PDM-Workbench

PDM-Workbench
Release 3.2
for Aras Innovator

Installation &
Administration Manual

Copyright

Contact

T-Systems International GmbH
PDC PLM
Fasanenweg 5
70771 Leinfelden-Echterdingen
Germany

http://www.pdm-workbench.com

☎ +49-711-972-4 96 57
✉ +49-711-972-9 59 75

mail : *cmi_support@t-systems.com*

Manual History

Version	Date
1.0	April 2005
2.0	November 2006
2.1	November 2007
2.2	September 2008
2.5	September 2010
3.0	October 2011
3.1	February 2012
3.1 SR1	February 2012
3.2	March 2012

This edition obsoletes all previous editions.

Your Comments are Welcome

Please feel free to tell us your opinion; we are always interested in improving our publications. Mail your comments to:

T-Systems International GmbH
PDC PLM
Fasanenweg 5
70771 Leinfelden-Echterdingen
Germany

mail: *cmi_support@t-systems.com*

Preface

About this Manual

This manual provides installation and configuration information for the PDM-Workbench. Before using this guide, be sure you understand:

- the Microsoft Windows operating system
- the administration of the CATIA V5 system
- the administration of the Aras Innovator system

Related Documents

The following manuals contain information about installation, administration, usage and customization of the *PDM-Workbench*:

Manual Title	Version
PDM-Workbench Installation & Administration Manual	3.2
PDM-Workbench User Manual	3.2

Trademarks

Table of Contents

Table of Figures

CHAPTER 1

Overview

This chapter provides basic information about the installation of the *PDM-Workbench*.

System and Software Requirements

Server Installation of Aras Innovator 9.3 on the following operation systems:

Windows Server 2008

CATIA V5 Client Version V5R19, V5R20, and V5R21 on the following operation systems:

Windows XP (32 Bit, 64 Bit), Windows 7 (64 Bit)

Installation steps

This section describes which PDM-Workbench modules (client and server) need to be installed.

On the client and the server two steps need to be performed each:

- Client installation: CATIA V5 Add-in (chapter 2)
- Client installation: License Manager. (For the installation of “licman20” please refer to the *Licman 2.0 Installation Manual*.)
- Server installation: PDM-Workbench data model and server methods (chapter 3)
- Server installation: PDM-Workbench server DLL (chapter 4)

The order of the installation does not matter.

CHAPTER 2

Adapting CATIA V5

The PWBCATV5 module provided by T-Systems International GmbH extends the CATIA V5 functionality to communicate with the Aras Innovator PDM system.

You should perform the following steps with your CATIA system administrator.

The **PWBCATV5_Rxx_xxVxx** module includes all of the supported platform data in a compressed file. Thus, you should choose an installation location for all CATIA V5 clients.

In the following example sections it is supposed that the software will be installed within the directory **C:\Program Files\T-Systems\PWBCATV5_Rxx_xxVxx_Aras_xx** on Windows but you can surely choose any other destination for the module.

Within the installation you will need to supply the PDM specific installation package. The file name follows the naming convention **PWBCATV5_xxVxx_Aras_xx**. Where the substring _xxVxx_ matches the corresponding substring of the CATIA module name.

Loading PWBCATV5 Software from CD-ROM

Windows XP/7

Use the Windows Explorer to locate the D**:\pwbcatv5\PWBCATV5_[Rxx]_[xxVxx].tar.Z** file on the CD. Extract the content of the archive file to a temporary installation location.

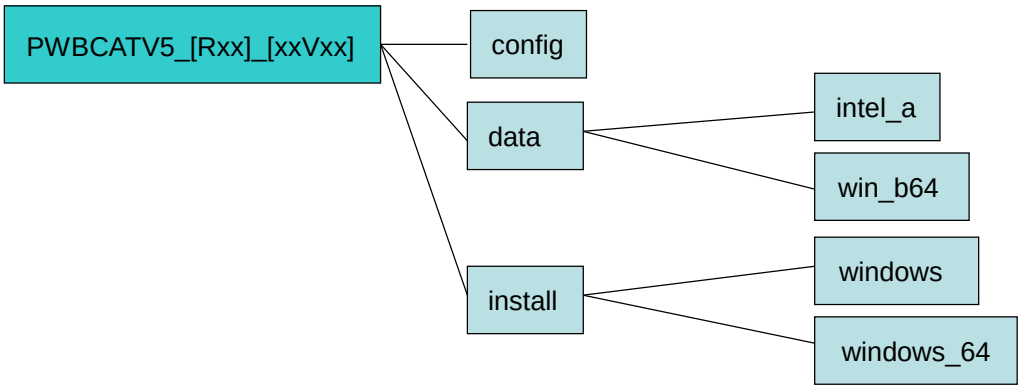
☞ **Caution: WinZip™ versions before 8.0 do not support the tar file correctly. We recommend to use WinZip™ 8.1 or above.**

PWBCATV5 Installation

After you have successfully transferred the installation files to your installation host; the following steps will install the files and configure your installation.

Configuring the installation

The **PWBCATV5_[Rxx]_[xxVxx]** Installation Directory has the following structure:



Picture 1: Directory structure of the PDM-Workbench installation files

The **config** directory contains readme files and special files needed by the installer or the installed program.

The **data** directory contains the binary distributions for the PWBCATV5 module for the supported operating system mnemonics.

The supported operation systems and their mnemonics are:

Windows XP (32 Bit)	intel_a
Windows XP (64 Bit)	win_b64
Windows 7 (64 Bit)	win_b64

The **install** directory contains the sub directories **windows**, and **windows_64** with all necessary data for the installer program.

Windows XP (32 Bit/64 Bit)/Windows 7 (64 Bit)

On **Windows XP (32 Bit)** use the Windows Explorer to run the **setup.exe** in the directory **PWBCATV5_[Rxx]_[xxVxx]\install\windows** of the installation package.

On **Windows XP (64 Bit)/Windows 7 (64 Bit)** use the Windows Explorer to run the **setup.exe** in the directory **PWBCATV5_[Rxx]_[xxVxx]\install\windows** of the installation package if you have installed the 32 Bit version of CATIA V5.

On **Windows XP (64 Bit)/Windows 7 (64 Bit)** use the Windows Explorer to run the **setup.exe** in the directory **PWBCATV5_[Rxx]_[xxVxx]\install\windows_64** of the installation package if you have installed the 64 Bit version of CATIA V5.

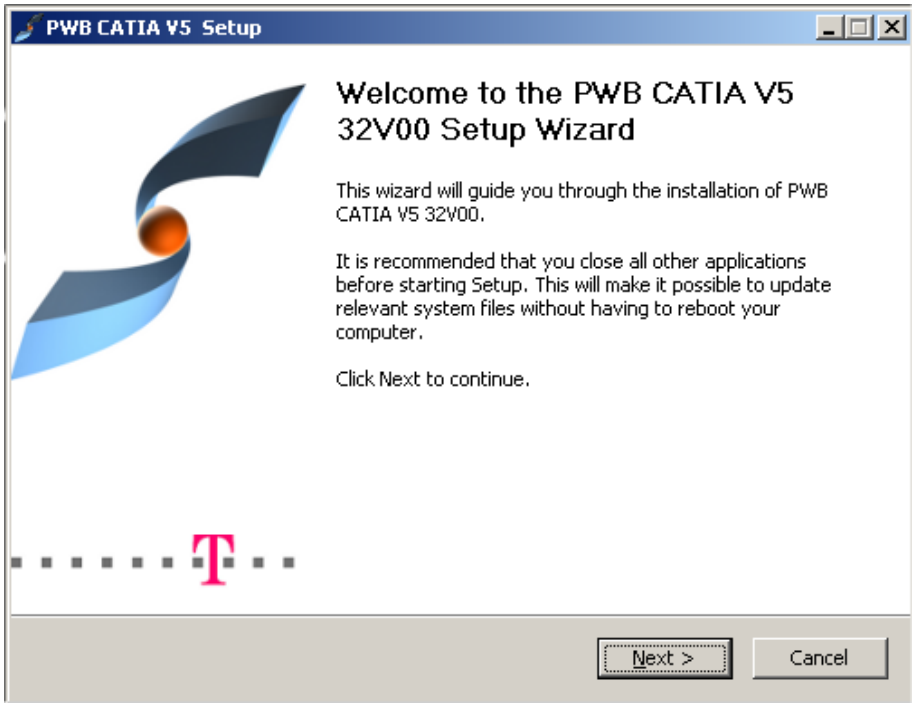
On **Windows Vista/Windows 7** the User Account Control (UAC) will be triggered and you will have to agree that the setup program may make changes to the computer. The installer is signed with a "**T-Systems International GmbH**" certificate to ensure its integrity and source.

The setup will **NOT** modify the native installation of CATIA V5.

The licman20 license manager has to be installed on the CATIA V5 client host. For the installation of the license manager please refer to the *Licman 2.0 Installation Manual*.

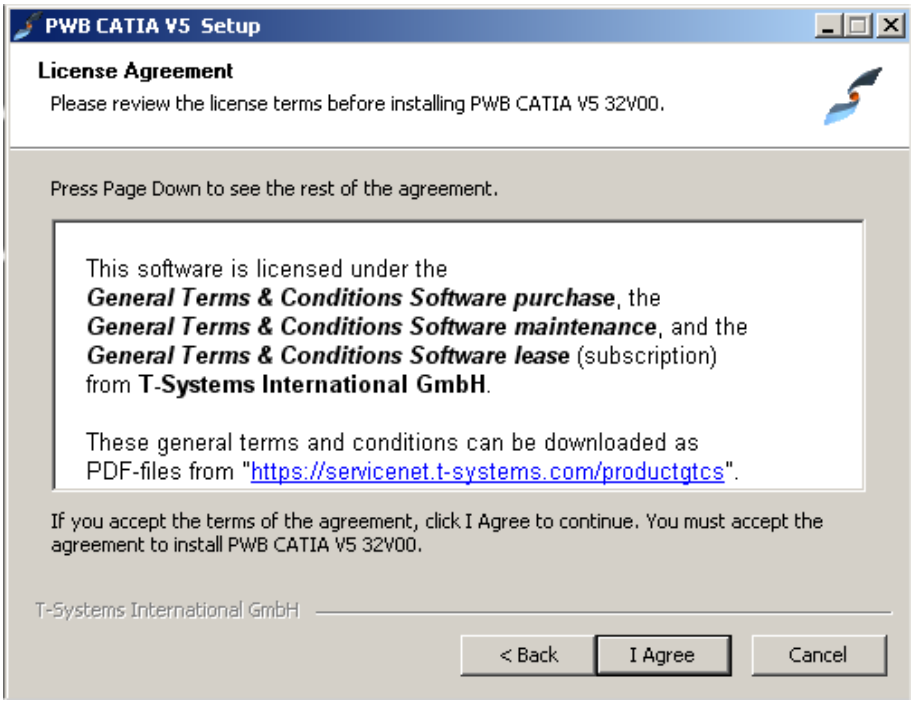
In the following the setup is shown step-by-step.

Installation process:



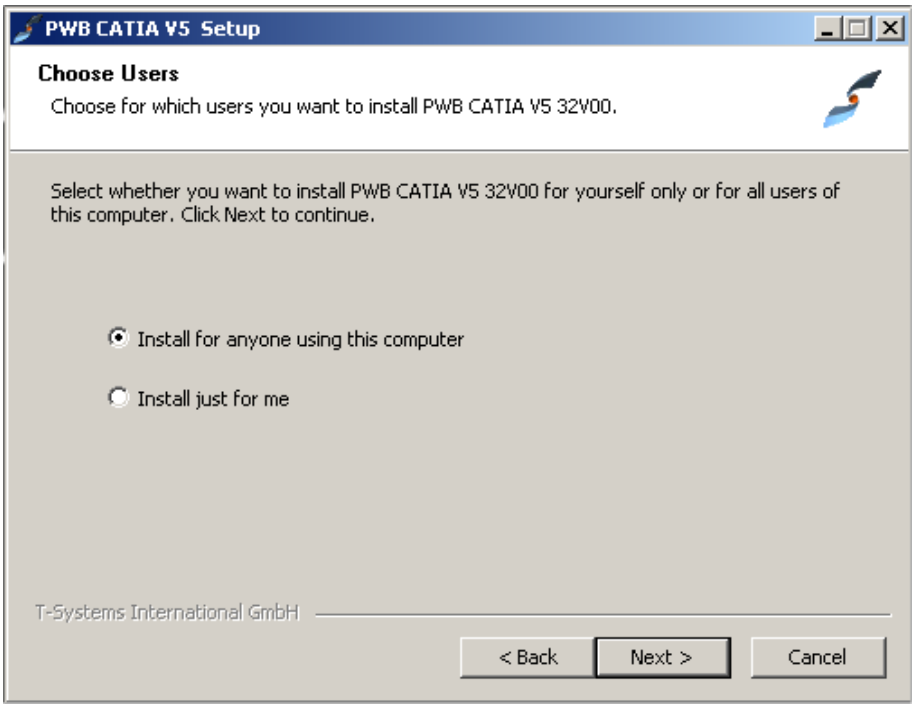
Picture 2: Welcome to the Installation

The installer software asks to approve the license terms (see *Picture 3: License Agreement*).



Picture 3: License Agreement

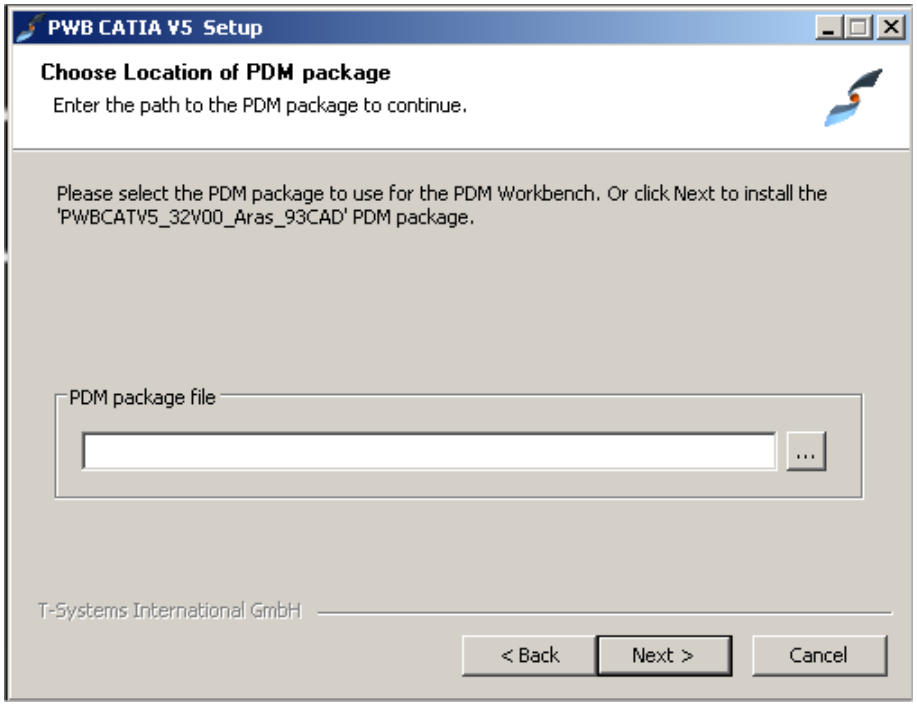
The installer software asks for the following input: **User scope.**
Next the installer will ask you to define the scope of the installation (see *Picture 4: Choose installation scope*). You can choose between an installation for anyone using the computer or just for the current user.



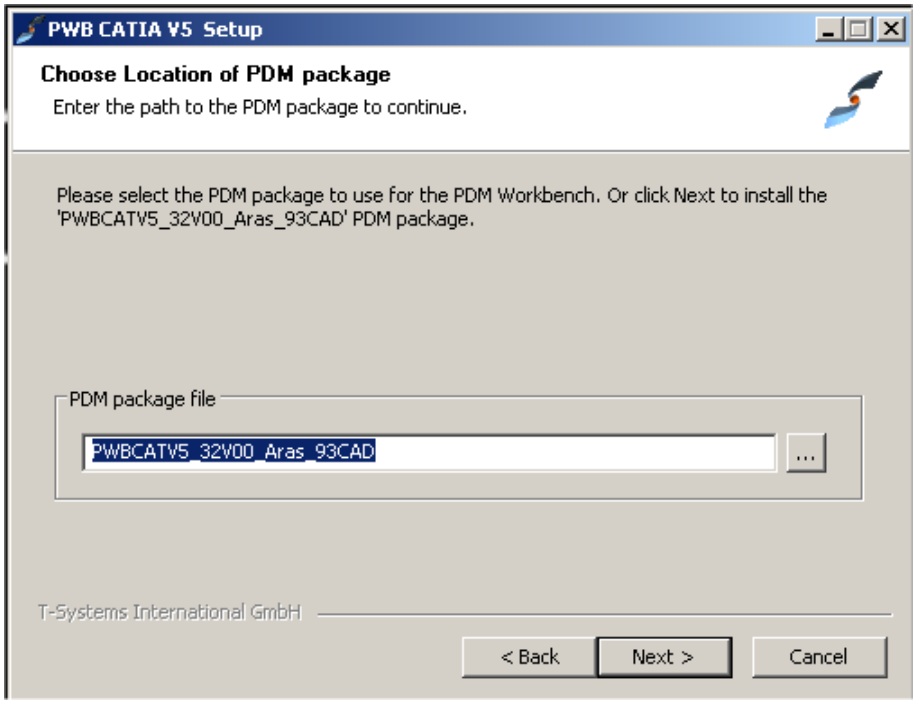
Picture 4: Choose installation scope

The installer software asks for the following input: **Location of the PDM package.**

The installer asks for the location of the PDM package to use (see *Picture 5: Choose PDM package*). If an PDM package has previously been unpacked within the installer it will be offered to install this package directly (see *Picture 6: Choose PDM package (with proposal)*).



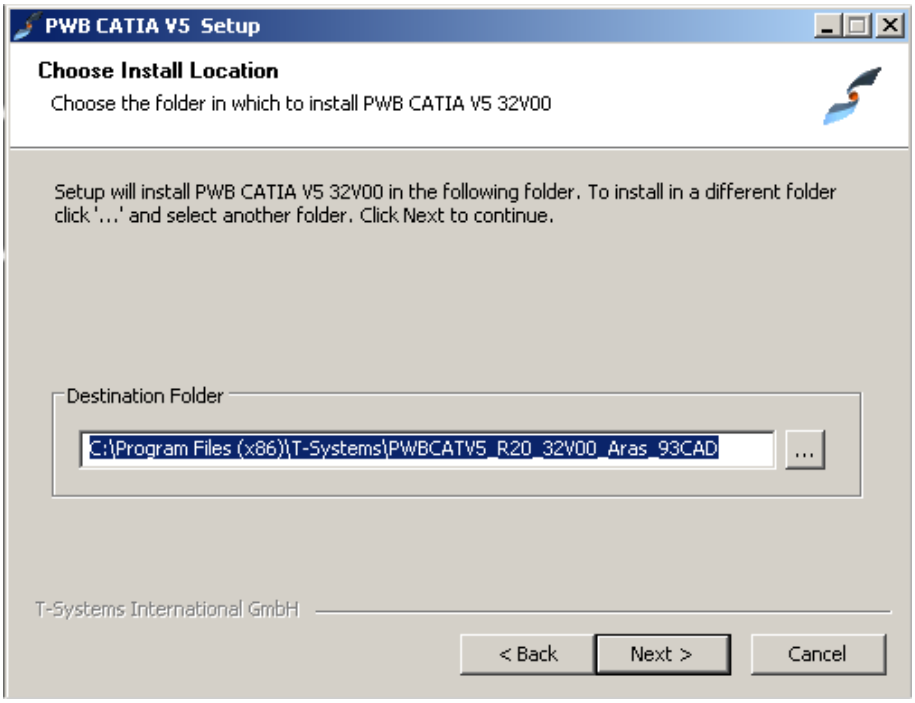
Picture 5: Choose PDM package



Picture 6: Choose PDM package (with proposal)

The installer software asks for the following input: **Installation directory.**

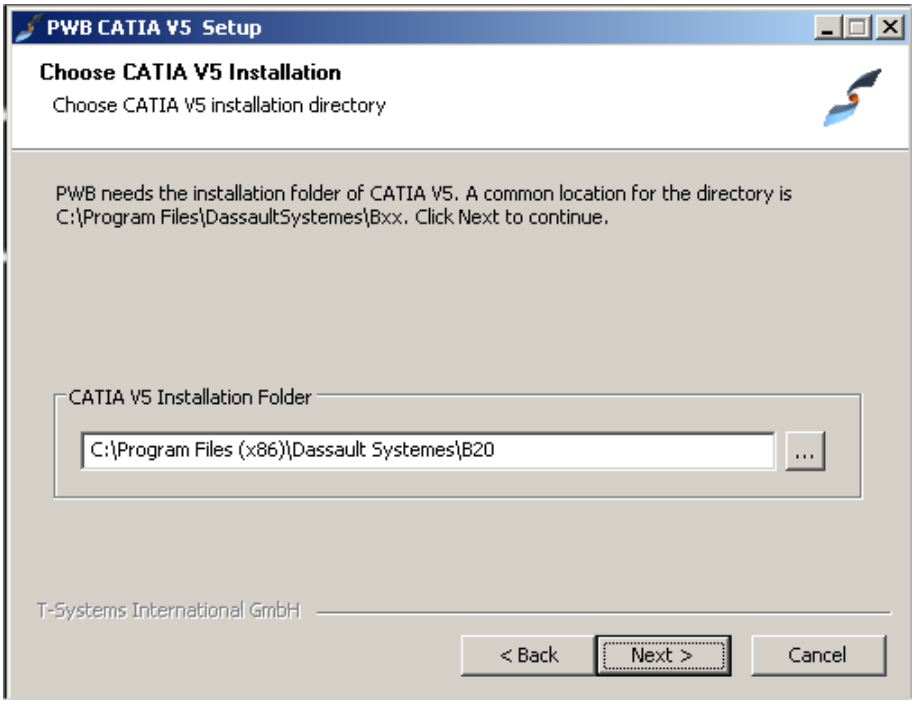
Next the installer will ask you for the target directory for the installation. You can use the given standard location or choose any other location (see *Picture 7: Choose Install Location*). The chosen folder must be empty or not existent.



Picture 7: Choose Install Location

The installer software asks for the following input: **CATIA installation directory**

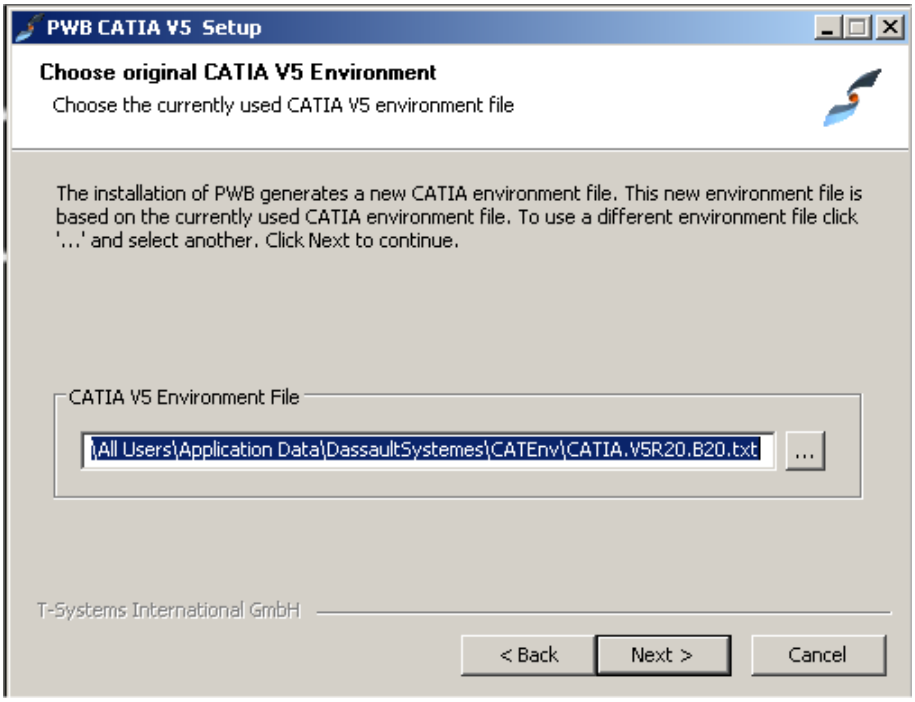
The installation path of the CATIA to use needs to be specified (see *Picture 8: Choose CATIA Installation*).



Picture 8: Choose CATIA Installation

Afterwards you will be asked for your CATIA V5 environment file (see *Picture 9: CATIA V5 Environment File Selection*).

The installation of PDM-Workbench generates a new CATIA V5 environment file. This new environment file is based on the currently used CATIA V5 environment file.

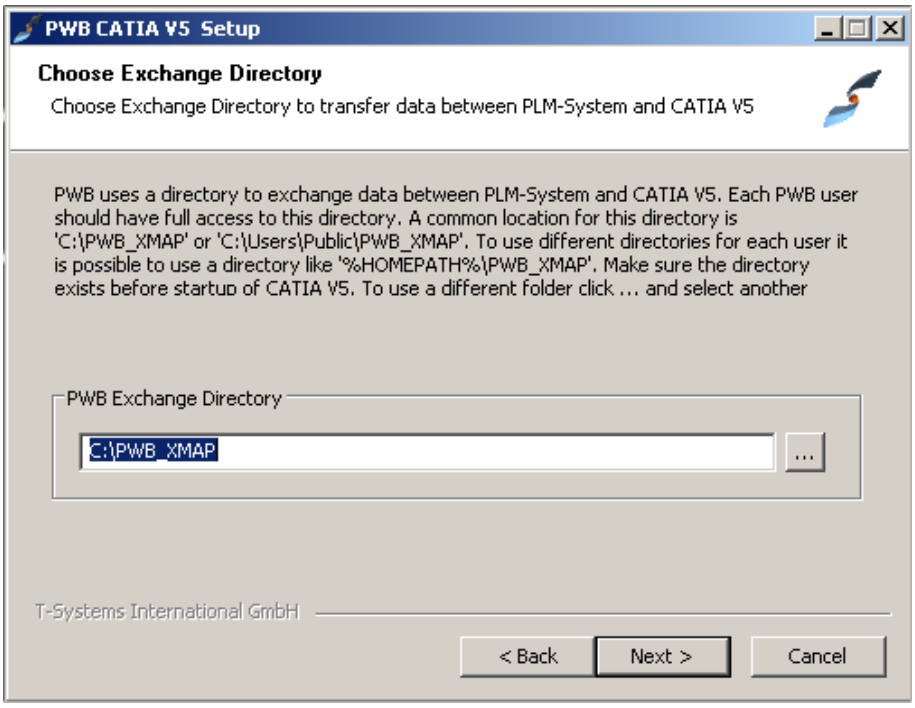


Picture 9: CATIA V5 Environment File Selection

The installer software asks for the following input: **PWB Exchange Directory**.

The PDM-Workbench needs a temporary directory to perform the file transfer between CATIA and the PDM system. Make sure this directory exists for every PDM-Workbench user on the CATIA client machine.

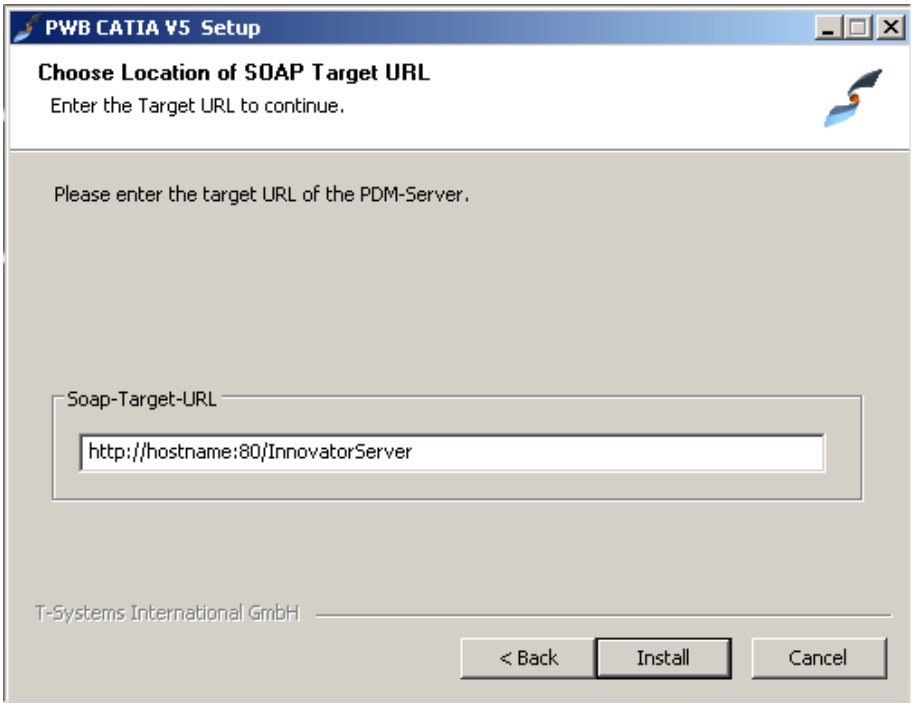
You can either use the standard location or choose any other location (see *Picture 10: PWB Exchange Directory*).



Picture 10: PWB Exchange Directory

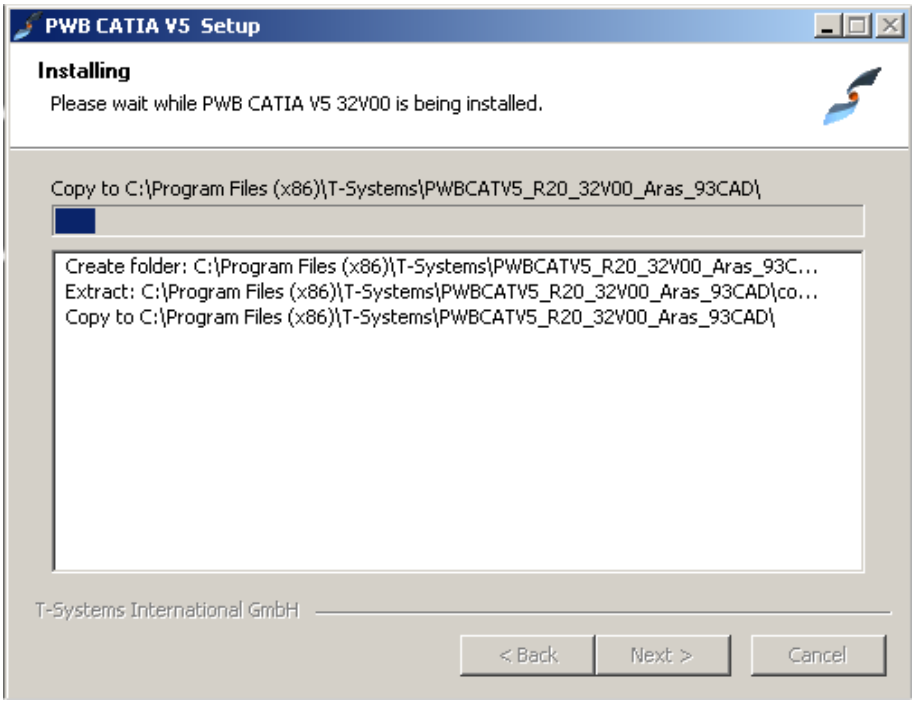
Finally you have to define the so called “Soap Target URL” for the PDM Server (see *Picture 11: Location of SOAP Target URL*).

This URL defines the host and port on which the PDM Server is reachable.

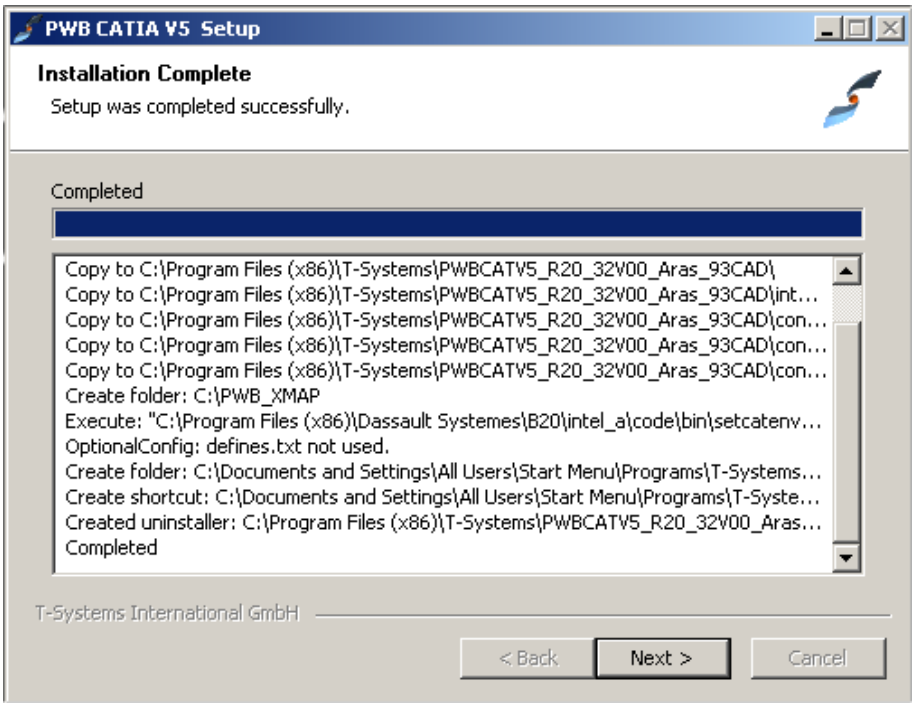


Picture 11: Location of SOAP Target URL

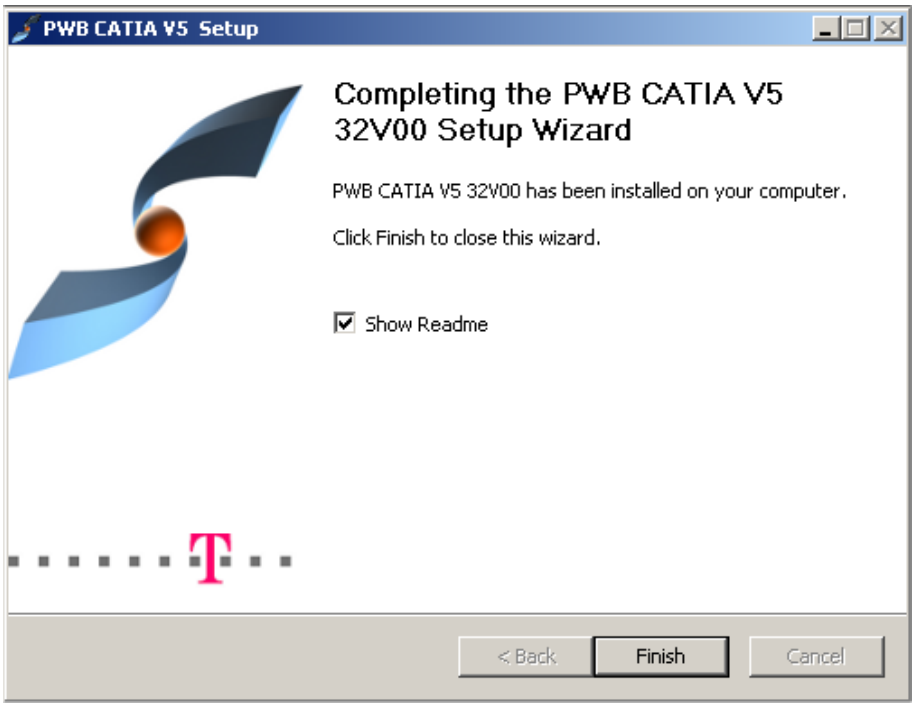
The installer will proceed in its process. The taken actions will be journalized (see *Picture 12: Installation progress* and *Picture 13: Installation progress end*).



Picture 12: Installation progress



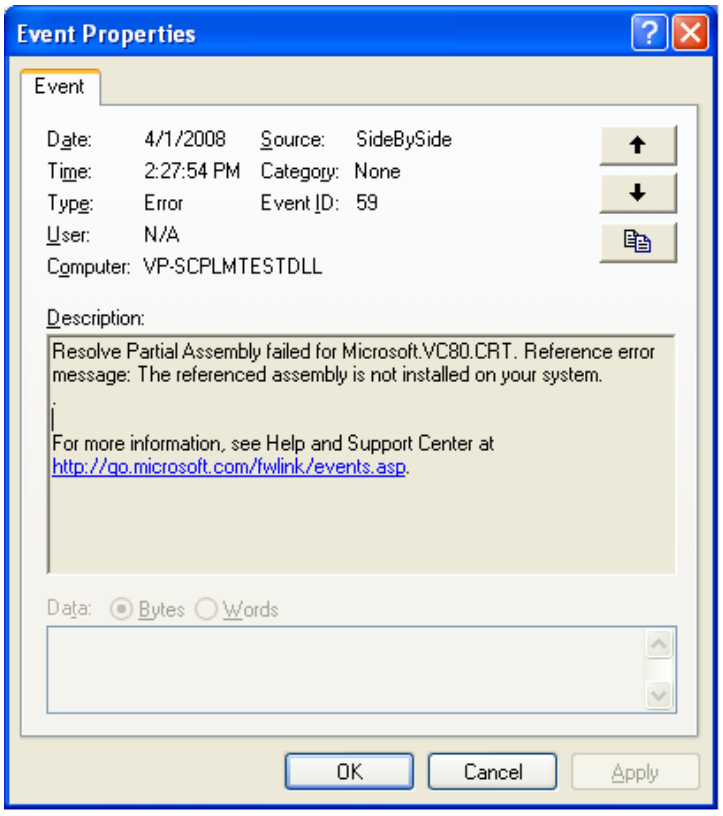
Picture 13: Installation progress end



Picture 14: Installation finished

Troubleshooting R18 (R17 (64 Bit))

If your PDM-Workbench Toolbar does not appear when starting up CATIA V5, please check Windows Event Viewer → System. Check if there is a SideBySide error like this.



Picture 15: Event Properties

This error indicates that you need additional Windows runtime libraries.

You can find these runtimes in one of the following directories:

```
PWBCATV5_R17_31V00\Windows_Runtime\x64\
(required for 64 Bit CATIA only)
PWBCATV5_R18_31V00\Windows_Runtime\x64\
(64 Bit CATIA)
PWBCATV5_R18_31V00\Windows_Runtime\x86\
(32 Bit CATIA)
```

There are two possibilities to install the new runtime:

Install the runtime libraries into the Windows XP installation (recommended).

This may need system privileges.

For 32 Bit PWB/CATIA you have to extract the package `vcredist_x86.zip` and execute the setup routine `vcredist_x86\vcredist_x86.exe`

For 64 Bit PWB/CATIA you have to extract the package `vcredist_x64.zip` and execute the setup routine `vcredist_x64\vcredist_x64.exe`

or

Copy the additional libraries in a sub directory of the PWB CATIA V5 installation.

For 32 Bit PWB/CATIA you have to extract the package `Microsoft.VC80_x86.zip`. Copy the folder `Microsoft.VC80.CRT` to the binary location of the PWB CATIA module: `PWBCATV5_R18_31V00\intel_a\code\bin\Microsoft.VC80.CRT`

For 64 Bit PWB/CATIA you have to extract the package `Microsoft.VC80_x64.zip`. Copy the folder `Microsoft.VC80.CRT` to the binary location of the PWB CATIA module: `PWBCATV5_R18_31V00\win_b64\code\bin\Microsoft.VC80.CRT`

Testing the installation

Windows

Use: Start **Programs→T-Systems→PWBCATV5_R21_31V00→PWB_START** to launch CATIA V5

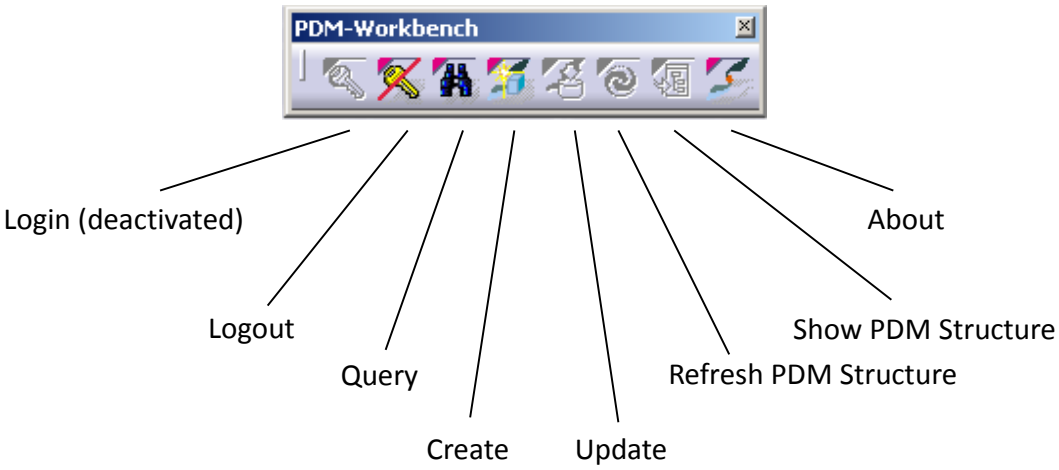
After the CATIA V5 has started the following message should appear in the command window:

```
debug on (level 1)
PDM-Workbench: Module Number 1030 :
license successfully allocated
```

The License Module Number may vary.

Within CATIA V5 the following toolbar has to be visible:

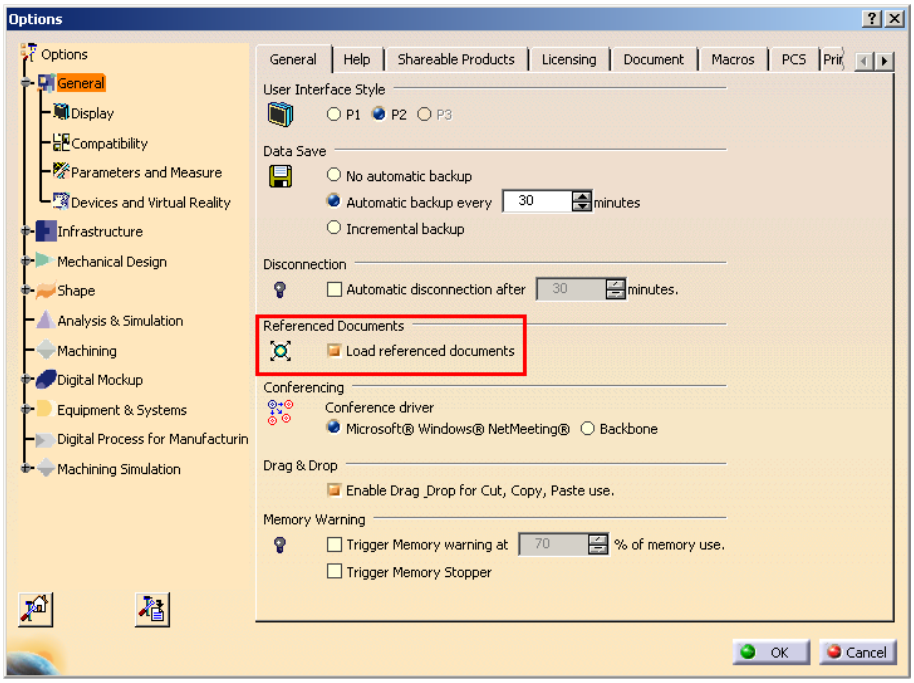
Picture 16: PDM-Workbench toolbar before login



Picture 17: The PDM-Workbench toolbar after the login

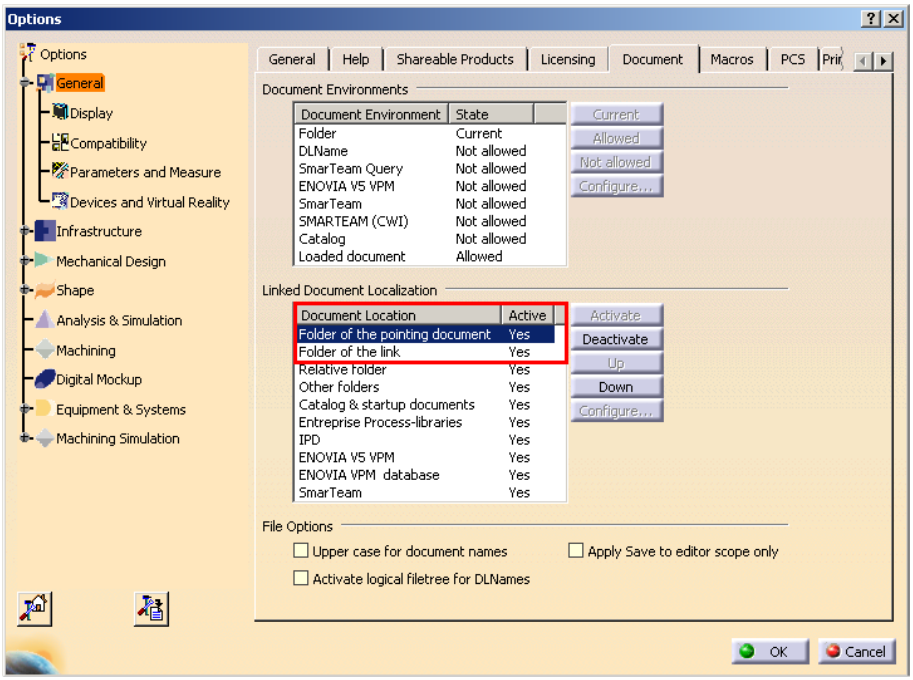
In the CATIA V5 Settings the following options have to be set as described below:

The *Load Referenced documents* option must be set in **Tools→Options→General** settings (see *Picture 18: CATIA V5 General→General Settings*).



Picture 18: CATIA V5 General→General Settings

In the *Linked Document Localization* the Options ***Folder of the pointing document*** and ***Folder of the link*** must be set to yes, and should be in this order (see *Picture 19: CATIA V5 General→Document Settings*).



Picture 19: CATIA V5 General→Document Settings

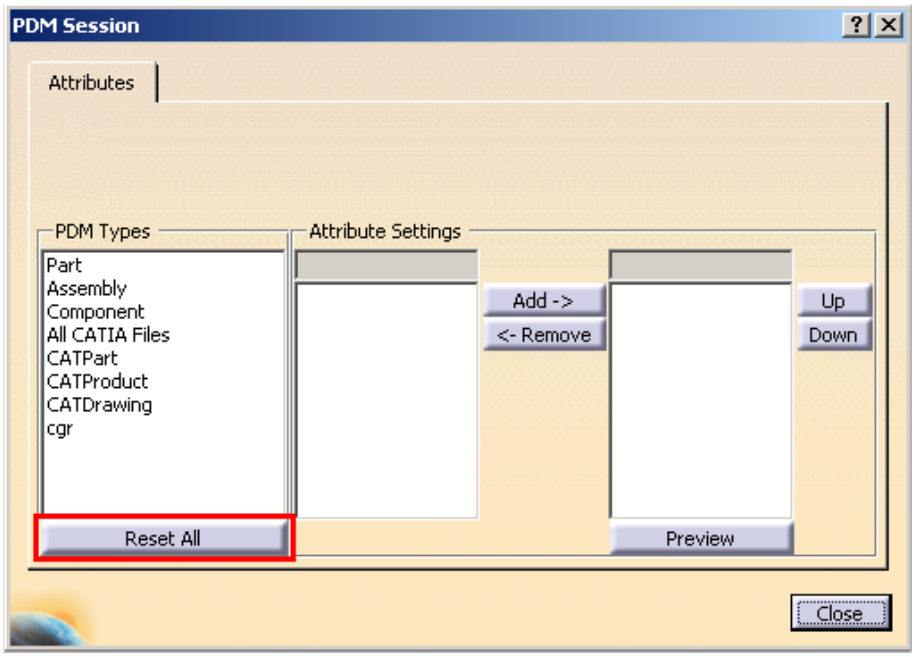
For PDM-Workbench functionality please refer to the *PDM-Workbench User Manual.*

PWBSchema modification

In case of an update of the CATIA client or changes in the PDM-Workbench Schema definition file PWBSchema.xml it is necessary to refresh the List View Column definition for all classes.

Please log in into the PDM system.

Choose in CATIA V5 *Tools→Options* and there *General – Compatibility – PDM-Workbench*. Click on “Customize List View”. The Configuration dialog opens (see *Picture 20: PDM Session Configuration dialog*).



Picture 20: PDM Session Configuration dialog

Please click the button “Reset All” in order to refresh the changes from the Schema definition file.

Setting of Environment Variables

The PDM-Workbench software will use the following environment variables in the CATIA environment:

Environment variable	Comment
PWB_XMAP	The location of the exchange map directory.
PWB_SCHEMA_FILE	Path including file name of the XML configuration file.
PWB_SOAP_TARGET_URL	The URL of the web service. Host and Port, e.g. edmg119:9070
PWB_DEBUG	Set to "ON" to receive PWB debug output in the console.
PWB_ADDTEMP_PREFIX	The prefix for the rename of the Part Numbers and File Names for the "Add Temp" and "Open File Temporary" command. Default value is "TMP".

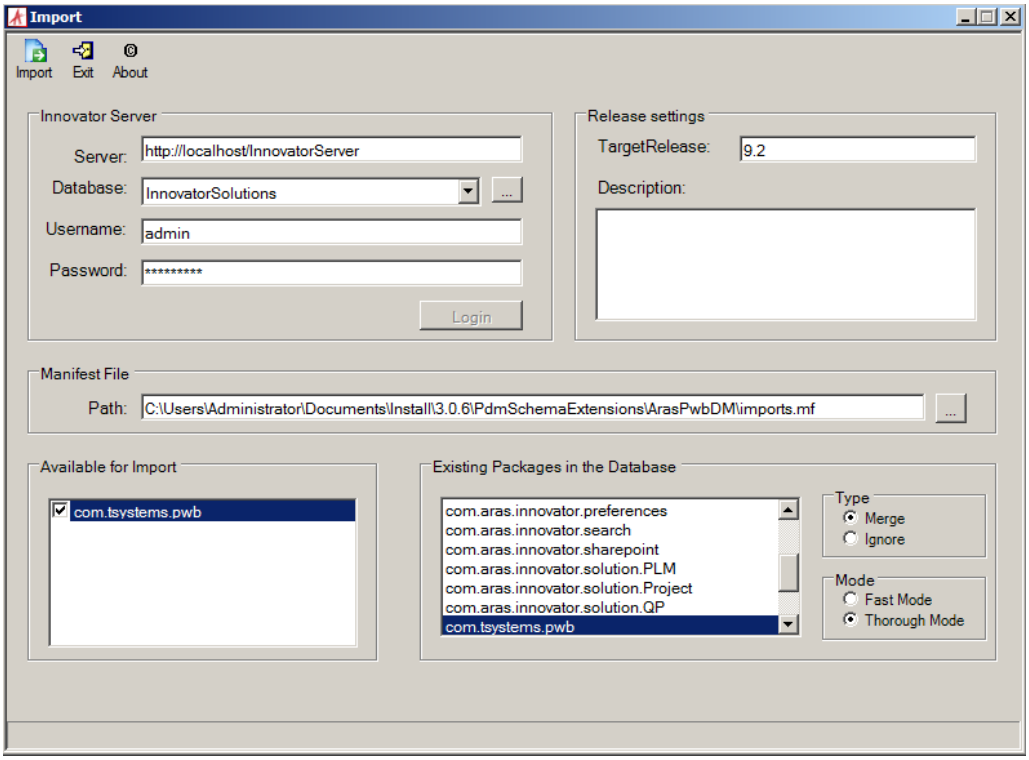
CHAPTER 3

PDM-Workbench Data Model

Installation

The PDM-Workbench data model and several server-side methods which call and support the main server functionality defined in the PDM-Workbench server DLL (see chapter 4) need to be imported to Aras Innovator.

For this the “InnovatorPackage.zip” file needs to be unpacked first. Then three packages need to be imported to Aras Innovator with the Aras Innovator import utility[1]:



Picture 21: PDM Aras Innovator import utility

Please select the manifest files

- PdmSchemaExtensions\ArasPwbDM\imports.mf
- PdmSchemaExtensions\ArasPwbDM_PLM\imports.mf

[1] The import utility has to be downloaded from the Aras homepage and to be installed. Link: www.aras.com --> Community --> Projects --> Package Import Export Utility

- PdmSchemaExtensions\ArasPwbDM_Core\imports.mf

in the import utility in this order and perform the import (Type is “Merge”, Mode is “Thorough Mode”) (see *Picture 21: PDM Aras Innovator import utility*).

CHAPTER 4

PDM-Workbench Server DLL

Please copy the file “PwbServerAddin.dll” to the Aras Innovator server directory

`C:\Program Files\Aras\Innovator\Innovator\Server\bin`
or to the corresponding directory if the Aras Innovator server has been installed in a different directory.

Also, please modify the file

`C:\Program Files\Aras\Innovator\Innovator\Server\method-config.xml`
by adding the red lines:

```
...
<MethodConfig>
  <ReferencedAssemblies>
    <name>System.dll</name>
    <name>System.XML.dll</name>
    <name>System.Web.dll</name>
    <name>System.Data.dll</name>
    <name>$(binpath)/IOM.dll</name>
    <name>$(binpath)/InnovatorCore.dll</name>
    <name>$(binpath)/CoreCS.dll</name>
    <name>$(binpath)/SPConnector.dll</name>
    <name>$(binpath)/PwbServerAddin.dll</name>
  </ReferencedAssemblies>
...

...
<![CDATA[
using System;
using System.IO;
using System.Xml;
using System.Text;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Net;
using System.Web;
using System.Web.SessionState;
using System.Globalization;
using Aras.IOM;
using PwbServerAddin;
using PwbServerAddin.Base;

namespace $(pkgname)
{
...
```

CHAPTER 5

Client Customization

The display names in the CATIA V5 workshop can be changed.

The data model of the backend PDM system has to be defined for the CATIA V5 workshop.

The variable $CATIA_INSTALL_DIR defines the installation directory of the PDM-Workbench CATIA client.

Display names

The Native Language Support (NLS) files for the CATIA V5 workshop are placed in the following directory:

$CATIA_INSTALL_DIR\intel_a\resources\msgcatalog

or

$CATIA_INSTALL_DIR\win_b64\resources\msgcatalog

and the sub directories for the different languages.

There are several NLS files for the dialogs and commands.

The displays for the PDM schema file (see above) are defined in the following files:

- PWBSchemaDisplayNames.CATNls
- PWBSchemaDisplayNames_Aras_Aras.CATNls where the first “Aras” corresponds with the “system” value and the second "Aras" corresponds with the “customization” value of the PDM systems in the PDM schema file.

The displays for the error messages are defined in the following file:

- PWBUserErrors.CATNls

The displays in these NLS files can be changed.

Icons

The icons for the objects for the CATIA V5 workshop are placed in the following directory:

$CATIA_INSTALL_DIR\intel_a\resources\graphic\icons\normal

or

$CATIA_INSTALL_DIR\win_b64\resources\graphic\icons\normal

The icon names correspond to the names used in the PDM schema file.

Data model definition

The configuration of the data model for Aras Innovator has to be done in the configuration file (xml schema file) to be used by the PDM-Workbench module within CATIA V5.

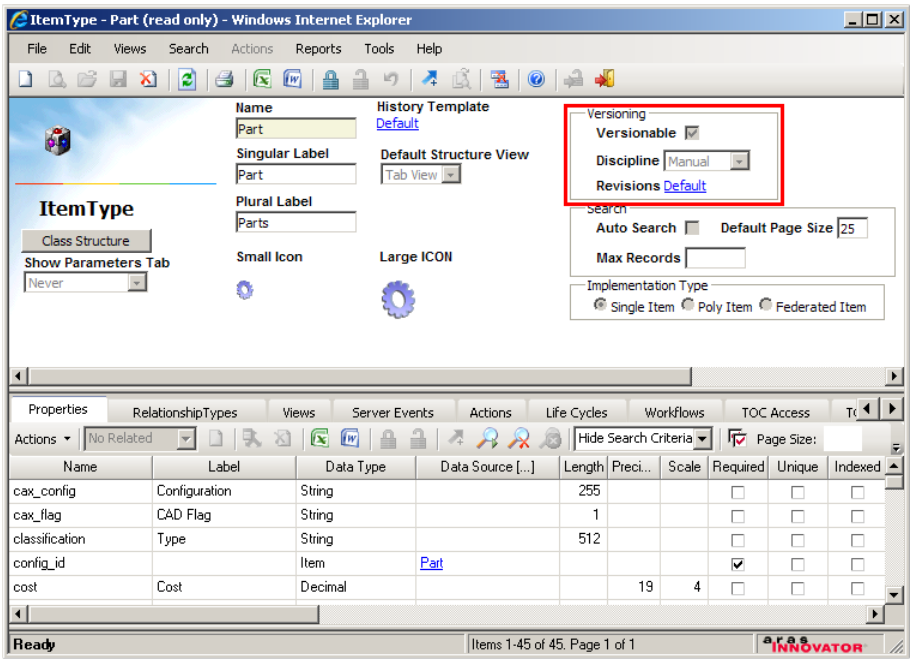
CHAPTER 6

Server Configuration

This chapter describes the configuration of the server side of the PDM-Workbench integration.

Versioning

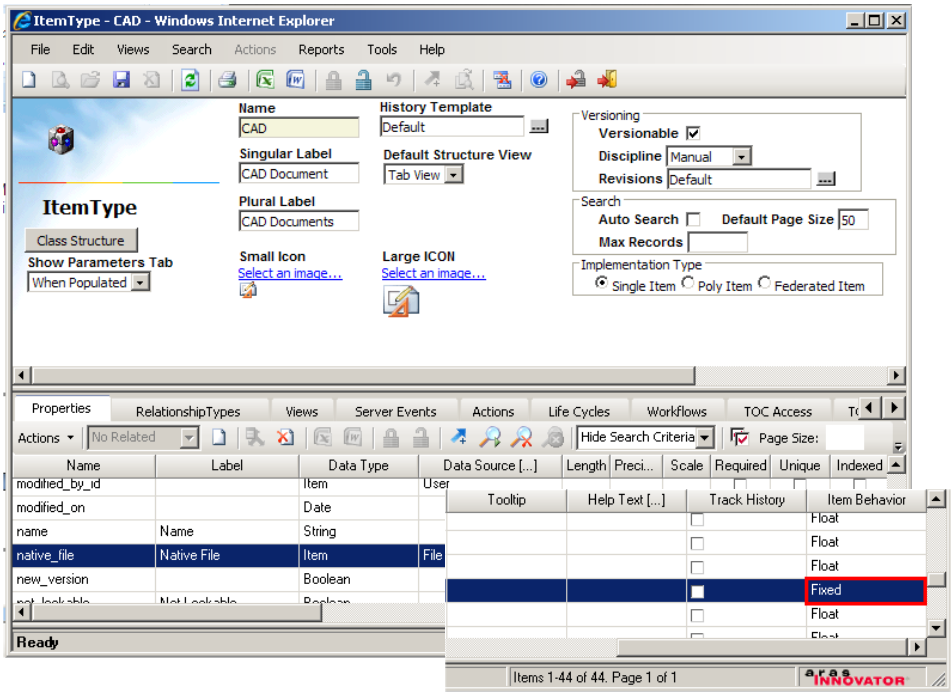
Please set the versioning discipline for the item types "Part" and "CAD" to "Manual". Now the new generation of a part and CAD document will not be created automatically in case of an update (see *Picture 22: Item Type "Part"*).



Picture 22: Item Type "Part"

Accessibility of old file versions

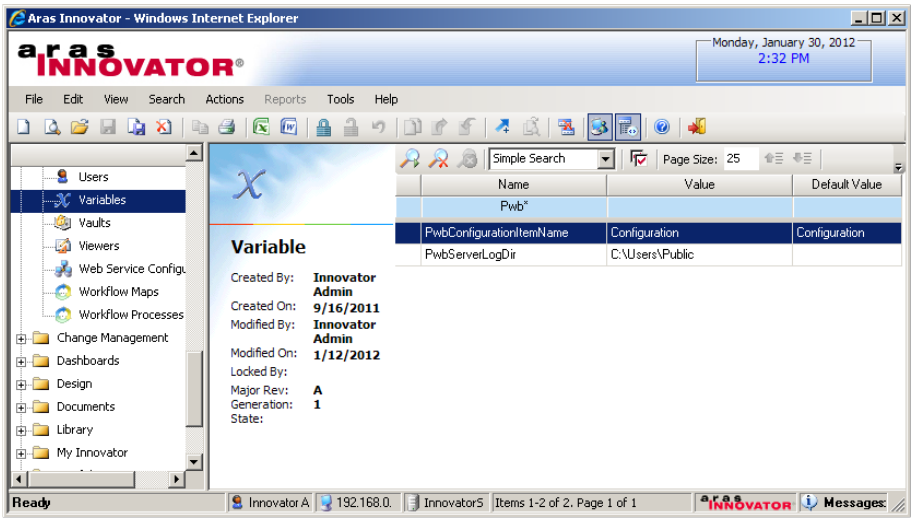
Please set the "Item Behavior" of the property "native_file" for the item type "CAD" to "Fixed". Now the old file versions can be accessed (see *Picture 23: Item Type "CAD"*).



Picture 23: Item Type "CAD"

Configuration Variables

The following Aras Innovator server configuration variables need to be set for PDM-Workbench to work correctly:



Picture 24: Aras Innovator server configuration variables

- PwbConfigurationItemName

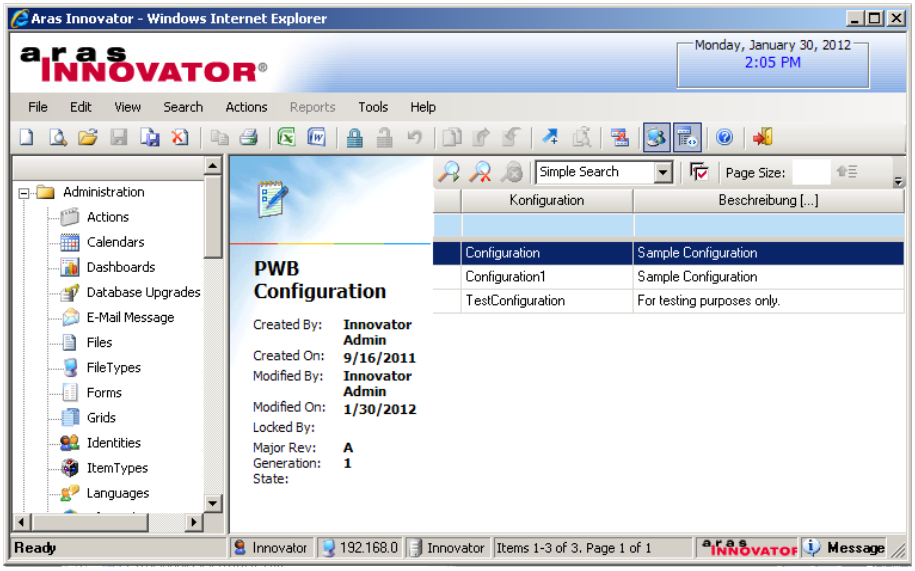
  The name of the PDM-Workbench configuration item which contains additional configuration information, like the attribute mapping configuration. Please see “Configuration Items” for more details.

- PwbServerLogDir

The absolute path of the directory into which the server log file should be written. If this variable is empty then no server log file will be written.

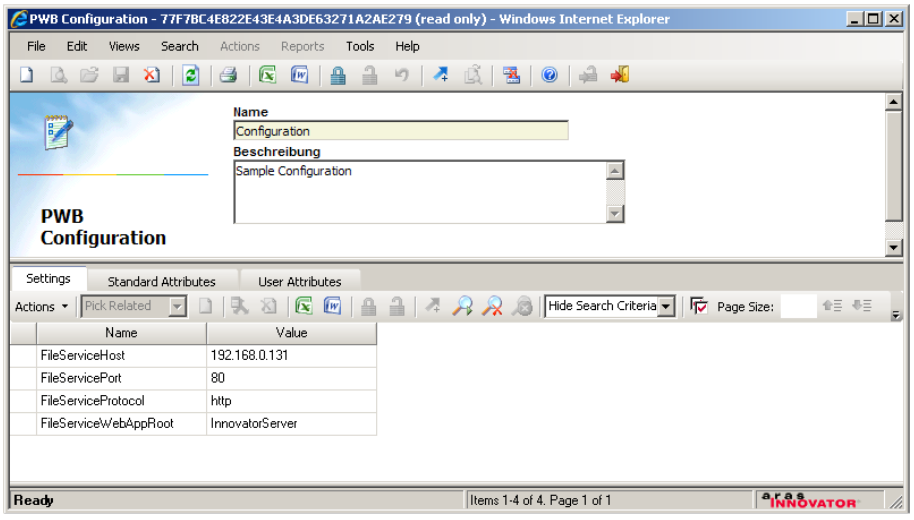
Configuration Items

In order to define the environment variables and to configure the mapping of attributes between Aras Innovator and CATIA V5 a special configuration item (see *Picture 25: PWB Configuration item in Aras Innovator*) has to be used:



Picture 25: PWB Configuration item in Aras Innovator

The following Aras Innovator server configuration variables need to be set for the PDM-Workbench to work correctly:



Picture 26: Aras Innovator server configuration variables

- FileServiceHost

  The IP address of the host on which the file vault server runs. This can, but does not have to be, the same as the Aras Innovator server host.

- FileServicePort

The port number of the Aras Innovator file vault process. The process has to run on the host whose IP address is defined in the variable “FileServiceHost”.

- FileServiceProtocol

  The file service protocol, e.g. "http".

- FileServiceWebAppRoot

  The web application root string of the file service http process.

- CadDocNumberAttr

  Default value: item_number

  Optional. The attribute containing the CAD document number of the document class.

- PartNumberAttr

  Default value: item_number

  Optional. The attribute containing the part number of the part class.

- CustomMethod_PreProcCreDlgAttrs

  Optional.

- InitialVersionCadDoc

  Optional. The initial version string for the item type "CAD Document".

- InitialVersionPart

  Optional. The initial version string for the item type "Part".

- PromoteSourceStates

  Default value: Preliminary|In Review

  Optional. A list of the promote source states, separated by "|"..

- PromoteTargetStates

  Default value: In Review|Released

  Optional. A list of the promote target states, separated by "|".

- ShowCreateDialogsDuringUpdate

  Optional. Has to be set to "true" in order to show the create dialogs during the update process.

- UseBomPartStructure

  Optional. Has to be set to "true" in order to use the "BOM Part Structure" data model. Otherwise the document data model will be used.

.

- CreateThumbnailsFromTypes

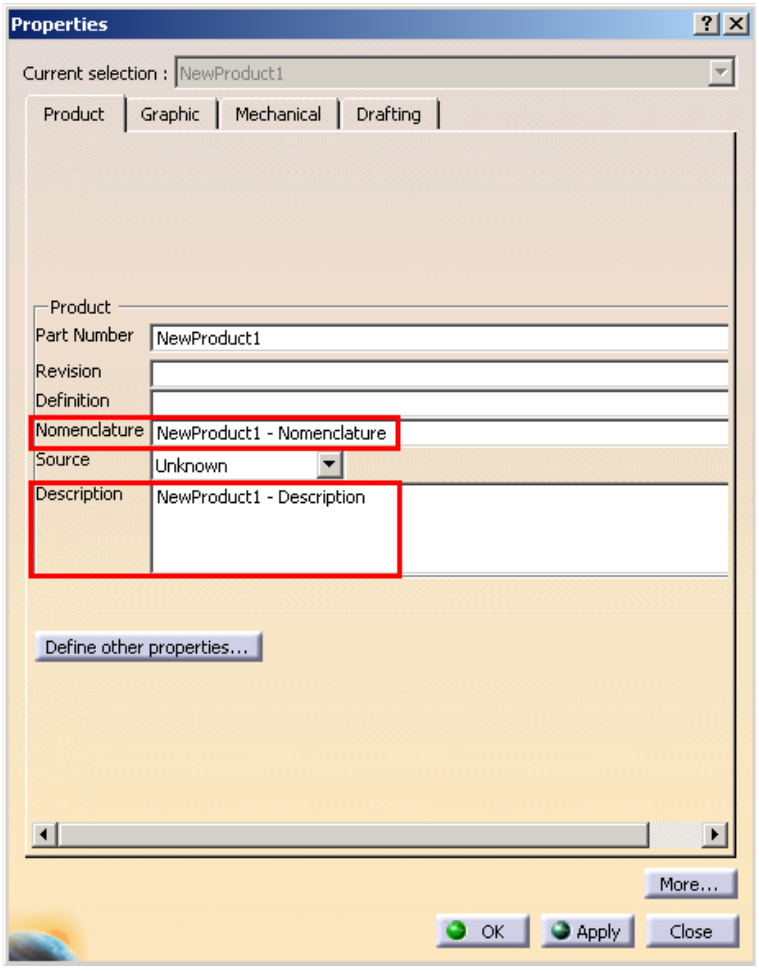
  Default value: .CATPart|.CATDrawing

  Optional. A list of file extensions for which the thumbnails should be created by update, separated by "|".

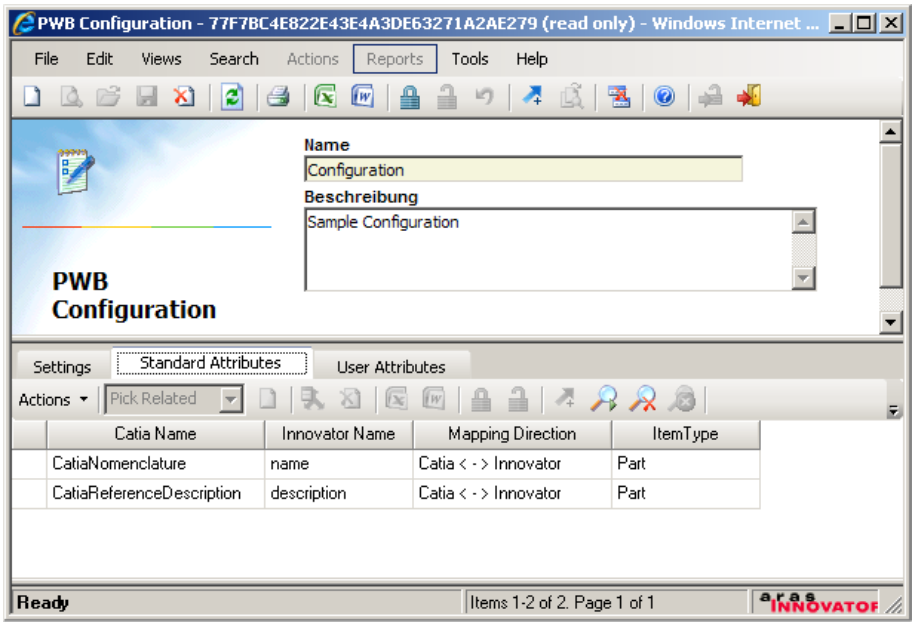
Here is a description of how to configure the attribute mapping:

CATIA standard and user-defined properties can be mapped to PDM attributes.

In the following example the standard CATIA attributes “Nomenclature” and “Description” are mapped to the attributes “name” and “description” of the Aras Innovator part object (see *Picture 27: Standard attributes in the “Properties” dialog* and *Picture 28: Configuration of standard attributes in Aras Innovator*).

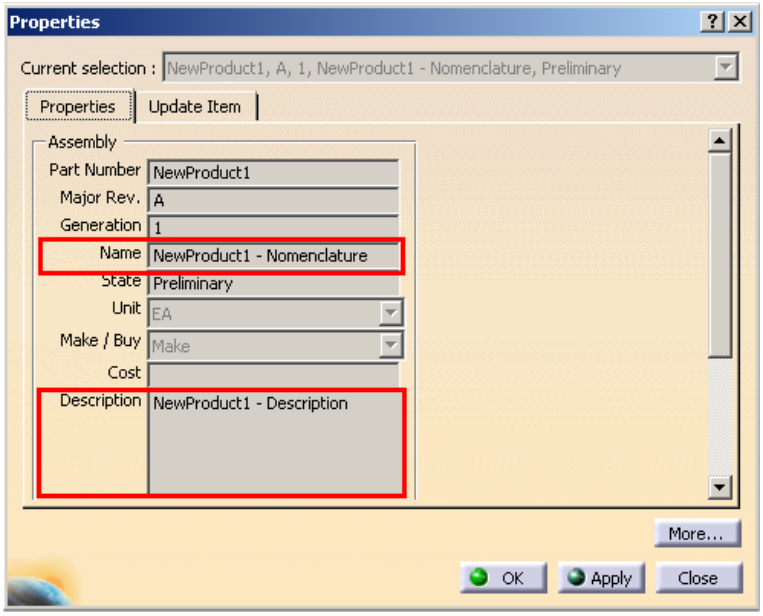


Picture 27: Standard attributes in the “Properties” dialog

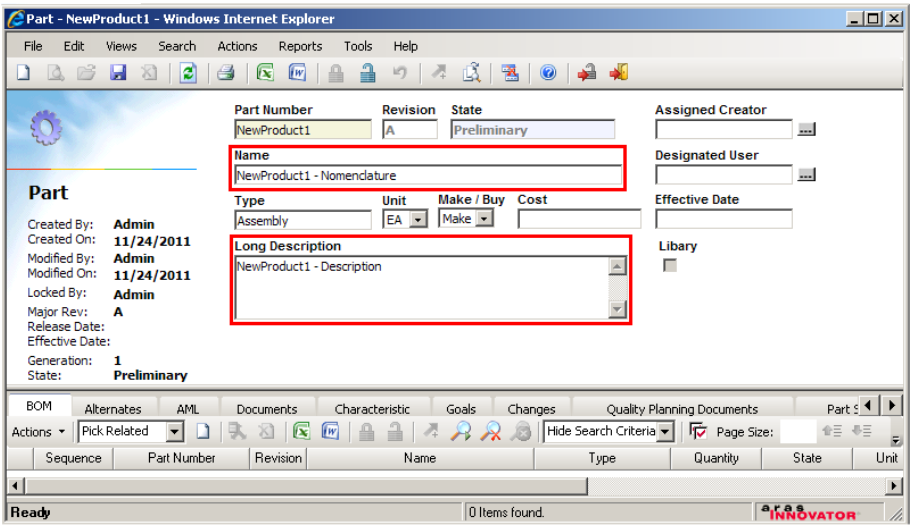


Picture 28: Configuration of standard attributes in Aras Innovator

After creating the part with Update the defined CATIA attribute values have been written to the PDM part object (see *Picture 29: Standard attributes in the "Properties" dialog of the PDM node* and *Picture 30: Standard attributes in Aras Innovator window*).

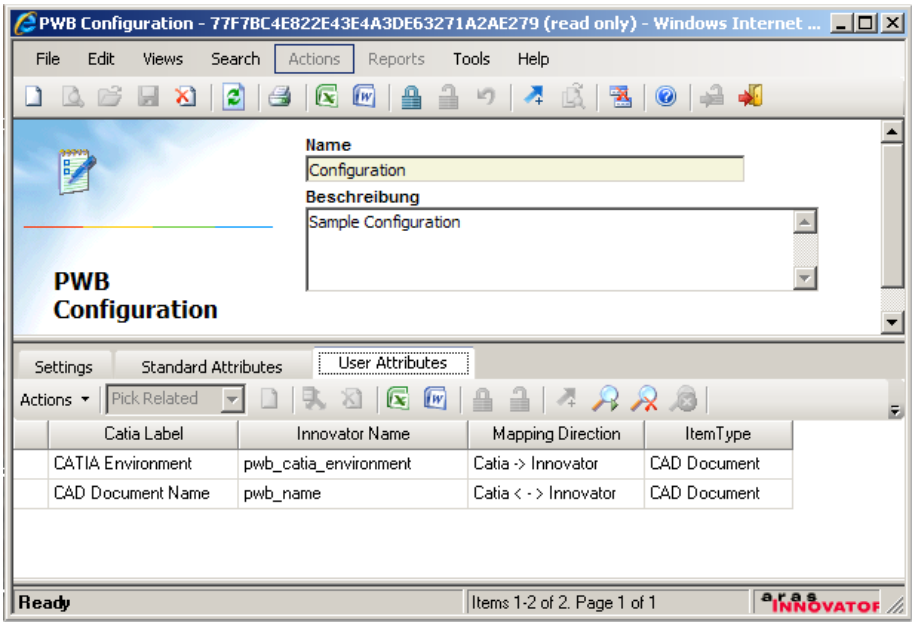


Picture 29: Standard attributes in the "Properties" dialog of the PDM node



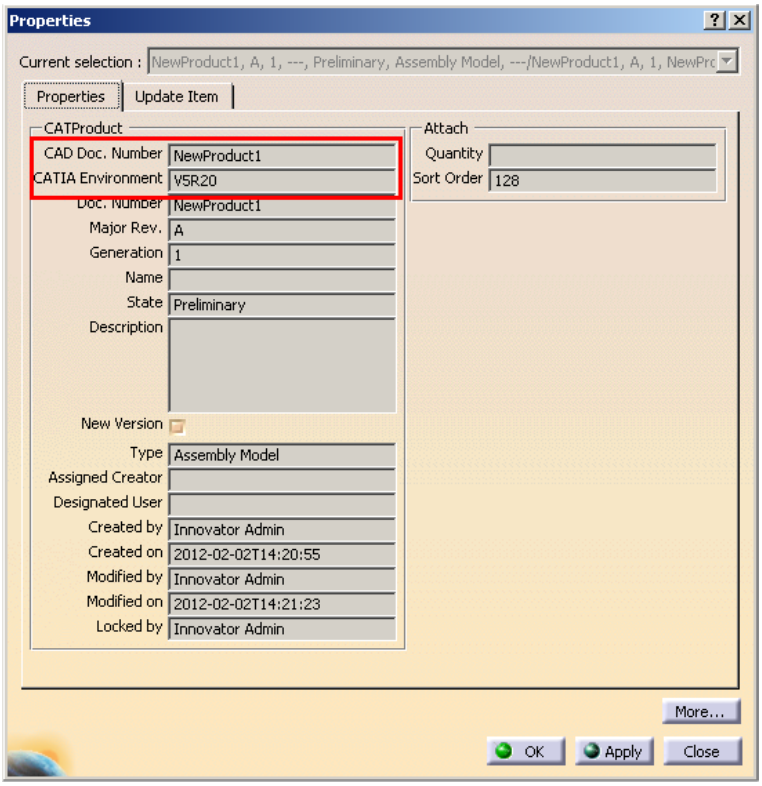
Picture 30: Standard attributes in Aras Innovator window

User-defined CATIA properties can also be mapped (see *Picture 31: Configuration of user-defined attributes in Aras Innovator*).

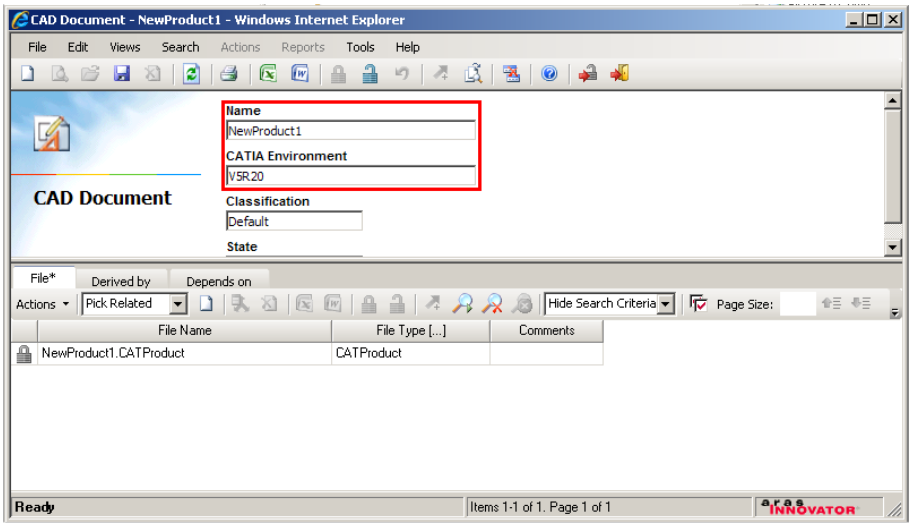


Picture 31: Configuration of user-defined attributes in Aras Innovator

While the structure is imported the values are written to the defined attributes of the Aras Innovator CAD document object (see and *Picture 32: User-defined attributes in the "Properties" dialog of the PDM node* and *Picture 33: User-defined attributes in Aras Innovator window*).

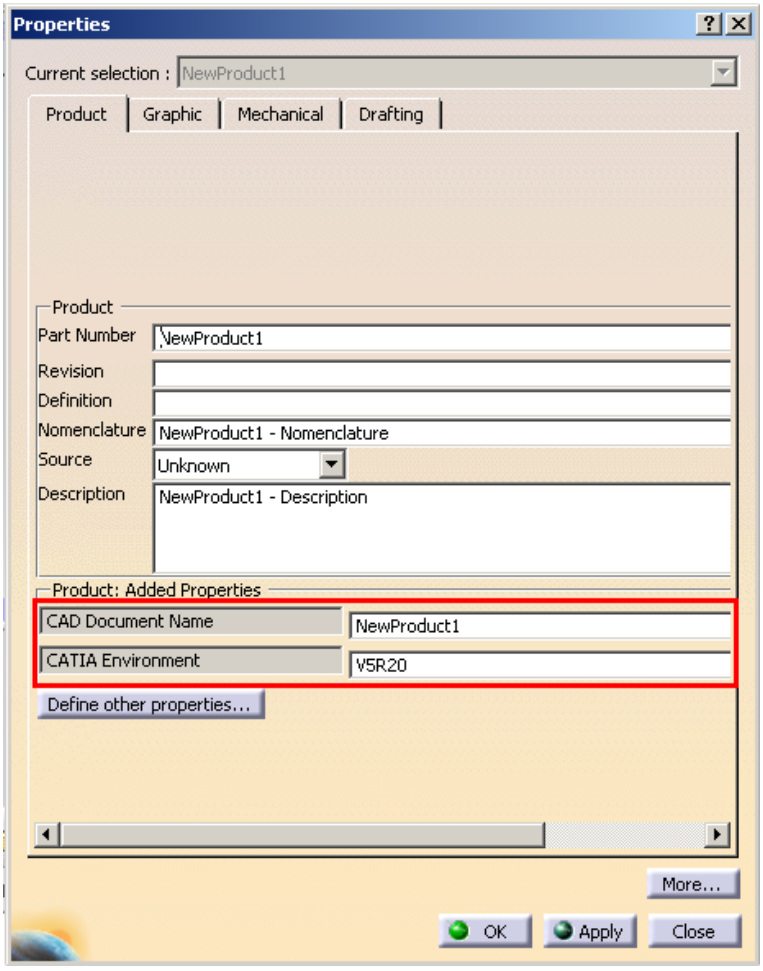


Picture 32: User-defined attributes in the "Properties" dialog of the PDM node



Picture 33: User-defined attributes in Aras Innovator window

After the import or after loading the structure it can be shown that the values are written from the PDM attributes into the CATIA files (see *Picture 34: User-defined attributes in the "Properties" dialog*).



Picture 34: User-defined attributes in the “Properties” dialog

CHAPTER 7

Client Schema File Configuration

This chapter describes the configuration of the client side of the PDM-Workbench integration.

Structure of the Schema File

The main purpose of the PDM-Workbench Schema File is to define which subset of the objects, relations, and attributes in the PDM system should be made available to the design engineer who is working with CATIA V5 and who needs to save the CATIA files he is working on in a PDM system.

The classes of PDM objects that the user can query, create, etc. will be defined in the Schema file, as well as the dialogs which contain these objects' attributes and the PDM relations which relate the PDM objects to each other.

The Schema file can be edited with a text editor, or a XML editor.

At the root of the Schema XML file, there is the tag "PWBSchemata". Its child tags are named "PWBSchema". The information about every PDM system that can be accessed is defined inside this "PWBSchema" tag. There is one "PWBSchema" tag for every PDM system and every PDM system customization that can be accessed from the PDM-Workbench.

```
<!-- root tag -->
<PWBSchemata>
   <!-- out-of-the-box Aras -->
   <PWBSchema system="Aras" customization="Aras"
              displayName="NLS_System" visibleLength="15">
   ...
   </PWBSchema>

   <!-- customization of Aras -->
   <PWBSchema system="Aras" customization="PDM-Customization"
              displayName="NLS_System" visibleLength="15">
   ...
   </PWBSchema>
</PWBSchemata>
```

Attributes of the tag "PWBSchema":

- "system" — Contains the short name of the PDM system. Supported is "Aras" for Aras Innovator.
- "customization" — Contains the name of the customization. If the PDM system is used out of the box without any customization, then the convention is to use the short name as defined for the

“system” attribute.

- "displayName" Contains the NLS (native language support) name of the PDM system or customization that is defined in the "PWBSchema" XML tag.
- "visibleLength" Contains the visible length of the display name to be shown in the dialogs of CATIA V5.
- "allowedLength" Contains the allowed length of the values inserted in the text editor widgets in characters.

NLS Support for Display Names

Many XML tags (PWBSchema, frame, language, object, relation, attribute, etc.) have an attribute with the name "displayName". The string that represents the value of that attribute defines the language-specific display name for that object that the PDM-Workbench users can see. The language-specific name is defined in the files "PWBSchemaDisplayNames.CATNls" and "PWBSchemaDisplayNames_SYSTEM_CUSTOMIZATION.CATNls", where SYSTEM is the value of the "system" attribute and CUSTOMIZATION is the value of the "customization" attribute. For system="Aras" and customization="Aras" the name of the CATNls file would be "PWBSchemaDisplayNames_Aras_Aras.CATNls". That file contains the NLS names specific for that PDM system or customization, while "PWBSchemaDisplayNames.CATNls" contains the general definitions that apply to all PDM systems.
In this case, the value for the frame's display name "NLS_UserData" is defined in the file "PWBSchemaDisplayNames_Aras_Aras.CATNls":

File "**PWBSchema.xml**":

```
...
<PWBSchema system="Aras" customization="Aras" displayName="NLS_System"
           visibleLength="15">
...
```

File "**PWBSchemaDisplayNames_Aras_Aras.CATNls**":

```
...
NLS_System = "Aras Innovator";
...
```

Contents of a PWBSchema XML tag

"xmap": optional

The value contains the absolute path of the exchange map directory, where the downloaded CATIA files are stored.

Example:

```
<xmap value="C:\PWB_XMAP" />
```

If the exchange map value is defined by the CATIA V5 environment variable "PWB_XMAP", then that takes precedence. The definition in the Schema file takes effect only if such a CATIA V5 environment variable does not exist.

"soapTargetUrl": optional

The value contains the URL of the server process that the PDM-Workbench client uses for its SOAP requests.

Example:

```
<soapTargetUrl value="http://epdmd11:80/InnovatorServer" />
```

If the soap target URL value is defined by the environment variable "PWB_SOAP_TARGET_URL", then this one takes precedence. The definition in the Schema file takes effect only if such a CATIA V5 environment variable does not exist.

"externalSoapClientCallPathEnvVar": mandatory

The name of the environment variable which contains the path to the Aras Innovator SOAP client executable.

"externalFileClientCallPathEnvVar": mandatory

The name of the environment variable which contains the path to the Aras Innovator file client executable.

"addTempPrefix": optional

The prefix for the rename of the Part Numbers and File Names for the "Add Temp" and "Open File Temporary" command.

Example:

```
<addTempPrefix value="TMP" />
```

If the add temp prefix value is defined by the environment variable "PWB_ADDTEMP_PREFIX", then this one takes precedence. The definition in the Schema file takes effect only if such a CATIA V5 environment variable does not exist.

The default value is "TMP".

"pwbWindowColor": optional

This tag contains the red, green, and blue values (0 - 255) of the PWB window background.

Example:

```
<pwbWindowColor red="143" green="155" blue="177" />
```

If not set then the CATIA V5 standard color will be used.

"maxExpansionLevel": optional

The value contains the maximum expansion level, starting from a root node the PDM-Workbench will display.

Example:

```
<maxExpansionLevel value="30" />
```

If not set then all levels of the assembly will be expanded.

"showDisplayStringInQueryListView": optional

Description.

"maximumDescriptionAttributeLength": optional

Description.

"showDataSourceNlsValues": optional

Description.

"removeToolbarIcons": optional

This tag includes the names of the toolbar entries to be removed.

Example:

```
<removeToolbarIcons>
   <icon name="Register" />
   <icon name="Synchronize" />
   <icon name="SetSessionConfig" />
```

```
   <icon name="NewPwbWindow" />
   <icon name="DynamicActions" />
</removeToolbarIcons>
```

“queryDialogActions”: optional

This tag includes the names of the buttons in the query dialog.

Example:

```
<queryDialogActions>
   <action name="OpenInPwbWindow" />
</queryDialogActions>
```

"sessionSettings ": optional

The value contains the session settings of the PDM-Workbench. The following entries are supported:

```
<queryMode name="listViewWindow" />
<queryMode name="pwbWindow" />
<relationDisplayMode name="relDisplay" />
<relationDisplayMode name="noRelDisplay" />
<passwordEncryption name="WinAuth" />
<passwordEncryption name="MD5" />
<passwordEncryption name="none" />
```

"installedLanguages": mandatory

This tag contains the name and date format of the installed languages (NLS files).

Example:

```
<installedLanguages visibleLength="15">
   <language name="en_us" displayName="NLS_EN"
             dateFormat="PWB_Standard" />
</installedLanguages>
```

"dateFormat": mandatory

This tag contains the definition of the date format for the installed languages.

Example:

```
<!-- PWB_Standard = YYYY-MM-DD -->
<dateFormat name="PWB_Standard" separator="-">
   <dateValue name="year" length="4" />
   <dateValue name="month" length="2" />
   <dateValue name="day" length="2" />
</dateFormat>
```

The order of the “dateValue” tags defines the format of the date. For each part of the date the length is defined in the tag.

"form " (Login Form): mandatory

This tag contains a description of the Login form. It defines the attributes needed for logging in to the PDM system. Generally it contains the attributes "login name", "password", and “database” at least, though other attributes like "group" can be defined if it is necessary for the PDM system.

Example:

```
<form name="Login" info="ShowOnlyLoginData">
```

```
   <frame displayName="NLS_UserData">
      <pwbFormAttribute name="PWBLoginUser" widgetType="ComboBox"
                        mode="update" visibleLength="10" required="true"
                        entryAllowed="true" dataSource="UserNames" />
      <pwbFormAttribute name="PWBLoginPassword"
                        widgetType="SingleLineEditor" mode="update"
                        visibleLength="10" required="false" />
      <formAttribute name="LoginDatabase" widgetType="ComboBox"
                     mode="update" visibleLength="10" required="true"
                     entryAllowed="false" />
   </frame>
</form>
```

The XML tags inside the "frame" tag describe how the attributes "user" and "password" are displayed in the login dialog.

"partClasses": optional

The value describes a list of class names (possibly of size 1) of all part classes. It has to be defined if the class can have sub parts.

Example:

```
<partClasses>
   <partClassName name="/Part/Assembly" canHaveSubParts="true"/>
   <partClassName name="/Part/Component" canHaveSubParts="false"/>
</partClasses>
```

"multiQuantityRelationAllowed": mandatory

The value indicates if the part-part relation created during the synchronize process will be created as multi quantity relation.

If there exists already a part-part relation between the two objects the same relation type will be used, not respecting the value.

Example:

```
<multiQuantityRelationAllowed value="true" />
```

For Aras Innovator the value has to be “true”.

"createRelationType": mandatory

The value indicates the type of the newly created part-part relation during the synchronize process.

Even if the multi quantity relation is allowed (see above) then a single quantity relation can be created.

Examples:

```
<createRelationType value="single"/>
<createRelationType value="multi"/>
```

For Aras Innovator the value has to be “multi”.

“catiaNodeBehaviorDefinitions": optional

Description.

Example:

```
<catiaNodeBehaviorDefinitions>
   <catiaNodeBehavior catiaNodeType="EmbeddedComponent"
                      partNumberPrefix="" behavior="SkipNode"/>
</catiaNodeBehaviorDefinitions>
```

"templateCatiaFilePath": optional

The path to the directory which contains the template CATIA files.

Example:

```
<templateCatiaFilePath value="C:\files\PDM-Workbench\Templates" />
```

"object": 1-n

This tag contains the definition of a PDM object class which can be used (queried, created, etc.) by the user.

The definition of PDM object classes, their corresponding dialogs and the actions that can be performed on them are described in the chapter **PDM Objects**.

"relation": 1-n

This tag contains the definition of PDM relation classes that can be accessed (expanded, created, etc.) by the user.

The definition of PDM relation classes, their corresponding dialogs and the actions that can be performed on them are described in the chapter **PDM Relations**.

"attribute": 0-n

The definition of PDM attributes that are referenced in dialogs.

Attributes and dialog forms are explained in the chapter **PDM Attributes and Form Attributes**.

"pwbAttribute": 0-n

The definition of attributes that do not correspond directly to PDM attributes of PDM objects.

Attributes and dialog forms are explained in the chapter **PDM Attributes and Form Attributes**.

"dataSource": 0-n

Data sources contain attribute values. By assigning data sources to attributes default values for these attributes can be defined.

Data sources are explained in the chapter **Data Sources**.

PDM Attributes and Form Attributes

Every PDM attribute that is displayed in a dialog form should be defined in an "attribute" tag.

The attribute definition contains the following attributes:

- "name" Mandatory, must correspond to the PDM attribute's name.
- "displayName" Mandatory. As described in "NLS Support for Display Names" the NLS string for the "displayName" XML attribute is defined in the CATNls file specific to the PDM system and the customization.
- "dataSource" Optional. The data source includes the possible values for

this attribute.

- "isFileName" Optional. If is set to "true" the value of the corresponding input filed is checked about illegal[2] characters when creating a file.
- "isPartNumber" Optional.
- "autoName" Optional.
- "isDerived" Optional.

Example:

```
<attribute name="name" displayName="NLS_Name" isFileName="true"
           isPartNumber="true" autoName="true"/>

<attribute name="current" displayName="NLS_current"
           dataSource="LifeCycleStates"/>

<attribute name="revision" displayName="NLS_Revision" />
```

A form definition contains form attributes which reference the previously defined PDM attribute.

The form attribute definitions contain the following attributes:

- "name" Mandatory, must correspond to the PDM attribute's name.
- "displayName" Optional. As described in "NLS Support for Display Names" the NLS string for the "displayName" XML attribute is defined in the CATNls file specific to the PDM system and the customization. If not defined here the display name of the "attribute" tag will be used.
- "mode" Possible values are "output" (read-only), "update" (can be modified), or "select" (e.g. for combo boxes).

  Default is "output".
- "visibleLength" Optional, the length of the text editor widget in characters.
- "allowedLength" Optional, the length of value that can be inserted in the text editor widget in characters.
- "required" "true" or "false". If "true", then a value must be set.

  Default is "false".
- "widgetType" Possible values are "SingleLineEditor", "MultiLineEditor", "ComboBox", "SingleCheckBox", "CheckBoxes", "RadioButtons", "SingleSelectorList", "MultiSelectorList", "NameValueList", "Date".

  Default is "SingleLineEditor".
- "embeddedObjAttr" Optional. If the PDM attribute refers to a different PDM attribute in a contained object attribute, then this XML attribute's value contains that object attribute's name.

[2] Filenames must not contain control characters, non printable characters and any of the following characters: `*?:;\/<>|`

- "embeddedAttribute" — Optional. The name of the PDM attribute of the embedded object. If "embeddedObjAttr" is set, then "embeddedAttribute" must be set, too.
- “dataSource” — Optional. The value defines the link to a data source that is more special than the linked data source in the <attribute> tag.
- "listViewRelevant" — "true" or "false". If "true", then the attribute will appear in the query list view.

  This attribute should only be added to "Query" forms. Please refer to the user manual for more information.

  Default is "false".
- "sort" — Possible values are "true" or "false". If "true", then the attribute can be sorted in the query result window.
- "displayOnly" — Possible values are "true" or "false". If "true", then the display value of the value of the data source will be used.
- "entryAllowed" — Possible values are "true" or "false". If "true", then the user can enter a text additional to the attached data source.

Example:

```
<form name="Query">
   <formAttribute name="name" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />
```

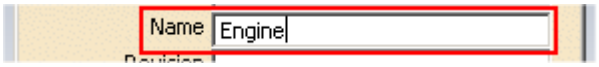
Form definitions generally refer to classes of PDM objects (query form, properties form, etc.). The definition of PDM object classes is described in chapter **PDM Objects**.

Description of the Widget Types

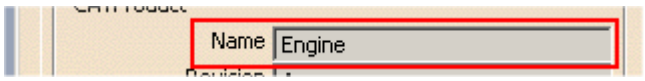
There are ten different widget types available to build up dialogs with. All widget types except “SingleLineEditor”, “MultiLineEditor” and one mode of “NameValueList” can only be used on attributes that have certain kinds of Data Sources attached. Data Sources are a container of a limited set of values.

The detailed explanation of Data Sources you can find in chapter **Data Sources**.

SingleLineEditor — Supports "update" and "output" mode.
Can be used for attributes with no data source attached and also for attributes with data sources of type “SingleValue”.

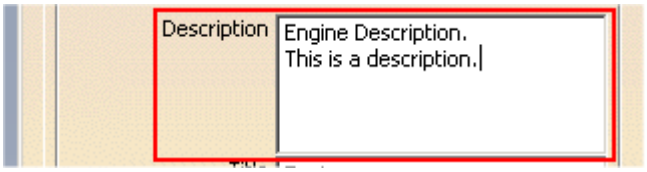


Picture 35: Single Line Editor Widget, update mode



Picture 36: Single Line Editor Widget, output mode

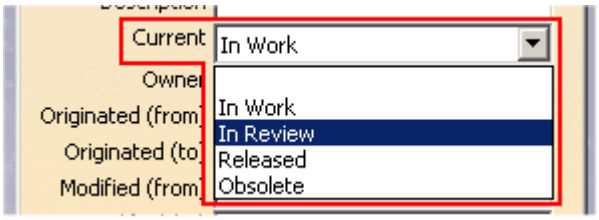
MultiLineEditor — Supports "update" and "output" mode.
Can be used for attributes with no data source attached and also for attributes with data sources of type “ValueList”.



Picture 37: Multi Line Editor Widget, update mode

ComboBox

Supports “select” and “output” mode.
This widget type can only be used for attributes with data sources of type “ValueList”, “BooleanValueList” or “invokeMessage” if this message returns a set of values.



Picture 38: Combo Box Widget, select mode

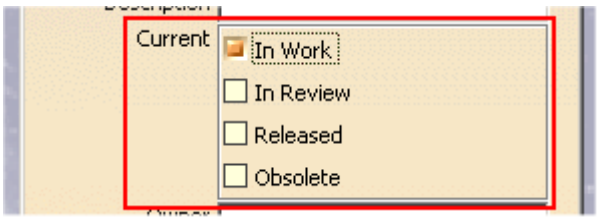
SingleCheckBox

Supports “select” and “output” mode.
Needs an attribute with a data source of type "BooleanValueList".
This widget should be used only for required attributes or for attributes that are only displayed, already set to a value and cannot be updated.

Picture 39: Single Check Box Widget, select mode

CheckBoxes

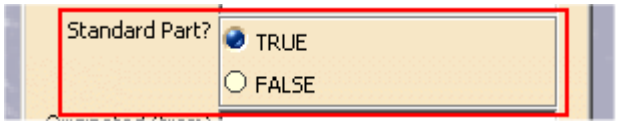
Supports “select” and “output” mode.
Possible for attributes with data sources which contain several values (type "ValueList”, “BooleanValueList” or “invokeMethod”) and where the user can select more than one value.



Picture 40: Check Boxes Widget, select mode

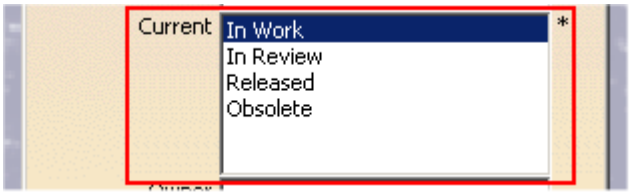
RadioButtons

Supports “select” and “output” mode.
Possible for attributes with a data source of type "ValueList", “BooleanValueList” or “invokeMethod” where only one value can be selected at one time.
They should be used for required attributes only, because one value has always to be selected.



Picture 41: Radio Buttons Widget, select mode

SingleSelectorList Supports “select” and “output” mode.
It represents a list with one column where one item is selectable.
It can be used for all attributes with attached data sources of type "ValueList", “BooleanValueList” or “invokeMethod”.



Picture 42: Single Selector List Widget, select mode

MultiSelectorList Supports “select” and “output” mode.
It represents a list with one column where several items are selectable.
(The multi-selector list looks like the single-selector list, except that more than one item of the list can be selected.)
It can be used for all attributes with attached data sources of type "ValueList", “BooleanValueList” or “invokeMethod”.

NameValueList Supports “select”, “update” and “output” mode.
It represents a list with two columns (e.g. for name value sets)

→ when working in update mode the widget type can be used for all attributes within the Schema file (no data source needed). The user can change every single column item (columns can be empty).

→ when working in select mode the widget type can only be used for attributes with data sources of type “NameValueList” attached. (widget acts as a filter then)

PDM Objects

Object XML tags define PDM object classes that can be used in the PDM-Workbench application. They represent the subset of objects defined within the PDM system which are needed in a PDM-CAD integration.

An "object" XML tag contains the following attributes:

- "name" The internal PDM class name.
- "displayName" The class name that is shown to the user.
- "icon" The icon that represents the class in the PDM window and the list window.

Example:

```
<object name="/Part/Assembly" displayName="NLS_Assembly" icon="Aras_Part">
```

For the icon to be displayed correctly in the PDM window a bitmap file with the name of the icon (in this example "Aras_Part.bmp") must exist in the subdirectory "resources\graphic\icons\normal" of the CATIA V5 directory (e.g. "intel_a" on Windows 32 Bit CATIA installation, and “win_b64” on Windows 64 Bit CATIA installation).

The list view window needs a bitmap file with the file name (icon name) + "16x16.bmp", in this example "Aras_Part16x16.bmp".

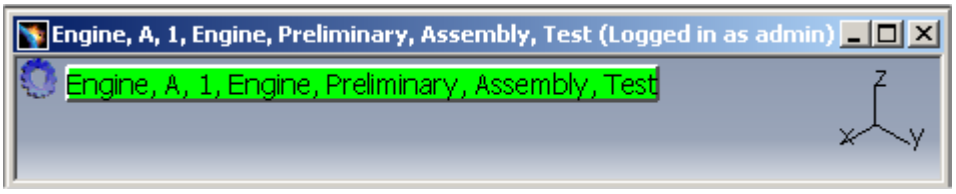
As described in "NLS Support for Display Names" the NLS string for the "displayName" XML attribute is defined in the CATNls file specific to the PDM system and the customization.

Description of PDM Objects

The "description" tag defines which of the attributes of the class should be displayed beside the icon. In this example, these are the attributes “item_number”, “major_rev”, “generation”, “name”, and “state”.

Example:

```
<description>
   <descAttribute name="item_number" />
   <descAttribute name="major_rev" />
   <descAttribute name="generation" />
   <descAttribute name="name" />
   <descAttribute name="state" />
   <descAttribute name="classification" />
   <descAttribute name="description" />
</description>
```



Picture 43: PDM Node in PWB window

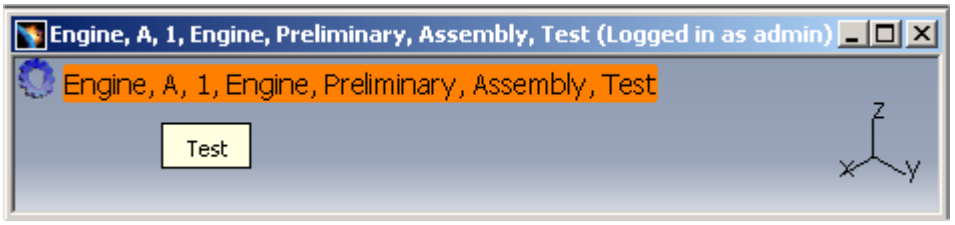
Tooltip of PDM Objects

The "tooltipAttribute" tag defines which string has to be shown as tooltip for the object. The value can be the name of an attribute of the object. If this value is empty then the description defined in the chapter above will be used.

Example:

```
<tooltipAttribute name="description" />
```

The attribute “description” has the value “Test”.



Picture 44: Tooltip of PDM Node in PWB window

Actions on PDM Objects

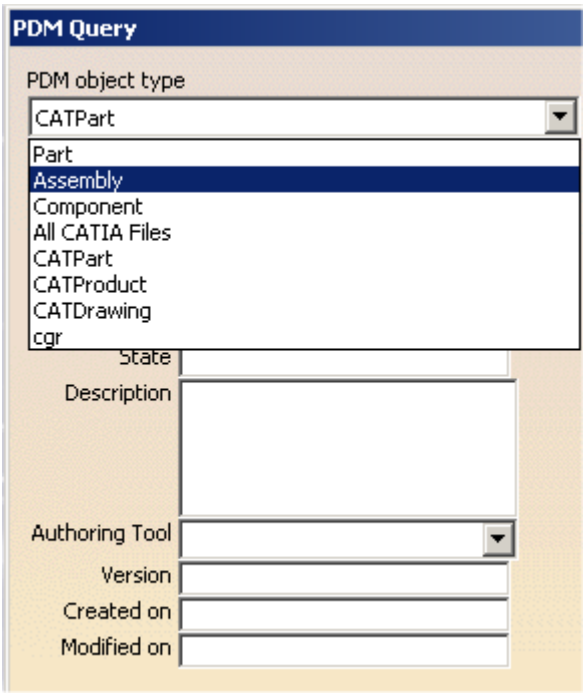
Actions that can be performed with PDM objects have to be defined in the Schema file. There are two kinds of actions: So-called toolbar actions, which are started by clicking on an icon in the PDM-Workbench toolbar, and context actions, which are started by right-clicking on the node and selecting one of the context menu items.

Toolbar actions are defined with an "action" tag. The action "Query" can be defined on any object type.

Example:

```
<!-- * all PWB toolbar actions permitted for this object * -->
<action name="Query" />
```

If, for instance, the action "Query" is defined for the object type "/Part/Assembly", then, when the user clicks on the "Query" toolbar icon, the type "Assembly" (display name) is included in the query dialog list, otherwise it is not.



Picture 45: Select PDM object type in PDM Query dialog

Context Actions

Context actions are declared similarly to toolbar actions. For context actions, the tag "contextAction" is used.

Example:

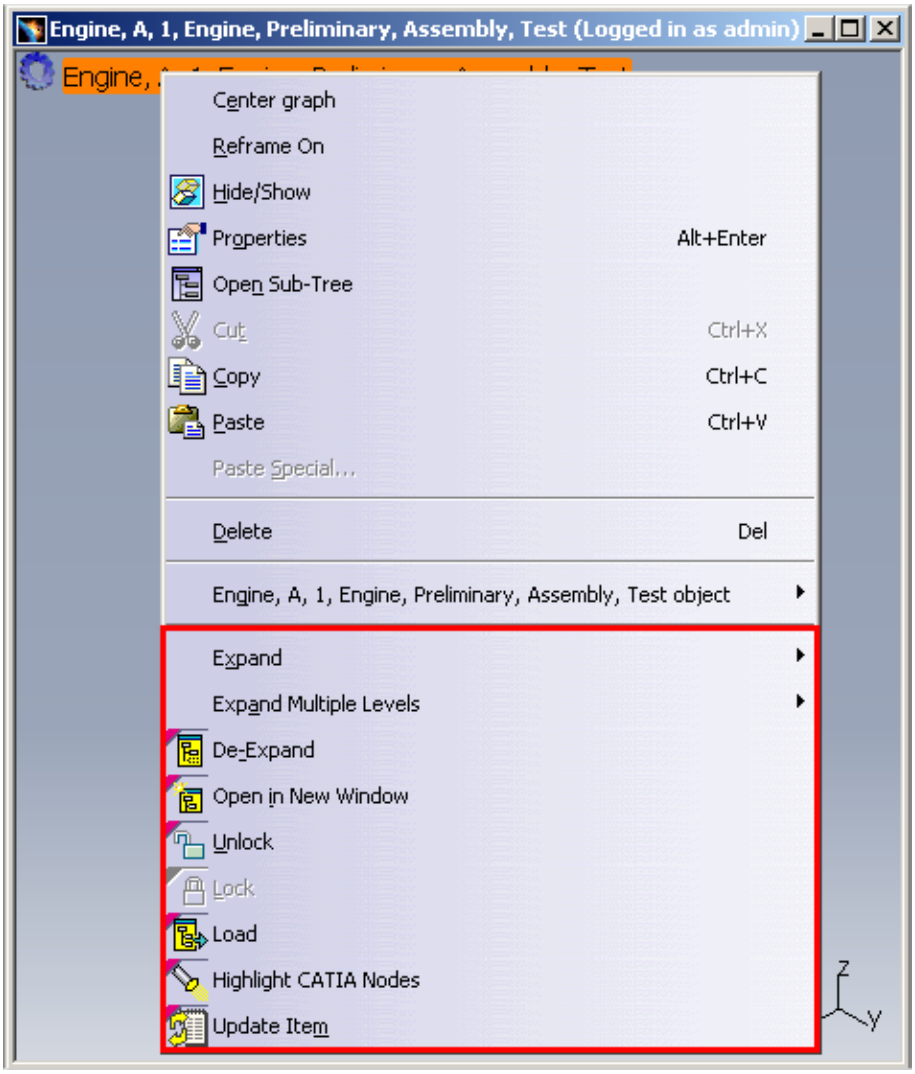
```
<!-- * all PWB context actions permitted for this object * -->
<contextAction name="Expand" usedIn="PdmWindow" />
<contextAction name="MultipleExpand" usedIn="PdmWindow" />
<contextAction name="DeExpand" usedIn="PdmWindow" />
<contextAction name="CheckIn" usedIn="PdmWindow|QueryDialog" />
<contextAction name="CheckOut" usedIn="PdmWindow|QueryDialog" />
<contextAction name="LoadStructure" usedIn="PdmWindow" />
<contextAction name="Highlight" usedIn="PdmWindow" />
<contextAction name="UpdateItem" usedIn="PdmWindow" />
<contextAction name="DeleteItem" usedIn="PdmWindow" />
```

The attribute "usedIn" defines where the context action is shown. The attribute value "PdmWindow" activates the action in the PDM window. The attribute value "QueryDialog" activates the action in the result list view of the query dialog. These two attribute values can be combined with the pipe "|".

The context action "Properties" has a window with two tabs for “Properties”, and "UpdateItem". The corresponding dialog forms must also be defined for that class.

The context action “Properties” is default and not explicitly defined in the Schema file.

If a certain context action is defined for a PDM object class in the Schema file, then the corresponding context menu entry for object nodes of that class exists.



Picture 46: Context actions for the type /Part/Assembly

Some things to keep in mind regarding the definition of context actions:

If the context action "Unlock" (CheckIn) is defined, then the dialog forms "CheckInNew" and "CheckIn" must also be defined.

If the context action "Lock" (CheckOut) is defined, then the dialog form "CheckOut" must also be defined.

The context action “Load” (LoadStructure) should only be defined on part objects.

PDM Object Forms

The following forms can be defined for an object class:

"Query", "Properties", "UpdateItem", "CheckInNew", "CheckIn", "CheckOut"

The "CheckIn" and "CheckOut" forms do not have to contain any attributes, they just need to be defined.

Example:

```
<form name="Query">

   <formAttribute name="item_number" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="major_rev" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />
```

```
    <formAttribute name="generation" widgetType="SingleLineEditor"
                   mode="update" visibleLength="15" required="false"
                   listViewRelevant="true" />

    <formAttribute name="name" widgetType="SingleLineEditor" mode="update"
                   visibleLength="15" required="false"
                   listViewRelevant="true" />

    <formAttribute name="state" widgetType="SingleLineEditor" mode="update"
                   visibleLength="15" required="false"
                   listViewRelevant="true" />

    <formAttribute name="unit" widgetType="ComboBox" mode="update"
                   visibleLength="15" required="false" />

    <formAttribute name="make_buy" widgetType="ComboBox" mode="update"
                   visibleLength="15" required="false" />

    <formAttribute name="description" widgetType="MultiLineEditor"
                   mode="update" visibleLength="15" required="false"
                   listViewRelevant="true" />

    <formAttribute name="created_on" widgetType="SingleLineEditor"
                   mode="update" visibleLength="15" required="false"
                   listViewRelevant="true" />

    <formAttribute name="modified_on" widgetType="SingleLineEditor"
                   mode="update" visibleLength="15" required="false"
                   listViewRelevant="true" />

</form>
```

PDM Relations

Relation XML tags define PDM relation classes that can be used in the PDM-Workbench application. They represent the subset of the relations defined in the PDM system that are needed in PDM-CAD integration.

As with the "object" XML tags, a "relation" XML tag contains the following attributes:

- "name" The internal PDM class name.
- "displayName" The class name that is shown to the user.
- "icon" The icon that represents the class in the PDM window and the list window.
- "createAllowed" Defines whether relations of this class can be created by the user. Relations are created by copying and pasting object nodes. If "createAllowed" is "true", then, at paste, the relation class is included in the list of applicable relations, otherwise it is not. Please refer to the user manual for more information.
- “expandAllowed” Defines if it is allowed to expand this relation class explicitly in the context action “Expand”.

Example:

```
<relation name="Part BOM" displayName="NLS_PartBOM"
          icon="TcEnt_PartPartRelation" createAllowed="true"
          expandAllowed="true">
```

As for the PDM object classes, a bitmap file for the icon must exist in the CATIA icon subdirectory ("IconName.bmp" and "IconName16x16.bmp").

As described in "NLS Support for Display Names" the NLS string for the "displayName" XML attribute is defined in the CATNls file specific to the PDM system and the customization.

Description of PDM Relations

As with PDM object definitions, the "description" tag defines which of the attributes of the class should be displayed beside the icon.

In this example the attributes “Class”, “major_rev”, and “generation” are defined as the description.

Example:

```
<description>
   <descAttribute name="Class" />
   <descAttribute name="major_rev" />
   <descAttribute name="generation" />
</description>
```

“Relationship” tags

The tags "leftToRightRelationship" and "rightToLeftRelationship" define the relationships for the two sides of the relation. They contain the following XML attributes:

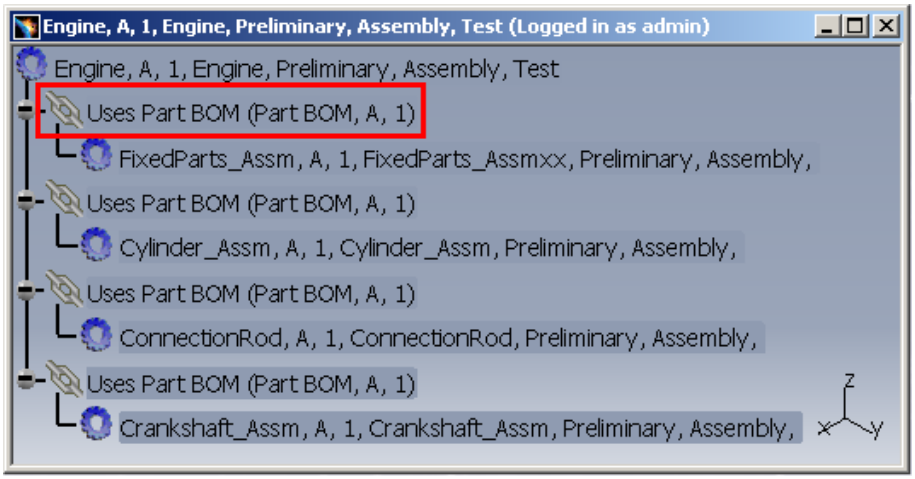
- "name" The internal PDM relationship name.
- "displayName" The relationship name that is shown to the user.
- "multipleExpand" If "multipleExpand" is "true", then the relationship appears in the "Expand Multiple Levels" context menu, otherwise it does not. For this the context action "MultipleExpand" must be defined on the PDM object. The default value is "false".

Example:

The Aras relation "Part BOM" has the relationships "Part BOM_LeftToRight" (display name is "Uses Part BOM") and "Part BOM_RightToLeft" (display name is "Is Used in Part BOM").

```
<leftToRightRelationship name="Part BOM_LeftToRight"
                         displayName="NLS_PartBOM_LeftToRight"
                         multipleExpand="true" />
<rightToLeftRelationship name="Part BOM_RightToLeft"
                         displayName="NLS_PartBOM_RightToLeft"
                         multipleExpand="true" />
```

The text that describes a relation icon is the display name of the expanded relationship ("Part BOM“ in this example) as well as the parameters defined in the "description" XML tag in parentheses “Class”, “major_rev”, and “generation” in this example).



Picture 47: Relation icon with relationship and description attribute

"Left and Right Object" Classes

The tags "leftObject" and "rightObject" define which object classes are valid for this relation.

In the following example the relation is between CATIA parts:

```
<leftObject name="/Part/Assembly" />
<rightObject name="/Part/Assembly" />
<rightObject name="/Part/Component" />
```

PDM Relation Forms

The following forms can be defined for a relation class:

"Properties", "UpdateItem", "Create"

The definition of the dialog forms for relations is similar to the definition of the dialog forms for objects.

Example:

```
<form name="UpdateItem">
   <formAttribute name="reference_designator"
                  widgetType="SingleLineEditor" mode="update"
                  visibleLength="15" />
</form>
```

Data Sources

Data sources describe a static set of values that are already known when writing the Schema file. The set of these values will never change during the lifetime of the PDM-Workbench.

Data Source "Value" tag

The value tag of static data sources contains the following XML tags:

- "name" The PDM name of the attribute.
- "displayName" The dialog display name of the attribute.
- "booleanValue" "true" or "false" to assign the correct value to the attribute names (this tag is only used for type BooleanValueList).
- "valueName" The PDM name of the value attribute (this tag is only used for type NameValueList).
- "displayValue" The dialog display name of the value attribute (this tag is only used for type NameValueList).

Static data sources can be of type:

SingleValue: the data source contains only one static element.

Example:

```
<dataSource name="Autoname" type="SingleValue">
   <value name="autoname" displayName="NLS_Autoname" />
</dataSource>
```

ValueList: the data source contains a set of static value elements.

Example:

```
<dataSource name="LifeCycleStates" type="ValueList">
```

```
   <value name="Preliminary" displayName="NLS_Preliminary"/>
   <value name="Review" displayName="NLS_Review"/>
   <value name="Approve" displayName="NLS_Approve"/>
   <value name="Released" displayName="NLS_Released"/>
</dataSource>
```

BooleanValueList: the data source contains exactly the value pair “true” and “false”.

Example:

```
<dataSource name="PlusOrMinus" type="BooleanValueList">
   <value name="+" displayName="NLS_ValueSetPlus" booleanValue="true" />
   <value name="-" displayName="NLS_ValueSetMinus" booleanValue="false" />
</dataSource>
```

NameValueList: the data source contains a list of name-value pairs.

Example:

```
<dataSource name="Test" type="NameValueList">
   <value name="test1" displayName="NLS_test1" valueName="test1_value"
          displayValue="NLS_test1_value" />
   <value name="test2" displayName="NLS_test2" valueName="test2_value"
          displayValue="NLS_test2_value" />
</dataSource>
```

Complete example of using a data source tag:

The attribute “CheckedOut” can be assigned exactly to “true” or “false”. Within the dialog this will be expressed by showing a “+” or a “-“ sign. Therefore we define a data source called “PlusOrMinus” and attach this container to the attribute description.

```
<attribute name="CheckedOut" displayName="NLS_CheckedOut"
           dataSource="PlusOrMinus" />
<dataSource name="PlusOrMinus" type="BooleanValueList">
   <value name="+" displayName="NLS_ValueSetPlus" booleanValue="true" />
   <value name="-" displayName="NLS_ValueSetMinus" booleanValue="false" />
</dataSource>
```