are filtered from the current class configuration. Only visible and editable fields in Agile PLM are available.

Column	Description
CAX Name	Displays additional information about the PLM Field selected. This column is read only in the initial and update save tabs.
Attribute ID	Displays additional information about the PLM Field selected. This column is read only
Property Type	This column lets you select from SOLIDWORKS internal integration parameters and from SOLIDWORKS properties. If you set the value to **Attribute**, you must specify a SOLIDWORKS property name in the Parameter column.
Parameter	You can specify a SOLIDWORKS property name in this column. A parameter with the given name is searched in the configuration specific properties first. If there is no configuration specific property with that name, the standard or custom properties of the file are scanned.
Format	Provide basic formatting options for values mapped. See Formatting values during mapping (p. 51) for more details.
Format Parameter	Provide basic formatting options for values mapped See Formatting values during mapping (p. 51) for more details.

The available property types are CAD dependent and listed in the table below.

Property Type	Purpose
ModelFullName	File name including path
ModelName	File name with extension but without path
ModelExtension	File extension
ModelType	File type, equal to the CAD file extension in most CAD systems
ModelPath	File path location
ModelNameOrConfiguration	File name or configuration name
ModelNameAndConfiguration	File name plus configuration name
ObjectName	File name without path and extension
ObjectName.Type	File name without path
CreatingSystem	CAD version the file is created with
Attribute	Retrieve the CAD Property defined in the Parameter column
String	Set the string defined in the Parameter column
Code	Execute the CAD callback code defined in the Parameter column
$USER	Set the current login user name as value

Property Type	Purpose
$USERID	Set the current login user ID as value
Configuration	Configuration name
ModelConfigurationNames	Contained configuration names
ModelStamp	Internal timestamps of the file
ModelFamilyType	Part family type or configuration type
ModelFamily	Part family or configuration master or generic
ModelLinkType	Linked references type
ModelLinks	Linked source file
HelperPartIdent	Helper part property

Where updates in the SOLIDWORKS connector are written into both SOLIDWORKS and Agile PLM, we recommend mapping attributes in the **Initial Save**, **Update Save** and **CAD Properties**. If the names of the attributes are changed in Agile PLM, you must manually recreate the mappings in the Mapping editor.

Mapping Agile PLM values to SOLIDWORKS Properties

The mapping of Agile PLM values back into SOLIDWORKS properties is defined in **CAD Properties** and **TitleBlock** tabs.

The **CAD Properties** tab defines the mappings to SOLIDWORKS properties. Some CAD systems support special logic for drawing title blocks, especially if the displayed texts in the drawing cannot be linked to CAD properties. For this use case the second **TitleBlock** tab is used by some CAD tools.

The following columns are available:

Column	Description
Property Name	The name of the target SOLIDWORKS property. This could be a SOLIDWORKS file attribute or SOLIDWORKS property or configuration-specific property
PLM SuperClass	Switches between the target Agile PLM superclass objects; Design or Item
PLM Field	Depending on the selected PLM superclass object, the available Agile PLM fields are filtered from the current class configuration. Only visible and editable fields in Agile PLM are available.
CAX Name	Set with the default symbolic name or the attribute ID if no such symbolic name exists. **CAX Name** column is editable to support editing complex legacy logic for data extraction in drawing title blocks.
Format	Provide basic formatting options for values mapped See Formatting values during mapping (p. 51) for more details.
Format Parameter	Provide basic formatting options for values mapped See Formatting values during mapping (p. 51) for more details.

Formatting values during mapping

The **Format** and **Format Parameter** columns provide basic formatting options for values mapped between SOLIDWORKS and Agile PLM and vice versa.

Valid format options are listed in the table below. The date formats work only if the value to be formatted is given in an integer value.

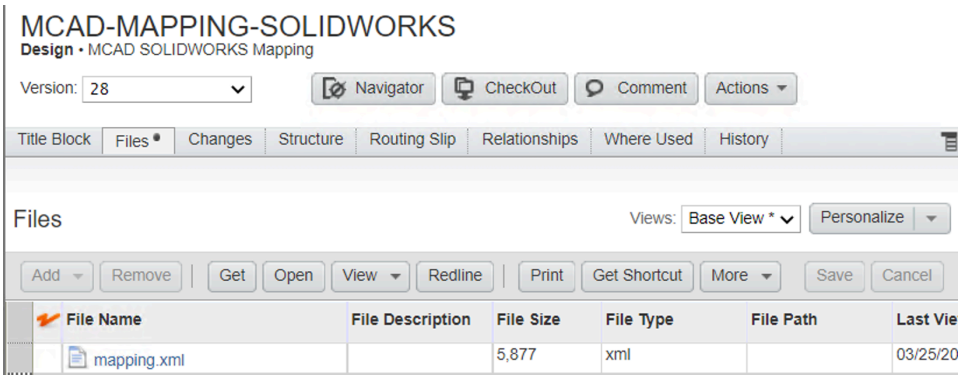
Format	Description
None	No formatting action is executed
toLowerCase	Convert the value to lower case characters
toUpperCase	Convert the value to uppercase characters
subString	Cut a sub string from the value with the start and end index defined in the Format Parameter column. Valid values are for example 0-end, 3-end, 0-50.
Prefix	Append the prefix defined in the Format Parameter column in front of the value
Suffix	Append the suffix defined in the Format Parameter column at the end of the value
Code	Execute the CAD callback code defined in the Format Parameter column to format the value
DateFormat DD.MM.YYYY HH:MI:SS	Format the Date like 15.12.2010 23:30:00
DateFormat DD.MM.YYYY	Format the Date like 15.12.2010
DateFormat DD.MM.YY	Format the Date like 15.12.10
DateFormat DD-MM-YY	Format the Date like 15-12-10
DateFormat MM/DD/YY	Format the Date like 12/15/10
DateFormat DD-MMM-YY	Format the Date like 15-Dec-2010

See DateFormats Section (p. 94) for more details on how to manually modify the date format.

5.3.3.2 MCAD-MAPPING folders

Describes how mapping is handled.

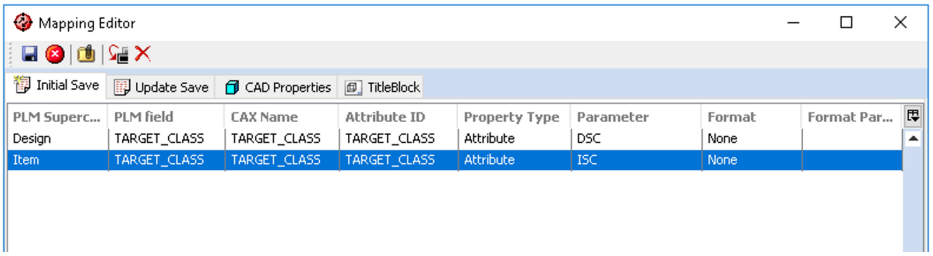
The system creates a design in Agile PLM called `MCAD-MAPPING-SOLIDWORKS` and attaches the mapping to that Design. On the next login of any user with the same SOLIDWORKS system, the mapping is downloaded automatically and used on the client machine.



5.3.3.3 Mapping the Sub-Class for New Objects to Agile PLM (TARGET_CLASS Attributes)

The MCAD connector provides a feature that allows you to define the class assignment of an object through mapping.

When creating objects in Agile PLM, the class assignment of an object is usually taken from the corresponding pre-set value in the Preferences dialog. However, there is also a way to define the class assignment of an object through mapping: The TARGET_CLASS attributes listed in the **PLM field** columns of the Mapping Editor control the sub-class that is used when creating an object through the MCAD connector. An example is given below.



The TARGET_CLASS attributes should only be used during object creation, means in the **Initial Save** tab of the Mapping Editor.

If TARGET_CLASS mapping is not set or CAD file property defining the sub-class is empty (=null string) the default class from the Preferences dialog is used.

5.3.3.4 Mapping Empty Values to and from Agile PLM Fields

Since version 3.6.1, the MCAD connector allows mapping empty values to and from Agile PLM attributes.

Configuration file: `CaxConfig.xml`

Every attribute that should be allowed to contain empty values needs to be added to one of the following white-lists:

FF_WHITE_SPACE_LIST	For objects of the **File Folders** base-class or any sub-class thereof.
ITEM_WHITE_SPACE_LIST	For objects of the **Items** base-class or any sub-class thereof.
CHANGE_WHITE_SPACE_LIST	For objects of the **Changes** base-class or any sub-class thereof.

Each of the white-list represents an XML `Structure` object containing a list of `Fields`.

- The value `1` indicates that empty values are allowed for the corresponding Agile PLM attribute
- The value `0` means that empty values are not allowed.

If an attribute is not listed in the white list, the MCAD connector implicitly assumes that empty values are not allowed as well.

The `Name` tag of the XML `Field` must contain the **Base ID** of the Agile PLM attribute in question.

Example:

```
<Structure>
    <Name>FF_WHITE_SPACE_LIST</Name>
    <FieldCollection>
        <Field><Name>6174</Name><Value>1</Value></Field> <!-- empty values allowed -->
        <Field><Name>1303</Name><Value>0</Value></Field> <!-- empty values not allowed
 -->
    </FieldCollection>
</Structure>
```

Related links
CAXConfig.xml Settings (p. 84)

5.3.3.5 Mapping Values on Save As

In MCAD release 3.5.0.0, the functionality for customizing attribute assignments in the MCAD connector has been removed from the product (Preferences dialog, sub-class tabs).

This was done to prevent incompatibilities to the default MCAD data model which could be introduced by improper configurations. However, the attribute customization also allowed customers to extend the data model of MCAD to their needs, for example, by configuring an attribute to which MCAD would write the source object of a Save As process. Since MCAD 3.5.0.0, this functionality is not available any longer.

To restore the previously available functionality regarding the Save As source object, a new mapping functionality was introduced to the MCAD connector.

Configuration file: `CaxConfig.xml`

This mapping functionality allows users to define attributes to which the source objects of a Save As process should be written during the execution of that process.

```
<Structure>
  <Name>CAX_NAMES_BY_ID</Name>
    <FieldCollection>
      <Field><Name>ObjectType.AttributeID</Name><Value>CAX_COPY_FROM</Value></Field>
      <Field><Name>ObjectType.AttributeID</Name><Value>CAX_COPY_FROM</Value></Field>
    </FieldCollection>
</Structure>
```

`ObjectType` is the type of object to which the value should be written. This can be `FF` (FileFolder or Design) or `Item` (any kind of Item). `AttributeID` is the attribute ID of the target PLM attribute. The actual function value `CAX_COPY_FROM` defines the source object of an MCAD Save As process. For Design objects this is the Design number, for Items the Item number.

Example configuration:

```
<Structure>
  <Name>CAX_NAMES_BY_ID</Name>
    <FieldCollection>
      <Field><Name>FF.2471842</Name><Value>CAX_COPY_FROM</Value></Field>
      <Field><Name>Item.1301</Name><Value>CAX_COPY_FROM</Value></Field>
    </FieldCollection>
</Structure>
```

List of automatically handled attributes on Save As

The following attributes are handled automatically on Save As and should not be mapped:

Title block tab, Page 1 and Page 2:

- Design number
- Revision
- Version
- Check out status
- CAD filename
- Design system
- File type
- Part number

Relationship tab:

- Link type

Structure tab:

- Identifier
- Component
- Revision
- Version
- Quantity

5.3.3.6 Mapping read only Item BOM attributes

Since release 3.6.4.3, the MCAD connector allows mapping read only Item BOM attributes.

Configuration file: `CaxConfig.xml`

The new structure `READONLY_BOM_FIELDS` is introduced. It is absent by default and must be manually added. This XML structure defines all Item BOM fields which should not be updated by the MCAD connector (read-only fields), defined by their field ID.

```
<Structure>
  <Name>READONLY_BOM_FIELDS</Name>
  <!-- Values are not used -->
  <FieldCollection>
    <Field><Name></Name><Value></Value></Field>
  </FieldCollection>
</Structure>
```

Only attribute ID's may be used.

Though the XML definition is based on a `name-and-value` syntax, the `value` field itself is not evaluated by the connector.

The `Name` tag of the XML `Field` must contain the **Base ID** of the Agile PLM attribute in question.

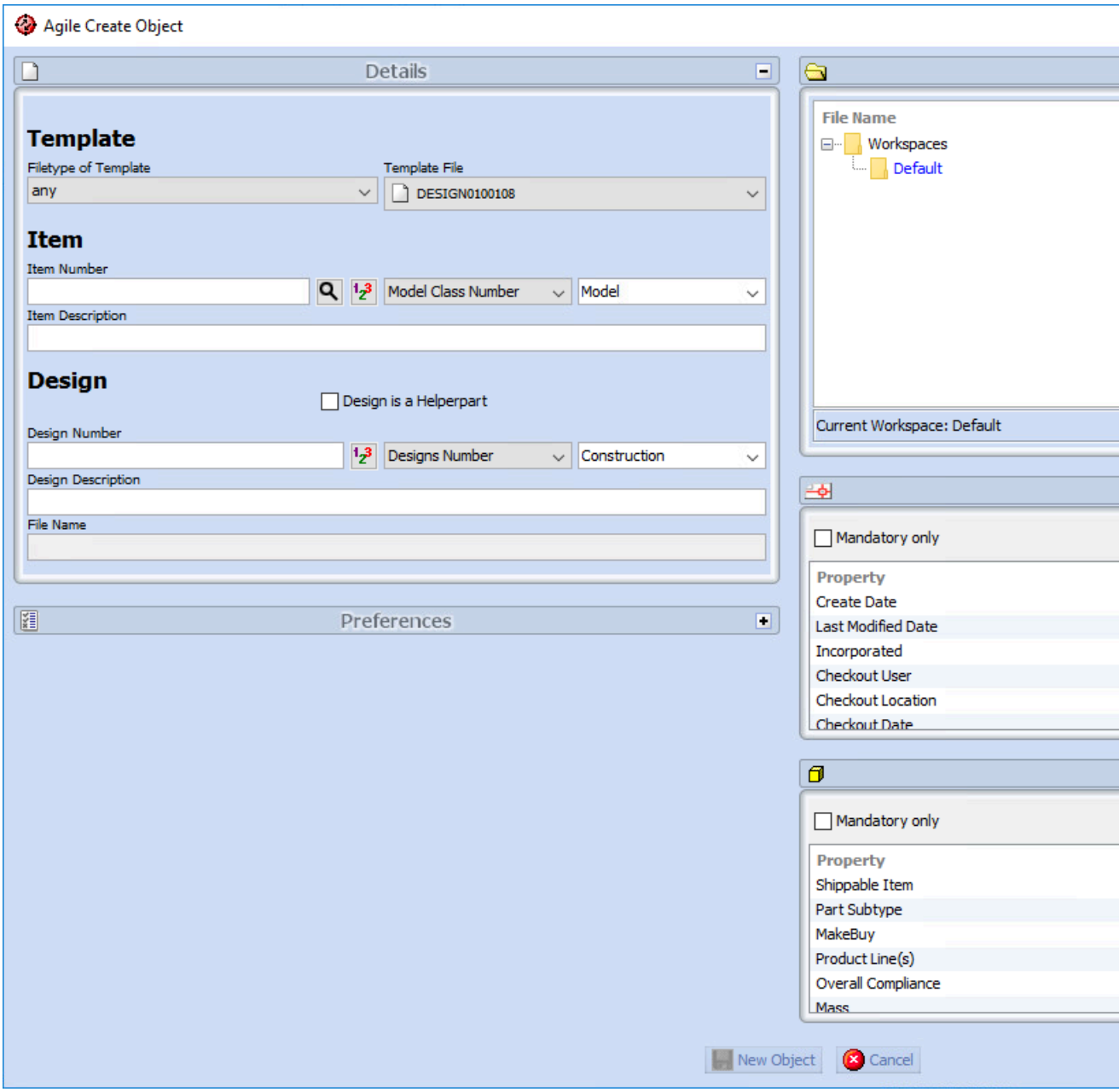
The IDs or CAX names listed in the structure will be filtered out by BOM CUO actions.

Related links
CAXConfig.xml Settings (p. 84)

5.3.4 Template Management

This section provides detailed information about all necessary steps to enable object creation using templates saved in Agile PLM.

The Create Object dialog is used create new objects simultaneously in SOLIDWORKS and Agile PLM using template files stored in Agile PLM. The dialog is displayed by the **Agile** > **Create** command within the integration menu.



See User Guide for more information.

5.3.4.1 Template Structures in Agile PLM

The Create Object dialog uses a certain data structure in the Agile PLM as the basis of object creation.

All templates that should be available to the user must be stored in that structure. The structure must be named `SITE-START-SOLIDWORKS` where `SITE` is a variable that can be defined in the `CAXConfig.xml` file, its default value is `MCAD` if it is not defined in that file.

It is possible to store template structures of more than one CAD system in Agile PLM. If the `SITE` variable is assigned for each client, subdivisions of the same CAD can use templates in the same PLM without interfering each other. For example `LONDON-START-SOLIDWORKS` and `ROME-START-SOLIDWORKS` may exist in the same Agile PLM for SOLIDWORKS clients with the different `SITE` variables `LONDON` and `ROME` where the `LONDON` division can only use the structure `LONDON-START-SOLIDWORKS` and vice versa.

The `SITE` variable is assigned by editing the file `%xPlmRootDir%\CAXConfig.xml`. To set the value, after the beginning of the first occurrence of the line `FieldCollection` edit the following line:

```
<Field><Name>SITE</Name><Value>Custom_site_value</Value></Field>
```

`Custom_site_value` is the string that defines the site, here you can enter any string desired, but it must not contain spaces. Save the `CAXConfig.xml` after finishing the edit. These steps must be repeated for each client that should belong to a site.

5.3.4.2 Create a Template Structure in Agile PLM

You must create valid template structures in Agile PLM to use the Create New object feature.

About this task

This task describes how to to create a Template Structure in Agile PLM.

You must have the appropriate rights and roles to create new Design Objects in Agile PLM.

Procedure

1. Login to the Agile PLM Web Client .
2. Click on the **Create New** drop-down list and select **File Folders** > **Designs**.
   ➔ A pop-up window opens
3. On the pop-up window, select the type **Design** and enter the appropriate name of the template structure in the **Number** field for example `MCAD-START-SOLIDWORKS`.
4. Click **Save**.

Result

A Template structure has been created in Agile PLM and can be loaded with SOLIDWORKS files.

5.3.4.3 Disable Autonumber Use for New Design Objects

To use create template structures in Agile PLM, you must switch off forced AutoNumber usage when creating new Design objects in Agile PLM.

Before you start

You must have access Java Client access and privileges associated with an Administrator type role.

About this task

This task describes how to disable forced AutoNumber use when creating new Design objects in Agile PLM.

Procedure

1. Log in to Agile PLM Java Client.
2. In the tab pane on the upper left-hand side, choose the **Admin** tab.
3. In the feature tree within that tab, open the node **Data Settings**.
4. Double click on the sub-node **Classes**.
   ➔ The **Classes** window is opened.

5. In the **Classes** window, select and double click on the subclass **Design**.
   ➔ **Subclass:Design** window is opened.
6. In the **General Information** tab set the option **AutoNumber Required** to **No**.
7. Click **Save**.

Result

Autonumber use when creating new Design objects has been disabled in Agile PLM.

Next steps

Either return to the Agile PLM Web Client to create the new Design **MCAD-START-SOLIDWORKS** or create it from the Java Client.

Once the Design object is created. Repeat step 6 and 7 but set the value back to **Yes** and then close the Java Client.

5.3.4.4 Adding Template Files to the Structure

A template file is a normal SOLIDWORKS file that is made available in a template structure in Agile PLM. When you use the Create Object frame for object creation, the selected template file is downloaded to your workspace, renamed and opened in SOLIDWORKS. Due to this, template files can be created in the same way as any other SOLIDWORKS file.

Before you start

You must first create the Design template structure for example, **MCAD-START-SOLIDWORKS** in Agile PLM.

Create a SOLIDWORKS file that will be used as the template and save it to your local workspace.

About this task

This use case describes how to add a template file to the template structure in Agile PLM.

Procedure

1. On Agile PLM Web Client, navigate to your template structure.
   ➔ Agile PLM Web Client displays the design object of the template structure.
2. Click **CheckOut**
   ➔ The Design object of the template structure has been checked out.
3. Go to the **Structure** tab of the Design Object and Click **Add**.
   ➔ A text field opens.
4. In the text filed, click on the button **Create to add**.
   ➔ A pop-up window opens.
5. In the pop-up window select a **Type** and enter a **Number**. Optionally, you can also add a **Description**
6. Click **Add**.
   ➔ A new Design object for a template is created.
7. Click on the Number of the newly created Design object to open it.
   ➔ Agile PLM Web Client displays the newly created Design object.

8. Go to the **Files** tab of the created template file object and upload a SOLIDWORKS file.

   a) Click **Add**.

      ➔ The **File Upload Selector** pop-up opens.

   b) Click **Browse for Files**.

      ➔ Windows File explorer opens.

   c) Search and select a SOLIDWORKS file for upload and click **Open**.

      ➔ The selected file is added to the **File Upload Selector** window.

   d) Click **Upload**.

      ➔ The SOLIDWORKS file is attached to the **Files** tab of the newly created template file object. That SOLIDWORKS file becomes the template file.

      The template file object is a Design object in the template structure, not the structure itself which is a Design object itself!

   e) Select the **Title Block** tab and enter at least the properties **Design System** and **Filetype**. All other properties can optionally be left empty.

      - The **Design System** is the name of the CAD from which the template file was created
      - **Filetype** is the file ending of the template file in uppercase (without a dot).

   f) **Optionally:** Fill in the **Subtype** to assign a subtype to the template file.

   g) Save the changes made to that template object.

Result

The template structure is operational and template creation is possible from the Create Object dialog.

Related links
Template Structures in Agile PLM (p. 56)
Subtypes (p. 59)

5.3.4.5 Subtypes

A subtype is a SOLIDWORKS file type that extends, or in other words, specializes another SOLIDWORKS file type.

Every subtype file has the same file ending as its supertype and acts as a normal SOLIDWORKS file. However, in Agile PLM a subtype behaves slightly different. Every subtype is defined by the string that is entered in the **Subtype** property in the **Title Block** tab of a Design object. If a Design should not be assigned to a subtype, the property field can be left empty.

A valid subtype entry must consist of at least two characters (spaces do not count as characters in this case). If only one character is entered in that property field, it is treated as if it was empty.

To create a new subtype, you need to assign a string that was not already used by a Design object in the template structure. It is not necessary to create a new Design object when creating a new subtype. Changing, or deleting the subtype of an already existing Design is also possible. To do so, you need to change or delete the entry of the field **Subtype** in the Design's **Title Block**. Likewise, the subtype of a Design object can be changed to an already existing subtype by simply changing the **Subtype** property. It is possible to assign the same subtype to template objects of different file types as well.

Every subtype that is used in the template structure gets its separate entry in the **Filetype of Template** combo box in the Create Object dialog. The entry is displayed the same way as the entry for the supertype with the name of the subtype appended. If a subtype entry is selected, only the template files

that belong to the selected file type **and** the selected subtype are displayed in the **Template File** combo box. However, if the entry for the supertype is selected, all template files of the selected file type are displayed, no matter to which subtype they belong.

The entry for a supertype in the **Filetype of Template** combo box does only appear if there is at least one template file of the corresponding file type that does not belong to a subtype.

Subtypes enable you to separate template files from each other according to certain criteria. They may be used to provide templates of the same file type to the user in a structured way, for example separated by projects, locations, names of clients and so on. In fact, every property that can be expressed as a string could be used as a subtype making a finely graded template classification possible.

5.3.4.6 Structure Resolution

The Create functionality of MCAD does not perform any kind of structure resolution for the elements of the template structure.

Therefore, when a template is downloaded, only the actual SOLIDWORKS file selected in the Create Object dialog is downloaded and opened in the SOLIDWORKS regardless of its structure.

5.3.5 Viewable Files Management

Viewable files are 2D or 3D interchange file formats that are automatically created and attached in Agile PLM along with the native SOLIDWORKS file.

5.3.5.1 How to Configure Viewable Formats Creation

You can define the viewable file formats available in the Preferences dialog and hence the viewable formats created by the MCAD connector.

Configuration file: `CAXConfig.xml`

Structure: Viewables

The supported viewable types are defined in the `Viewables` structure in a semi-colon separated list:

```
<Structure>
   <Name>Viewables</Name>
      <FieldCollection>
         <Field><Name>ViewablesDrawing</Name><Value>PDF;TIF;CGM</Value></Field>
         <Field><Name>ViewablesModel</Name><Value>CGR;WRL;STEP;IGES;3DXML;JT;PDF</
Value></Field>
      </FieldCollection>
</Structure>
```

Setting	Description
ViewablesDrawing	Defines viewable file types for drawing files visible in the Preferences dialog. **Possible values:** List of viewable file endings (use semi-colons as list separator).

Setting	Description
ViewablesModel	Defines viewable file types for 3D models files visible in the Preferences dialog. **Possible values:** List of viewable file endings (use semi-colons as list separator).

The supported formats include:

Drawing Formats	Model Formats
■ PDF ■ EDRW ■ TIF ■ DXF	■ STEP ■ X_T ■ IGES ■ EPRT ■ EASM

The MCAD connector uses the SOLIDWORKS's PDF generation tool by default. So any settings to control what ends up in the PDF should be described in the SOLIDWORKS tool's documentation.

Related links
Viewables Section (p. 96)

5.3.5.2 Viewable File Naming

You can configure the file names that should be used for viewable files created by the MCAD connector.

Configuration file: `CAXConfig.xml`

Structure: FileNaming

Depending on the file extension different rules are possible. However, for each file extension only one rule is supported. The Field `Name` represents the file ending of the viewable file while the Field `Value` represents the actual naming rule.

```
<Structure>
 <Name>FileNaming</Name>
 <FieldCollection>
  <Field><Name>PDF</Name><Value>%Item.NUMBER%_%Item.REV%.PDF</Value></Field>
  <Field><Name>STP</Name><Value>%Item.NUMBER%_%Item.REV%.STP</Value></Field>
  <!--Field><Name>JT</Name><Value>%NUMBER%_%REV%.JT</Value></Field-->
  <Field><Name>PNG</Name><Value>thumbnail.png</Value></Field>
 </FieldCollection>
</Structure>
```

Some CAD systems use the file ending .STEP/IGES instead of .STP/IGS for STEP/IGES viewables. In this case, the rule defined for STP/IGS is not applied for viewable files with the file ending .STEP/IGES. Therefore, you must also provide a naming rule for .STEP/IGES files.

The file naming rule is parsed during the file transfer process and can contain values from the Design object or from the Item object:

- **Design Values:** `%NUMBER%_%REV%.JT` - combines Design Number and Version.

- **Item Values:** `%Item.NUMBER%_%Item.REV%.JT` - combines Item Number and Revision.
- **Fix Values:** `thumbnail.png` - does not contain values from the the Design or Item objects.

Since version 3.4, it is possible to remove file endings from any valid Agile PLM data set using the `_ROOT` extension. Appending the `_ROOT` flag to an attribute removes its CAD file ending. For example:

Definition:

```
<Field><Name>PDF</Name><Value>%NUMBER%_%Item.Number%.PDF</Value></Field>
```

- **Resulting filename:** *DESIGN123456.SLDPRT_PART1234.PDF*

Definition:

```
<Field><Name>PDF</Name><Value>%NUMBER_ROOT%_%Item.Number%.PDF</Value></Field>
```

- **Resulting filename:** *DESIGN123456_PART1234.PDF*

If one of the values is missing or commented out, the standard viewable naming rule is applied. The rule uses the file name of the SOLIDWORKS file, from which the viewable file originated and the revision.

```
<!-- Field><Name>PDF</Name><Value>%Item.NUMBER%_%Item.REV%.PDF</Value></Field -- >
```

Renaming the JT viewables can cause problems. Therefore, it is commented out by default.

When you have Designs without Item objects, we recommend that you use `%NUMBER_ROOT%_%REVISON%.xxx` to ensure that you have an understandable file name. In this case, the Design must have a revision value.

Sheet Number Handling

The connector is also able to handle creating multiple viewables for the same model for example multi-sheet drawings. where each created file gets a unique filename based on the sheet number.

Example for DXF viewables for a multi-sheet drawing where multiple DXF files are generated:

```
<Field><Name>DXF</Name><Value>%Item.NUMBER%_%Item.REV%_%SHEET_NO%.DXF</Value></Field>
```

Related links
FileNaming Section (p. 102)

5.3.5.3 Model Viewable Combo Box Options

Since release 3.6.4, only the options **Top CAD file** and **Disabled** are available by default for the Models combo box in the Viewable Creation section of the Preferences dialog.

The options **All CAD Files**, **All Parts** and **All Assemblies** were removed but can be activated if desired.

Configuration file: `CAXConfig.xml`

Structure: BrowserDisplay

You can configure the valid options by adding the following `Field` to the `BrowserDisplay` structure:

```
<Field><Name>ModelViewableOptions</Name><Value>TOP</Value></Field> <!-- values: TOP,
 ALL, ASSEMBLIES, PARTS -->
```

You can also add more than one option by using semi colons to separate the list:

```
<Field><Name>ModelViewableOptions</Name><Value>TOP;PARTS;ASSEMBLIES</Value></Field>
```

The Field `Value` represents the actual option to be added. Valid options include:

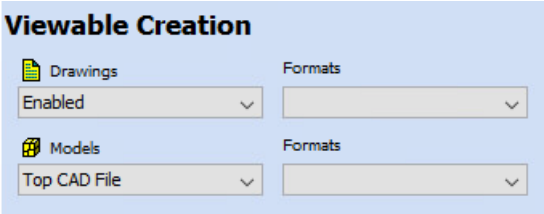
- TOP - for **Top CAD file**
- ALL - for **All CAD Files**
- ASSEMBLIES - for **All Assemblies**
- PARTS - for **All Parts**

5.3.5.4 Activate Viewable Creation on the Preferences Dialog

The types of viewable files that are automatically created are controlled in the Preferences dialog **Viewable Creation** section.

You can activate viewable creation for 2D and 3D files. This can be set independently for Drawings and Models (part and assembly files) and can be set to generate the viewable files for all CAD files, only the top CAD file, or no CAD files. See Model Viewable Combo Box Options (p. 62) for more details.

Additional configuration may be necessary to automatically create the viewable files.



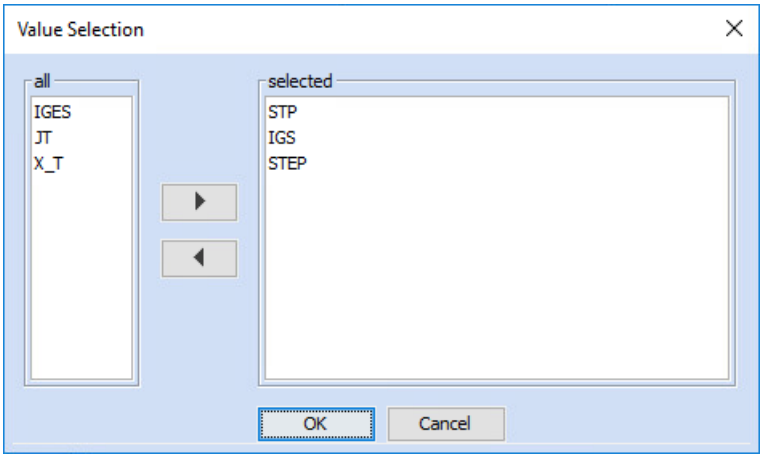
The combo boxes titled **Drawings** and **Models** define in which situations the viewables selected in the **Formats** combo boxes should be created. If no format is selected, no viewables are created regardless of the selected option in these combo boxes.

Option	Description
Drawings	Controls viewable generation for 2D model files. Valid options include: ■ **Enabled** - viewables of the selected formats are generated for drawing files. ■ **Disabled** - No viewables are generated for drawing files.

Option	Description
Models	Controls viewable generation for 3D model files. Valid options are: ■ **Top CAD file** - viewables of the selected formats are generated only for the top SOLIDWORKS file in the structure. ■ **All CAD Files** - viewables of the selected formats are generated for all SOLIDWORKS files in the structure. ■ **All Parts** - viewables of the selected formats are generated only for part files. ■ **All Assemblies** - viewables of the selected formats are generated only for assembly files. ■ **Disabled** - No viewables files are generated. Since release 3.6.4, only the options **Top CAD file** and **Disabled** are available by default.

Enabling viewable generation for 3D models can cause a huge load to processes in the SOLIDWORKS session on save and have a huge impact on the save performance. It is NOT recommended to enable and set as default for all users.

The **Formats** combo boxes launch a pop up dialog that displays all available viewable formats. The pop-up lets you select the desired formats to be created.



Related links
BrowserDisplay Section (p. 87)
Viewables Section (p. 96)

5.3.5.5 Manage viewable files in the local workspace

You can configure and define the viewable file types (file endings) that should be deleted by the context menu item **Remove Viewables** available in the Workspace Manager.

Configuration file: `CAXConfig.xml`

Structure: WorkspaceDeleteViewables

The **Remove Viewables** function deletes viewable files stored locally in the selected workspace.

See WorkspaceDeleteViewables Section (p. 96) for more information.

5.3.6 Thumbnail Support

Inside the Web Components thumbnails are extracted from the SOLIDWORKS native file.

The SOLIDWORKS system stores thumbnail views into the SOLIDWORKS binary data that are extracted using the same routines used by Windows Explorer.

The connector does not generate the thumbnail content but retrieves it from SOLIDWORKS files. Please refer to the documentation of your SOLIDWORKS tool on how to enable thumbnail generation in the SOLIDWORKS files. The connector extracts the same thumbnail visible in the thumbnail view of Windows Explorer.

The thumbnails are only held locally and are shown inside the Workspace Manager in small icons. A bigger fly-out is shown on the small icon, on a node in the browser views or on the file name columns in the table views. In the Load Preview the thumbnail from Agile PLM is downloaded if no local thumbnail is available. In all other dialogs, only thumbnails generated locally are used.

5.3.6.1 Activating Thumbnails

The MCAD connectors support the thumbnail functionality of Agile PLM.

About this task

This use case describes how to activate Thumbnails on Agile PLM Java Client.

Procedure

1. On the Java Client, navigate to the **Admin** tab and expand the **Server Settings** options by clicking on the plus (+) sign besides the folder name.
2. Double click on **Preferences**

   ➔ The Preferences dialog is opened displaying the **General Information** tab.
3. Change the option setting **Thumbnail Display** to **Enabled** from the drop down list.
4. Change the option setting **Thumbnail and Streaming File Pre-Generation** to **Enabled** from the drop down list.

Result

Thumbnails are activated.

5.3.6.2 AutoVue configuration for CAD thumbnails

To show the SOLIDWORKS thumbnail in the Web Client you should disable the generation of thumbnails in AutoVue for the SOLIDWORKS native file formats and enable the .PNG file rendering in AutoVue only.

Otherwise, the AutoVue generated thumbnails can overwrite the SOLIDWORKS thumbnail. Since the SOLIDWORKS file is not loaded for thumbnail generation, this reduces the load on the AutoVue server and database.

5.3.6.3 Transfer SOLIDWORKS thumbnails to Agile PLM

To transfer thumbnails to Agile PLM, you must configure the connector.

- **Configuration file:** `CAXConfig.xml`
- **Parameter:** `UploadThumbnails`
- **Default Value:** `ecs`

The option setting `UploadThumbnails` is used to control the transfer of thumbnails to Agile PLM. The possible values include:

Value	Description
ecs	If set to `ecs`, an API function is used to store the thumbnails in the Agile PLM database to prevent file transfer overhead. This is the default value.
fileserver	If set to `fileserver`, SOLIDWORKS thumbnails are transferred to Agile PLM by uploading them to the file vault. A preview .PNG file is visible in the **Files tab** of the Design object in this case.
false	If set to `false`, the thumbnail transfer to Agile PLM is switched off.

5.3.6.4 Thumbnails in the Load Preview

The thumbnails in the load preview use the Agile PLM thumbnails if available and download the file from the vault on demand (on a show fly out request).

If local files and thumbnails are available on the local disk already, then the thumbnail file is not downloaded from Agile PLM and the local representation is used.

5.3.6.5 Configure Thumbnail size in Agile PLM

Additionally, you can configure the size of the Thumbnail image in Agile PLM.

- **Configuration file:** `CAXConfig.xml`
- **Parameter:** `THUMBNAILS`

The default configuration is set in the internal `OraclePLM\jar\agile9\a93ws_connector.jar\com\xplm\agile93\cax\xml\CAXConfig.xml` file and should NOT be changed:

```
<Structure>
   <Name>THUMBNAILS</Name>
       <FieldCollection>
           <Field><Name>OMIT_GENERATION</Name><Value>false</Value></Field>
           <Field><Name>DEFAULT</Name><Value>200X200</Value></Field>
        </FieldCollection>
</Structure>
```

The external `OraclePLM\ini\CAXConfig.xml` can be extended to overwrite the default settings by adding a `Structure` called `THUMBNAILS` as follows:

```
<Structure>
  <Name>THUMBNAILS</Name>
    <FieldCollection>
      <Field><Name>DRW</Name><Value>100x300</Value></Field>
      <Field><Name>ASM</Name><Value>250X150</Value></Field>
    </FieldCollection>
</Structure>
```

where

- `Name` is the file extension in capital letters
- `Value` is Width x Height of the image in pixels

If the thumbnail size is reconfigured, thumbnail folders must be deleted for the changes to take effect. For example , delete `C:\AgileEC\wspaces\Default\.thumbnails`, otherwise the connector reuses the cached thumbnails.

5.3.7 Check Out Attributes functionality

This functionality re-reads a pre-defined list of Design attributes when executing a Check Out by the MCAD connector.

About this task

Since release 3.6.3.1, you can configure the attributes re-read from Agile PLM when the MCAD connector executes a Check Out.

The IDs of attributes to re-read are listed in the transaction `CheckOutAttributes` in the internal file `Agile93Templates.xml` in the package `com\xplm\agile93\ws\cad\xml`.

Do not remove the provided default attributes, you can only extend the list by adding required IDs.

The IDs are restricted on **TitleBlock** and **Page Two** Design attributes because the **Page Three** attributes are special for each subclass and they are not presented in the main table in the Save Preview.

If used in conjunction with a PX: The event subscriber used needs to have the following values selected:

- Trigger Type = Pre Execution
- Mode = Synchronous

You need administrative rights on the workstation to make the following changes.

Procedure

1. Copy `%xPlmRootDir%\jar\agile9\a934ws_proxy.jar` archive to a directory with write access for example `C:\Users\%USERPROFILE%\Documents\` folder
   a) Open the archive with 7-Zip, and navigate to the `xml` folder for example: `C:\Users\%USERPROFILE%\Documents\a934ws_proxy.jar\com\xplm\agile93\ws\cad\xml\`
   b) Copy the `Agile93Templates.xml` file out of the archive for example by dragging and dropping to your Documents folder.
   c) Open the `Agile93Templates.xml` file in a text editor and extend the transaction `CheckOutAttributes` by adding `Fields` with the **Base ID** of the required Design attributes. For example:

```
<!-- Description -->
<Field>
  <Name>6174</Name>
  <Value></Value>
</Field>
<!-- Text13 -->
<Field>
  <Name>1303</Name>
  <Value></Value>
</Field>
```

   d) Save the changes made to `Agile93Templates.xml`.
   e) Copy the `Agile93Templates.xml` file back into the archive. You can do this by dragging and dropping the file back into the archive open in 7-Zip.
2. Restore the `a934ws_proxy.jar` archive to its original location.

Result

The configured Design attributes are now set to be re-read when the connector performs Check Out of the corresponding Design Objects.

Mapped CAD values can be overwritten if they are listed in the `CheckOutAttributes` transaction.

6 Update

6.1 Modifying installation

Complete these steps to modify an existing installation and add for example new components or remove existing.

About this task

During modification, no existing files are overwritten and only missing files are added.

Procedure

1. Close all open applications related to the integration.
2. Start the current installer `setup.exe` with administrator rights.
   ➔ Visual C++ runtimes are checked/installed again and the installation wizard appears.
3. Click **Next** to start the wizard.
   ➔ The step *Modify, repair or remove installation* appears.
4. Click **Modify**.
   ➔ The step *Installation path* appears.
5. In this step, you cannot change the installation path, but applying overlay packages or making backups are possible.
   - To avoid later file access problems caused by installing in `C:\Program Files`, install into a writable directory that is not protected by Windows.
   - **Optional**: To overwrite the installation files with an overlay package after completion, enable the option **Apply custom files after installation**. If an overlay package from XPLM Solution GmbH is delivered as a ZIP file, unzip it first.

     When unzipping, often a "double" directory structure is created, for example `custom_files\custom_files\xml`. Always select the directory that corresponds to the installation directory that contains the configuration.
   - **Optional**: To backup the existing installation, enable the option **Backup current installation**.

     Always create a backup copy when you update or change your installation. This makes it easier to compare and merge the files later.
6. Click **Next** and update components, if required.
   ➔ The step *Ready to install* appears.
7. To start installation, click **Install**.

   During the installation process, a CMD window appears showing the progress of the installation. Do not click into this window or the installation will not continue. If you clicked in by mistake, press **Enter** or **Esc** to continue.
8. To close the wizard after installation, click **Finish**.

Result

The installation is modified. Start the product and verify everything works as expected.

Related links
Using overlay packages (p. 123)
Silent installation (p. 123)

6.2 Repairing installation

Complete these steps to repair an existing installation if the product does not work correctly, for example fixing missing or corrupt files, or incorrect shortcuts and registry entries.

About this task

During repair, existing files are overwritten and components registered again.

Procedure

1. Close all open applications related to the integration.
2. Start the current installer `setup.exe` with administrator rights.
   ➔ Visual C++ runtimes are checked/installed again and the installation wizard appears.
3. Click **Next** to start the wizard.
   ➔ The step *Modify, repair or remove installation* appears.
4. Click **Repair**.
   ➔ The step *Installation path* appears.
5. In this step, you cannot change the installation path, but applying overlay packages or making backups are possible.
   - To avoid later file access problems caused by installing in `C:\Program Files`, install into a writable directory that is not protected by Windows.
   - **Optional**: To overwrite the installation files with an overlay package after completion, enable the option **Apply custom files after installation**. If an overlay package from XPLM Solution GmbH is delivered as a ZIP file, unzip it first.

     When unzipping, often a "double" directory structure is created, for example `custom_files\custom_files\xml`. Always select the directory that corresponds to the installation directory that contains the configuration.
   - **Optional**: To backup the existing installation, enable the option **Backup current installation**.

     Always create a backup copy when you update or change your installation. This makes it easier to compare and merge the files later.
6. Click **Next**.
   ➔ The step *Ready to install* appears.
7. To start installation, click **Install**.

   During the installation process, a CMD window appears showing the progress of the installation. Do not click into this window or the installation will not continue. If you clicked in by mistake, press **Enter** or **Esc** to continue.
8. To close the wizard after installation, click **Finish**.
9. **Optional**: Carefully compare and merge the changes from the backup directory with the newly installed files.

Result

The installation is repaired. Start the product and verify everything works as expected.

Related links
Using overlay packages (p. 123)
Silent installation (p. 123)

6.3 Updating installation

Complete these steps to update an existing installation.

About this task

XPLM strongly recommends using appropriate services for an update. This ensures that existing functionality and modifications are correctly transferred to the new product. Contact support@xplm.com for assistance.

Procedure

1. Close all open applications related to the integration.
2. Start the new installer `setup.exe` with administrator rights.
   ➔ Visual C++ runtimes are checked/installed again and the installation wizard appears.
3. Click **Next** to start the wizard.
4. Click **Next**.
   ➔ The step *License agreement for end-users* appears.
5. Accept the license agreement and click **Next**.
   ➔ The step *Installation path* appears.
6. In this step, you cannot change the installation path, but applying overlay packages or making backups are possible.
   - To avoid later file access problems caused by installing in `C:\Program Files`, install into a writable directory that is not protected by Windows.
   - **Optional**: To overwrite the installation files with an overlay package after completion, enable the option **Apply custom files after installation**. If an overlay package from XPLM Solution GmbH is delivered as a ZIP file, unzip it first.

     When unzipping, often a "double" directory structure is created, for example `custom_files\custom_files\xml`. Always select the directory that corresponds to the installation directory that contains the configuration.
   - **Optional**: To backup the existing installation, enable the option **Backup current installation**.

     Always create a backup copy when you update or change your installation. This makes it easier to compare and merge the files later.
7. Click **Next** and update components, if required.
8. To start installation, click **Install**.

   During the installation process, a CMD window appears showing the progress of the installation. Do not click into this window or the installation will not continue. If you clicked in by mistake, press **Enter** or **Esc** to continue.
9. To close the wizard after installation, click **Finish**.
10. Carefully compare and merge the changes from the backup directory with the newly installed files.

Result

The installation is updated. Start the product and verify everything works as expected.

Related links
Using overlay packages (p. 123)
Silent installation (p. 123)

7 Uninstallation

7.1 Uninstalling integration

You can uninstall the integration in different ways.

Installation directory

Go to `%xPlmRootDir%` and double-click the **Uninstall** link.

Windows Control Panel

Go to *Programs and Features* and uninstall **XPLM Integration for Oracle Agile PLM**.

Silent mode

Command line calls runtime:

```
START /WAIT vcredist_[version]_[architecture].exe /uninstall /q
```

Command line calls MSI:

```
START /WAIT msiexec /uninstall [file name].msi /quiet
```

Copy the following code

To uninstall the integration in silent mode:

1. Copy the sample code and paste it in a batch file:
2. Save the file as `uninstall.bat` to the directory that contains the installation program.
3. Edit `uninstall.bat` again and change the values for the parameter names to reflect the configuration.

Example to uninstall MSXML 4.0 and Visual C++ Redistributable 9.0

```
START /WAIT msiexec /uninstall msxml_4.30.2100_x86.msi /quiet
START /WAIT vcredist_9.0.30729.6161_x64.exe /uninstall /q
```

8 Troubleshooting

8.1 Troubleshooting integration problems

About this task

In case of integration problems, proceed as follows:

Procedure

1. Close all related programs.
2. Activate logging.
3. Restart the integration and reproduce the problem.
4. Send an email with the problem description and attached log files to Oracle Global Customer Support (GCS) at www.oracle.com/support, or My Oracle Support at https://support.oracle.com.

Related links
Logging (p. 81)

8.2 License error codes

Table 1: License error codes

Error code	Description
1	Component license not found
10	License expired
100	Wrong MAC address
1000	Wrong company identifier
1111	License file exist but is not valid
3333	License file not found

8.3 Solving issues in mixed environments

If you plan to use ECAD and MCAD integrations together on the same client computer, several installation and licensing issues may occur.

Naming conventions for integration versions:

- Before ECAD product version 1.0.0, the separate FlexLM licensing mechanism was used together with the license file `license.dat`. We call such ECAD integrations the **first generation**.
- Starting with ECAD product version 1.0.0, the separate local license mechanism, called License Client, is used together with the license file `*.lic`. We call such ECAD integrations the **second generation**.
- Starting with ECAD product version 1.2.0, the official MCAD installer is used, which already contains and installs License Client. We call such integrations the **third generation**.


The integration described here corresponds to the third generation..

- Starting from 2022, a unified installer based on the third generation is gradually being introduced for all XPLM products. We call such integrations the **fourth generation**. Integrations installed with this installer can run fully in parallel with other products on the same client computer, eliminate redundancies in certain core components, and provide improved uninstall and upgrade behavior.

Issue	Solution
How can I identify an installed integration version?	■ First generation ECAD: □ In the installation directory of the integration, the license file `license.dat` exists. ■ Second generation ECAD: □ The environment variable `xPlmRootDir` exists and points to the installation directory of License Client where the license file `*.lic` is located. Typical locations for the license file in second-generation products are `%xPlmRootDir%\LicenseClient\xml` or `%xPlmRootDir%\xcl\xml`. □ The integration itself is not installed under the `xPlmRootDir` path but in another directory. □ In the installation directory of the integration, the file `uninst.exe` exists. ■ Third generation ECAD/MCAD: □ The integration is installed under the `xPlmRootDir` path. □ In the installation directory `xPlmRootDir`, the shortcut `Uninstall_*` exists. □ MCAD and certain ECAD installers are named `xxx_<version>_x64.msi`. □ ECAD installers in general are named `Integrate2_<PLM>_<ECAD>_<version & build>.exe` ■ Fourth generation ECAD/MCAD: □ The integration is installed under the `xPlmRootDir` path. □ In the installation directory `xPlmRootDir`, there is no shortcut `Uninstall_*`. □ MCAD installers are named `<PLM>_<MCAD>_<version>.7z.exe`. □ ECAD installers are named `<PLM>_<ECAD>_<version>.7z.exe`. □ The directory `C:\ProgramData\XPLM Solution GmbH` exists. □ The registry entry *HKLM\SOFTWARE\XPLM Solution GmbH\{00000000-0000-0000-0000-000000000000}* exists.

Issue	Solution
I plan to install an additional ECAD integration on the same client computer where an MCAD integration is already installed. What do I have to consider regarding licensing and installation?	■ MCAD = third generation and ECAD = first generation  This combination should not exist today. ■ MCAD = third generation and ECAD = second generation 1. Contact https://support.oracle.com and request a combined license file that serves both integrations. 2. Install the ECAD integration in a different directory than `%xPlmRootDir%`. 3. Replace the license file in the directory `%xPlmRootDir%\xml`. ■ MCAD and ECAD = third generation  This combination should not exist today. ■ MCAD = third generation and ECAD = fourth generation  Installations are not compatible! A fourth generation can only be installed by uninstalling the previous installation. In this case, the previous installation must be available as fourth generation as well. Contact https://support.oracle.com for assistance. ■ MCAD = fourth generation and ECAD = second generation 1. Contact https://support.oracle.com and request a new ECAD license file. 2. Install the ECAD integration in a different directory than `%xPlmRootDir%`. 3. Copy the new ECAD license file to the directory `%xPlmRootDir%\xml`. Fourth-generation products support multiple license files. In this case, License Client is used to serve both MCAD and ECAD integrations. ■ MCAD = fourth generation and ECAD = third generation  This combination should not exist today. ■ MCAD/ECAD = fourth generation 1. Contact https://support.oracle.com and request a new ECAD license file. 2. Install the ECAD integration in the same directory `%xPlmRootDir%`. Fourth-generation products can be used completely in parallel and do not interfere with each other. 3. Copy the new ECAD license file to the directory `%xPlmRootDir%\xml`. Fourth-generation products support multiple license files.

Issue	Solution
I plan to install an additional MCAD integration on the same client computer where an ECAD integration is already installed. What do I have to consider regarding licensing and installation?	■ ECAD = first generation and MCAD = third/fourth generation 1. Contact https://support.oracle.com and request a new MCAD license file. 2. Install the MCAD integration. 3. Copy the new MCAD license file to the directory `%xPlmRootDir%\xml`. ■ ECAD = second generation and MCAD = third generation This combination should not exist today. ■ ECAD = second generation and MCAD = fourth generation 1. Contact https://support.oracle.com and request a new MCAD license file. 2. Back up the existing ECAD license file in directory `%xPlmRootDir%\xml`. 3. Uninstall the existing License Client. 4. Install the MCAD integration. 5. Copy both license files to the directory `%xPlmRootDir%\xml`. Fourth-generation products support multiple license files. In this case, License Client is used to serve both MCAD and ECAD integrations. ■ ECAD and MCAD = third generation This combination should not exist today. ■ ECAD = third generation and MCAD = fourth generation Installations are not compatible! A fourth generation can only be installed by uninstalling the previous installation. In this case, the previous installation must be available as fourth generation as well. Contact https://support.oracle.com for assistance. ■ ECAD/MCAD = fourth generation 1. Contact https://support.oracle.com and request a new MCAD license file. 2. Install the MCAD integration in the same directory `%xPlmRootDir%`. Fourth-generation products can be used completely in parallel and do not interfere with each other. 3. Copy the new MCAD license file to the directory `%xPlmRootDir%\xml`. Fourth-generation products support multiple license files.

8.4 Resolving common errors

Following table shows a list of errors that may occur and possible solutions to resolve these errors.

Errors	Possible Reason(s) and Solution(s)
Installation errors	

Errors	Possible Reason(s) and Solution(s)
Error message *.NET 4.x not detected* is displayed.	**Reason:** Systems requirement Microsoft .NET Framework is not installed or the required version is missing **Solution:** Install or update the .NET Framework to the requested version.
This upgrade is not possible due to compatibility reasons error displayed during installation	**Reason:** Some versions of the MCAD connector cannot be upgraded automatically as the underlying architecture of the setup wizard might have changed. **Solution:** Completely uninstall the previous version of the MCAD connector that is installed on the environment in question and afterwards run the setup wizard of the new MCAD connector version.
Connection errors	

Login dialog does not appear	**Reason:** If it occurs on Windows 10 environment, could be 8.3 MS DOS file naming being deactivated. Use the following command (**requires a command prompt with elevated privileges!**) to verify if 8.3 file naming is activated on the system in question;
	`fsutil 8dot3name query C:`
	This assumes that drive `C:` is the drive on which the MCAD connector is installed. Adjust the drive to your specific environment.
	Solution: If 8.3 file naming is deactivated, the problem can be solved by uninstalling the MCAD connector and deleting all files and directories that have been written by the installation wizard (usually, this means the directory `C:\Program Files\XPLM Solution GmbH\` and all its contents). Next, activate 8.3 file naming using the following command (**requires a Windows command prompt with elevated privileges!**);
	`fsutil behavior set disable8dot3 0`
	After that, reinstall the MCAD connector and verify that the login dialog appears

Login dialog appears but connection is not established	**Reason:** If the URL ends with a forward slash (/), connection is not possible **Solution:** Check the URL entered in the login dialog and ensure it does not end with a forward slash (/)
Login failed pop up message opens in the background	If a user enters the wrong credentials and attempts to login, it may happen that the *Login failed* pop up message is opened in the background leaving the SOLIDWORKS system unresponsive. **Workaround:** Pressing **Alt-Tab** will let you navigate to the background dialog and close it.

Errors	Possible Reason(s) and Solution(s)
RUP 20 breaks compatibility with the MCAD connector for the first time	**Reason:** Older MCAD connector versions do not work with RUP 20 due to an update of certain server-side libraries which breaks backwards compatibility. Servers running an RUP version older or equal than RUP 19 remain compatible with the MCAD connector. **Solution:** MCAD version 3.6.4.1 (or later) is required for Agile 9.3.6 RUP 20

MCAD connector can not connect to servers running Agile 9.3.6 RUP 20 (or newer) as well as to servers running RUP 19 (or older)	**Reason:** The libraries `httpclient.jar`, `httpcore.jar`, `slf4j-api.jar` (from RUP20 and above) are unavailable on the Agile PLM server for customers using older RUP versions.
	Solution: To allow using of local jars, the java option has to be specified in `acx.bat` `-Dfsclient.local=TRUE`
	For each jar the location can be provided using java options in `acx.bat` as below. The default locations are used if java options are not provided.
	`-Dfsclient.jar=<full_path_to_jar>`
	`Default: <CAX_ROOT>/jar/agile9/ecfsclient.jar`
	`-Dhttpclient.jar=<full_path_to_jar>`
	`Default: <CAX_ROOT>/jar/commons/httpclient.jar`
	`-Dhttpcore.jar=<full_path_to_jar>`
	`Default: <CAX_ROOT>/jar/commons/httpcore.jar`
	`-Dslf4j-api.jar=<full_path_to_jar>`
	`Default: <CAX_ROOT>/jar/commons/slf4j-api.jar`

User interface errors	
Some characters are not correctly displayed on the Save Preview and Workspace Manager	**Reason:** This is a known issue where **Non-Latin** CAD file name characters are not correctly displayed on the Save Preview/Workspace Manager in Windows systems **Solution:** Install the corresponding Language Pack. For Example, Korean characters are displayed as boxes. This is because Java does not use the Windows 10 font (Malgun Gothic) for displaying Korean characters which is available without the Korean Language Pack installed. In this case, installing the Korean Language Pack resolves the issue.
MCAD dialog windows are opened behind the SOLIDWORKS system.	**Reason:** This is a known SOLIDWORKS 2019 issue where MCAD dialogs are opened in the background. **Solution:** Update to SolidWorks 2019 SP 5.0 solves the issue.
Errors while saving	
Save Preview does not appear or crashes	**Reason:** This error may occur when there are errors in the `CAXConfig.xml` file. Removing any of the attributes configured in the `TableDisplay` section for example, will cause the Save Preview dialog to deadlock.

Errors	Possible Reason(s) and Solution(s)
Error message *"Failed to update structure # Error adding rows to Structure table: String value….. exceeds the maximum length of 50 characters. # Design is already checked out # Design is already checked out #*	**Reason:** Reference Attribute in the Design Structure is system limited to 50 characters. **Solution:** This issue needs to be solved in Agile PLM and is not a connector issue. Oracle provides SQL scripts that can be used to extend the attribute fields to more characters as needed. Contact Oracle support for more information.
Error message *Error adding rows to Structure table: String value exceeds the maximum length of 50 characters*	**Reason:** This error occurs if the description of a Standard Part is longer than 50 characters and you attempt to save the file to Agile PLM. The MCAD connector writes the description of a standard part to Agile PLM attribute **2000008380**. The maximum length of this attribute is fixed to 50 characters in the database of Agile PLM and it is not possible to easily expand it. **Workaround** - You can configure the MCAD connector to truncate attributes to the maximum allowed Agile PLM attribute length by adding the following option to the `ConnectionProperties` structure of the `CAXConfig.xml` file. `<Field><Name>checkMaxLength</Name><Value>true</Value></Field>`
Java process unexpectedly terminates after a Save Process	**Reason:** This is an installer issue caused by lack of memory to process the request. The error cannot be fixed/compensated. The Java process has to be stopped and restarted at the next connection to the Agile PLM. A modal user notification message added. If Java process cannot fork a new system processes it will be terminated with a message: `Java has to be terminated due to <Exception Message Placeholder|Unknown Error>! The login data will be requested again at the next connection to the PLM.` **Workaround**: Users will be simply asked to login at the next Agile PLM action.
Errors while loading	
Loading does not work when clicking **Load in CAD** in the Load Preview. No error message is displayed, the process just fails silently.	**Reason:** This error may occur when there are Characters that are not allowed in Windows in a file name. For example having quotation marks (**""**) in the **CAD Filename** attribute of one or more of the Designs attempted to load from Agile PLM. **Solution:** The values in the **CAD Filename** attribute need to be corrected manually in this case. Refer to **Doc ID 2798278.1** for more information.

Errors	Possible Reason(s) and Solution(s)
Load Preview does not appear after clicking Load to CAD in the web client.	Check if Windows UAC is activated and the browser is started with admin privileges. In this case, do not start the browser with admin privileges. Check the file association of .CADXML files. InvokeCallback.exe must be the default application for opening such files. You can set this with the **Open with** dialog in Windows. If you are not able to set InvokeCallback.exe as the default application for opening .CADXML files, it maybe that the InvokeCallback.exe is incorrectly registered, if the wrong path is given in the Windows registry under `HKEY_CLASSES_ROOT\Applications\InvokeCallback.exe\shell\edit\command` In this case, Windows does not allow users to select an application (`.exe` file) in the **Open with** dialog. Correcting the startup path under the above mentioned registry key is required to resolve the issue in this case.
Errors during Object Creation	
Error message *No templates found in database. New objects cannot be created until templates are created in the database.*	**Reason:** No valid templates available in Agile PLM. **Solution:** Make sure that the template structure in Agile PLM has been properly created and contains files.
A template object does not appear in the Template File combo box although it is part of the template structure in Agile PLM.	**Reason:** Damaged template objects in Agile PLM are not displayed in the Create Object dialog, neither in the **Filetype of Template** combo box nor in the **Template File** combo box. **Solution:** Check if all properties of the template object's Title Block are properly set. Refer to the chapters Adding Template Files to the Structure and Subtypes for more information.
Template files that contain family tables do not work	**Reason:** CAD files with family tables of any kind (such as Configurations) should not be used as template files because references between or within these files cannot be resolved when downloading a template file using the Create function. Due to this it is not guaranteed that the family tables work in the downloaded template. **Solution:** Do not use family tables in template files.
Java process crashes when using Create repeatedly	**Reason:** SOLIDWORKS and the connector deadlock if users start the Create dialog directly after having started SOLIDWORKS, close it and then open it again. This issues has been reported for users with Windows 10 version 1607 . **Solution:** Upgrade Windows 10 to version 1909
Item and BOM related errors	

Errors	Possible Reason(s) and Solution(s)
Error message *Error updating rows of BOM table: BOM Table is Read Only! #*	**Reason:** This error may appear when saving/publishing the BOM. **Solution:** Check the configuration of the Agile fields in the BOM tab: Item Number (ID: 1011), BOM Quantity (ID: 1035), CAD Filename (ID: 1341), Identifier (ID:2175), Component (ID: 2176), Reference (ID:2177). These fields must be visible (enabled). If they are not, the BOM does not get published. Until Agile 9.3.4, no error message is issued by EC services if these fields are not enabled/accessible. Also check if those fields allow long enough values (e.g., at least 150 chars for CAD Filename).
File transfer errors	
Files cannot be transferred. Max number of attempts exceeded	**Reason:** This error message can have a number of reasons; ■ **File server not running or not configured correctly** □ Check that the file server is running by attempting to upload and download files via the Web Client. ■ **File Size Mismatch** □ The file sizes defined in Agile PLM and the actual file size in the vault are different. If this is the case, an error message in the `a9handlers.log` should be visible. The **Oracle Utility vault** tool can be used to fix this error. ■ **Incompatible MCAD and Server version** □ Check that your server version is supported by the MCAD version in use. ■ **Missing privileges** □ Check that the user has the required privileges to download the files from Agile PLM ■ **Corrupted migrated data** □ In this case the data corruption needs be fixed.

8.5 Logging

The MCAD connectors for Agile PLM provides component specific logging.

8.5.1 Web Component Logs

Complete these steps to enable debug logging.

About this task

Configuration file: `%xPlmRootDir%\ini\Logfile.xml`

This use case describes how to activate web component logging in debug mode. The Web Component log files are written to `%USER_HOME%\AgileCache\` directory on the users workstation.

Procedure

1. Close the SOLIDWORKS system and make sure all MCAD processes especially the Java process have terminated.
2. Go to `%USER_HOME%\AgileCache\` and delete all the `.log` files. These include:
   - `a9handlers2.log` - Contains Java Exceptions thrown during program execution.
   - `a9proxy2.log` - Contains logging information on the client-server communication.
   - `a9gui2.log` - Contains GUI related logging information.
   - `a9_performance2.log` - Contains performance related logging information.
   - `a9_cad_proxy2.log` - interface between CAD Script Engine and Java Module
   - `a9ccon2.log`
   - `a9_ft2.log` - Contains logging information for file transfer utility.

     This log file was added in version 3.6.4.4 of the SOLIDWORKS connector.

   It is highly recommended to delete or rename the log files before starting to capture logging output to prevent overly long log files.
3. Go to `%xPlmRootDir%\ini` and make the following changes:
   a) Rename `Logfile.xml` to `Logfile.xml_STD`
   b) Rename `Logfile.xml_DEBUG` to `Logfile.xml`
4. Restart the SOLIDWORKS system including the MCAD connector.

Result

Web component logging in debug mode is now activated on your workstation.

Next steps

Logging messages are continuously written to the log files. After capturing the required logging output, revert the changes made in Step 3 above. Debug logging may cause a huge load on the system affecting performance.

8.5.2 Script Engine Logs

Script engine logging is turned off by default.

About this task

This use case describes how to activate Script engine logging.

Procedure

1. Open `XPlmSolidWorksA9Connector.xml` in a text editor for example Notepad ++ .
2. Set `EnableScriptEngineLogging` to `true`.
3. Set the file name for the log file. Usually, `xacw.log` is set by default.

   The path is relative to `%USER_HOME%\AgileCache\`, therefore just the filename is given.
4. Define the level of detail of the log file in the `ScriptengineLogLevel` setting.
5. Save changes made to the `xml` file.

Result

Script engine logging is enabled on your workstation.

Next steps

After capturing the required logging output, set `EnableScriptEngineLogging` back to `false`.

8.5.3 CAD Connector Logs

CAD connector logging is turned off by default.

About this task

This use case describes how to activate CAD Connector logging.

Procedure

1. Open `XPlmSolidWorksConnector.xml` in a text editor for example Notepad ++
2. Set `EnableSolidWorksLogging` to `true`.
3. Set the location and file name for the log file by inserting a valid path to `SolidWorksLogFile` setting. For example `C:\SolidWorks.log`

   The given directory must exist. We recommend using `%USER_HOME%\AgileCache\` directory.
4. Define the level of detail of the log file in the `SolidWorksLogLevel` setting.
5. Save changes made to the `xml` file.

Result

CAD Connector logging is enabled on your workstation.

Next steps

After capturing the required logging output, set `EnableSolidWorksLogging` back to `false`.

9 References

9.1 Configuration files

9.1.1 CAXConfig.xml Settings

The `CAXConfig.xml` file controls general and numbering options for load and save. The different sections control the communication between client and server, logic for display in the client, parameters for part families and the numbering schemes and change process.

9.1.1.1 Basic Section

Setting	Description
SITE	Defines the global parameter for the MCAD connector configuration. MCAD connectors with different `SITE` setting use different configuration objects in Agile PLM. **Default:** MCAD **Possible Values:** Any string value
checkRequired	If set to `true`, the MCAD connector verifies that all Agile PLM fields labeled as mandatory contain a value. If not, an error message is displayed to the user during save. **Default:** false **Possible Values:** true | false
NonExistingPartsFromCAD	This option setting only affects property mapping. ■ If set to `allowed`, a property mapping towards the Item number can be used to create new Item objects, if the value mapped to the Item number does not identify an existing Item. ■ If set to `remove`, Item objects are not created in this case. **Default:** allowed **Possible Values:** allowed | removed
AllowManualItemNumber	Makes the Item Number text field in the Details pane editable (true) or non-editable (false). **Default:** true **Possible Values:** true | false
VersionsToSearch	Defines how many older Design versions are searched if a **Where used** query is executed. Only retrieves Design versions which are older than the selected one. **Default:** 10 **Possible Values:** Any integer value

Setting	Description
VersionsToSearchFromCAD	**Default:** 10 **Possible Values:** Any integer value
RelationsFromCAD	**Default:** true **Possible Values:** true | false
DocumentSupport	Defines if sub-classes of the Documents class are visible in the MCAD connector (For example in the Preferences dialog). **Default:** false **Possible Values:** true / false Entry does not exist in default installation and must be manually added to the configuration file.

9.1.1.2 ConnectionProperties Section

Setting	Description
timeout	Defines the timeout value for web services responses from the Agile PLM server in milliseconds. **Default:** 900000 **Possible Values:** Any number value
bulksize	The server calls are divided into packages of this number of objects per call, depending on the server and network performance (use 20 for slower environments and 100 for faster ones). **Default:** 50 **Possible Values:** 20 | 50 | 100
call-threads	Fallback value for the thread related option settings `call-threads-get`, `call-threads-cuo-save`, `call-threads-cuo-commit`. If no value is given for any of these previously named option settings, the value given for `call-threads` is used. **Default:** 4 **Possible Values:** 1 | 2 | 4
call-threads-get	Number of concurrent threads for **GET** requests with which the EC services interface is called. **Default:** 4 **Possible Values:** 1 | 2 | 4

Setting	Description
call-threads-cuo-save	Number of concurrent threads for save requests with which the EC services interface is called. **Default:** 2 **Possible Values:** 1 | 2
call-threads-cuo-commit	Number of concurrent threads for commit requests, structure and relationships creating requests, BOM requests with which the EC services interface is called. **Default:** 2 **Possible Values:** 1 | 2
UploadThumbnails	Controls thumbnail upload to Agile PLM. ■ **fileserver** – upload the SOLIDWORKS thumbnail from the client to the vault and attach the .PNG to the Design in Agile PLM. To display the thumbnail in Agile PLM configure AutoVue to: □ NOT render the SOLIDWORKS native files for thumbnails □ Render .PNG files for thumbnails ■ **false** – no thumbnail is uploaded from the SOLIDWORKS client to the server. ■ **ecs** – hand over the thumbnail to Agile PLM via web service **Default:** ecs **Possible Values:** false | fileserver | ecs
resolution_max_depth	This is a hidden setting. It needs to be added to `CaxConfig.xml` manually in case that it should be changed. We strongly advice against changing this setting. Defines the depth of the structure resolution that is executed when loading from Agile PLM. In other words, how many levels of an assembly structure are retrieved from Agile PLM: ■ **0** - only retrieve the root element ■ **1** - retrieve the root element and the first level of child objects ■ **2** - retrieve the root element and two levels of child elements ■ **50** - retrieve any number of levels **Default:** 50 **Possible Values:** 0 | 1 | 2 | 50 Since Agile 9.3.6 RUP 10, the default value for this option setting should be used. If CaxConfig.xml contains a different value for this option setting, we recommend deleting this option setting from CaxConfig.xml to apply the default.

9.1.1.3 BrowserDisplay Section

HideSuccessSummary	If set to `true`, the Save Summary dialog is only displayed for save processes where errors occurred. If set to `false`, the Save Summary dialog is displayed at the end of each save process. **Default:** false **Possible Values:** true | false
LazyLoad	Activates or de-activates the MCAD connector's LazyLoad feature. ■ If set to `true` (activated)- Agile PLM information in the MCAD GUI is only retrieved from Agile PLM and displayed on user request (click the object to initiate the process). ■ If set to `false` (de-activated) - all Agile PLM information for all objects visible in the MCAD GUI is retrieved and displayed as soon as a GUI widget appears on screen. Thus, activated LazyLoad may shorten onset times especially for many Agile PLM-known SOLIDWORKS objects. **Default:** true **Possible Values:** true | false
Tree.Menu.Zip	Defines, if the **Zip and Upload Workspace** context menu item is visible in Workspace Manager. ■ `true` - show the context menu item ■ `false` - hide the context menu item **Default:** true **Possible Values:** true | false
Tree.Menu.Remove	Defines, if the **Delete Workspace** context menu item is visible in Workspace Manager. ■ `true` - show the context menu item ■ `false` - hide the context menu item **Default:** false **Possible Values:** true | false  Entry does not exist in default installation and must be manually added to the configuration file.

AllowExistingChangesOnly	If set to `true`, when manually entering a Change number into the Change number field, the Change assignment is only executed if the number of an already existing Change has been given. If set to `false`, entering a Change number into the Change number field causes the MCAD connector to create a new Change object, if none with the given number already exists. **Default:** false **Possible Values:** true | false
DisableChangesAutonumbers	Activates or de-activates the Change autonumber button in the Save Preview dialog or Assign Change dialog window. ■ `true` - de-activates the button ■ `false` - activates the button **Default:** false **Possible Values:** true | false
ShowTreeView	Defines the pre-set value for showing or hiding the tree view widget. This value only defines the default value of this option setting. The value is overwritten by the user-specific option setting in the `$HOME/AgileCache/GUIConfig.xml` configuration file. ■ `true` - pre-set value to show the Tree View widget ■ `false` - pre-set value to hide the Tree View widget **Default:** true **Possible Values:** true | false
WorkspaceFolderClass	Defines the class of the Agile PLM object that is used for uploading workspace folders via the MCAD connector's **Zip and Upload Workspace** functionality. **Default:** Design **Possible Values:** Any Agile PLM class object
WorkspaceFolderAutonumber	Defines the autonumber generator which is used to retrieve the Design number for the object that is used for uploading workspace folders via the MCAD connector's **Zip and Upload Workspace** functionality. **Default:** Design Number Removing both entries causes the MCAD connector to create Design objects with a number derived from the current user name and time stamp.

WorkspaceFolderDescription	Defines the description given to the Design object created when uploading workspace folders via the MCAD connector's **Zip and Upload Workspace** functionality.The order of the values does not play any role. **Default:**host;path;file
ShowFlyoutInBrowser	Activates or de-activates the thumbnail fly-out dialog in the MCAD connector's tree view widget. ■ `true` - activates the thumbnail fly-out dialog ■ `false` - de-activates the thumbnail fly-out dialog **Default:** true **Possible Values:** true | false
ShowFlyoutInList	Activates or de-activates the thumbnail fly-out dialog in the MCAD connector's list view widget. ■ `true` - activates the thumbnail fly-out dialog ■ `false` - de-activates the thumbnail fly-out dialog **Default:** true **Possible Values:** true | false
SinglePortThreshold	This option setting defines how many references a structure needs to have to be displayed in single port mode. Single port mode is a more compact view of the references in the Save Preview that is automatically enabled if there are more than the number of references given in the `SinglePortThreshold` option are displayed in the Tree View/Graph View. ■ `true` - activates single port mode ■ `false` - de-activates single port mode **Default:** true **Possible Values:** true | false Entry does not exist in default installation and must be manually added to the configuration file.
ModelViewableOptions	Defines the combo box options for model viewable creation in the Preferences dialog. **Default:** TOP **Possible Values:** TOP / ALL / ASSEMBLIES / PARTS Entry does not exist in default installation and must be manually added to the configuration file. See Model Viewable Combo Box Options (p. 62) for more information.

ShowCheckInOption	Defines if a separate Check In save option should be displayed in the Java dialogs context menus. Setting this option setting to `true` causes the MCAD GUI to display a separate Check In save option without Incorporate in addition. **Default:** false **Possible Values:** true | false Entry does not exist in default installation and must be manually added to the configuration file.
DesignTableHeight	Since release 3.6.4.3 the connector allows resizing the height of the panes in the Side Panel of the Save Preview. This option setting configures the height of the **Design Properties Table** in the side panel of the Save Preview. The values are integer and provides the height for property panes in pixels. The value can be set between 150 and 500 pixels otherwise it will be set to the max/min value respectively. **Default:** 250 **Possible Values:** Any pixel Integer value between 150 and 500 Entry does not exist in default installation and must be manually added to the configuration file.
ItemTableHeight	Since release 3.6.4.3 the connector allows resizing the height of the panes in the Side Panel of the Save Preview. This option setting configures the height of the **Item Properties Table** in the side panel of the Save Preview. The values are integer and provides the height for property panes in pixels. The value can be set between 150 and 500 pixels otherwise it will be set to the max/min value respectively. **Default:** 250 **Possible Values:** Any pixel Integer value between 150 and 500 Entry does not exist in default installation and must be manually added to the configuration file.
WorkspaceTableHeight	Since release 3.6.4.3 the connector allows resizing the height of the panes in the Side Panel of the Save Preview. This option setting configures the height of the **Workspace Table** in the side panel of the Save Preview. The values are integer and provides the height for property panes in pixels. The value can be set between 150 and 500 pixels otherwise it will be set to the max/min value respectively. **Default:** 305 **Possible Values:** Any pixel Integer value between 150 and 500 Entry does not exist in default installation and must be manually added to the configuration file.

SearchColumn	Defines the default search criterion for the search feature in the Workspace Manager. **Default:** DESCRIPTION **Possible Values:** Any property value available in the Workspace tab column headers. Entry does not exist in default installation and must be manually added to the configuration file.
ShowServerLoadDrawing	Defines if Load Drawings Check Box should be displayed in in the Load preview. **Default:** true **Possible Values:** true | false Entry does not exist in default installation and must be manually added to the configuration file.

9.1.1.4 FlowControl Section

This section controls behavior during processes during the Save and Load Preview windows.

Setting	Description
cleanUnreserved	Reverts the save option to **do not save** for all SOLIDWORKS objects, that are not checked out by the current user if the **Force User Checkout** option is activated. ■ `true` - revert the save option ■ `false` - do not revert the save option **Default:** false **Possible Values:** true | false
AllowCheckoutOfNonLatest	Permits users to check out a Design version which is not the latest one in Agile PLM via the MCAD connector. ■ `true` - check out of a non-latest Design version is allowed ■ `false` - check out of a non-latest Design version is not allowed **Default:** true **Possible Values:** true | false
HelperPartAttachment	Defines, if attachments should be published for helper parts during the Publish to Item process. ■ `true` - publish attachments for helper parts ■ `false` - do not publish attachments for helper parts **Default:** true **Possible Values:** true | false

Setting	Description
HelperPartBOM	Defines, if the BOM should be published for helper parts during the Publish to Item process. ■ `true` - publish the BOM for helper parts ■ `false` - do not publish the BOM for helper parts **Default:** false **Possible Values:** true | false
InstanceBOM	Defines, if the BOM should be published for instances of part families (configurations) during the Publish to Item process. ■ `true` - publish the BOM for part family instances ■ `false` - do not publish the BOM for part family instances **Default:** true **Possible Values:** true | false
GenericBOM	Defines, if the BOM should be published for the generic of a part family during the Publish to Item process. ■ `true` - publish the BOM for the generic ■ `false` - do not publish the BOM for the generic **Default:** true **Possible Values:** true | false
SuppressUpdateFromBOM	Suppresses the attribute update during BOM creation for existing child elements of a BOM structure (**Title Block**, **Page Two**, **Page Three**) during BOM creation. ■ `true` - suppress attribute update ■ `false` - allow attribute update **Default:** false **Possible Values:** true | false
DrawingResolutionAction	Defines the **Save option**, that is applied to any drawing Design object added to the Save Preview dialog using the **Save As** confirmation pop-up. The Save As confirmation pop-up appears every time a Save As is executed in the Save Preview, allowing users to have the MCAD connector perform a where-used query in Agile PLM to add drawings related to the structure displayed in the Save Preview. **Default:** Save As **Possible values:** Save As | Check In | Save | Ignore Entry does not exist in default installation and must be manually added to the configuration file.

Setting	Description
ExcludeInstancesFromDFCO	Since release 3.6.2.1 the connector allows adding file-less Design objects to DFCOs by generating dummy files on the fly when assigning a DFCO through the MCAD connector. Has no effect for CAD systems that represent part family instances/configuration by individual files. The part family/configuration instances for these CAD systems are always added to DFCOs. ■ `true` - do not add part family instances/configurations to DFCOs ■ `false` - add part family instances/configurations to DFCOs **Default:** true **Possible Values:** true | false
SaveReservedOnly	If set to `true` and Preference setting **Check Out During Save** is set to **Automated Check Out**, an effective **Save Option** will be set only for designs that are checked out by the current user. **Default:** false **Possible values:** true | false
ExistingItemWorkflow	Since release 3.6.4.6, the connector allows deactivating *Item already exists* pop up prompts while assigning existing items in Agile PLM through the *Save preview*. ■ `ExistingItemDef` - No *Item already exists* pop up prompts are displayed. Item is directly assigned and CAD properties are kept by default. ■ `NONE` - *Item already exists* pop up prompts are shown each time an existing item is assigned to confirm assignment and to choose the property values to be kept. **Default:** NONE **Possible values:** NONE | ExistingItemDef Entry does not exist in default installation and must be manually added to the configuration file.
FF_REUSE_CLASS_BY_SAVE_AS ITEM_REUSE_CLASS_BY_SAVE_AS	If set to `true`, for File Folders or Items respectively the sub-class assignment is based on the existing sub-classes of the source objects of the Save As process overriding the default classes selected in the *Preferences dialog*. **Default:** false **Possible values:** true | false Entry does not exist in default installation and must be manually added to the configuration file.

9.1.1.5 DateFormats Section

Defines a set of date and time formatting parameters based on the Java `SimpleDateFormat` specification. These formatting definitions can be applied to date and time values mapped between SOLIDWORKS and Agile PLM.

```
<Structure>
 <Name>DateFormats</Name>
 <FieldCollection>
  <Field><Name>yyyy-MM-dd HH:mm:ss z</Name><Value>CAD</Value></Field>
  <Field><Name>yyyy-MM-dd</Name><Value>CAD</Value></Field>
  <Field><Name>dd.MM.yyyy HH:mm:ss z</Name><Value>CAD</Value></Field>
  <Field><Name>dd.MM.yyyy</Name><Value>CAD</Value></Field>
 </FieldCollection>
</Structure>
```

You can override these settings by configuring the `mapping.xml` file. For example:

```
<Field>
<Name>FORMAT_TYPE</Name>
<Value>DateFormat MM/dd/yyyy</Value>
</Field>
```

Related links

MCAD-MAPPING folders (p. 51)

9.1.1.6 TableDisplay Section

This section contains all columns that should be displayed in the tables of the Save and Load Preview windows.

The fields initially listed here should not be deleted, this may cause errors in program execution.

However, it is possible to add additional PLM fields to customize the view. To add an additional column to the table a new `field` must be added to the `FieldCollection` of the `TableDisplay` section.

A `field` of a PLM Design object is added by insertion of a new line of XML code that can be derived from the following example:

```
<Field><Name>FieldID</Name><Value>0</Value></Field>
```

If a PLM field of an Item object should be used, the string `Item.FieldID` must be used as field name instead of just `FieldID`:

```
<Field><Name>Item.FieldID</Name><Value>0</Value></Field>
```

The name of the field must contain the `FieldID` (internal ID) of the PLM field that should be added, its value must always be `0` (zero) by convention. The `FieldID`, also called **Base ID**, can be found in the Agile Java Client (**Data Settings** > **Classes** > **Double-click on a Design or Part class** > **User Interface Tabs** > **Double-click on a list entry** > **Attributes tab** > **Column: Base ID**).

After having added one or more column entries the file `%USERHOME%\GUIConfig.xml` must be deleted to make the changes appear in the GUI.

The following code example shows the `TableDisplay` section with two customized columns. To the tables the PLM fields `1305` of the Design object and `1313` of the Item object were added as separate columns:

```
<Structure>
 <Name>TableDisplay</Name>
 <FieldCollection>
            <!-- snip -->
              <!-- Customized fields: -->
             <Field><Name>1305</Name><Value>0</Value></Field>
             <Field><Name>Item.1313</Name><Value>0</Value></Field>
 </FieldCollection>
</Structure>
```

To make fields editable in the Save Preview the `Value` tag is set to `1` instead of `0`.

Not all `fields` are supported to be editable while others are always editable. The following fields are always **read-only**, regardless of the corresponding `Value` tag:

- CAX_FIL_NAME
- CAX_FULL_NAME
- IS_INCORPORATED
- DCO • DCO_SEQUENCE
- DCO_STATUS
- CHECKOUTUSER
- WORKFLOW_STATUS
- CAX_MODIFIED
- PLM_MODIFIED
- SAVED
- HAS_PRIVILEGE
- FILTER
- ASSIGNED
- CAX_MODEL_TYPE
- CAX_MODEL_REF
- CAX_LINK_TYPE
- CAX_LINK_REF
- CAX_TYPE

To remove columns from the Save and Load Preview windows, delete the corresponding field from the `FieldCollection` of the `TableDisplay` section. The column must be removed from both the external `OraclePLM\ini\CAXConfig.xml` and the internal `OraclePLM\jar\agile9\a93ws_connector.jar\com\xplm\agile93\cax\xml\CAXConfig.xml` files.

To add columns only external `CAXConfig.xml` should be modified.

To control the columns visible in the Save Preview and Load preview windows, click on the drop-down button on the right-hand side after the last column, and select and deselect columns as needed.

9.1.1.7 WorkspaceTableDisplay Section

This section contains information on the columns of the table displayed in the Workspace Manager window.

It can be manipulated in the same way as the `TableDisplay` section. Refer to TableDisplay Section (p. 94) for details, all information given in this chapter apply for the `WorkspaceTableDisplay` section as well, apart from the functionality of the `Value` tags. `Fields` displayed in the Workspace Manager are **never** editable.

9.1.1.8 OverrideConfiguration Section

The MCAD connectors allow filtering the sub-classes and number generators available in the GUI based on CAD file types.

See Filtering SubClasses and AutoNumber Generators Displayed by the MCAD GUI (p. 40) for more information.

9.1.1.9 Viewables Section

This section controls the viewable types that can be created by the MCAD connector.

The supported viewable types are defined in a semi-colon separated list.

Setting	Description
ViewablesDrawing	Defines viewable file types for drawing files visible in the Preferences dialog. **Possible values:** List of viewable file endings (use semi-colons as list separator).
ViewablesModel	Defines viewable file types for 3D models files visible in the Preferences dialog. **Possible values:** List of viewable file endings (use semi-colons as list separator).

Refer to How to Configure Viewable Formats Creation (p. 60) for more details.

9.1.1.10 WorkspaceDeleteViewables Section

Defines the viewable file types (file endings) the MCAD connector's **Delete Viewables** functionality deletes from a workspace folder.

The viewable types defined in the `value` tag in this section should match those given the `viewables` section.

Setting	Description
ViewablesDrawingExtensions	Defines the drawing file formats, which should be deleted from the local workspace by the **Remove Viewables** function in the Workspace Manager.
ViewablesModelExtensions	Defines the 3D file formats, which should be deleted from the local workspace by the **Remove Viewables** function in the Workspace Manager.

9.1.1.11 PartFamilies Section

Setting	Description
FamilySelection	If set to `true`, operations (for example, Check Out) applied to a component of a part family (configuration) through the MCAD connector are implicitly applied to all other components of the same part family. **Default:** false **Possible values:** true | false
FamilyInstanceNumbering	Defines the numbering schema for Design objects of instances of part families (configurations). ■ **GENERIC_INDEX** – Instances get the same Design number as the part family generic assigned, appended with a numerical index. ■ **GENERIC_CONFIG** – Instances get the same Design number as the part family generic assigned, appended with the CAD specific name of the instance. ■ **false** – Instances get independent Design numbers assigned. **Default:** GENERIC_INDEX **Possible values:** GENERIC_INDEX | GENERIC_CONFIG | false
CountDelimiter	Separator character used for appending part family (configurations) related appendices (see FamilyInstanceNumbering) or indices for multiple drawings which are related to the same model (see DrawingNumberFromModel). **Default:** _
PartInstanceDashNumbering	`true` - the Part item corresponding to part family instance get the suffix from the design instance (like *P001-001*,…) `false` - the Part item corresponding to part family instances get own item numbers, no instance suffix. **Default:** false **Possible values:** true | false Entry does not exist in default installation and must be manually added to the configuration file.

Setting	Description
PartGenericHasDashNumber	`true` – the Item object corresponding to the generic Design gets a number with dash assigned. For example, *P001-000* for generic and *P001-001* for instances. `false` - the Item object corresponding to the generic Design does not get a number with dash assigned. For example, *P001* for generic and *P001-001* for instances. **Default:** false **Possible values:** true | false Entry does not exist in default installation and must be manually added to the configuration file.
PartInstanceDelimiter	Separator character between the number and the counter for part variants **Default:** - Entry does not exist in default installation and must be manually added to the configuration file.

9.1.1.12 ChangeProperties Section

Setting	Description
InitialRevision	Initial Item revision value. **Default:** Introductory
RevisionSequence	Defines the proposed Revision values the MCAD connector implicitly assigns to Change objects on Change creation. You can have more than one `RevisionSequence` based on LifeCyclePhases. For example `RevisionSequence_Preliminary` **Default:** Introductory,A,B,C,D,E,F,G,H,I,J,K,L,M….. **Possible values:** List of Revision values (strings) separated by comma The RevisionSequence must always start with the value `Introductory` as that is the value implicitly used by the server.
RevisionSequenceEditor	Defines, if the MCAD connector provides a combo box (drop down list) for editing the Item Revision columns. The entries of the combo box are taken from the option setting `RevisionSequence`. If set to `false` and if the Revision field is editable, users can enter free text into the Revision field. **Default:** false **Possible values:** true | false

Setting	Description
SetLifecyclePhase	Defines, if the MCAD connector provides a combo box (drop down list) for editing the **LifecylcePhase** column. If set to `false` and if the LifecylcePhase field is editable, users can enter free text. **Default:** true **Possible values:** true | false
DesignRevisionLogic	Comma-separated list of the following values. If set the value controls the following behavior: ■ `increment` - increase minor number on checkout ■ `editable` - the Design revision field is editable in the save preview. ■ `publish` - the Design revision is reset on publish to fit the item revision code. The minor revision is calculated or reset. ■ `noparentheses` - remove any parentheses from the design revision **Default:** editable,publish,noparentheses
InitialDesignRevision	If no Item object is assigned to a Design, this value defines the `Item Revision` string displayed for the versions of this Design object. **Default:** 1 **Possible values:** Any string value
DesignRevisionForIntroductory	Defines the string, that replaces the `Introductory` Item Revision in the Design version drop-down list. By changing this option setting, it is possible to use other string values for representing the Introductory Revision. **Default:** 1 **Possible values:** Any string value
AssignChangeCheckout	If `true`, this triggers an implicit check out of the related Design object when assigning a Change object. Does not work if **ForceUserCheckout** is set to `true`. **Default:** false **Possible values:** true | false
VersionSeparator	Separator character used for separating the major and minor Item Revision displayed in the Design version drop-down list. Has no effect if the minor version is de-activated. **Default: .**

Setting	Description
PartDesignNumbering	If set to `true`, on Item number retrieval (for example, when pressing the Item autonumber button), the MCAD connector implicitly assigns the retrieved Item number to the Design number of the related Design object. **Default:** true **Possible values:** true | false
DrawingNumberFromModel	Defines the numbering schema for Design objects of CAD drawings: If set to true, the Design number of the drawing's Design object gets a number equal to the Design number of the underlying model, appended with a numerical index. **Default:** true **Possible values:** true | false
PushPartRevisionToDesign	If set to `true`, when changing the Item Revision in the MCAD connector's GUI, the changed Revision is automatically transferred to the Item Revision displayed in the Design version drop-down list. **Default:** true **Possible values:** true | false
PushDesignRevisionToPart	If set to `true`, when changing the Design version in the MCAD connector's GUI, the changed version is automatically transferred to the Item Revision. **Default:** true **Possible values:** true | false
PublishAttachments	Defines the attachment types, identified by their file ending, that are attached to an Item object when executing a publish process. **Default:** any **Possible values:** any | Comma separated list of file endings (without "."), for example `PDF,STEP,IGES,IGS`
PublishAttachmentType	The value in this option setting is written to the **Attachment Type** PLM field when executing a publish process. If set to `NONE`, nothing is written to that field. **Default:** NONE **Possible values:** none | Any string except `none`
PublishIntroductory	If set to `true`, the MCAD connector also executes publish processes for Items without assigned Change objects (means for Items with `Introductory` Item Revision). **Default:** true **Possible values:** true | false

Setting	Description
PublishPartSite	Defines the `Site` to which BOM changes are written when executing a publish process, if **Sites** are enabled in Agile PLM. Use the value `none` if **Sites** are disabled. **Default:** none **Possible values:** none | Any string except `none`
MaxDesignNumberLength	Defines the maximum allowed length of the Design number field values. Any Design number entries longer than the specified length are truncated by the MCAD connector. **Default:** 50 **Possible values:** Any integer value
AllowedNumberCharacters	Defines a set of characters which may be used for Item and Design numbers. The MCAD connector replaces any character not contained within this list with underscores ("_") in the Item and Design number fields. Use the value `ANY` to disable this functionality. This functionality does not affect autonumber assignment, Autonumber retrieved from Agile PLM are never changed by the MCAD connector. **Default:** ANY **Possible values:** ANY | Set of characters without delimiters, for example: `abcdefghijklmnopqrstuvwxyz1234567890`
Change_Category	Defines the default Change Category value for Item Change Orders created by the MCAD connector. **Default:** none **Possible values:** Any string except `none`, which exists in the drop-down list of the Change Category attribute for Item Changes in Agile PLM.
PublishErrorHandling	Defines the behavior of the MCAD connector if errors during the publish process occur. ■ If set to `ERROR`, the save, check in and publish process is completely cancelled if an error occurs. ■ If set to `WARNING`, an error report is displayed in the Save Summary window and only the publish process is cancelled, not the save or check in. **Default:** WARNING **Possible values:** ERROR | WARNING

Setting	Description
PublishIntegerQuantities	If set to `true`, the MCAD connector transfers quantity values (Design structure, BOM) in integer representation instead of a number with a decimal place. **Default:** false **Possible values:** true | false
LCPPolicy	Since release 3.6.4.6, you can defines the Life-cycle phase (LCP) set when assigning changes through the *Save Preview*. ■ `DEFAULT` - The LCP of an assigned change is set to the default Life-cycle phase defined in the *Preferences dialog* ■ `OLD` - The new LCP of an assigned change is gotten from the previous version if not empty or set to `Preliminary`, otherwise, it is gotten from default LCP value set in the *Preferences dialog*. If item has an pending ECO created from *Introductory* revision, the LCP will be set to `Preliminary` because it is the old LCP for such ECOs. **Default:** DEFAULT **Possible values:** DEFAULT | OLD

9.1.1.13 FileNaming Section

This section provides the configurations for the file names that should be used for viewable files created by the MCAD integration.

Depending on the file extension different rules are possible. However, for each file extension only one rule is supported. The Field `Name` represents the file ending of the viewable file while the Field `Value` represents the actual naming rule.

```
<Structure>
  <Name>FileNaming</Name>
  <FieldCollection>
    <Field><Name>PDF</Name><Value>%Item.NUMBER%_%Item.REV%.PDF</Value></Field>
    <Field><Name>STP</Name><Value>%Item.NUMBER%_%Item.REV%.STP</Value></Field>
    <!--Field><Name>JT</Name><Value>%NUMBER%_%REV%.JT</Value></Field-->
    <Field><Name>PNG</Name><Value>thumbnail.png</Value></Field>
  </FieldCollection>
</Structure>
```

Refer to Viewable File Naming (p. 61) for more information.

9.1.1.14 READONLY_FF_FIELDS Section

This XML structure defines all File Folder PLM fields which should not be updated by the MCAD connector (read-only fields), defined by their field ID.

Though the XML definition is based on a `name-and-value` syntax, the `value` field itself is not evaluated by the connector.

```
<Structure>
 <Name>READONLY_FF_FIELDS</Name>
 <FieldCollection>
  <Field><Name>2000025502</Name><Value></Value></Field>
  <Field><Name>2000025503</Name><Value></Value></Field>
            <!-- snip -->
 </FieldCollection>
</Structure>
```

The fields already defined in this section should not be removed from `CaxConfig.xml` file.

9.1.1.15 READONLY_ITEM_FIELDS Section

This XML structure defines all Item fields which should not be updated by the MCAD connector (read-only fields), defined by their field ID.

Though the XML definition is based on a `name-and-value` syntax, the `value` field itself is not evaluated by the connector.

```
<Structure>
 <Name>READONLY_ITEM_FIELDS</Name>
 <FieldCollection>
  <Field><Name>YYY</Name><Value></Value></Field>
 </FieldCollection>
</Structure>
```

9.1.1.16 CAX_NAMES_BY_ID Section

Allows executing an attribute mapping during save as operations.

Refer to Mapping Values on Save As (p. 53) for more information.

9.1.2 Settings in XPlmSolidWorksConnector.xml

This section describes the available settings and valid values for the connector in `XPlmSolidWorksConnector.xml`.

Setting	Description
EnableSolidWorksLogging	Enables and disables logging for the SOLIDWORKS connector. **Default:** false **Possible values:** true | false

Setting	Description
SolidWorksLogFile	Path and name of the log file for the SOLIDWORKS connector. **Default:** C:\SolidWorks.log **Possible values:** any valid path
SolidWorksLogLevel	Defines the level of detail of the log file. A higher number indicates more log messages. **Default:** 10 **Possible values:** 10 | 100
EnableSolidWorksAddinLogging	Enables or disables logging for the SOLIDWORKS Addin. **Default:** false **Possible values:** true | false
SolidWorksAddinLogFile	Path and name of the log file for the SOLIDWORKS Addin. **Default:** C:\SolidWorks_Addin.log **Possible values:** any valid path
SolidWorksAddinLogLevel	Defines the level of detail of the log file. **Default:** 300 **Possible values:**
IgnoreMissingParts	If set to `true`, missing parts are ignored. If set to `false`, an error message is thrown for missing parts. **Default:** true **Possible values:** true | false
SolidWorksStandardPartDir	Directory where standard parts should be recognized during check in. **Default:** C:\SOLIDWORKS Data
AllowRecursiveStructure	If set to `true`, transfer of recursive structure is allowed. **Default:** true **Possible values:** true | false

Setting	Description
SolidWorksDefaultSaveName_1 SolidWorksDefaultSaveName_2 SolidWorksDefaultSaveName_3	Defines the save name for a SOLIDWORKS file that is not saved locally to disk and the integration does the first local save. If the default save name already exists in the workspace folder, the integration automatically adds an increment number. **Default:** SolidWorksDefaultSaveName_1: Part.sldprt SolidWorksDefaultSaveName_2: Assembly.sldasm SolidWorksDefaultSaveName_3: Drawing.slddrw
SWAddins	Additional Add-ins to be loaded. **Default:** %XPLM_SWX_ADDIN%
SolidWorksSearchOptionPath	Allows the manipulation of the SOLIDWORKS search path (**Options** > **File Paths** > **Search Paths**) during integration load processes. The path given in this option setting is added to the standard search path. Any valid directory path can be used as option value. **Default:** -no value-
SolidWorksScriptEngine	Do not change! **Default:** intern
SolidWorksCSVSeparator	It is possible to import SOLIDWORKS object meta data via CSV data. CSV file needs same file name like SOLIDWORKS object. Therefor this property defines the separator in the CSV file. **Default:** @ **Possible values:** any special character
SolidWorksEvent_StartNotify	If set to `true`, integration starts up immediately. If set to `false`, the integration starts on demand when any integration function is called from the Agile ribbon inside SOLIDWORKS. The integration must be started before **Load to CAD** can be used. **Default:** true **Possible values:** true | false
SolidWorksEvent_ PartModifyNotify	If set to `true` and strict workflow is configured user will be informed when modifying a file which is not checked out. **Default:** false **Possible values:** true | false

Setting	Description
SolidWorksEvent_ AssemblyModifyNotify	If set to `true` and strict workflow is configured user will be informed when modifying an assembly which is not checked out. **Default:** false **Possible values:** true | false
SolidWorksEvent_ DrawingModifyNotify	If set to `true` and strict workflow is configured user will be informed when modifying a drawing which is not checked out. **Default:** false **Possible values:** true | false
SolidWorksEvent_ PromptForFilenameNotify SolidWorksEvent_ Assembly_OnNewSelection SolidWorksEvent_ RenameItemNotify SolidWorksEvent_ PreRenameItemNotify SolidWorksEvent_ FileOpenPreNotify	SOLIDWORKS events which are not used in the standard integration. They can be used to call methods in a customer script engine, if needed. **Default:** false **Possible values:** true | false We strongly recommend that you contact XPLM for implementation.
SolidWorksTiffDPI	Dots per inch of tiff **Default:** 300
SolidWorksLoad DrawingAsyncCheckInterval	Defines time in milliseconds to wait for checking whether a drawing has finished opening in SOLIDWORKS in asynchronous mode. Can be useful to open drawings which require a lot of CPU time and no user input. If set to `0`, drawings will be opened in synchronous mode. **Default:** 0 **Possible values:** any integer value
ExcludeVirtualFromMissingFiles	If set to `true`, excludes virtual components from the list of missing files. **Default:** false **Possible values:** true | false

Setting	Description
BulkLoaderHandling	Set this to `true` when traversing model data with document manager API (in BulkLoader scenarios) Ensures the integration gets structure information even for attached non-SOLIDWORKS CAD models (e.g. external references) **Default:** false **Possible values:** true | false

9.1.3 Settings in XPlmSolidWorksA9Connector.xml

This section describes the available settings and valid values for the connector in `XPlmSolidWorksA9Connector.xml`.

Setting	Description
EnableScriptEngineLogging	If set to true, logging is enabled. **Default:** false **Possible values:** true | false
ScriptengineLogfile	Path to a log file, required if logging is enabled. If you do not specify a path the log file is written to `%USERPROFILE%\AgileCache` folder. **Default:** xacw.log
ScriptengineLogLevel	Defines the level of detail of the log file. A higher number indicates more log messages. **Default:** 10 **Possible values:** 10 | 100
SolidWorksMenuFiles	The add-in menu file in the `xml` directory. **Default:** XPlmSolidworksA9Addin.xml
RenameOnCreate	If renaming of files created by the **Create New Object** function is requested, set value to a renaming rule for example, NUMBER **Default:** blank
RenameOnLoad	■ `NUMBER:` Renamed file names are equal to the Agile PLM number. ■ `CAX_FIL_NAME:` Do not rename on load. **Default:** NUMBER **Possible Values:** Any other value

Setting	Description
RebuildOnLoad	If `true`, before opening them at the end of the load process, all assemblies in which referenced documents were renamed, will be rebuilt. **Default:** false **Possible values:** true | false
RenameOnInitialSave	If set to `true`, files are renamed on initial save by copying them and then saving the copies to Agile PLM. CLB files are created for both the original files and the copied and renamed files Checked into Agile PLM. **Default:** false **Possible values:** true | false Do not use this option if you have large assemblies or external references and suppressed components. Use the `RenameOnLoad` feature in these cases!
RenameOnSave	If set to `true`, files are renamed on **Save As**. **Default:** false **Possible values:** true | false Do not use this option if you have large assemblies or external references and suppressed components. Use the `RenameOnLoad` feature in these cases!
RenamingRule	Renaming rule for building the filename. This is a reserved option setting, always set the default value. **Default:** %CAX_NEW_NAME%
SolidWorks_Allow ReopenOnRename	If set to `true`, renamed files are reopened in CAD. **Default:** true **Possible values:** true | false
ConfiguredDefault	Controls the default behavior for handling configurations. ■ If set to `true`, each configuration is treated as a separate Design object. ■ If set to `false`, no Design object is created for each configuration. The default can be overridden by the file property set in the `ConfiguredProperty`. **Default:** true **Possible values:** true | false

Setting	Description
ConfiguredProperty	If the given property name is contained in a SOLIDWORKSfile and set to `No` then no configurations are created in Agile PLM. If set to `Yes` then configurations are created in Agile PLM. **Default:** Configured
ConfiguredProperty2	Additional property name to identify configured files. If property name is contained in a file and set to `No` then no configurations are created in Agile PLM. **Default:** Configured2
ConfiguredValue_Configured	Optional value of the `ConfiguredProperty` or `ConfiguredProperty2` that would be interpreted as `Yes` (configured). **Default:** yes **Possible values:** yes | no
ConfiguredValue_NotConfigured	Optional value of the `ConfiguredProperty` or `ConfiguredProperty2` that would be interpreted as `No` (not configured). **Default:** no **Possible values:** yes | no
MasterConfigProperty	If the given property name is contained in the configuration specific properties and the value of this property points to an existing configuration in the same file, the linked configuration is used, and no extra configuration object is created in Agile PLM. **Default:** MasterConfig
FindPLMObject	Whether to search for existing configurations and files in Agile PLM. **Default:** true **Possible values:** true | false
SolidWorks_Disable UpdateDrawingBOM	Whether to reset the update flag in drawings to suppress updating the parts list on Load. ■ If `false`: SOLIDWORKS drawing BOM will be updated when you add or delete components in the associated assembly. This may happen during Save or Load process. ■ If `true`: SOLIDWORKS drawing BOM will NOT be updated. **Default:** false **Possible values:** true | false

Setting	Description
SolidWorks_Always ExtractExternalReferences	If set to `true`, external references are always traversed. If set to `false`, external references are traversed on demand with user prompt only. **Default:** false **Possible values:** true | false
AppendPLMFieldsTo ViewableNames	Whether to append additional Agile PLM fields into the viewables file-names. **Default:** true **Possible values:** true | false
AppendingRuleViewables	FieldID or CAX field names to append. Format like. **Default:** %REV%
SolidWorks_BulkLoad_ AllConfigurations	Bulkloader switch to create all or only use configurations on initial import. **Default:** false **Possible values:** true | false
SolidWorks_BulkLoad_ CollectDrawings	If set to true, drawings of related model are collected during bulk load to be saved to PLM, too. If set to false, integration does not search for related drawings during bulk load. **Default:** false **Possible values:** true | false
SolidWorks_InstanceViewables	Viewable type to create for configurations that have no real 3D model file. If you set this option, depending on the assembly complexity the generation is very time and resource consuming. It is not recommended to use this on big assemblies because SOLIDWORKS needs to regenerate each saved configuration, which can lead to unstable SOLIDWORKS behavior. **Default:** false **Possible values:** X_T | false
SolidWorks_Rebuild DrawingOnSave	Perform a rebuild on saving a drawing. Should be set to true if sometimes model attributes do not reflect immediately into drawing title blocks. **Default:** false **Possible values:** true | false

Setting	Description
SolidWorks_Always SaveTopAssembly	If set to `true`, the top assembly will always be saved. **Default:** false **Possible values:** true | false
SolidWorks_ReserveBefore Modify_EnableUseCase	■ **0**- Use case *modify only CAD files which were checked out* is disabled, thus, status of SOLIDWORKS files (read only/editable) will not be changed by Agile EC . ■ **1**- Use case *modify only CAD files which were checked out* is enabled. During check out/cancel check out the file status will be adapted (read only/editable) depending from resulting checkout status. **Default:** 0 **Possible values:** 0 | 1 See Strict CAD Modification Workflow (p. 35) for more information.
SolidWorks_ReserveBefore Modify_NotifyUser	If it is set to `1` and `SolidWorks_ReserveBeforeModify_EnableUseCase` is also set to `1`, the user will be informed when modifying files which are not checked out. **Default:** 0 **Possible values:** 0 | 1 See Strict CAD Modification Workflow (p. 35) for more information.
SolidWorks_ReserveBefore Modify_AutoReserve	If it is set to `1` and `SolidWorks_ReserveBeforeModify_EnableUseCase` is also set to `1`, files which are modified will be reserved automatically. **Default:** 0 **Possible values:** 0 | 1 See Strict CAD Modification Workflow (p. 35) for more information.
SolidWorks_StartWSM WithoutCADStructure	If set to `1`, structure of active SOLIDWORKS document is NOT analyzed and an empty Workspace Manager **CAD** tab is shown. If set to `0`, the structure of active SOLIDWORKS document is analyzed before opening Workspace manager and displayed in the **CAD** tab. **Default:** 0 **Possible values:** 0 | 1

Setting	Description
SolidWorks_BatchCreate DuplicateJobOnFailure	Relevant for Batch Queue processing. One batch job can contain more than one task (action). If this is the case and one of n tasks fails (n>1) then the original job will get status `Error`. If set to `true`, a new job will be created containing only the erroneous task. This new job can later be easily started to be executed after repairing the error cause. **Default:** true **Possible values:** true | false
SolidWorks_BatchRun PropertyUpdate	If `true`, SOLIDWORKS file properties will be updated from PLM values during batch job execution. **Default:** true **Possible values:** true | false
SolidWorks_ExcludeBroken ReferencesFromSave	Excludes external references that are labelled as broken in SOLIDWORKS (**External References dialog** > **Break Selected or Break All**) from the save process of the MCAD connector. If this option setting is enabled (value: true), the references in question do not appear in the Save Preview and subsequently they are not saved to Agile PLM. **Default:** false **Possible values:** true | false

9.2 Tools and add-ons

9.2.1 About this chapter

This chapter provides a basic understanding of the tools that come with the installer or are available separately as add-ons.

This chapter covers the following components:

9.2.2 Batch Queue Admin

9.2.2.1 Introduction

Please consult XPLM services for setup and configuration of this component.

9.2.3 Batch Server

9.2.3.1 Introduction

Please consult XPLM services for setup and configuration of this component.

9.2.4 Encryption Helper

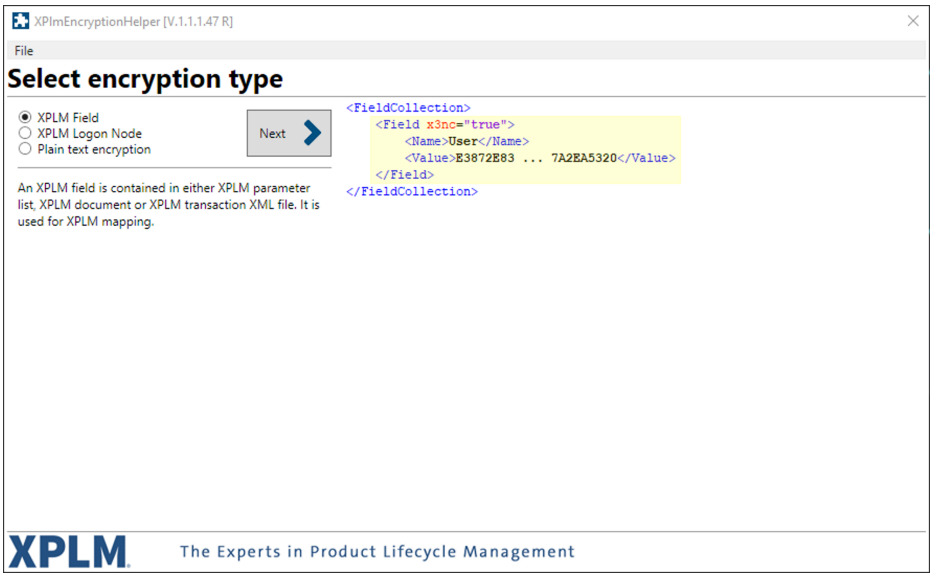
9.2.4.1 Introduction

The Encryption Helper is used to encrypt fields that must not be read in plain text, for example passwords.

You can encrypt required passwords and other information to prevent the value from appearing as plain text in configuration files. This is true for most configuration files, at least for login parameters and for regular transactions written in the current data model.

Make sure that you keep the original password in a safe place if you want to encrypt it. It is not possible to recover the original value after encryption.

Figure 1: User interface



9.2.4.2 Usage

The application is installed from the integration installer in the step *Tool components*. After installation, a shortcut to start the application is placed on the desktop.

About this task

The Encryption Helper can encrypt a defined field, a logon node or just any plain text. It provides examples on how to use the encrypted information.

Procedure

1. Start Encryption Helper via the shortcut on the desktop.
2. Select an encryption type and click **Next**. Click **Back** to select another type.
3. **XPLM Field**: Enter the name of the field and its value, and click **Encrypt**.
4. **XPLM Logon Node**: Enter the name of the field and its value, and click **Encrypt**.
5. **Plain text encryption**: Enter the plain text and click **Encrypt**.

Result

The entered information is encrypted. Click **Copy** to copy the information to the clipboard and use it in the desired file.

9.2.5 Integration Creator

9.2.5.1 Introduction

Integration Creator is a tool for compiling integration code manually. This additional installation step is required if certain legal agreements or contractual conditions prohibit XPLM to ship already compiled integration code in the installer.

Who needs Integration Creator?

- **New customers** must use Integration Creator to compile the integration code for the listed integrations themselves. After compilation, the overlay must be distributed to the client computers running the integration.
- **Existing customers** updating the integration do not require Integration Creator and will receive an overlay from XPLM that already contains the compiled integration code with other updated files. This overlay must be distributed to the client computers running the integration.

If you are working as a contractor for a customer, you must perform this code compilation in the customer's environment using their own licenses.

Scope of Integration Creator

Integration Creator must be used for the following integrations:

- SOLIDWORKS to
  - Aras Innovator (`Aras-SolidWorks_*.7z.exe`)
  - Oracle Agile EDM (`OracleEDM-SolidWorks_*.7z.exe`)
  - Oracle Agile PLM (`OraclePLM-SolidWorks_*.7z.exe`)
  - SAP PLM (`SAP-MCAD_*.7z.exe`)

System requirements for build machine running Integration Creator

- Windows 10/11 x64 with local administrator rights
- Min. 40 GB free disk space
- Visual Studio Build Tools 2019 (components to be installed require > 30 GB disk space)
- SOLIDWORKS in a supported and same version as on the client computers, see System requirements (p. 12)

Administrator tasks (new customers)

- Compiling integration code (p. 118)
- Requesting and entering SOLIDWORKS API key (p. 119)
- Distributing compiled integration code (p. 120)

Administrator tasks (existing customers)

Distribute the overlay you have received from XPLM as part of an update to the client computers running the integration. See Distributing compiled integration code (p. 120) for more information.

9.2.5.2 Compiling integration code

To compile integration code with Integration Creator, complete the following steps.

Procedure

1. On the build machine, install SOLIDWORKS in a supported version.

   Select the **Sample files** when installing SOLIDWORKS. These files are required by Integration Creator.

2. Copy the installer archive `*.7z.exe` to the build machine.
3. Extract the archive and start `setup.exe` with administrator rights.
4. Select components as required, and select in the step *Tools* the option **Integration Creator**.
5. Complete installation.

   ➔ After installation, a shortcut to Integration Creator is placed on the desktop.

6. Install Visual Studio build tools:

   a) Download the latest build tools for Visual Studio 2019 from the Microsoft website.

   b) Execute the downloaded installer `vs_BuildTools.exe` with administrator rights and follow the instructions.

   ➔ The install window *Visual Studio Build Tools 2019* appears.

   c) Don't select anything and click **Install**.

   ➔ After installation, the window *Visual Studio Installer* appears with the installed product *Visual Studio Build Tools 2019*.

   d) In the section **Visual Studio Build Tools 2019**, select **More** > **Import configuration** and select the file `%xPlmRootDir%\accessories\xplm.vsconfig`.

   e) Click **Review details** and commit the dialog of missing component IDs.

   ➔ The install window *Visual Studio Build Tools 2019* appears again with the required components selected under **Installation details**.

   f) Click **Modify**.

   ➔ Required components are downloaded and installed. If existing components are no longer supported, a window appears. Click **Continue** to proceed with installation.

7. Start the shortcut Integration Creator with administrator rights.

   ➔ A single window with the button **Create** appears.

8. Click the button **Create**.

   ➔ The integration code is compiled and copied with other updated files to the installation directory `%xPlmRootDir%`.

9. After compilation, a success message appears. In this step you can add the compiled code to a new overlay, or add it to an existing one. Click **Yes** and select a directory where the overlay should be created.
10. Provide the SOLIDWORKS API key, see Requesting and entering SOLIDWORKS API key (p. 119) for more information.

Result

The integration code is compiled and required files are added to the overlay.

9.2.5.3 Requesting and entering SOLIDWORKS API key

To run the integration, customers must request and enter a valid key for the SOLIDWORKS document manager API. Without this key, the integration fails to start.

Procedure

1. Click the link https://www.solidworks.com/support/subscription/key-request/ and login with your credentials.
2. Select the following options and create the key:
   - Product version: Select the correct SOLIDWORKS version in use.
   - Requested functionality: Select all options.
   - Reason for use: Internal use only
   - Specify application: SOLIDWORKS
   - Partner DM category: PLM Integration
3. Edit file `%xPlmRootDir%\xml\XPlmSolidWorksDocumentmanager.xml` and enter the key in the section that corresponds to the SOLIDWORKS version in use, for example SOLIDWORKS 2023:

```
<Field>
  <Name>SolidWorksDocumentManagerKey_SW2023</Name>
  <Value>KEY GOES HERE</Value>
</Field>
```

4. Save the file.
5. Add the file to the created overlay under the directory `xml`, for example:

```
myOverlay
├──bin
│   ├──x64
│   └──x86
└──xml
```

Result

The SOLIDWORKS API key is defined in the XML file and added to the overlay.

9.2.5.4 Distributing compiled integration code

The compiled integration code and other updated files in the overlay must be distributed to the client computers running SOLIDWORKS.

New customers

1. On the build machine, complete tasks as described in chapter Initial setup (p. 16).
2. Test the integration.
3. Add changed files also to the created overlay.
4. Distribute the overlay as part of a new installation:

   If you want to distribute the overlay on a larger scale, you should use silent-mode. For only a few installations, you can use the installer or manually copy the files over an existing installation.

   - To install the integration and apply the overlay in silent-mode, create a batch script for silent-mode installation calling the individual MSIs and the overlay, and run it on the client computers. See the related links for more information.
   - To install the integration and apply the overlay using the installer, start the installer. In the step *Installation path*, enable the option **Apply custom files after installation** and enter the path to the overlay directory.
   - To apply the overlay manually over an existing installation, copy the content from the overlay to the installation directory `%xPlmRootDir%`, overwriting existing files.

Existing customers

During an integration update, you will receive the overlay directly from XPLM. To distribute the overlay, use the established deployment methods in your company.

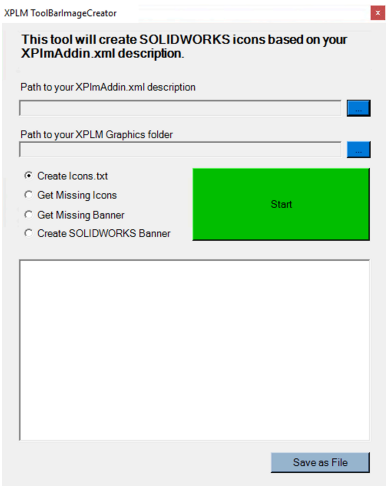
Related links
Using overlay packages (p. 123)
Silent installation (p. 123)

9.2.6 Toolbar Image Creator for Dassault Systemes SOLIDWORKS

9.2.6.1 Introduction

To show button icons in the ribbon toolbar, SOLIDWORKS uses combined banner images with all icons defined for ribbon groups. If the add-in is modified, the *Toolbar Image Creator for Dassault Systemes SOLIDWORKS* is used to re-create these banner images and to show the buttons correctly in the menu.

Figure 2: User interface



9.2.6.2 Configuration

This chapter describes how the add-in in SOLIDWORKS is configured. Understanding this is required to make changes to the add-in and to use the application *Toolbar Image Creator for Dassault Systemes SOLIDWORKS*.

Button texts

Configuration file: `%xPlmRootDir%\xml\XPlmSolidworksA9AddinResource.xml`

This file contains the language-specific definitions for ribbon groups, button captions, tooltips and status bar texts. This file is the basis for the file `XPlmSolidworksA9Addin.xml` in the same directory.

Buttons and ribbon groups

Configuration file: `%xPlmRootDir%\xml\XPlmSolidworksA9Addin.xml`

In this file, buttons and the ribbon groups are defined.

The buttons are defined in the section `Buttondefinition`:

```
...
  <Buttondefinition>
    <Button id="..." caption="..." tooltip="..." statusBarText="..."
    callbackMethod="..." bitmap="..." <optional settings>
  </Buttondefinition>
...
```

- `id`: Defines the unique button ID.
- `caption`, `tooltip`, `statusBarText`: Defines references to resource keys in the file `%xPlmRootDir%\xml\XPlmSolidworksA9AddinResource.xml`.
- `callbackMethod`: Defines a function to execute when clicking the button.

- `bitmap`: Defines the location of the icon in the directory `%xPlmRootDir%\graphics\A9-SolidWorks`.
- `<optional settings>`: Defines additional button configuration, for example calling a different script engine, passing a parameter, executing pre- or post-actions or a macro.

The ribbon shown in SOLIDWORKS when opening a part, an assembly, a drawing, or if nothing is opened, is defined in the section `Ribbondefinition`:

```
...
  <Ribbondefinition>
    <Ribbon environment="..." tag="...">
      <Group id="..." name="...">
        <Button id="..." imageSize="..." />
      </Group>
    </Ribbon>
  </Ribbondefinition>
...
```

Each group also has a unique ID and references a button defined in section `Buttondefinition`.

To remove functions, you can comment single `Button` definitions or whole `Group` definitions. To create a new group, the group needs its own ID, a resource string and button definitions.

Icons

Graphic directory: `%xPlmRootDir%\graphics\A9-SolidWorks`

The button icons used in the integration menu are stored in this directory.

9.2.6.3 Usage

If you change the order of buttons or groups, execute the application Toolbar Image Creator for Dassault Systemes SOLIDWORKS. The application is installed from the integration installer in the step *Tool components*. After installation, a shortcut to start the application is placed on the desktop.

About this task

If you only make changes to the text definitions, these changes will take effect after you restart SOLIDWORKS.

Procedure

1. On the desktop, start the shortcut *SOLIDWORKS Toolbar Image Creator* with administrator rights.

   ➔ The application appears.

2. Define the path to the add-in file, for example `%xPlmRootDir%\xml\XPlmSolidworksA9Addin.xml`.
3. Define the path to the graphic directory, for example `%xPlmRootDir%\graphics\A9-SolidWorks`.
4. To just get icon information, select one of the following options and click **Start**:
   - **Create Icons.txt**: Creates an overview of the button icons and banner images used.
   - **Get Missing Icons**: Reports any icons that are defined but not found in the graphic directory.
   - **Get Missing Banner**: Reports any banner images that are not found in the graphic directory. Note that this function does not check if the content of the banner is correct.

   ➔ The information is compiled and shown in the window below. Click **Save as File** to export the result.

5. To create new banner images, select **Create SOLIDWORKS Banner** and click **Start**.
6. Restart SOLIDWORKS.

Result

The ribbon contains the buttons and groups as defined.

9.3 Using overlay packages

An overlay package usually contains modified configuration files. You can select an option in the installer to apply an overlay package as the last step of the installation process, copying the modified configuration over the installed files.

About this task

You can also perform this task in silent-mode.

Procedure

1. Manually install the product on a client computer using the installer in GUI-mode.
2. After installation, configure the product as required.
3. Create a new directory outside `%xPlmRootDir%`, for example `C:\my_config_files`.
4. Copy the modified files from the directories under `%xPlmRootDir%` to `C:\my_config_files`, creating also their relevant parent directories as in the original location, for example `C:\my_config_files\xml`.
5. Apply overlay package in GUI-mode:
   a) Start `setup.exe` and click **Modify** to change the installation.
   b) In the step *Installation path*, select the option **Apply custom files after installation** and the path to the overlay package.

   You can use UNC paths or network shares for overlay packages. This enables a fully automated deployment across the company.

Result

Custom files from an overlay package are copied as the last step of the installation over existing files.

9.4 Silent installation

You can install the integration in silent mode. Silent mode has the advantage that you can easily install the integration on client computers without user interaction.

9.4.1 Preparing silent installation

Complete these steps to install the integration in silent-mode without user interaction.

About this task

Create the file `silent.bat` with the installation configuration. The configuration would normally be defined in the installation wizard. In a silent installation, you use defined keywords and values in `silent.bat` to define the configuration.

Procedure

1. Copy the sample code and paste it in `silent.bat`:

```
START /WAIT msiexec /i packages\acx_full_3.6.4.2704_x64 /quiet INSTALLDIR="C:\Test"
 CHECK_NX="1" VERSION_NX="2206"
```

2. Adjust parameters as described in chapter Silent installation parameter (p. 124).
3. In the file `silent.bat`, edit values for parameter names to reflect the configuration.
4. Save the file as `silent.bat` in the same directory that contains the `setup.exe`.
5. Execute the file `silent.bat`.

Result

The integration is installed successfully without user interaction.

Next steps

The following code shows an example to install MSXML 4.0 and Visual C++ Redistributable 9.0:

```
START /WAIT msiexec /i msxml_4.30.2100_x86.msi /quiet
START /WAIT vcredist_9.0.30729.6161_x64.exe /q
```

9.4.2 Using custom files in silent installation

Complete these steps to create a silent installation using an overlay package with custom files.

Procedure

1. Create an overlay package with custom files and verify it works.
2. Prepare the silent installation and use parameter `CHECK_CUSTOM` to enable an overlay package and parameter `CUSTOM_FILES` to point to the directory containing the files.

Result

Custom files are applied correctly after the main silent installation is complete.

9.4.3 Silent installation parameter

The following table explains different parameters that can be used for executing a silent installation of the integration.

Table 2: Parameter and description

Parameter	Description
INSTALLDIR	Defines installation directory (for example `INSTALLDIR="C:\Test"`).
CHECK_CUSTOM	Copying custom files over created installation (value must be `"1"` to enable, for example `CHECK_CUSTOM="1"`)
CUSTOM_FILES	Place where custom files are located (for example `CUSTOM_FILES="C:\custom_files"`)
JAVA_HOME_X86	Defines installation path of Java 32-bit installation (for example `JAVA_HOME_X86="C:\Program Files (x86)\Java\jre[version]"`)

Parameter	Description
JAVA_HOME_X64	Defines installation path of Java 64-bit installation (for example `JAVA_HOME_X64="C:\Program Files\Java\jre[version]"`)
REBOOT_AFTER_FINALIZE	Performs a reboot after installation is completed (for example `REBOOT_AFTER_FINALIZE="1"`).
CHECK_ACD	Enables installation for Autodesk AutoCAD Mechanical (value must be `1` to enable, for example `CHECK_ACD="1"`).
VERSION_ACD	Defines version of Autodesk AutoCAD Mechanical (value must be `autocad` followed by the version shown in the GUI, for example `VERSION_ACD="autocad[version]"`).
CHECK_INV	Enables installation for Autodesk Inventor (value must be `1` to enable, for example `CHECK_INV="1"`).
VERSION_INV	Defines version of Autodesk Inventor (value must be `inventor` followed by the version shown in the GUI, for example `VERSION_INV="inventor[version]"`).
CHECK_CV5	Enables installation for Dassault Systèmes Catia V5 (value must be `1` to enable, for example `CHECK_CV5="1"`).
VERSION_CV5	Defines version of Dassault Systèmes Catia V5 (value must be the version shown in the GUI, for example `VERSION_CV5="[version]"`).
CHECK_PROE	Enables installation for PTC Creo Parametric (value must be `1` to enable, for example `CHECK_PROE="1"`).
VERSION_PROE	Defines version of PTC Creo Parametric (value must be the version shown in the GUI, for example `VERSION_PROE="[version]"`).
CHECK_PROE_ADDONS	Enables installation for Batch Server/Bulk Loader support (value must be `1` to enable, for example `CHECK_PROE_ADDONS="1"`).
CHECK_NX	Enables installation for Siemens NX (value must be `1` to enable, for example `CHECK_NX="1"`).
VERSION_NX	Defines version of Siemens NX (value must be the version shown in the GUI, for example `VERSION_NX="[version]"`).
CHECK_SED	Enables installation for Siemens Solid Edge (value must be `1` to enable, for example `CHECK_SED="1"`).
VERSION_SED	Defines version of Siemens Solid Edge (value must be `solidedge` followed by the version shown in the GUI, for example `VERSION_SED="solidedge[version]"`).
CHECK_SWX	Enables installation for SOLIDWORKS (value must be `1` to enable, for example `CHECK_SWX="1"`).

Parameter	Description
VERSION_SWX	Defines version of SOLIDWORKS (value must be `sw` followed by the version shown in the GUI, for example `VERSION_SWX="sw[version]"`).
CHECK_BATCH_QUEUE_ADMIN	Enables installation for Batch Queue Admin (value must be `1` to enable, for example `CHECK_BATCH_QUEUE_ADMIN="1"`).
CHECK_BATCH_SERVER	Enables installation for Batch Server (value must be `1` to enable, for example `CHECK_BATCH_SERVER="1"`).
BATCH_SERVER_INSTALL_TYPE	Defines type of Batch Server installation (possible values listed below, for example `BATCH_SERVER_INSTALL_TYPE="1"`). ■ Windows Interface: 1 ■ Windows Service: 2
CHECK_SWX_XCSH	Enables installation for the SOLIDWORKS xCSH support (value must be `1` to enable, for example `CHECK_SWX_XCSH="1"`).
CHECK_TOOLBAR_IMAGE_CREATOR	Enables installation for the SOLIDWORKS image toolbar creator (value must be `1` to enable, for example `CHECK_TOOLBAR_IMAGE_CREATOR="1"`).
CHECK_ENCRYPTION_HELPER	Installs Encryption Helper (value must be `1` to enable, for example `CHECK_ENCRYPTION_HELPER="1"`)