

Installation & Administration Guide

Autodesk Inventor integration
for
SOLIDWORKS PDM

Valid for product version: 2023 SP2.0 (1.6.2)

Published: 09.05.2023 | Build: 168 | Revision: 10605

Legal information

©1995-2020, Dassault Systèmes SolidWorks Corporation, a Dassault Systèmes SE company, 175 Wyman Street, Waltham, Mass. 02451 USA. All Rights Reserved.

The information and the software discussed in this document are subject to change without notice and are not commitments by Dassault Systèmes SolidWorks Corporation (DS SolidWorks).

No material may be reproduced or transmitted in any form or by any means, electronically or manually, for any purpose without the express written permission of DS SolidWorks.

The software discussed in this document is furnished under a license and may be used or copied only in accordance with the terms of the license. All warranties given by DS SolidWorks as to the software and documentation are set forth in the license agreement, and nothing stated in, or implied by, this document or its contents shall be considered or deemed a modification or amendment of any terms, including warranties, in the license agreement.

Patent notices

SOLIDWORKS® 3D mechanical CAD and/or Simulation software is protected by U.S. Patents 6,219,049; 6,219,055; 6,611,725; 6,844,877; 6,898,560; 6,906,712; 7,079,990; 7,477,262; 7,558,705; 7,571,079; 7,590,497; 7,643,027; 7,672,822; 7,688,318; 7,694,238; 7,853,940; 8,305,376; 8,581,902; 8,817,028; 8,910,078; 9,129,083; 9,153,072 and foreign patents, (for example, EP 1,116,190B1 and JP 3,517,643).

eDrawings® software is protected by U.S. Patent 7,184,044; U.S. Patent 7,502,027; and Canadian Patent 2,318,706.

U.S. and foreign patents pending.

Trademarks and product names for SOLIDWORKS products and services

SOLIDWORKS, 3D ContentCentral, 3D PartStream.NET, eDrawings, and the eDrawings logo are registered trademarks and FeatureManager is a jointly owned registered trademark of DS SolidWorks.

CircuitWorks, FloXpress, PhotoView360, and TolAnalyst are trademarks of DS SolidWorks.

FeatureWorks is a registered trademark of Geometric Ltd.

SOLIDWORKS 2018, SOLIDWORKS Standard, SOLIDWORKS Professional, SOLIDWORKS Premium, SOLIDWORKS PDM Professional, SOLIDWORKS PDM Standard, SOLIDWORKS Workgroup PDM, SOLIDWORKS Simulation, SOLIDWORKS Flow Simulation, eDrawings, eDrawings Professional, SOLIDWORKS Sustainability, SOLIDWORKS Plastics, SOLIDWORKS Electrical, SOLIDWORKS Composer, and SOLIDWORKS MBD are product names of DS SolidWorks.

Other brand or product names are trademarks or registered trademarks of their respective holders.

COMMERCIAL COMPUTER SOFTWARE - PROPRIETARY

The Software is a “commercial item” as that term is defined at 48 C.F.R. 2.101 (OCT 1995), consisting of “commercial computer software” and “commercial software documentation” as such terms are used in 48 C.F.R. 12.212 (SEPT 1995) and is provided to the U.S. Government 14 (a) for acquisition by or on behalf of civilian agencies, consistent with the policy set forth in 48 C.F.R. 12.212; or (b) for acquisition by or on behalf of units of the Department of Defense, consistent with the policies set forth in 48 C.F.R. 227.7202-1 (JUN 1995) and 227.7202-4 (JUN 1995).

In the event that you receive a request from any agency of the U.S. Government to provide Software with rights beyond those set forth above, you will notify DS SolidWorks of the scope of the request and DS SolidWorks will have five (5) business days to, in its sole discretion, accept or reject such request. Contractor/Manufacturer: Dassault Systèmes SolidWorks Corporation, 175 Wyman Street, Waltham, Massachusetts 02451 USA.

Copyright notices for SOLIDWORKS PDM Professional product

Outside In [®]Viewer Technology, © 1992-2012 Oracle

©2011, Microsoft Corporation. All rights reserved.

Contents

Legal information.....	ii
Glossary.....	6
1 General information.....	7
1.1 Introduction.....	7
1.2 How the integration works.....	8
2 Installation.....	9
2.1 Pre-installation information.....	9
2.1.1 Operating system support.....	9
2.1.2 Supported SOLIDWORKS PDM releases.....	9
2.2 Installing integration.....	9
3 Post-installation.....	11
3.1 SOLIDWORKS PDM.....	11
3.1.1 Add-in registration.....	11
3.2 Inventor.....	12
3.2.1 Add-In registration.....	12
3.3 Additional information.....	12
4 Configuration.....	13
4.1 SOLIDWORKS PDM.....	13
4.1.1 Setup tasks in SOLIDWORKS PDM add-in.....	13
4.1.2 Improving check in performance.....	14
4.1.3 Configuration files.....	14
4.1.4 Property mapping.....	17
4.1.5 BOM creation.....	21
4.1.6 Document creation.....	21
4.1.7 Enabling logging.....	22
4.2 Inventor.....	23
4.2.1 Configuration Files.....	23
4.3 Renaming.....	26
5 Update.....	29

5.1	Modifying installation.....	29
5.2	Repairing installation.....	29
5.3	Updating installation.....	30
6	Uninstallation.....	32
6.1	Uninstalling integration.....	32
7	Troubleshooting.....	33
7.1	Troubleshooting integration problems.....	33
7.2	License error codes.....	33
7.3	Adding drawings to vault leads to infinite loop.....	33
8	References.....	34
8.1	Using custom files.....	34
8.2	Silent installation.....	34
8.2.1	Preparing silent installation.....	34
8.2.2	Using custom files in silent installation.....	35
8.2.3	Silent installation parameter.....	35

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, parts 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.

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 SOLIDWORKS PDM 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 SOLIDWORKS PDM 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 General information

1.1 Introduction

The Inventor integration for SOLIDWORKS PDM provides functions to save and load design data to and from SOLIDWORKS PDM.

Concurrent engineering is supported through the use of reservation of individual objects and structures.

The integration uses the API from both applications. It is designed to extend existing Inventor and SOLIDWORKS PDM functionality.

The handling of Inventor structure and objects is done in Inventor itself. Inventor is the technical master application, whereas SOLIDWORKS PDM is the organizational master application. SOLIDWORKS PDM manages Inventor objects and structures that are important in the design and approves a possible workflow for individual design objects.

Functionalities of the integration are available via additional toolbars or extended menus in Inventor. They access both the available functions of Inventor and SOLIDWORKS PDM.

1.2 How the integration works

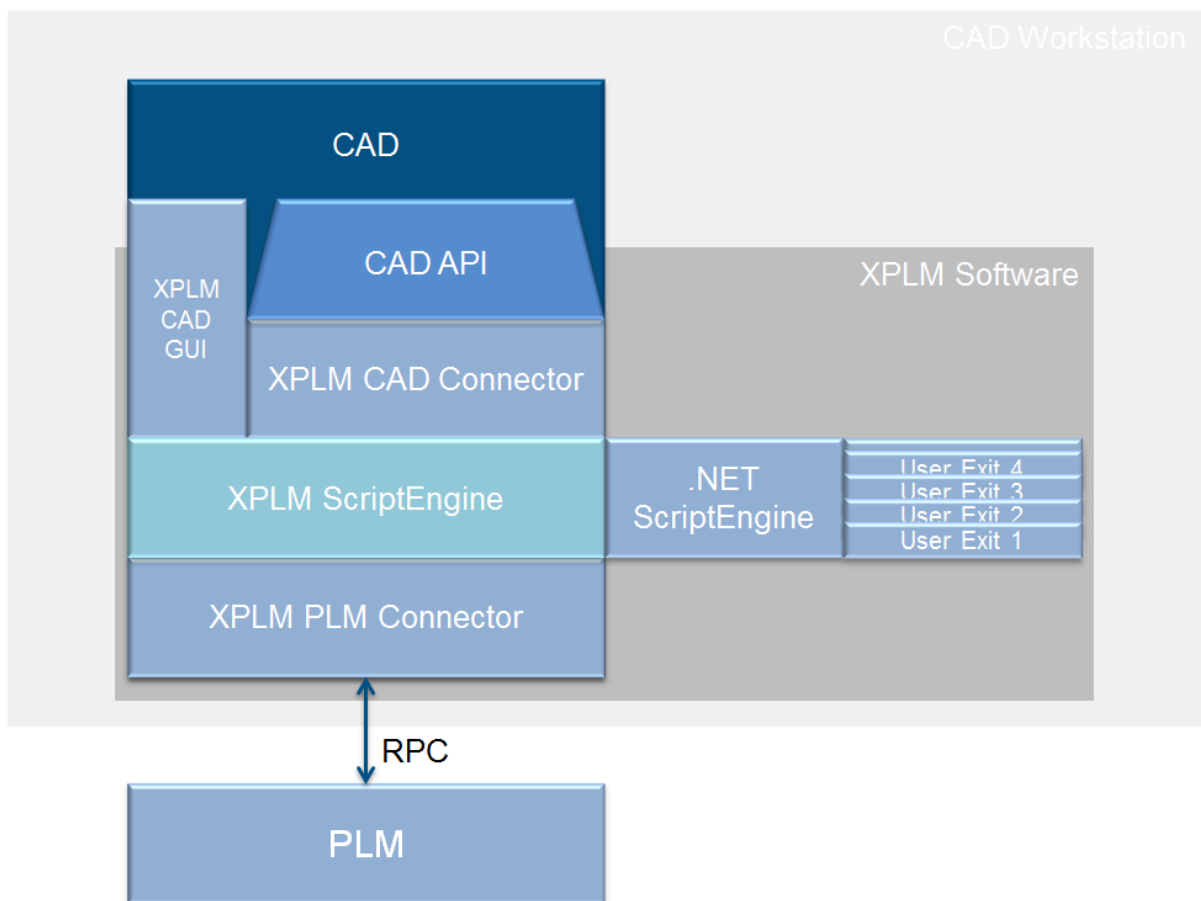
The integration is the interface between Inventor and SOLIDWORKS PDM and allows a consistent data exchange between the applications.

The Inventor connector encapsulates the CAD specific API into XPLM's standard methods to access and manipulate CAD data. Examples: retrieve structure and parameter information, update parameters, change object names.

The SOLIDWORKS PDM connector encapsulates the SOLIDWORKS PDM specific API into XPLM's standard methods to access and manipulate SOLIDWORKS PDM data. Examples: create and update CAD object metadata.

XPLM's script engine contains the integration logic between Inventor and SOLIDWORKS PDM. Some of the transactions are exposed in user exits. The user exits receive the full document information at the moment of the execution. The information is passed in a complex data structure. They allow the read access to all document properties and context information as well as the possibility to modify these. The user exits are executed within an embedded VB.NET script engine.

Figure 1: Integration architecture



2 Installation

This chapter provides information around the installation of the integration.

2.1 Pre-installation information

Data cards

To ensure that the integration works properly, select the data card **Inventor** in dialog *Select configuration details* during the creation of a new vault.

Optional preview application

For data formats supported by eDrawings, the integration can use the eDrawings viewer as visualization tool for previewing Inventor files.

For more information about eDrawings, refer to [eDrawings for Autodesk Inventor – Frequently Asked Questions](#)

2.1.1 Operating system support

Support of specific operating system version might be limited by support from specific CAD system.

2.1.2 Supported SOLIDWORKS PDM releases

Name	Version
Dassault Systèmes SOLIDWORKS PDM Professional	2017-23

2.2 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 `<install_dir>`. 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 custom files](#) on page 34 for more information..
7. In the step *MCAD components*, select the authoring tool in use and, if required, a corresponding version. Click **Next**.
8. Click **Install** to start the installation.
9. 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 Post-installation

This chapter describes post-installation steps that need to be performed after installation of the XPLM integration.

3.1 SOLIDWORKS PDM

In the following, SOLIDWORKS PDM-related post-installation steps are described.

3.1.1 Add-in registration

Register the add-in as described in the following chapter.

Before you start

The required files for 32-bit SOLIDWORKS PDM servers are located in the installation directory under `<install_dir>\bin\x86\pdmp20xx` and for 64-bit servers under `<install_dir>\bin\x64\pdmp20xx`.

Procedure

1. To register the add-in, open the SOLIDWORKS PDM *Administration* tool, navigate to the desired vault and then to *Add-ins*. Perform a right-click and select **New Add-in...**



If the integration should only be active on one test workstation, the Add-in has to be registered on this workstation as a *Debug Add-in*.

2. Navigate to `<install_dir>\bin\x86 (or x64) \pdmp[version]` and select the following files:
 - EPDM.Interop.epdm.dll
 - Interop.XPlmComConnector.dll
 - Interop.XPlmComDataTypes.dll
 - Interop.XPlmComService.dll
 - SOLIDWORKSPDMPAddin.dll
 - XplmComlog.dll

3. After confirming, a warning message appears. Confirm the message.

Result

Now navigate to the selected vault in the the SOLIDWORKS PDM *Administration* tool and check if the add-in *SOLIDWORKSPDMADDIN* is available under *Add-ins*. It might be possible that the explorer process needs to be restarted first.

Next steps

For 64-bit and 32-bit installations: 32-bit and 64-bit components have to be installed in 64bit environment to give all clients access to the installation. Errors could occur if the SOLIDWORKS PDM Administration misses rights. Please run SOLIDWORKS PDM as Administrator. If a second add-in should be registered, perform the steps described in [SOLIDWORKS API Help](#).



On work stations with the SOLIDWORKS PDM client which does not use the integration and where no installation should be carried out, the Visual C++ Runtime library has to be installed (vcredist_x86.exe, additionally vcredist_x64.exe at 64-bit), otherwise the start of the client cause an error message.

3.2 Inventor

In the following, Inventor-related post-installation steps are described.

3.2.1 Add-In registration

Activate the add-in as described in following chapter.

On first startup of Inventor after integration installation, dialog *Add-in Manager Security Alert* appears. Press **Open Add-In Manager** and search for application *XPlmAddin Class* in opened dialog. Set *Load Behavior* from **Block** to **Loaded/Unloaded** and **Load Automatically**. Do not unblock *SolidWorks Enterprise PDM Inventor AddIn*.

3.3 Additional information

Language settings

The language of the integration can be changed using the XML tag `<Language>` in the `<install_dir>\xml\XPlmConnector.xml` file. The following languages are supported:

Table 1: Available languages

Language	Value to enable the option
English	EN
German	DE

4 Configuration

4.1 SOLIDWORKS PDM

In the following, further SOLIDWORKS PDM-related configuration possibilities are described.

4.1.1 Setup tasks in SOLIDWORKS PDM add-in

With tasks, Inventor files can be converted by right-clicking a file and selecting a convert command.

Before you start

Set `InventorEvent_DocumentChange` to `false` in `XPlmInventorConnector.xml` to ensure that tasks work properly.

About this task

To setup the tasks defined in `<install_dir>\xml\PDMPInventorAddin.xml`, do the following:

Procedure

1. Open SOLIDWORKS PDM *Administration* tool.
2. Navigate to the desired vault and then to *Tasks*.
3. Perform a right-click and select **New Task....**
4. Enter a name for the new task (e.g. Convert PDF).
5. For *Add-in* select the *PDMP Addin* and click **Next**.
6. Within the *Execution Method* screen, the computer is shown in the *Computers supporting the task* list. If not perform a right-click on the SOLIDWORKS PDM icon in Windows hidden icons and select **Task Host Configuration**.
 - a) On the top right hand side, select **File Vault**.
 - b) In the table, set the **Permit** flag for the SOLIDWORKS PDM add-in and click **OK**.
7. Now click **Refresh List** in the task UI.
Computer is now visible in the table.
8. Enable the computer and click **Next**.
9. In the *Scheduling* section, the execution time for the task can be defined. In this example we use **This task is not scheduled**. So just click **Next**.
10. In the *Converter Configuration* section, the data from the `ScriptEngine` is defined. Make sure that the values for *Menuitems* and *Targetformat* are set correctly and click **Next**.
Possible values for *Targetformat* are:
 - For the conversion of drawings to the PDF format use: PDF
 - For the conversion of 3D models to the STEP format use: STEP
11. In *Permissions*, *Success Notification* and *Error Notification* no changes have to be made. So click **OK**.

Result

Now start an explorer, go to the SOLIDWORKS PDM vault and check if the menu entries are available in the context menu. It might be possible that the explorer process needs to be restarted first.

Next steps

Do the above described process for all desired tasks and close the *Administration* tool.

4.1.2 Improving check in performance

About this task

SOLIDWORKS PDM can automatically include drawings referencing a part or assembly when the latter is checked in. The available options in SOLIDWORKS PDM manage this functionality. Including drawings may make the check in command slower. To prevent this, perform the following steps.

Procedure

1. Start the *Administration* tool of SOLIDWORKS PDM.
2. Double-click the vault in question.
3. Double-click **Users** and then double-click current user.
→ *Admin - Properties* dialog opens.
4. Click **Settings** (on the lower left hand side).
→ *Settings* dialog opens.
5. Select **Check In**.
→ Option **Look for drawings in the entire vault is enabled**.
6. Disable this option to improve the performance.

Result

Check in performance is improved.

4.1.3 Configuration files

The SOLIDWORKS PDM integration for Inventor is configured mainly using following configuration files:

- **PDMPCConnector.xml**: Base configuration of the SOLIDWORKS PDM connector
- **PDMPIInventorConnector.xml**: Contains mostly Inventor related settings
- **PDMPIInventorTransaction.xml**: Contains the configured transactions for the save, load and update processes.

Settings in PDMPCConnector.xml

Table 2: Settings and values

Setting	Purpose and available values
Settings for SOLIDWORKS PDM logging	

Setting	Purpose and available values
EnablePDMPLogging	<p>If set to <code>true</code>, logging is enabled. Refer to chapter Enabling logging on page 22 for more information.</p> <p>It is recommended to set this setting to <code>false</code>, because the logging can negatively affect the performance of the integration.</p> <p>Default: false</p> <p>Possible values: true false</p>
PDMLogFile	<p>Is the definition of the destination of the log file.</p> <p>Important is that the destination is a folder for which the user has write permissions.</p> <p>It is recommended to set the file extension to <code>*.xml</code>.</p> <p>For example C:\Users\User1\PDMLog.xml</p>
PDMLogLevel	<p>Definition of the granularity of the log messages.</p> <p>Possible values for the log level are <code>1</code> to <code>10</code>.</p> <p>Levels between <code>1</code> and <code>4</code> are reserved for logging in the business components, while levels <code>5</code> to <code>9</code> include logging information with respect to the data exchange with SOLIDWORKS PDM.</p> <p>Level <code>10</code> provides the maximum number of log messages.</p> <p>The higher the level, the more data is generated.</p> <p>Default: 5</p> <p>Possible values: 1-10</p>
PDMWorksCancel OnError	<p>If set to <code>true</code>, any process is aborted on error.</p> <p>Default: true</p> <p>Possible values: true false</p>
PDMLogIncludeDate	<p>If set to <code>true</code>, every log entry contains date and time. If set to <code>false</code>, date and time are not written into the log file. This allows comparing two log files.</p> <p>Default: true</p> <p>Possible values: true false</p>

Settings in PDMInventorConnector.xml

Table 3: Settings and values

Setting	Purpose and available values
Menu definition reference (can be extended)	

Setting	Purpose and available values
PDMWorksMenuFiles	Name of add-in XML file. Default: PDMPInventorPDMPAddin.xml
Handling of error message	
PDMErrorSilent	Defines, if error is shown or not. Default: 0 Possible values: 0 (interactive) 1 (silent)
Settings for Inventor Add-in Files	
InventorMenuFiles	Name of add-in xml file. Default: PDMPInventorAddin.xml
Settings for communication component	
RunMacro	Defines the used communication component. Default: extern Possible values: extern (XPlmCOMService) intern (XPlmUtilities)
Setting for using timestamps	
PDMWorksUse PDMTimestamp	Defines, if SOLIDWORKS PDM uses time stamp from the SOLIDWORKS PDM data set or the local file when determining the metadata. It is activated by default. If it is set to <code>false</code> , the <i>Data Card</i> dialog does not appear when adding files to the vault via Inventor. Default value: true Possible values: true false
Settings for main script engine	
ActivePDMPScriptEngine	Name of main script engine. Default: PDMPInventorScriptengine.CScriptEngine
UseSerialNumber	It is possible to use serial numbers for file names. To use serial numbers, set this option to <code>true</code> and add the name of the number generator to transaction <code>createSerialNumbers</code> , field <code>functionmodule</code> in file <code>PDMPInventorTransaction.xml</code> .

Settings in PDMPInventorTransaction.xml

Table 4: Transactions

Transaction	Description
getAttribGetStructureCollection	See chapter Property mapping on page 17.

Transaction	Description
getAttribSetStructureCollection	

4.1.4 Property mapping

This chapter describes settings for property mapping in configuration file

`PDMPInventorTransaction.xml`.

getAttribGetStructureCollection

The function **AttribGet** reads the properties from the Inventor file and transfers them to SOLIDWORKS PDM.

The mapped variables in the data card are filled with properties.

The purpose is to prepare properties so that SOLIDWORKS PDM can use them.

Therefore, a parameter list with block and attribute names of the common mapping is created after the intermediate mapping with transaction `getAttribGetStructureCollection`.

The transaction contains all Inventor standard property blocks with its standard properties. These properties can be mapped here to some data card properties as required.

```
<Structure>
<Name>Summary</Name>>  <!-- block name in file card -->
<FieldCollection>
<Field>
  <Name>[Name]</Name>  <!-- attribute name in file card, e.g. Comments -->
  <Type>XPlmDocument</Type>
  <Subtype>StructureCollection</Subtype>
  <Structurename>SummaryInformation_EN</Structurename>
  <Attribut>[Attribut]</Attribut>  <!-- attribute name in Inventor, e.g. Comments -->
</Field>
```

The transaction can be extended if desired. But Inventor attribute names are fix, except names at the custom block. There it is possible to use own attribute names.

Troubleshooting is supported by log messages.

getAttribSetStructureCollection

The function **AttribSet** saves variable values from SOLIDWORKS PDM into the CAD file properties of Inventor (changes via data card).

Data coming from SOLIDWORKS PDM must be converted into a compliant format for updating file properties.

The mapping looks as following example:

```
<StructureCollection>
  <Structure>
    <Name>Summary</Name>
    <FieldCollection>
      <Field>
        <Name>[Name]</Name>  <!-- attribute name in file card, e.g. Comments -->
        <Value>[Value]</Value>  <!-- attribute name in Inventor, e.g. Comments -->
      </Field>
    </FieldCollection>
  </Structure>
</StructureCollection>
```

Customizing property mapping

If property mapping has to be customized, please follow these steps:

1. Copy file `PDMPInventorTransaction.xml` to `Customer_PDMPInventorTransaction.xml`
2. It is recommended removing all transactions from the new file that should not be modified, so that the file `Customer_PDMPInventorTransaction.xml` contains the two transactions `getAttribGetStructureCollection` and `getAttribSetStructureCollection`.
3. Define the property mapping in the new configuration file.
4. To map a custom property from Inventor to SOLIDWORKS PDM, add definitions to transaction `getAttribGetStructureCollection`, structure `CustomProperty` as in following example:

```
<Transaction>
  <Aliasname>getAttribGetStructureCollection</Aliasname>
  <Import>
    <Parameter>
      <StructureCollection>
        ...
      <Structure>
        <Name>CustomProperty</Name>
        <FieldCollection>
          <Field>
            <Name>[property name]</Name> <!-- e.g. AssemblyNo -->
            <Type>XPlmDocument</Type>
            <Subtype>StructureCollection</Subtype>
            <Structurename>Inventor User Defined Properties</Structurename>
            <Attribut>[attribute name]</Attribut> <!-- e.g. AssemblyNo -->
          </Field>
        </FieldCollection>
      </Structure>
    </StructureCollection>
    ...
  </Parameter>
</Import>
</Transaction>
```

5. To map defined property (step 4) from SOLIDWORKS PDM to Inventor file properties, add definition to transaction `getAttribSetStructureCollection`, structure `CustomProperty` as in following example:

```
<Transaction>
  <Aliasname>getAttribSetStructureCollection</Aliasname>
  <Import>
    <Parameter>
      <StructureCollection>
        ...
        <Structure>
          <Name>CustomProperty</Name>
          <FieldCollection>
            <Field>
              <Name>AssemblyNo</Name>
              <Value>AssemblyNo</Value>
            </Field>
          </FieldCollection>
        </Structure>
      </StructureCollection>
      ...
    </Parameter>
  </Import>
</Transaction>
```

Fallback

In case transactions `getAttribGetStructureCollection` and `getAttribSetStructureCollection` are not available, variables are declared in the corresponding data card via the Dassault Systèmes SOLIDWORKS PDM Professional administration tools.

These file properties are provided by the Inventor connector:

```
Inventor Summary Information
  Title
  Subject
  Author
  Keywords
  Comments
  Last Saved By
  Revision Number
  Thumbnail

Inventor Document Summary Information
  Category
  Manager
  Company

Design Tracking Properties
  Creation Time
  Part Number
  Project
  Cost Center
  Checked By
  Date Checked
  Engr Approved By
  Engr Date Approved
  User Status
  Material
  Part Property Revision Id
  Catalog Web Link
  Part Icon
  Description
  Vendor
  Document SubType
  Document SubType Name
  Proxy Refresh Date
  Mfg Approved By
  Mfg Date Approved
  Cost
  Standard
  Design Status
  Designer
  Engineer
  Authority
  Parameterized Template
  Template Row
  External Property Revision Id
  Standard Revision
  Manufacturer
  Standards Organization
  Language
  Defer Updates
  Size Designation
  Categories
```

4.1.5 BOM creation

This chapter describes settings for bill of materials in configuration file `PDMPInventorTransaction.xml`.

- The SOLIDWORKS PDM-BOM is configurable (see transaction `updateCADBOMs` in transaction file `PDMPInventorTransaction.xml`, properties BOM name, type etc.).
- The SOLIDWORKS PDM-BOM has eight default columns (Filename, Dir, IDs, Configuration, TreeLevel, Qty, Version) which are always created by the connector.
- Further columns can be added depending on the CAD (property `BOMColumnDefinition` in transaction `updateCADBOMs`).
- The content of BOMLines is configurable (see transaction `updateCADBOMPosition`, the fields in BOMLine must match default and custom columns, the name is crucial).



`updateCADBOMPosition` is the transaction for standard components. The transaction for special components is called `updateCADBOMPosition_[name of the component]`, e.g. `updateCADBOMPosition_virtualComponent`.

4.1.6 Document creation

This chapter describes settings for the document creation via Inventor function **New** in configuration file `PDMPInventorTransaction.xml`.

The transaction `getNewDocumentSettings` controls the behavior for the document creation via the Inventor function **New**. Following settings are available:

Table 5: Settings in transaction `getNewDocumentSettings`

Setting	Purpose and available values
FieldCollection	
showDatacard	<p>If set to <code>true</code>, data card is shown when creating documents via Inventor function New.</p> <p>Default: true</p> <p>Possible values: true false</p>
maxIteration	<p>Defines the maximum index of iterations for adding files to the vault.</p> <p>Default: 100</p> <p>Possible values: minimum value is 1</p>
StructureCollection SerialNumber	

Setting	Purpose and available values
createSerialNumber	<p>If set to <code>true</code>, name of the created file is preallocated in save dialog. The serial number needs to be created in SOLIDWORKS PDM Administration tool and added to <code>usedSerialNumber</code>. If the serial number does not exist, an appropriate error message appears.</p> <p>However, the preallocated name can be overwritten by the user.</p> <p>Default: true</p> <p>Possible values: true false</p>
usedSerialNumber	<p>Defines the serial number for the preallocation, see <code>createSerialNumber</code>.</p> <p>Default: [empty]</p>

Fallback

In case transactions `getNewDocumentSettings` is not available, the default values are as follows:

Table 6: Fallback values

Setting	Value
FieldCollection	
showDatacard	true
maxIteration	1000
StructureCollection SerialNumber	
createSerialNumber	false
usedSerialNumber	[empty]

4.1.7 Enabling logging

If required, you can activate logging for the components used in the integration. Logging should not run permanently, as a large amount of log messages are generated and performance is affected. Deactivate logging again, after the issue is solved.

Execute following steps to enable logging:

1. Edit the file `PDMPConnector.xml`:
 - A. Set `EnablePDMPLogging` to `true`.
 - B. Set value `PDMPLogFile` to a path on local disc.
 - C. Set value of `PDMPLogLevel` accordingly.
2. Edit the file `XPlmInventorConnector.xml`:
 - A. Set `EnableInventorLogging` to `true`.
 - B. Set value `InventorLogFile` to a path on local disc.
 - C. Set value of `InventorLogLevel` accordingly.

4.2 Inventor

In the following, further Inventor-related configuration possibilities are described.

4.2.1 Configuration Files

The SOLIDWORKS PDM integration for Inventor is configured mainly using the following configuration files:

- **XPlmInventorConnector.xml** - base configuration of the Inventor connector.
- **PDMPInventorAddin.xml** – contains the menu definition and add-in registration. **This file should not be changed.**

Settings in XPlmInventorConnector.xml



The configuration of the `XPlmInventorConnector.xml` depends on the environment.

When converting it should be ensured that the environment is not used for client operations and vice versa.

For better user handling, particular events are used in client environments. These events check changes of CAD files and check if files are checked in or out (e.g.). Through interactions (message boxes) the user can make decisions.

During task operations in the conversion environment, the appearance of message boxes causes the task to terminate.

For this reason, most events need to be deactivated in conversion environments.

Table 7: Settings and values

Setting	Purpose and available values
Environmental Adjustments - Client	
ApprenticeReducedAttributeSet	<p>The property controls whether a reduced set of attributes is read when traversing. If set to <code>true</code>, then a reduced set of attributes is read.</p> <p>Default: true</p> <p>Possible values: true false</p> <p>Do not change.</p>
InventorApprenticePropertyDefaultValues	<p>This property controls if default date values are converted. If set to <code>true</code>, the property converts the default date value of Inventor to an empty string. If set to <code>false</code>, the default date value of Inventor is set.</p> <p>Default: true</p> <p>Possible values: true false</p>
EnableInventorLogging	<p>If set to <code>true</code>, logging is enabled.</p> <p>Default: false</p> <p>Possible values: true false</p>

Setting	Purpose and available values
InventorLogLevel	Describes the log level (higher number means more log messages). Default: 10
InventorLogFile	Value is the full path to the log file, required if logging is enabled. For example C:\tmp\Inventor.log
InventorIgnoreMissingParts	The property controls whether the traversing of CAD structures is aborted with errors if files are not found. Default: true Possible values: true false Do not change.
InventorEvent_ExitNotify	This option is linked to a callback event created by CAD. If such an option is activated, the corresponding transaction defined in the XML file PDMPInventorTransaction.xml is executed on event occurrence. It is recommended to set this option to <code>true</code> in client environments. Default: true Possible values: true false
InventorEvent_DocumentChange	The property controls whether Inventor throws a corresponding event when changes are made to a loaded file. This setting is important for the change use case. Default: true Possible values: true false Do not change.
InventorEvent_OpenDocument	The property controls whether Inventor throws a corresponding event when a file is opened. Default: false Possible values: true false Do not change.
InventorEvent_ActivateDocument	The property controls whether Inventor throws a corresponding event when a loaded model is activated. Default: false Possible values: true false Do not change.

Setting	Purpose and available values
InventorEvent_SaveDocument	<p>The property controls whether Inventor throws a corresponding event when saving a loaded model.</p> <p>Default: false</p> <p>Possible values: true false</p> <p>Do not change.</p>
InventorEvent_BrowserPane_OnActivate	<p>The property controls whether Inventor throws an event when the Inventor Browser Pane is activated. This event is used to update the Edge bar content.</p> <p>Default: true</p> <p>Possible values: true false</p> <p>Do not change.</p>
Environmental Adjustments - Conversion (Tasks)	
InventorEvent_StartNotify	<p>This option is linked to a callback event created by CAD. If such an option is activated, the corresponding transaction defined in the XML file PDMPInventorTransaction.xml is executed on event occurrence. It is recommended to set this option to <code>false</code> in conversion environments.</p> <p>Default: false</p> <p>Possible values: true false</p>
InventorEvent_ExitNotify	<p>This option is linked to a callback event created by CAD. If such an option is activated, the corresponding transaction defined in the XML file PDMPInventorTransaction.xml is executed on event occurrence. It is recommended to set this option to <code>true</code> in conversion environments.</p> <p>Default: true</p> <p>Possible values: true false</p>
InventorEvent_DocumentChange	<p>The property controls whether Inventor throws a corresponding event when changes are made to a loaded file. This setting is important for the change use case.</p> <p>Default: false</p> <p>Possible values: true false</p> <p>It is recommended to set this option to <code>false</code> in conversion environments.</p>

Setting	Purpose and available values
InventorEvent_OpenDocument	<p>The property controls whether Inventor throws a corresponding event when a file is opened.</p> <p>Default: false</p> <p>Possible values: true false</p> <p>Do not change.</p>
InventorEvent_ActivateDocument	<p>The property controls whether Inventor throws a corresponding event when a loaded model is activated.</p> <p>Default: false</p> <p>Possible values: true false</p> <p>Do not change.</p>
InventorEvent_SaveDocument	<p>The property controls whether Inventor throws a corresponding event when saving a loaded model.</p> <p>Default: false</p> <p>Possible values: true false</p> <p>Do not change.</p>
InventorEvent _BrowserPane_OnActivate	<p>The property controls whether Inventor throws an event when the Inventor Browser Pane is activated. This event is used to update the Edge bar content.</p> <p>Default: true</p> <p>Possible values: true false</p> <p>It is recommended to set this option to <code>false</code> in conversion environments.</p>

4.3 Renaming

Renaming of Inventor models

Renaming of Inventor models causes issues in SOLIDWORKS PDM if the Inventor Apprentice version is not matching the Inventor file version. Therefore we recommend to ensure the following:

- Before initial imports into SOLIDWORKS PDM all Inventor files must be on the same Inventor version as the integrations you want to use. Run the Inventor upgrade tools before the import.
- Avoid working with duplicate file names in Inventor. Inventor allows the same filename in different folders. This feature should not be used in SOLIDWORKS PDM installations since it can cause issues with further processes (like pack&go) and would require to rename components later.
- Prevent renaming, copy and move of Inventor files in SOLIDWORKS PDM. Use Inventor CAD **Save As** capabilities instead. If you are using SOLIDWORKS PDM for renaming, copy or move ensure a refile of the affected assemblies to the matching Inventor version.

The following table describes the actions in the Inventor integration that are influenced by the Inventor version and Inventor file version.

Table 8: Influenced actions

Feature	Inventor version matches Inventor file version	Inventor version is higher than Inventor file version	File Format Plugin usage (Inventor Apprentice)
Load in Inventor	OK	OK	-
Save in Inventor	OK	OK	-
Get properties	OK	OK	x
Set properties	OK	OK	x
Check in	OK	OK	(partial)
Check out	OK	OK	-
Analyze structure	OK	OK	x
Renaming (Explorer)	OK	Fails – Workaround: perform a refile and resave from Inventor before	x
Copy (Explorer)	OK	Fails – Workaround: perform a refile and resave from Inventor before	x
Move (Explorer)	OK	Fails – Workaround: perform a refile and resave from Inventor before	x
Get version	OK	Can fail loading an old version if renamed components are used – Workaround: Upgrade files in vault to matching Inventor version	x

Technical background

Renaming a component forces a *Replace Reference* in the latest version. On loading an older assembly version the renamed component file is replaced again in the older assembly version. The older assembly version could have been produced in an older Inventor version. The Inventor Apprentice API cannot replace the reference in Inventor assemblies that do not match to the Apprentice version.

Recommendation: Avoid renaming and move in SOLIDWORKS PDM during the product lifecycle. If you rename components in the database, make sure to version up and resave immediately the higher level assemblies and drawings from Inventor where these components are used. On Inventor upgrades make sure you upgrade the required Inventor data inside SOLIDWORKS PDM to the used Inventor version.

5 Update


5.1 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.
 - **Optional:** Enable the option **Apply custom files after installation** to overwrite the installation files with an overlay package after completion. 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 directly corresponds to `<install_dir>` and contains the configuration.
 - **Optional:** Enable the option **Backup current installation** to backup the existing installation. It is recommended to always make a backup when updating or changing the installation.
6. Click **Next** and update components, if required.
 - The step *Ready to install* appears.
7. To start installation, click **Install**.
8. To close the wizard after installation, click **Finish**.

Result

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


5.2 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.
 - **Optional:** Enable the option **Apply custom files after installation** to overwrite the installation files with an overlay package after completion. 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 directly corresponds to `<install_dir>` and contains the configuration.
 - **Optional:** Enable the option **Backup current installation** to backup the existing installation. It is recommended to always make a backup when updating or changing the installation.
6. Click **Next**.
→ The step *Ready to install* appears.
7. To start installation, click **Install**.
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.

5.3 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.
 - **Optional:** Enable the option **Apply custom files after installation** to overwrite the installation files with an overlay package after completion. 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 directly corresponds to `<install_dir>` and contains the configuration.
 - **Optional:** Enable the option **Backup current installation** to backup the existing installation. It is recommended to always make a backup when updating or changing the installation.
7. Click **Next** and update components, if required.
8. To start installation, click **Install**.
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.

6 Uninstallation

6.1 Uninstalling integration

You can uninstall the integration in different ways.

Installation directory

Go to `<install_dir>` and double-click the **Uninstall** link.

Windows Control Panel

Go to *Programs and Features* and uninstall **XPLM Integration for Dassault Systèmes SOLIDWORKS PDM Professional**.

Silent mode

Command line calls runtime:

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

Command line calls MSI:

```
START /WAIT msixexec /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:

```
START /WAIT msixexec /uninstall "Autodesk Inventor Integration 2020 SPx.0 (Build  
[version]).msi" /quiet
```

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


7 Troubleshooting

This chapter describes instructions for troubleshooting integration errors.

7.1 Troubleshooting integration problems

In case of integration problems, do the following.

1. Close related programs.
2. Enable logging.
3. Restart the integration and reproduce the problem.
4. Send an email with a problem description and log files to support.

7.2 License error codes

Table 9: 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

7.3 Adding drawings to vault leads to infinite loop

When adding files to the vault via Inventor, the data card appears. In case of drawings (.idw), adding may not result in the data card being displayed. The reason is a missing configuration of the SOLIDWORKS PDM vault. By default, only parts (.ipt) and assemblies (.iam) are considered for adding files to the vault. After adding the file extension (.idw) as described below, the data card appears.

Procedure

1. Open the SOLIDWORKS PDM Administration tool.
2. Expand node **Users** of the affected vault and perform a right-click on the user. Select **Settings** from the context menu.
3. In the left part of the upcoming dialog, select **Adding Files** and add ; `idw` (with semicolon) to the list of file extensions.
4. Restart the vault and the integration.
5. Add a drawing to the vault via Inventor.

Result

Data card appears for the drawing.

8 References

8.1 Using custom files

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 `<install_dir>`, for example `C:\my_config_files`.
4. Copy the modified files from the directories under `<install_dir>` 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.

8.2 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.

8.2.1 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 msixexec /i "Autodesk Inventor Integration 2020 SPx.0 (Build  
[version]).msi" /quiet INSTALLDIR="C:\Test" VERSION_PDMP="pdmp[version]" CHECK_INV="1"  
VERSION_INV="inventor[version]"
```

2. Adjust parameters as described in chapter [Silent installation parameter](#) on page 35.
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 msixexec /i msxml_4.30.2100_x86.msi /quiet  
START /WAIT vc_redist_9.0.30729.6161_x64.exe /q
```

8.2.2 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.

8.2.3 Silent installation parameter

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

Table 10: Parameter and description

Parameter	Description
INSTALLDIR	Place where the integration should be installed (for example <code>INSTALLDIR="C:\Programs\SOLIDWORKS PDM"</code>)
CHECK_CUSTOM	Copying custom files over created installation (value must be <code>"1"</code> to enable, for example <code>CHECK_CUSTOM="1"</code>)

Parameter	Description
CUSTOM_FILES	Place where custom files are located (for example <code>CUSTOM_FILES="C:\custom_files"</code>)
REBOOT_AFTER_FINALIZE	If set to <code>1</code> , a restart of the system is executed after installation, for example <code>REBOOT_AFTER_FINALIZE="1"</code> . It is set to <code>0</code> by default.
VERSION_PDMP	Version of SOLIDWORKS PDM (value must be <code>"pdmp"</code> followed by a supported version, for example <code>VERSION_PDMP="pdmp2020"</code> .
CHECK_INV	Installs Autodesk Inventor (value must be <code>"1"</code> to enable, for example <code>CHECK_INV="1"</code>)
VERSION_INV	Version of Autodesk Inventor (value must be <code>"inventor"</code> followed by the version shown in the GUI, for example <code>VERSION_INV="inventor2020"</code>)