



# **PDM Workbench NX**

## **PDM Workbench NX Release 18.0 for Aras Innovator**

---

## **Installation & Administration Manual**

Version 1



---

## Copyright

© 2005-2024 T-Systems International GmbH.  
All rights reserved. Printed in Germany.

---

## Contact

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

<http://plm.t-systems-service.com/en/pdm-workbench-nx>

☎ +49 (0) 40 30600 5544

✉ +49 (0) 3915 80125688

mail : [cmi\\_support@t-systems.com](mailto:cmi_support@t-systems.com)

---

## Manual History

Version	Date	Version	Date
1.2.5	September 2016	12.0	May 2020
2.1.0	April 2017	13.0	November 2020
2.2.0	Oktober 2017	14.0	June 2021
2.3.1	July 2018	15.0	January 2022
9.0	February 2019	16.0	November 2022
10.0	May 2019	17.0	May 2023
11.0	November 2019	18.0	May 2024

This edition 16.0 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  
Business Unit PLM  
Fasanenweg 5  
70771 Leinfelden-Echterdingen  
Germany

mail: [cmi\\_support@t-systems.com](mailto:cmi_support@t-systems.com)

---

# Preface

---

## About this Manual

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

- the Microsoft Windows operating system
- the administration of the Siemens NX 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 NX:

Manual Title	Version
<i>PDM Workbench NX Installation &amp; Administration Manual</i>	18.0
<i>PDM Workbench NX User Manual</i>	18.0

---

## Trademarks

NX is a registered trademark of Siemens Digital Industries Software.

Aras and Aras Innovator are registered trademarks of Aras Corporation.

Names of other products mentioned in this manual are used for identification purpose only and may be trademarks of their companies.



---

# Table of Contents

---

<b>CHAPTER 1</b> .....	<b>1</b>
<b>OVERVIEW</b> .....	<b>1</b>
SYSTEM HARDWARE AND SOFTWARE REQUIREMENTS.....	1
<i>Aras Server</i> .....	1
<i>NX Client</i> .....	1
INSTALLATION STEPS .....	2
<b>CHAPTER 2</b> .....	<b>3</b>
<b>ADAPTING NX</b> .....	<b>3</b>
LOADING PWB NX SOFTWARE FROM CD-ROM .....	3
PWB NX INSTALLATION.....	3
<i>Configuring the installation</i> .....	3
SILENT INSTALLATION .....	11
<i>Parameters</i> .....	12
<i>Usage</i> .....	12
SILENT UN-INSTALLATION .....	13
<i>Parameters</i> .....	13
<i>Usage</i> .....	13
REQUIRED NX OPTIONS .....	13
LICENSE MANAGER INSTALLATION .....	14
TROUBLESHOOTING .....	14
TESTING THE INSTALLATION.....	14
SETTING OF ENVIRONMENT VARIABLES.....	15
<b>CHAPTER 3</b> .....	<b>16</b>
<b>PDM WORKBENCH NX DATA MODEL</b> .....	<b>16</b>
INSTALLATION.....	16
<i>Standard Package</i> .....	17
<i>Open in NX</i> .....	17
<b>CHAPTER 4</b> .....	<b>19</b>
<b>PDM WORKBENCH SERVER DLL</b> .....	<b>19</b>
COPYING THE DLL.....	19
MODIFYING THE SERVER CONFIGURATION FILE.....	19
<b>CHAPTER 5</b> .....	<b>21</b>
<b>SERVER CONFIGURATION</b> .....	<b>21</b>
CONFIGURATION VARIABLES .....	22
<i>PWB Logging on Server</i> .....	22
CONFIGURATION ITEMS .....	23
<i>Attribute mapping</i> .....	24
<i>"CAD is Master for Instances" Functionality</i> .....	25
DERIVED FILES CONFIGURATION .....	26
<i>Derived viewable files</i> .....	26
<b>CHAPTER 6</b> .....	<b>27</b>
<b>CONFIGURATIONS FOR SPECIFIC FUNCTIONALITIES</b> .....	<b>27</b>
STANDARD CONFIGURATION .....	27
<i>Exchange map</i> .....	27
<i>SOAP target URL</i> .....	27
<i>Session settings</i> .....	27
<i>Create Part mode</i> .....	28

Update NX Data Structure from PDM Option.....	28
Reconnect At Update .....	28
Key attribute.....	28
Class attribute.....	28
Relation attribute.....	28
Relationship attribute.....	28
Left relationship attribute.....	28
Right relationship attribute.....	28
Left relation class attribute.....	29
Right relation class attribute.....	29
Extended relation class attribute.....	29
Last modification date attribute.....	29
DATA MODEL CONFIGURATION .....	29
Document data model .....	29
QUERY CONFIGURATION.....	29
The Query dialog attributes .....	29
QueryOrderByAttribute .....	30
MaxQueryResults .....	30
AUTONAME SUPPORT USING ARAS INNOVATOR SEQUENCE ITEMS.....	30
Autoname functionality can use a server method .....	32
OPEN IN ARAS FROM NX CLIENT .....	34
Configure web-browser .....	34
RECONNECT AT UPDATE .....	36
CREATE ONLY ONE GENERATION PER CLAIM .....	39
FAMILY PART HANDLING .....	42
STANDARD PART HANDLING.....	43
<b>CHAPTER 7.....</b>	<b>44</b>
<b>CLIENT SCHEMA FILE CONFIGURATION .....</b>	<b>44</b>
STRUCTURE OF THE SCHEMA FILE.....	44
Attributes of the tag "PWBSchema":.....	44
Display Names.....	45
Configuration settings.....	45
"object": 1 - n.....	45
"attribute": 0 - n.....	46
"pwbAttribute": 0 - n.....	46
"dataSource": 0 - n.....	46
PDM ATTRIBUTES AND FORM ATTRIBUTES .....	46
Description of the Widget Types.....	47
Login Form.....	49
PDM OBJECTS.....	49
Description of PDM Objects .....	50
Actions on PDM Objects.....	50
PDM Object Forms .....	50
DATA SOURCES.....	51
Data Source "Value" tag.....	51
Complete example of using a data source tag:.....	52
CUSTOMIZING PDM WORKBENCH NX MENU .....	52
VALIDATION RULES FOR UPDATE IN SCHEMA FILE .....	53
PREVENT UPDATE OF NX FILES WHICH WERE CREATED WITH AN OLDER RELEASE .....	54
<b>CHAPTER 8.....</b>	<b>55</b>
<b>TROUBLESHOOTING .....</b>	<b>55</b>
INVALID CAD INSTANCES WITH ARAS 3D CAD TO PDF CONVERSION .....	55
LOCKING CAD DOCUMENTS IN CATIA FAILS WHEN THE CONVERSION SERVER IS INSTALLED.	
.....	56
CONVERSION TASKS FAIL FOR UPDATED OR NEW CAD FILES FROM PDM WORKBENCH .....	56

---

# Table of Figures

---

PICTURE 1: DIRECTORY STRUCTURE OF THE PDM WORKBENCH INSTALLATION FILES .....	4
PICTURE 2: WELCOME TO THE INSTALLATION.....	5
PICTURE 3: LICENSE AGREEMENT .....	5
PICTURE 4: CHOOSE USERS .....	6
PICTURE 5: CHOOSE LOCATION OF PDM PACKAGE.....	6
PICTURE 6: CHOOSE LOCATION OF PDM PACKAGE (WITH PROPOSAL) .....	7
PICTURE 7: CHOOSE INSTALL LOCATION .....	7
PICTURE 8: CHOOSE NX INSTALLATION.....	8
PICTURE 9: CHOOSE EXCHANGE DIRECTORY .....	8
PICTURE 10: CHOOSE LOCATION OF SOAP TARGET URL .....	9
PICTURE 11: CHOOSE DATABASE NAME .....	9
PICTURE 12: CHOOSE DATA MODEL .....	10
PICTURE 13: SUBSUMPTION.....	10
PICTURE 14: INSTALLATION PROGRESS .....	11
PICTURE 15: INSTALLATION FINISHED .....	11
PICTURE 16: PDM ARAS INNOVATOR IMPORT UTILITY .....	16
PICTURE 17: ARAS INNOVATOR SERVER CONFIGURATION VARIABLES .....	22
PICTURE 18: LOGGING CONFIGURATION VARIABLES .....	23
PICTURE 19: PWB CONFIGURATION ITEM IN ARAS INNOVATOR.....	23
PICTURE 20: DEFINE ATTRIBUTE MAPPING .....	24
PICTURE 21: DEFINE GROUP ATTRIBUTE MAPPING – CAD DOCUMENTS.....	25
PICTURE 22: DEFINE GROUP ATTRIBUTE MAPPING – GROUP .....	25
PICTURE 23: DEFINE GROUP ATTRIBUTE MAPPING – PWB CONFIG.....	25
PICTURE 24: DERIVED FILES CONFIGURATION .....	26
PICTURE 25: SAMPLE USEBOMPARTSTRUCTURE CONFIGURATION .....	29
PICTURE 26: SAMPLE QUERYORDERBYATTRIBUTE CONFIGURATION .....	30
PICTURE 27: SAMPLE MAXQUERYRESULTS CONFIGURATION .....	30
PICTURE 28: SAMPLE SEQUENCE ITEM.....	30
PICTURE 29: SEQUENCE ITEMS USED IN EXAMPLE .....	31
PICTURE 30: SAMPLE USESERVERMETHODSFORAUTONAME CONFIGURATION.....	31
PICTURE 31: FILE “DEEPLINKING.TS OR .JS” .....	35
PICTURE 32: EMPTY TAB .....	35
PICTURE 33: ARAS INNOVATOR WEB CLIENT WITH CLIENT URL .....	36
PICTURE 34: WINDOW TITLE “ARAS INNOVATOR” .....	36
PICTURE 35: PWB CONFIGURATION SETTING “ARASWINDOWTITLESUBSTRING” .....	36
PICTURE 36: PWB CONFIGURATION SETTING “RECONNECTATUPDATEMETHOD” .....	37
PICTURE 37: ITEM TYPE “CAD” – ADD PROPERTY “PWB_ORIG_CAD_PARTNUMBER” .....	37
PICTURE 38: PWB CONFIGURATION – “ORIGCADPARTNUMBERATTR” .....	37
PICTURE 39: PWB CONFIGURATION SETTING “CLAIMEDISNEWGENATTR” .....	39
PICTURE 152: SETTING “CUSTOMMETHOD_GETCUSTOMITEMINFO” .....	40
PICTURE 152: CUSTOM METHOD “PwBCus_GETCUSTOMITEMINFONX” .....	40
PICTURE 42: CAD ITEM EXTENSION .....	42
PICTURE 43: CAD STRUCTURE ITEM EXTENSION .....	42
PICTURE 43: PART ITEM EXTENSION.....	43
PICTURE 39: SINGLE LINE EDITOR WIDGET, UPDATE MODE .....	48
PICTURE 40: SINGLE LINE EDITOR WIDGET, OUTPUT MODE .....	48
PICTURE 41: MULTI LINE EDITOR WIDGET, UPDATE MODE .....	48
PICTURE 42: COMBO BOX WIDGET, SELECT MODE.....	48
PICTURE 43: SINGLE CHECK BOX WIDGET, SELECT MODE.....	48
PICTURE 44: PASSWORD BOX WIDGET .....	48
PICTURE 45: URL WIDGET .....	48
PICTURE 52: “ARAS 3D CAD TO PDF CONVERSION” RULE.....	57
PICTURE 53: “UNCHECK LOCK DEPENDENCIES ARAS 3D CAD TO PDF CONVERSION” RULE ..	57

---





---

# CHAPTER 1

## Overview

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

---

### System Hardware and Software Requirements

#### *Aras Server*

##### *Hardware*

For the Aras Server hardware sizing please refer to Aras recommended hardware sizing document.

Please refer to <https://www.aras.com/support/documentation/>

and look for the Platform specifications document, e.g.

Aras Innovator 12.0 - Platform Specifications.pdf

##### *Software*

#### Aras Innovator 12

Server Installation of Aras Innovator 12.0 on the following operating systems:

- Windows Server 2012, Windows Server 2014, Windows Server 2016
- Detailed information see *Aras Documentation*: “Aras Innovator 12 Platform Specifications”

#### Aras Innovator 22 to 24

Server Installation of Aras Innovator on the following operating systems:

- Windows Server 2012, Windows Server 2016, Windows Server 2019
- Detailed information see *Aras Documentation*: “Aras Innovator xx Platform Specifications”

#### *NX Client*

##### *Hardware*

The T-Systems NX Aras Connector PDM Workbench does not introduce any additional requirements to the CAD Workstations. The CAD Workstation spec should be close to the proposed certified hardware spec as defined by Siemens PLM for NX.

Please refer to:

<https://www.plm.automation.siemens.com/global/de/support/certifications.html>  
for Operation System and CPU recommendations.

Select the respective NX Release and Windows 10 64 bit as Operating System.

To process large NX Product Structures the CAD Workstation should have at least 32 GB RAM. For fast processing a SSD of sufficient size (500 GB) is recommended.

---

## Software

### Aras Innovator

On the NX client computers .NET 4.7.2 must be installed.

Starting with Aras Innovator 17 the following .NET Core 3.1 module must be installed:

- .NET Desktop Runtime

Starting with Aras Innovator 22 the following .NET 6.0 module must be installed:

- .NET Desktop Runtime

### NX

NX Client Version 1926, Version 1953, and Version 2206 on the following operating systems:

- Windows 10 (64 Bit)

---

## Installation steps

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

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

- Client installation: NX 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 NX data model and server methods (chapter 3)
- Server installation: PDM Workbench NX server DLL (chapter 4)

---

# CHAPTER 2

## Adapting NX

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

The **PWBNX\_Vxx\_xx** module includes all of the supported platform data in a compressed file. Thus, you should choose an installation location for all NX clients.

In the following example sections it is supposed that the software will be installed within the directory **C:\Program Files\T-Systems\PWBNX\_Vxx\_xx\_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 **PWBCAD\_xx\_Aras\_xx.zip**.

The installation does three things:

1. It copies the software on the hard disk.
2. It connects the Aras Innovator-NX-Integration to the installed NX installation.
3. It configures some settings of the Aras Innovator-NX-Integration.

To simplify the installation you can use the provided **setup.exe**.

---

### Loading PWBNX Software from CD-ROM

*Windows 10*

Use the Windows Explorer to locate the **D:\pwbpx\PWBNX\_Vxx\_xx.zip** file on the CD. Extract the content of the archive file to a temporary installation location.

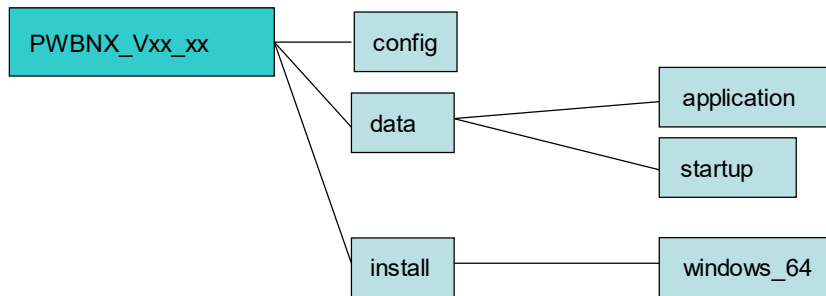
---

### PWBNX 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 **PWBNX\_Vxx\_xx** 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 PWB NX module.

The **install** directory contains the sub directory **windows\_64** with all necessary data for the installer program.

#### *Windows 10 (64 Bit)*

On **Windows 10 (64 Bit)** use the Windows Explorer to run the **setup.exe** in the directory **PWB NX\_Vxx\_xx\install\windows\_64** of the installation package if you have installed the 64 Bit version of NX.

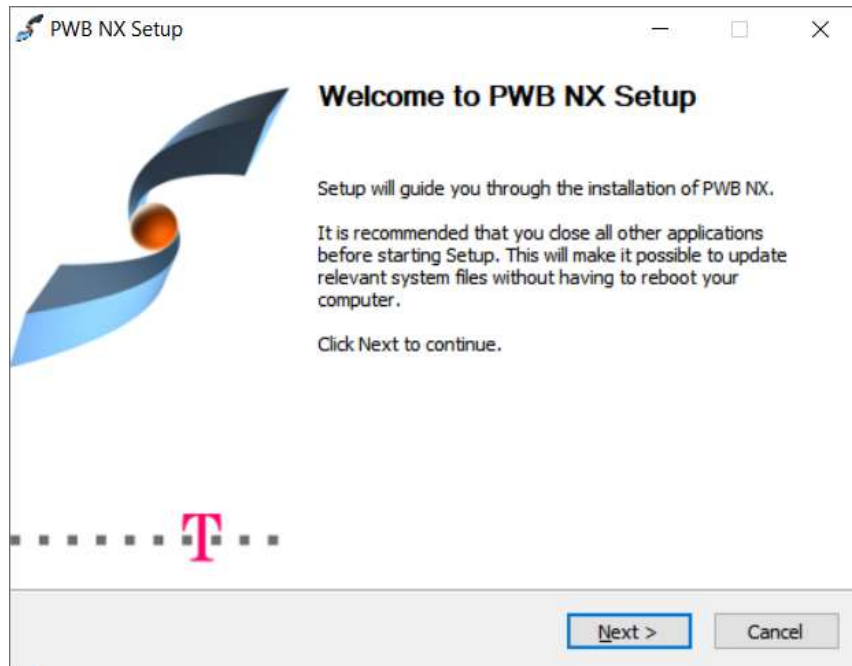
On **Windows Vista/Windows 7** the User Account Control (UAC) will be triggered and you will have to agree that the setup program may make changes to the computer. The installer is signed with a “**T-Systems International GmbH**” certificate to ensure its integrity and source.

The setup will **NOT** modify the native installation of NX.

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

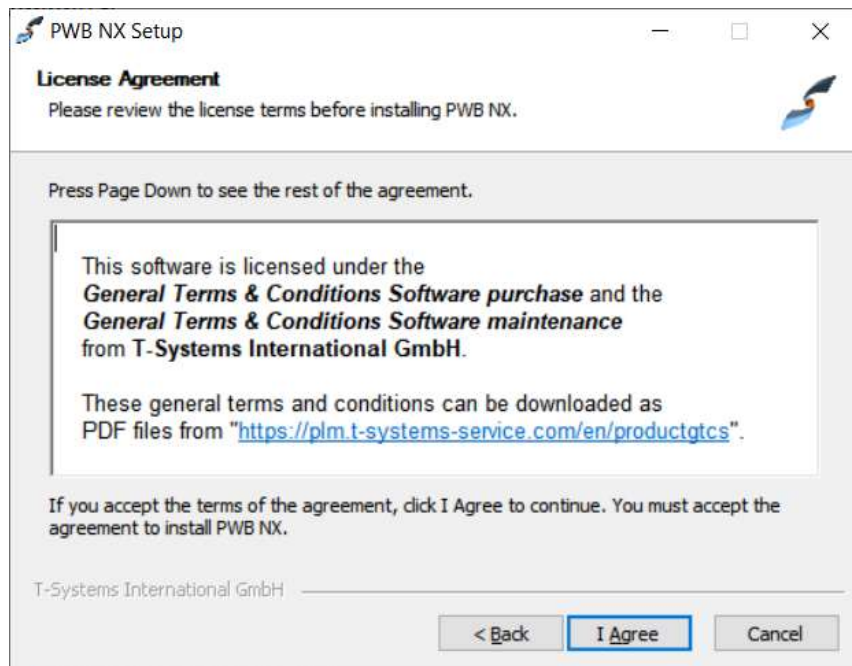
In the following the setup is shown step-by-step.

Installation process:



Picture 2: Welcome to the Installation

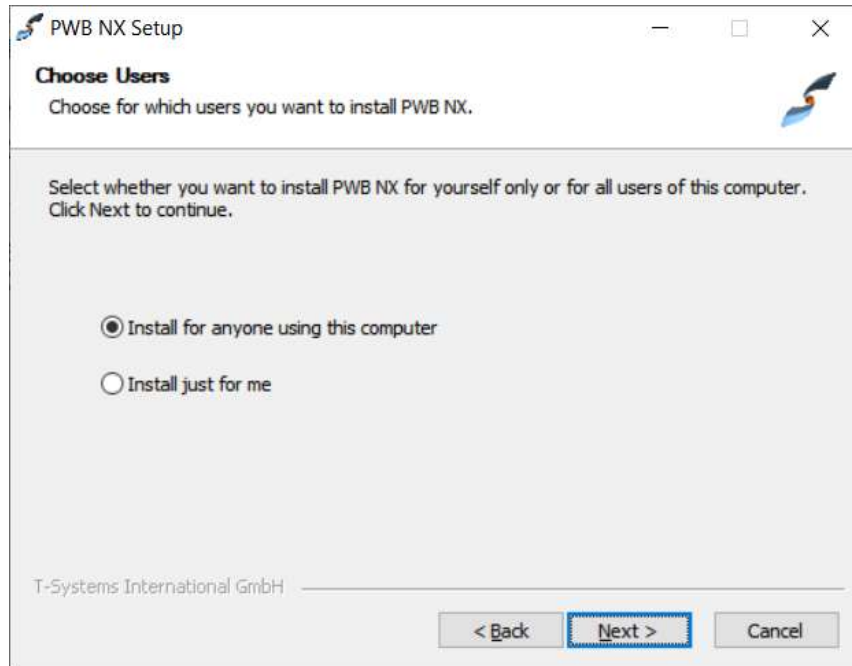
The installer software asks to approve the license terms (see *Picture 3: License Agreement*).



Picture 3: License Agreement

The installer software asks for the following input: **User scope**.

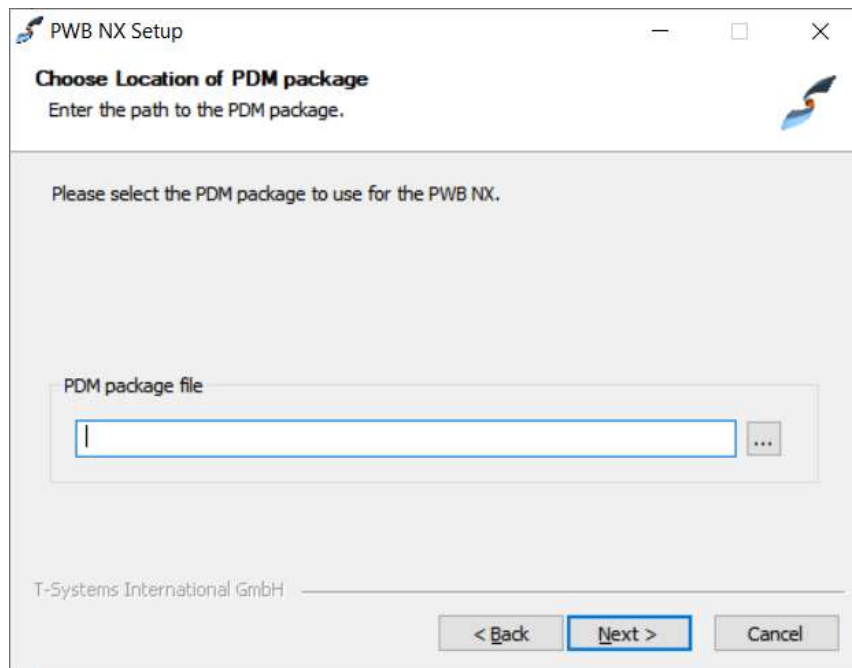
Next the installer will ask you to define the scope of the installation (see *Picture 4: Choose Users*). You can choose between an installation for anyone using the computer or just for the current user.



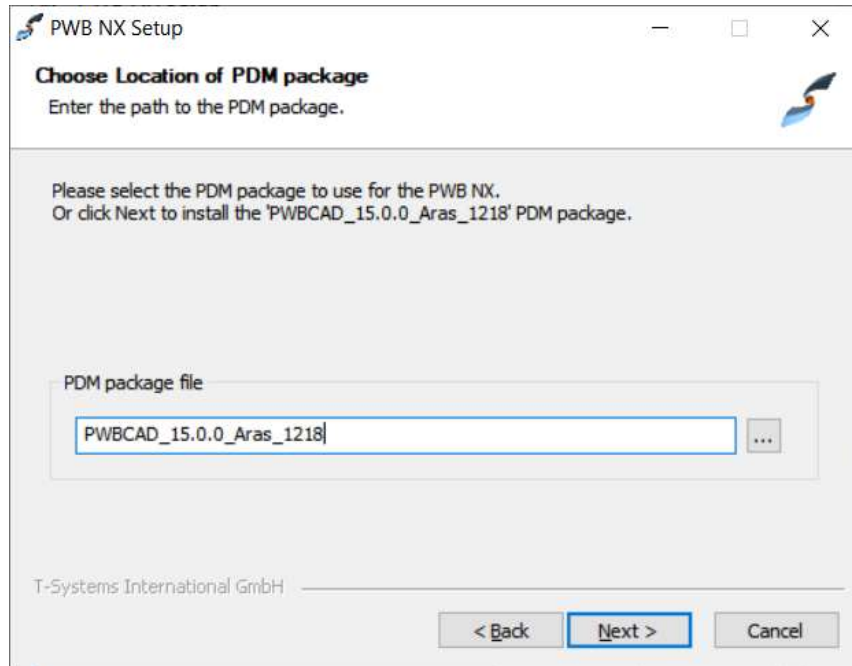
**Picture 4: Choose Users**

The installer 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 Location of PDM package*). If a PDM package has previously been unpacked within the installer it will be offered to install this package directly (see *Picture 6: Choose Location of PDM package (with proposal)*).



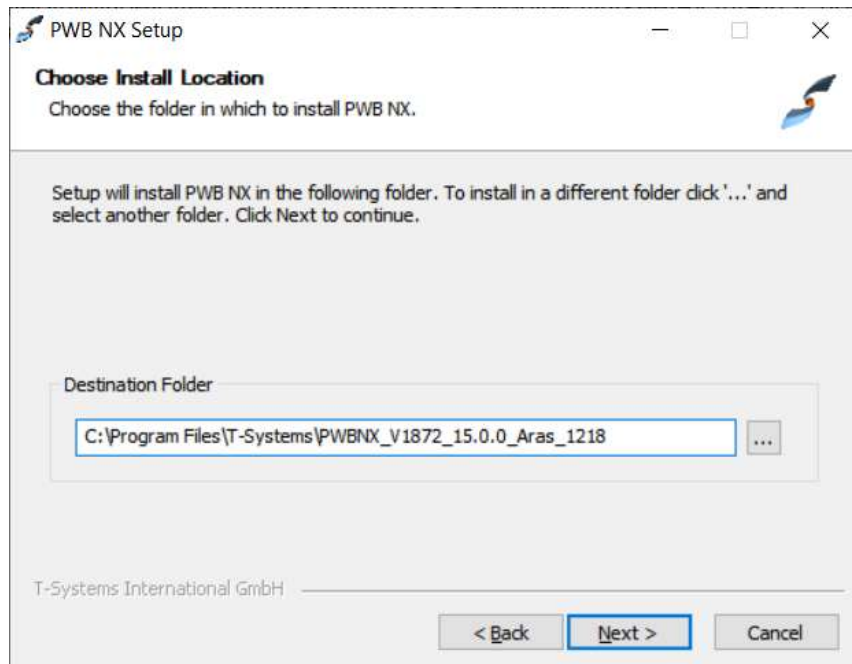
**Picture 5: Choose Location of PDM package**



**Picture 6: Choose Location of PDM package (with proposal)**

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

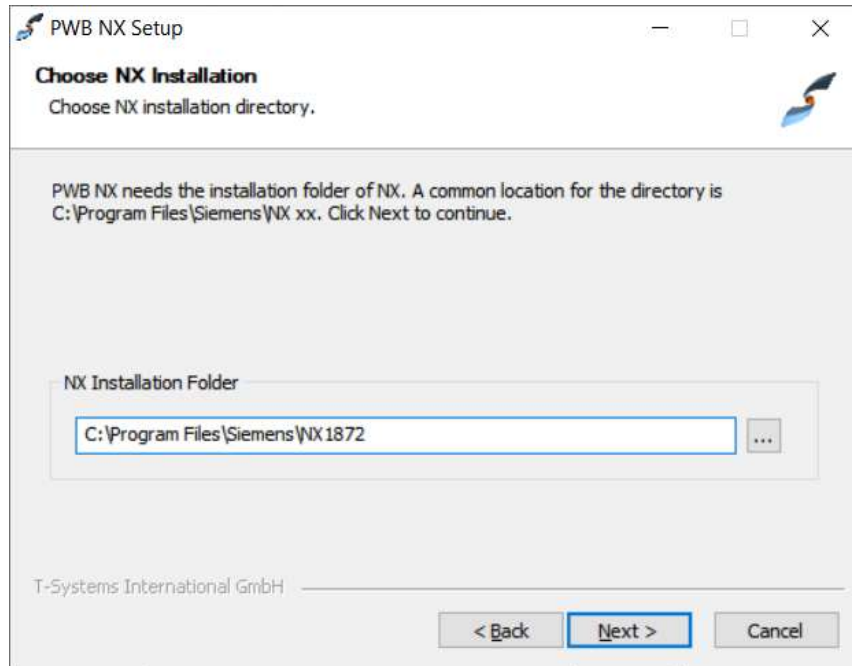
Next the installer will ask you for the target directory for the installation. You can use the given standard location or choose any other location (see *Picture 7: Choose Install Location*). The chosen folder must be empty or not exist.



**Picture 7: Choose Install Location**

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

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



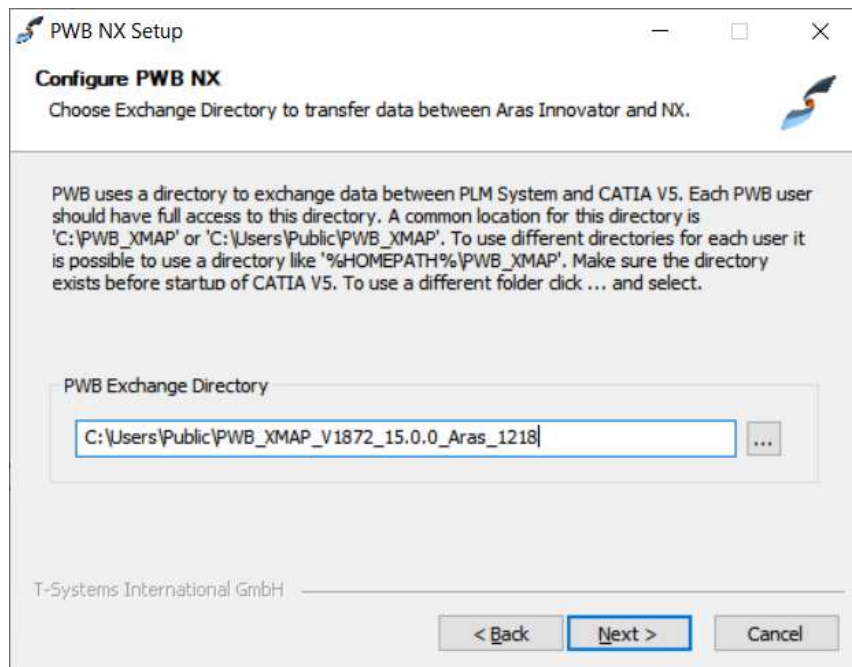
**Picture 8: Choose NX Installation**

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

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

You can either use the standard location or choose any other location (see *Picture 9: Choose Exchange Directory*).

If it is planned to run more than one NX session at a time each session must use its own PWB Exchange Directory!



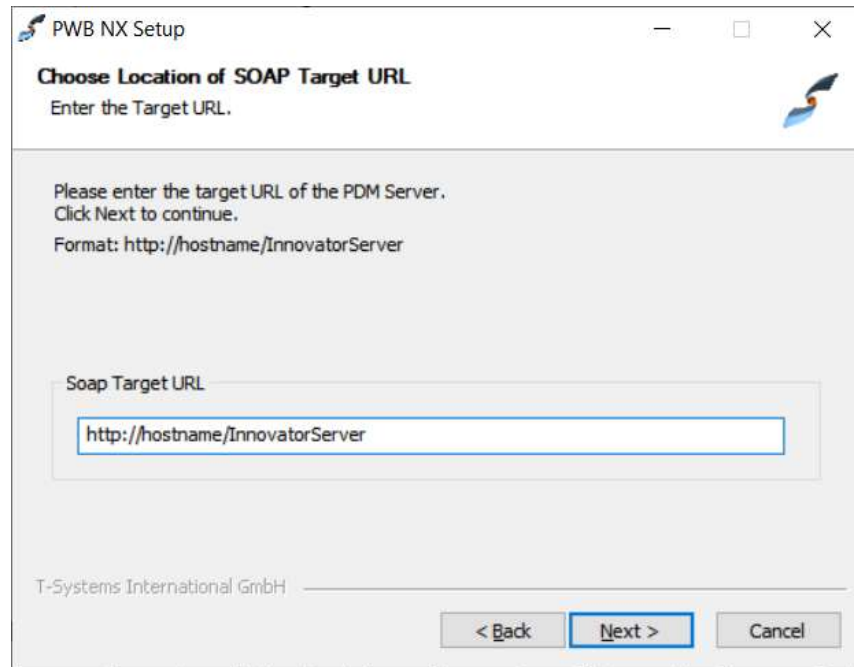
**Picture 9: Choose Exchange Directory**



The installer software asks for the following input: **SOAP Target URL**.

Finally you have to define the so called “Soap Target URL” for the PDM Server (see Picture 10: Choose Location of SOAP Target URL).

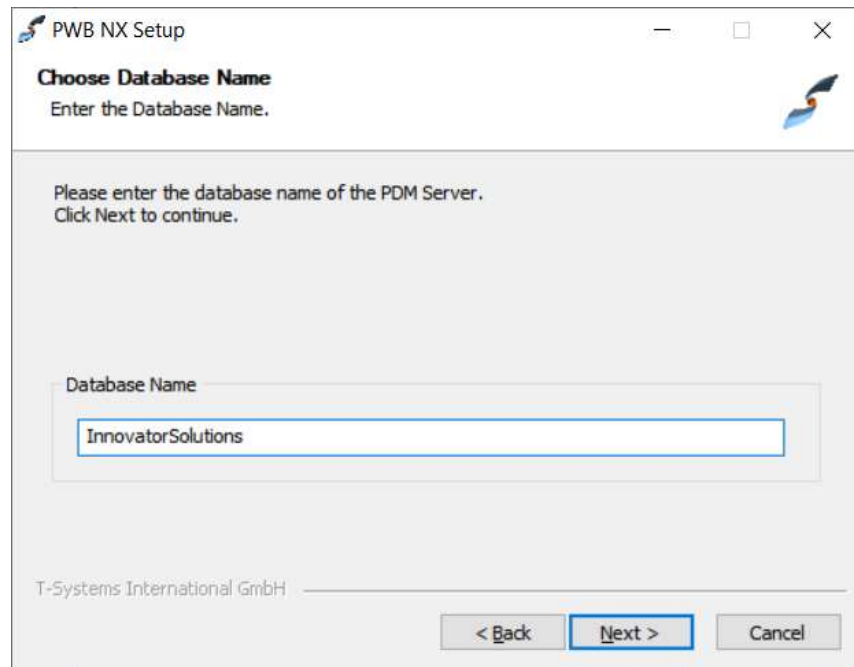
This URL defines the host on which the PDM Server is reachable.



**Picture 10: Choose Location of SOAP Target URL**

The installer software asks for the following input: **Database Name**.

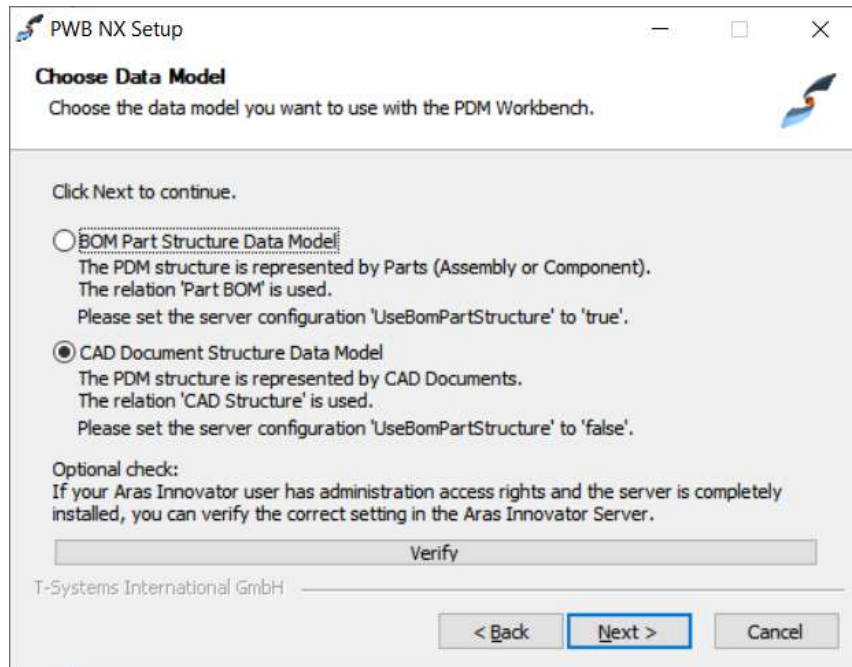
Now you have to add the Database Name (see Picture 11: Choose Database name).



**Picture 11: Choose Database name**

The installer software asks for the following input: **Data Model**.

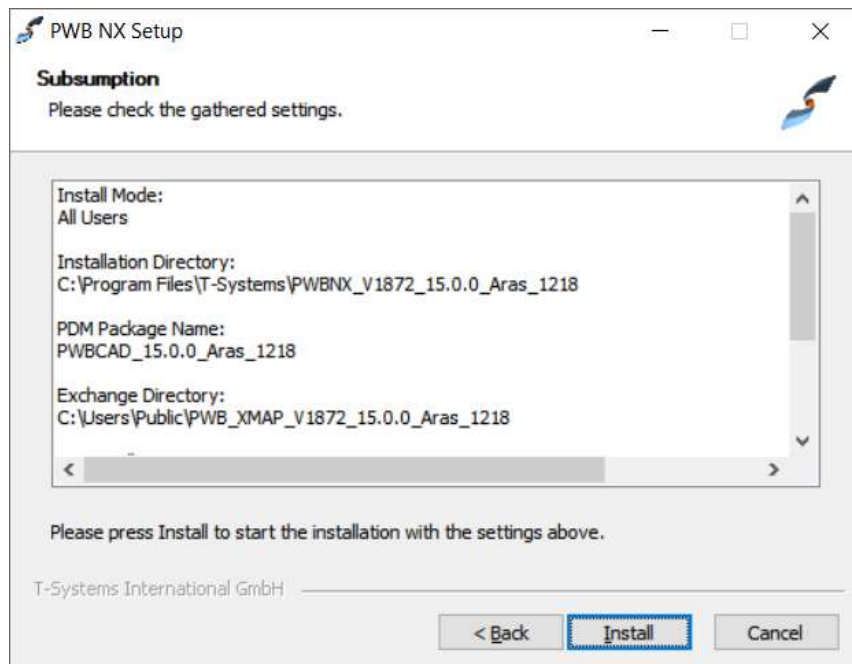
Finally you have to choose the data model to be used by the PDM Workbench (see *Picture 12: Choose Data Model*).



**Picture 12: Choose Data Model**

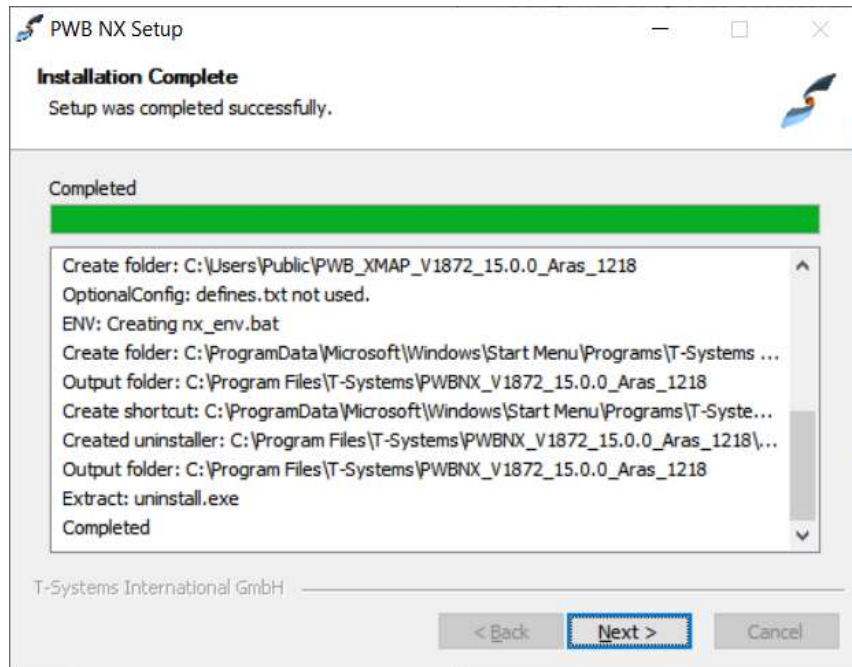
With the optional check you can directly log in into Aras Innovator with an administrative user and verify the setting of the data model (UseBomPartStructure) in the PWB configuration object.

After that you see the subsumption of your inputs before confirming them (see *Picture 13: Subsumption*).

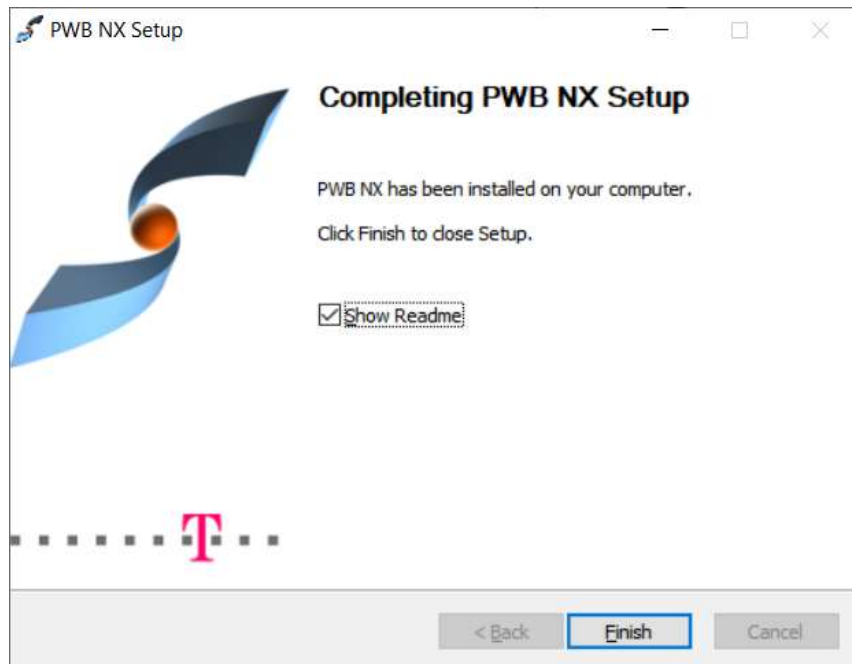


**Picture 13: Subsumption**

The installer will proceed in its process. The taken actions will be journalized (see *Picture 14: Installation progress*).



Picture 14: Installation progress



Picture 15: Installation finished

## Silent Installation

It is possible to use a silent installation for the client installation.

---

## Parameters

The following parameters are available for the silent installation:

Parameter name	Description	Sample value
/S	Activates the silent mode.	
/User= value	Installation only for yourself ("User") or for all users of the computer ("Admin"). Default is the highest possible value.	Admin
/PdmPackageNamePath= (File full path)	The full path of the zip file which includes the PDM package.	D:\aras_client\PWBCAD_15.0.0_Aras_1218.zip
/NxInstDir= (Directory path)	The directory of the NX installation.	C:\Program Files\Siemens\NX 1872
/ExchangeMap= (Directory path)	The directory of the Exchange Directory.	C:\Users\Public\PWB_XMAP
/SoapTargetURL= (URL)	The SOAP target URL of the Aras server.	http://localhost/InnovatorServer
/DatabaseName= (Database Name)	The Database Name of the Aras server.	InnovatorSolutions
/D=(Directory path)	The target directory of the installation.	C:\Program Files\T-Systems\PWBNX_V1872_15.0.0_Aras_1218

The parameters "/S" and "/SoapTargetURL" are required.

The parameter "/User" is optional. The highest possible value will be used as default value.

The parameter "/PdmPackageNamePath" is optional if the installation had run before. Then the last package will be used. If it is the first installation you have to provide the file path of the PDM package.

The value for the NX installation is optional; the value can be fetched from the Windows registry.

The parameter "/ExchangeMap" is optional. The directory "C:\Users\Public\PWB\_XMAP" will be used as default value.

The parameter "/DatabaseName" is optional. The value "InnovatorSolutions" will be used as default value.

The parameter "/D" is optional. A part of the value will be taken from the current directory. It must be the last parameter used in the command line and must not contain any quotes, even if the path contains spaces. Only absolute paths are supported.

If one value is not given and it is not possible to fetch a value from the system the installation process will be stopped and the error message can be found in the file *install.log*.

## Usage

For the silent installation please open a command line window as administrator.

Inside the temporary installation location, locate the folder

---

“PWBNX\_Vxx\_xx\install\windows\_64” for an installation on a client with Windows 64 Bit.

Start the silent installation with a command line like this example:

```
Setup.exe /S /User= Admin /PdmPackageNamePath=D:\aras_client\  
PWBCAD_15.0.0_Aras_1218.zip /NxInstDir="C:\Program  
Files\Siemens\NX 1872" /ExchangeMap= "C:\Users\Public\PWB_XMAP"  
/SoapTargetURL= "http://localhost/InnovatorServer"  
/DatabaseName="InnovatorSolutions" /D=C:\Program Files\T-  
Systems\PWBNX_V1872_15.0.0_Aras_1218
```

The log file of the installation will be stored in the current directory. There you can find the information about the installation process.

When the installation ended successful you will find the success message in this file.

---

## Silent Un-Installation

It is possible to use a silent un-installation for the client un-installation.

### Parameters

The following parameter is available for the silent un-installation:

Parameter name	Description	Sample value
/S	Activates the silent mode.	

The parameter “/S” is required.

### Usage

For the silent installation please open a command line window as administrator.

Start the silent installation with a command line like this example:

```
"C:\Program Files\T-  
Systems\PWBNX_V1872_15.0.0_Aras_1218\uninstall.exe" /S
```

Please use the full path of the file `uninstall.exe` of the installation you want to uninstall.

---

## Required NX Options

The following NX options are required:

- File → Preferences → Assembly Load Options... → use partial loading has to be switched off (NX 11), or

File → Preferences → Assembly Load Options... → Scope / Option: Fully Load should be set.

- File → Utilities → Customer Defaults... → Gateway → Assembly Navigator → All → Pack Components has to be switched off.

---

## License Manager Installation

The licman20 license manager has to be installed on the NX client host. For the installation of the license manager please refer to the *Licman 2.0 Installation Manual*.  
(For software and manual download see: <https://plm.t-systems-service.com/en/licman>)

---

## Troubleshooting

Problem	Solution
"PDM Workbench" menu and/or toolbar not visible within NX.	Check the content of the NX log-file (Help-menu → "Log File"). There should be some lines with "TSI.PDM...". If you don't find such lines the connection between the Aras Innovator-NX-Integration and NX is not established.
Starting NX or a command from "PDM Workbench" results in a licensing error.	Check the existence of a license using the "Licman Test" available in Windows Start menu → T-Systems → Licman.
The login dialog shows not the correct Aras Innovator server address.  The login dialog shows not the correct database.	Check the PDM Workbench NX Schema file, which is placed in the config directory of the installation.  The right server address must be defined within the "soapTargetUri" XML element.  The database must be defined within the "dataSource" XML element (<dataSource name="LoginDatabases" type="ValueList">)
The login fails.	Check the server address shown in the dialog.  Check your credentials.  In case of a server connection error:  Check whether it is possible to connect to the Aras Innovator server with the Innovator thin client.

---

## Testing the installation

For installation testing follow these steps:

1. Use: Start→Programs→T-Systems PWB NX 12.0 10.0.0 Innovator 1112→pwb\_nx\_start to launch NX.
2. Check that the menu "PDM Workbench NX" is visible within NX.
3. Press the "About" button in the "PDM Workbench NX" menu. The about dialog should be shown.
4. Press the "Login" button in the "PDM Workbench NX" menu. The login dialog should open and show the address of the Aras Innovator server entered during installation. Also the database from the installation must be selectable.

---

Once the steps 1. – 4. are successful the installation is ok.

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

---

## Setting of Environment Variables

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

Environment variable	Comment
PWB_XMAP	The location of the exchange map directory. The exchange map directory must be unique for every started NX session on the same client.
PWB_SCHEMA_FILE	Path including file name of the XML configuration file.
PWB_SOAP_TARGET_URL	The URL of the web service. <a href="http://localhost/InnovatorServer">http://localhost/InnovatorServer</a>
PWB_DATABASE_NAME	The name of the database, e.g. InnovatorSolutions
PWB_DEBUG	Set to "ON" to receive PWB debug output in the console.

---

# CHAPTER 3

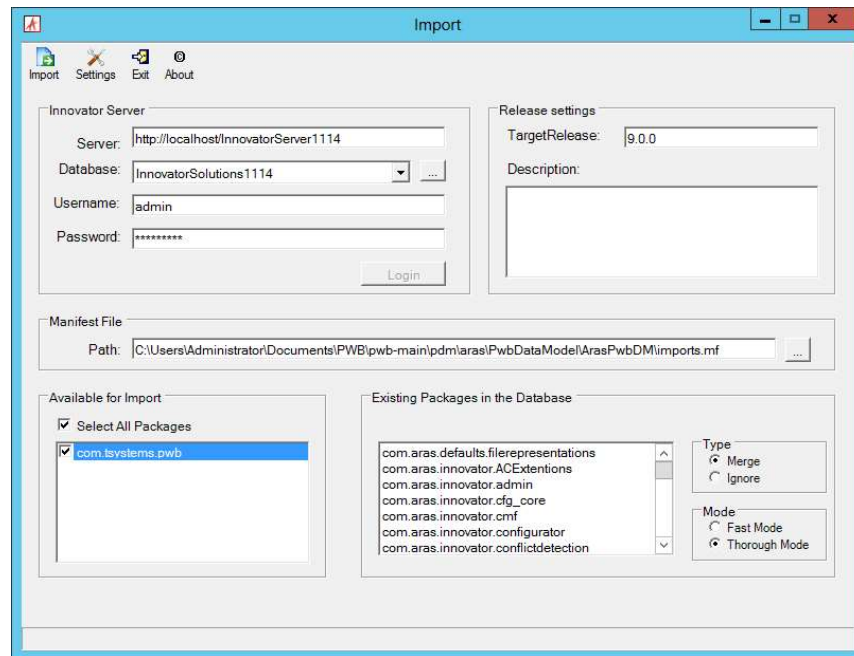
## PDM Workbench NX Data Model

---

### Installation

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

For this the “PwbDataModel\_[XX.X].zip” file needs to be unpacked first. Then six packages need to be imported to Aras Innovator with the Aras Innovator import utility<sup>1</sup>:



Picture 16: PDM Aras Innovator import utility

---

<sup>1</sup> The import utility has to be downloaded from the Aras homepage and to be installed. Link: [www.aras.com](http://www.aras.com) → Downloads → Download and Support → Additional Utilities → Package Import/Export Utility



---

## ***Standard Package***

Please select the manifest files

- PwbDataModel\_[XX.X]\ArasPwbDM\imports.mf
- PwbDataModel\_[XX.X]\ArasPwbDM\_PLM\imports.mf
- PwbDataModel\_[XX.X]\ArasPwbDM\_Core\imports.mf
- PwbDataModel\_[XX.X]\ArasPwbDM\_Cui\_default\imports.mf
- PwbDataModel\_[XX.X]\ArasPwbDM\_NX\_Additionals\imports.mf

in the import utility in this order and perform the import (Type is “Merge”, Mode is “Thorough Mode”) (see *Picture 16: PDM Aras Innovator import utility*).

## ***Open in NX***

For the “Open in NX” functionality in Aras Innovator client select the manifest file

- PwbDataModel\_[XX.X]\Specific\OpenInCAD\OpenInNX\imports.mf

in the import utility in this order and perform the import (Type is “Merge”, Mode is “Thorough Mode”) (see *Picture 16: PDM Aras Innovator import utility*).



---

# CHAPTER 4

## PDM Workbench Server DLL

---

### Copying the DLL

Please copy the files

```
PwbServerAddin.dll
PwbServerAddin.pdb (optional)
```

from the distribution package 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.

---

### Modifying the server Configuration file

Also, please modify the file

```
C:\Program Files\Aras\Innovator\Innovator\Server\method-config.xml
```

by adding the highlighted lines:

```
...
<MethodConfig>
  <ReferencedAssemblies>
    <name>System.dll</name>
    <name>System.XML.dll</name>
    <name>System.Web.dll</name>
    <name>System.Data.dll</name>
    <name>System.Core.dll</name>
    <name>System.Configuration.dll</name>
    <name>System.Web.Extensions.dll</name>
    <name>$(binpath)/IOM.dll</name>
    <name>$(binpath)/InnovatorCore.dll</name>
    <name>$(binpath)/SPConnector.dll</name>
    <name>$(binpath)/ConversionManager.dll</name>
    <name>$(binpath)/FileExchangeService.dll</name>
    <name>$(binpath)/Conversion.Base.dll</name>
    <name>$(binpath)/Aras.TDF.Base.dll</name>
    <name>$(binpath)/Aras.ES.dll</name>
    <name>$(binpath)/PwbServerAddin.dll</name>
  </ReferencedAssemblies>
  ...
  ...
  <Template name="CSharp" line_number_offset="35">
  <![CDATA[using Aras.IOM;
using System;
using System.Collections;
```

---

```
using System.Collections.Generic;
using System.Data;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Web;
using System.Web.SessionState;
using System.Xml;
using PwbServerAddin;

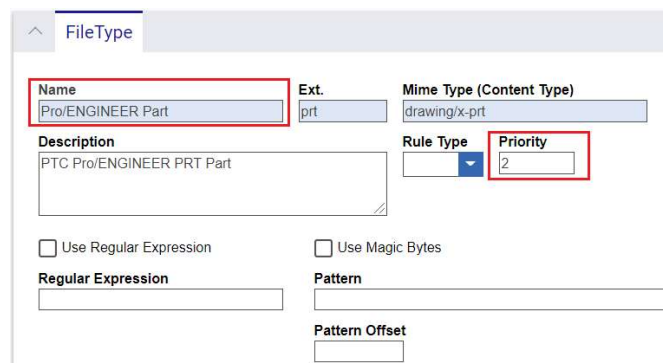
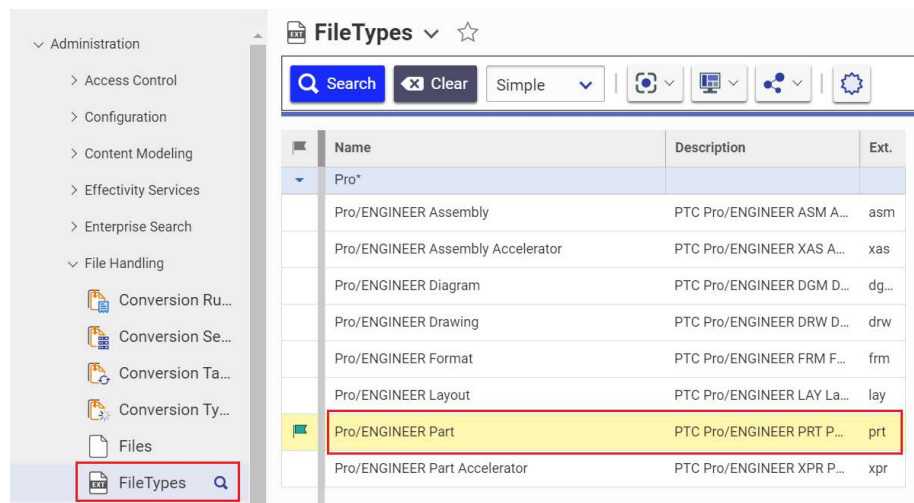
namespace $(pkgrname)
{
...
}
```

# CHAPTER 5

## Server Configuration

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

In order to make sure that files with the extension “prt” are defined as NX files, the priority of the file type “Pro/ENGINEER Part” has to be set to 2, and the priority of the “NX Model” file type should be set to 1:



## Configuration Variables

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

Name	Value	Default Value
pwb*		
PwbConfigurationItemName	Configuration	Configuration
PwbOpenInNXPort	8086	8086
PwbServerLogDir	C:\Users\Public\Documents	

Picture 17: 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.

This variable is only optionally as the pre-configured value “Configuration NX” is set as default in the client schema file:

```
SET PWB_SCHEMA_FILE=%PWBDIR%\config\PWBSchema_Aras_NX.xml
```

```
<PWBSchemata>
  <PWBSchema system="Aras" customization="Aras"
    serverConfig="Configuration NX"
    displayName="NLS_System"
    visibleLength="15"
    allowedLength="64">
```

- **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.

- **PwbOpenInNXPort**

The port to be used for the Open in NX method.

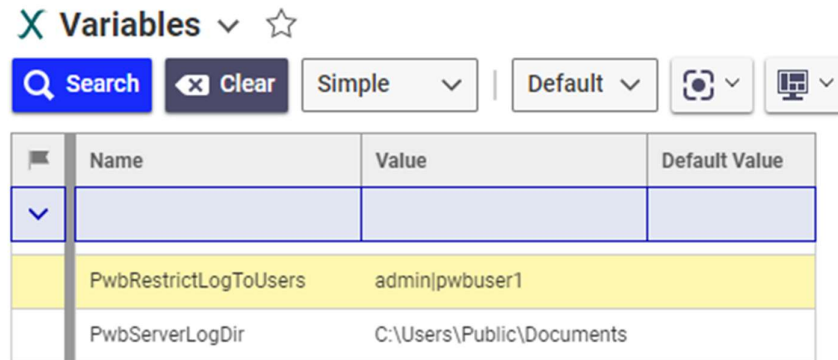
## PWB Logging on Server

There are two changes in the PWB logging functionality on the Aras server:

- The name of the logfile now also holds the database name:  
<configured log dir>PwbServerLog\_<database-name>\_<login-user>.txt
- The log can be restricted to specific users (this could be used to create a log file for a specific user in the production environment).

### Configuration

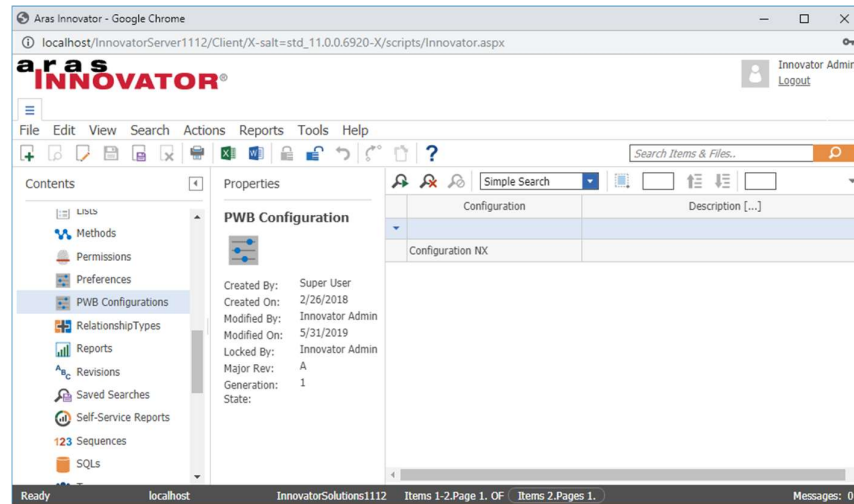
To enable general logging, set the Aras Innovator variable “PwbServerLogDir” to the log directory. To restrict the logging to a specific set of users, set the Aras Innovator variable “PwbRestrictLogToUsers” to a list of names (separated by ‘|’).



Picture 18: Logging configuration variables

### Configuration Items

In order to define the environment variables and to configure the mapping of attributes between Aras Innovator and NX a special configuration item (see *Picture 19: PWB Configuration item in Aras Innovator*) has to be used:



Picture 19: PWB Configuration item in Aras Innovator

Open this item. The following configuration variables might be set:

- ShowCreateDialogsDuringUpdate: This configuration value manages the availability of the create dialog for new objects during update.

- UseServerMethodesForAutoname: The configuration determines, if sequences items or server methods are used for autonaming. (See *Autoname Support using Aras Innovator Sequence Items*)
- UseBomPartStructure: This configuration defines the data model to be used to store the document structure.
- GroupAttributeNameForUserAttributes : The configuration determines the attribute field on the CAD document which represents the group for the attribute mapping
- ReconnectAtUpdateMethod: The configuration defines the server method for the reconnect function.
- OrigCadPartnumberAttr: This configuration determines the CAD document attribute, to retain the original file name for the reconnect function.
- ClaimsNewGenAttr: This configuration needs to be set to *pwb\_claimed\_is\_new\_gen*, if only one generation should be created per claim.
- CustomMethod\_GetCustomItemInfo: With help of this configuration, a server method might be defined implementing an own rule for the generation creation.
- UseCadlsMaster: The configuration determines whether the Workbench handles the CAD as master for instances in the BOMPartStructure Mode. If the BOMPartStructure mode is not used, CADMaster Mode is always used.

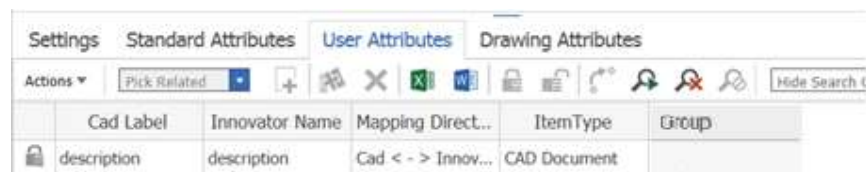
### Attribute mapping

User defined attributes which are defined at the Part (applied to Part) at the native file can be mapped to a specified attribute in Aras. Other user defined attribute (applied to a component, reference set ...) can not be mapped.

To define the attribute mapping, open the configuration item and switch to the “User Attributes” view.

Enter a new entry for each attribute you want to map. Fill in the following values:

- Cad Label: the name of the attribute in NX
- Innovator Name: the name of the attribute in Aras Innovator
- Mapping Direction: The direction in which you want to map. From CAD System to PLM, the other way around or bidirectional.
- ItemType: The item type (CAD Document or Part) to which the attribute should be mapped.



**Picture 20: Define Attribute Mapping**

It is possible to define attribute mapping groups. This field is optional.

If an attribute mapping group is set, the attribute is mapped only if the corresponding attribute at the cad document has the defined group value. Which cad attribute that is, is defined in the PWB Configuraiton with the GroupAttributeNameForUserAttributes

(see above). This can be used if there are specific attributes in the cad file, but are mapped to the same attribute in aras innovator.



e.g. there are the following Cad Documents:

Document Number	Revi...	Name	Type	State	Native File [...]	country
test54	A	test54	Mechanical/Part	Preliminary	test54.prt	DE
test55	A	test55	Mechanical/Part	Preliminary	test55.prt	BR

**Picture 21: Define Group Attribute Mapping – Cad Documents**

And the following mapping rules are defined:

Cad Label	Innovator Na...	Mapping Dire...	ItemType	Group
sizeDE	size	Cad < - > Inno...	CAD Document	DE
sizeBR	size	Cad < - > Inno...	CAD Document	BR

**Picture 22: Define Group Attribute Mapping – Group**

And the following attribute is configured in the PWB Config file.

GroupAttributeNameForUserAttributes	country
-------------------------------------	---------

**Picture 23: Define Group Attribute Mapping – PWB Config**

Whereas the Group as well as the country field at the cad document are of the list type PWBAttributeMappingGroup. The list is already installed with the PWB, the values has to be configured.

Then for the CAD Document “Test54” the cad attribute sizeDE is mapped to the innovator attribute size, as the cad attribute country has the value “DE”, and for “DE” the “sizeDE” is configured.

Whereas for the Group “BR” the attribute “sizeBR” is configured. So for the CAD Document “Test55”, where the country attribute is set to “BR”, the cad attribute “sizeBR” is mapped to innovator attribute “size”.

It must be insured that the cad document attribute which is configured in the PWB Configuration (country in this example), is set to the correct value in a cad document afterAdd Event.

### ***“CAD is Master for Instances” Functionality***

The PDM Workbench always controls instances by PDM. It reads the instance information from PDM (position, instance name, number of instances). It stores all instance information in PDM, by creating instances.

With this functionality, when a CAD structure is loaded from PDM, the instance information from the prt file is taken, the instance information from PDM is ignored.

At PDM Update the instance information in PDM is updated from the current values of the prt, as before. The difference is that the PDM Workbench Load process is not dependent of the correct, or even existing, instance information in the CAD structure to be loaded.

---

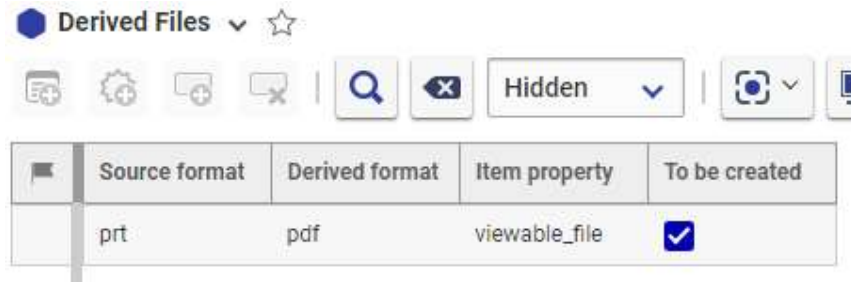
## Derived Files Configuration

### *Derived viewable files*

PDF files derived from the top assembly drawing can optionally be generated and uploaded during update.

And in the “Derived Files” tab in the PWB Configuration item there must be defined the derived file format and its storage property in Aras Innovator (e.g. viewable\_file or view\_file)

If the PDF should be created and uploaded the checkbox “to be created” must be checked.



**Picture 24: derived files configuration**

---

# CHAPTER 6

## Configurations for specific functionalities

This chapter describes the configuration of the PDM Workbench NX for specific functionalities.

---

### Standard configuration

#### *Exchange map*

In the PDM Workbench NX Schema file the absolute path of the exchange map directory, where the downloaded NX files are stored, can be configured.

Example:

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

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

Optional.

#### *SOAP target URL*

In the Schema file the URL of the server process, that the PDM Workbench NX client uses for its SOAP requests, can be configured.

Example:

```
<soapTargetUrl value="http://hostname/InnovatorServer" />
```

If the soap target URL value is defined by the environment variable "PWB\_SOAP\_TARGET\_URL", then that takes precedence. The definition in the Schema file takes effect only if such a NX environment variable does not exist.

Optional.

#### *Session settings*

In the Schema file the session settings of the PDM Workbench NX can be defined. The following entries are supported.

Example:

```
<sessionSettings>  
  <passwordEncryption name="MD5" />  
</sessionSettings>
```

Optional.

---

### **Create Part mode**

Depending on the value of the Schema file attribute createparts, Aras Innovator Part items are associated with each CAS Document item.

Example:

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

Optional.

### **Update NX Data Structure from PDM Option**

If the UpdateNXDataStructureFromPDM Setting is enabled, the NX Data structure will be updated with respect to the PDM during load.

Example:

```
<settings>  
  <setting value="false" name="UpdateNXDataStructureFromPDM"/>  
</settings>
```

Optional.

### **Reconnect At Update**

With the ReconnectAtUpdate setting optional reconnect during update will be enabled.

Example:

```
<settings>  
  <setting value="true" name="showReconnectAtUpdate"/>  
</settings>
```

Optional.

### **Key attribute**

Internal attribute, do not change.

### **Class attribute**

Internal attribute, do not change.

### **Relation attribute**

Internal attribute, do not change.

### **Relationship attribute**

Internal attribute, do not change.

### **Left relationship attribute**

Internal attribute, do not change.

### **Right relationship attribute**

Internal attribute, do not change.

---

### ***Left relation class attribute***

Internal attribute, do not change.

### ***Right relation class attribute***

Internal attribute, do not change.

### ***Extended relation class attribute***

Internal attribute, do not change.

### ***Last modification date attribute***

In the Schema file the name of the last modification date attribute can be defined.

Example:

```
<lastModificationDateAttribute name="last_mod_date" />
```

Default value: "last\_mod\_date"

Optional.

---

## **Data model configuration**

The PWB Configuration item setting "UseBomPartStructure" indicates which data model will be used with the PDM Workbench NX.

UseBomPartStructure	false
---------------------	-------

**Picture 25: Sample UseBomPartStructure configuration**

Default value: "false"

Mandatory.

### ***Document data model***

In the "CAD Document Structure Data Model" the PDM structure is represented by CAD Documents. The relation "CAD Structure" is used.

Each CAD Document includes the NX file.

---

## **Query configuration**

### ***The Query dialog attributes***

It is possible to configure the Query dialog. The Query dialog configuration is done with the "form name" tag "Query".

Within the "form name" tag the "formAttribute" tags are defined.

```
<form name="Query">  
  <formAttribute name="item_number"  
    ...  
/>
```

...  
</form>

### QueryOrderByAttribute

The PWB Configuration item setting “QueryOrderByAttribute” defines an attribute by which the query results are internally ordered. This is not noticeable by the user, but it can result in significant performance improvements when a query is performed if the attribute is in the database index.

QueryOrderByAttribute	id
-----------------------	----

Picture 26: Sample QueryOrderByAttribute configuration

Optional.

### MaxQueryResults

The PWB Configuration item setting “MaxQueryResults” defines the maximum number of items that are retrieved in a single query.

MaxQueryResults	100
-----------------	-----

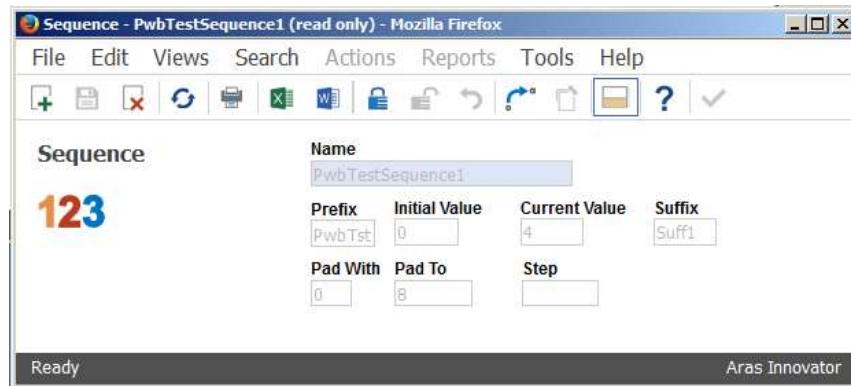
Picture 27: Sample MaxQueryResults configuration

Optional.

---

## Autaname Support using Aras Innovator Sequence Items

First sequence items which should be used for the autaname functionality need to be created:



Picture 28: Sample Sequence item

The following sequence items will be used in the configuration example:

Name	Prefix	Current Value	Suffix	Pad With	Pad To
Pwb* CAD*					
CAD Document	CAD-	0		0	8
PwbTestSequence1	PwbTst1	4	Suff1	0	8
PwbTestSequence2	PwbTst2	0	Suff2	0	8

Picture 29: Sequence items used in example

Then the sequence items to be used need to be configured in the schema file:

First an attribute with a data source which contains the names of the sequence items needs to be defined:

```
<attribute name="pwbAutonameRule" displayName="NLS_AutonameRule"
          dataSource="AutonameRules" />
```

```
<dataSource name="AutonameRules" type="ValueList">
  <value name="" displayName="None" />
  <value name="PwbTestSequence1" displayName="" />
  <value name="PwbTestSequence2" displayName="" />
  <value name="CAD Document" displayName="" />
</dataSource>
```

Then a corresponding form attribute has to be included in the login dialog.

```
<form name="Login" info="ShowOnlyLoginData" >
  <frame displayName="NLS_UserData">
    ...
    <formAttribute name="pwbAutonameRule"
                  displayName="Autoname Rule"
                  widgetType="ComboBox" mode="update"
                  visibleLength="15" required="false"
                  entryAllowed="false" />
  </frame>
</form>
```

The setting "UseServerMethodsForAutoname" has to be set to "false" in the active PWB Configuration item.

UseServerMethodsForAutoname	false
-----------------------------	-------

Picture 30: Sample UseServerMethodsForAutoname configuration

This will enable the user to select a sequence item name as an autoname rule at login.

---

### ***Autoname functionality can use a server method***

The autoname functionality can use a server method instead of using a sequence item directly for obtaining a PDM-generated part or document number value.

The autoname functionality has to be configured in the file "PWBSchema.xml".

The server method to be used needs to be configured in the Schema file:

First an attribute with a data source which contains the name of the server method needs to be defined:

```
<attribute name="pwbAutonameRule" displayName="NLS_AutonameRule"
          dataSource="AutonameRules" />
```

```
<dataSource name="AutonameRules" type="ValueList">
  <value name=" PwbAutonameMethod1" displayName="" />
</dataSource>
```

Then a corresponding form attribute has to be included in the login dialog ...

```
<form name="Login" info="ShowOnlyLoginData" >
  <frame displayName="NLS_UserData">
    ...
    <formAttribute name="pwbAutonameRule"
                  widgetType="ComboBox"
                  mode="update" visibleLength="15"
                  required="false" entryAllowed="false" />
  </frame>
</form>
```

... and in the "Set PDM Configuration" dialog.

```
<form name="PdmSessionConfig">
  <formAttribute name="pwbAutonameRule" widgetType="ComboBox"
                mode="update" visibleLength="15" required="false"
                listViewRelevant="true" />
</form>
```

This will enable the user to select a sequence item name as an autoname rule either at login or later while working in the PDM Workbench session.

In order for the "Set PDM Configuration" dialog to appear the setting "SetSessionConfig" has to be removed from the "removeToolBarIcons" definition:

```
<removeToolBarIcons>
  <!-- "LocalWorkspace", "Register", "Update", "Synchronize",
        "Refresh", "SetSessionConfig", "NewPwbWindow", "About"
  -->
  <!-- <icon name="LocalWorkspace" /> -->
  <icon name="Register" />
  <!-- <icon name="Update" /> -->
  <icon name="Synchronize" />
  <!-- <icon name="Refresh" /> -->
  <!-- <icon name="SetSessionConfig" /> -->
  <icon name="NewPwbWindow" />
  <!-- <icon name="About" /> -->
```



---

```
</removeToolBarIcons>
```

The setting "UseServerMethodsForAutoname" has to be set to "true" (default) in the active PWB Configuration item.

An additionally server method whose name corresponds to the name configured in the file "PWBSchema.xml" (e.g. "PwbAutonameMethod1"; "Server-Side"; "C#"; "Execution allowed to World") has to be defined on the Aras Innovator server.

The server method can use information from standard NX attributes of the NX files to be imported to PDM, or values from PDM Workbench dialogs.

This is an example of such a server method:

```
// CC0.Utilities.WriteDebug("_PwbOutput", "Entering 'PwbAutonameMethod1' ");

var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this);

// Preparing the input information
string Autoname = this.GetProperty("Autoname");
string PdmType = this.GetProperty("Type");
string PdmClassification = this.GetProperty("Classification");

Item CadStdPropsItem = this.GetPropertyItem("CadStdProps");
IDictionary<string, string> CadStdPropsDict = null;
if (CadStdPropsItem != null)
{
    CadStdPropsDict = PwbServerApiObj.DialogAttrsItemToDictionary(CadStdPropsItem);
}

Item CadDocInputDialogItem = this.GetPropertyItem("CadDocDialogAttrs");

IDictionary<string, string> CadDocInputDialogDict = null;
if (CadDocInputDialogItem != null)
{
    CadDocInputDialogDict =
        PwbServerApiObj.DialogAttrsItemToDictionary(CadDocInputDialogItem);
}

Item PartInputDialogItem = this.GetPropertyItem("PartDialogAttrs");
IDictionary<string, string> PartInputDialogDict = null;
if (PartInputDialogItem != null)
{
    PartInputDialogDict = PwbServerApiObj.DialogAttrsItemToDictionary(PartInputDialogItem);
}

// Place the custom algorithm here
string OutputLogInfo = "";

OutputLogInfo += "Autoname:" + Autoname + "" + "|";
OutputLogInfo += "PdmType:" + PdmType + "" + "|";
OutputLogInfo += "PdmClassification:" + PdmClassification + "" + "|";

if (CadStdPropsDict != null)
{
    OutputLogInfo += "CadStdProps:" + "|";

    var Enumerator = CadStdPropsDict.GetEnumerator();
    while (Enumerator.MoveNext() == true)
    {
        var CurrentVal = Enumerator.Current;
        OutputLogInfo += "" + CurrentVal.Key + "'->' + CurrentVal.Value + "" + "|";
    }
}

if (CadDocInputDialogDict != null)
{
    OutputLogInfo += "CadDocInputDialog:" + "|";
    var Enumerator = CadDocInputDialogDict.GetEnumerator();
    while (Enumerator.MoveNext() == true)
    {
        var CurrentVal = Enumerator.Current;
        OutputLogInfo += "" + CurrentVal.Key + "'->' + CurrentVal.Value + "" + "|";
    }
}
}
```

```

if (PartInputDialogDict != null)
{
    OutputLogInfo += "PartInputDialog:" + "|";

    var Enumerator = PartInputDialogDict.GetEnumerator();
    while (Enumerator.MoveNext() == true)
    {
        var CurrentVal = Enumerator.Current;
        OutputLogInfo += "\"" + CurrentVal.Key + "->" + CurrentVal.Value + "\"" + "|";
    }
}

// Getting the actual autoname value
string AutonameValue = "";
if (PdmType == "Part")
{
    AutonameValue = PwbServerApiObj.GetNextAutonameSequence("CAD Document");
}
else
{
    if ((PartInputDialogDict != null) &&
        (PartInputDialogDict.ContainsKey("item_number")))
    {
        AutonameValue = PartInputDialogDict["item_number"];
    }
    else
    {
        AutonameValue = PwbServerApiObj.GetNextAutonameSequence("CAD Document");
    }
}

OutputLogInfo += "done";

// Preparing the output
IDictionary<string, string> OutputInfoDict = new Dictionary<string, string>();

OutputInfoDict.Add("AutonameValue", AutonameValue);
OutputInfoDict.Add("LogLines", OutputLogInfo);

Item OutputInfoItem = PwbServerApiObj.DialogAttrsDictionaryToItem(OutputInfoDict);

return OutputInfoItem;

```

## Open in Aras from NX Client

Single CAD items can be loaded in the Aras Innovator web client from NX.

### Configure web-browser

By default, the open in Aras functionality uses the default web browser. Now it is also possible to configure a different browser to open the selected object in Aras.

If a special browser is configured, it is also possible to configure special browser options for this browser

To configure a web-browser the following environment variable must be set on the CATIA client (this can be done in the environment script of the PDM Workbench

```

catia_env.bat
SET PWB_WEBBROWSER=<browser>

```

Optional a second environment variable can be set to specify the browser options:

```

SET PWB_WEBBROWSER_OPTIONS=<browser options>

```

Example:

```

SET PWB_WEBBROWSER=msedge
SET PWB_WEBBROWSER_OPTIONS=--inprivate

```

### ***Close helper window***

When using “Open in Aras” an additional “helper window” is displayed even if the session is already open. It is possible to close this window automatically after the Item is opened.

Modify the file (beginning with Aras 24)

\Innovator\Client\Modules\aras\startup\deepLinking.ts in your Aras code tree.

Modify the file (Aras 14 to Aras 23)

\Innovator\Client\Modules\aras.innovator.core.MainWindow\deepLinking.ts in your Aras code tree.

Modify the file (up to Aras 12)

\Innovator\Client\Modules\aras.innovator.core.MainWindow\deepLinking.js in your Aras code tree.

Around line 26 you can add an additional snippet of code to close the window if an Aras Innovator instance is already open.

```
if (!deepWindowIframe.src) {
    deepWindowIframe.src = 'deepLinking.aspx';
    // Close this window if Innovator is already open
    var objWin = window.self;
    objWin.open('', '_self', '');
    objWin.close();
}
```

**Picture 31: File “deepLinking.ts or .js”**

Save the deepLinking.ts or .js file and restart the IIS.

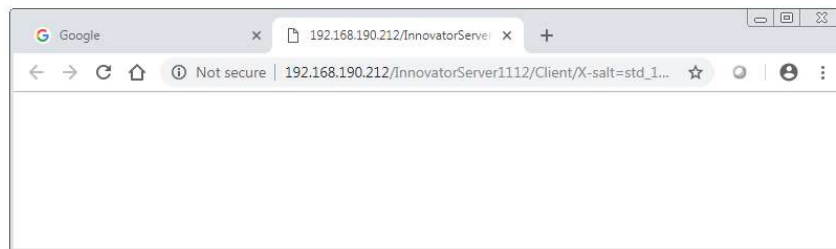
At all clients the browsers cache of the default browser must be cleared to enforce the use of the changed script.

Please note that this proposed change was tested in Aras Innovator 11.0 SP12, 11.0 SP15 and Aras 12.0 up to SP9 and Aras 17.

For later Aras Innovator version this behavior may be changed.

### **For Aras Innovator 11 only**

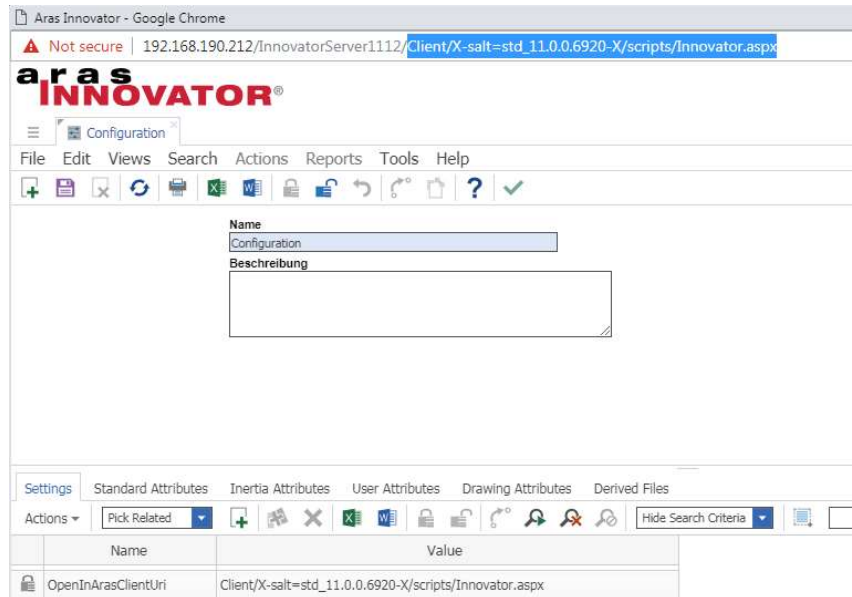
With Aras Innovator 11 the “Open In Aras” functionality opens a new empty tab in an additional default browser window.



**Picture 32: Empty tab**

To avoid this window, you must set the PWB Configuration “OpenInArasClientUri” to the client part of your Aras installation. You can obtain the value from the URL in the Aras Innovator Window:

[http://192.168.190.212/InnovatorServer1112/Client/X-salt=std\\_11.0.0.6920-X/scripts/Innovator.aspx](http://192.168.190.212/InnovatorServer1112/Client/X-salt=std_11.0.0.6920-X/scripts/Innovator.aspx)



**Picture 33: Aras Innovator web client with Client URL**

This setting must not exist in Aras Innovator 12.

To make sure the window handling works correctly an additional default browser window must exist (this could be the initial browser window of the Aras Innovator login).

### Window Management

If you use “Open in Aras” this functionality brings the Aras Innovator window on top.

By default the Aras Innovator window uses “Aras Innovator ...” as window title:



**Picture 34: Window title “Aras Innovator”**

If your customization uses a different window title you have to set the PWB Configuration “ArasWindowTitleSubString” to your window title.



**Picture 35: PWB Configuration setting “ArasWindowTitleSubString”**

### Reconnect at Update

To be able to reconnect NX parts during update to existing items on server, the following configurations need to be done on server:

Create a new Aras Innovator server method “PwbReconnectAtUpdate”, which searches for an existing CAD in Aras Innovator.

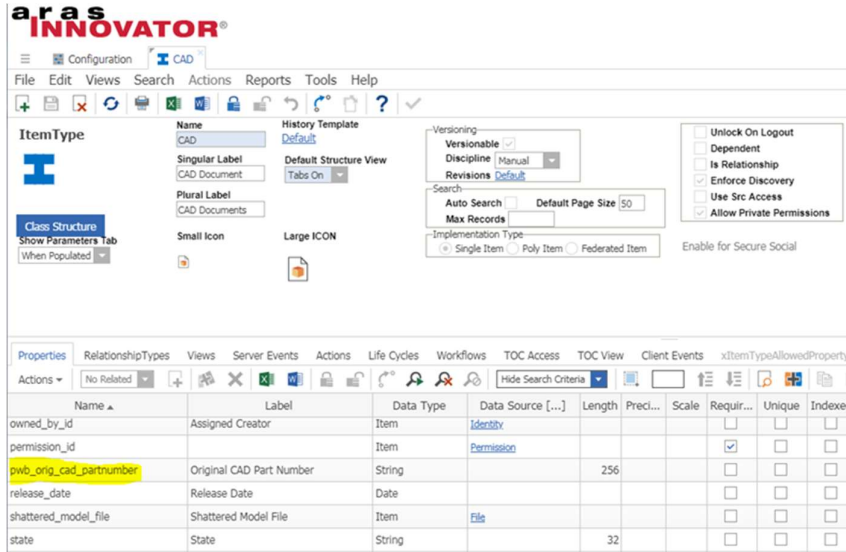
If you want to use a different name for the method, you have to set the name in the PWB Configuration setting “ReconnectAtUpdateMethod”:

ReconnectAtUpdateMethod	<Your Method name>
-------------------------	--------------------

**Picture 36: PWB Configuration setting “ReconnectAtUpdateMethod”**

If you want to use the original NXPart Number to find the items to be reconnected, you have to configure the following:

Create a new String property for CAD that holds the original CATIA Part number:



**Picture 37: Item Type “CAD” – Add property “pwb\_orig\_cad\_partnumber”**

If you use the “BOM Part Structure Data Model” you also have to create the String property in the Part type.

In the PWB Configuration you have to add the setting “OrigCadPartnumberAttr”.

OrigCadPartnumberAttr	pwb_orig_cad_partnumber
-----------------------	-------------------------

**Picture 38: PWB Configuration – “OrigCadPartnumberAttr”**

If this property is set, the original CATIA Part number is stored during create.

To use the original NX Part Number during reconnect, you can implement the method “PwbReconnectAtUpdate” like follows:

```
var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this);
Innovator InnovatorObj = this.getInnovator();

// ignore default CATIA Part Numbers for reconnect
var PartnumbersToIgnoreHashSet = new HashSet<string>();

PartnumbersToIgnoreHashSet.Add("model1");
PartnumbersToIgnoreHashSet.Add("model2");
PartnumbersToIgnoreHashSet.Add("model3");
PartnumbersToIgnoreHashSet.Add("model4");
PartnumbersToIgnoreHashSet.Add("model5");
PartnumbersToIgnoreHashSet.Add("model6");
PartnumbersToIgnoreHashSet.Add("modelmodel17");
PartnumbersToIgnoreHashSet.Add("model8");
PartnumbersToIgnoreHashSet.Add("model9");
PartnumbersToIgnoreHashSet.Add("assembly1");
PartnumbersToIgnoreHashSet.Add("assembly2");
PartnumbersToIgnoreHashSet.Add("assembly3");
PartnumbersToIgnoreHashSet.Add("assembly4");
PartnumbersToIgnoreHashSet.Add("assembly5");
PartnumbersToIgnoreHashSet.Add("assembly6");
PartnumbersToIgnoreHashSet.Add("assembly7");
```

```

PartnumbersToIgnoreHashSet.Add("assembly8");
PartnumbersToIgnoreHashSet.Add("assembly9");

// Preparing the input information
string ReconnectAtUpdate = this.getProperty("ReconnectAtUpdate");
string PdmType = this.getProperty("Type");
string PdmClassification = this.getProperty("Classification");

Item CatiaStdPropsItem = this.getPropertyItem("CadStdProps");
IDictionary<string, string> CatiaStdPropsDict = null;
if (CatiaStdPropsItem != null)
{
    CatiaStdPropsDict =
        PwbServerApiObj.DialogAttrsItemToDictionary(CatiaStdPropsItem);
}

Item CadDocInputDialogItem = this.getPropertyItem("CadDocDialogAttrs");
IDictionary<string, string> CadDocInputDialogDict = null;
if (CadDocInputDialogItem != null)
{
    CadDocInputDialogDict =
        PwbServerApiObj.DialogAttrsItemToDictionary(CadDocInputDialogItem);
}

Item PartInputDialogItem = this.getPropertyItem("PartDialogAttrs");
IDictionary<string, string> PartInputDialogDict = null;
if (PartInputDialogItem != null)
{
    PartInputDialogDict =
        PwbServerApiObj.DialogAttrsItemToDictionary(PartInputDialogItem);
}

string OrigCatiaPartNumber = null;

if (PdmType == "CAD")
{
    if (PdmClassification == "Mechanical/Assembly" ||
        PdmClassification == "Mechanical/Part" )
    {
        if (CadDocInputDialogDict != null &&
            CadDocInputDialogDict.ContainsKey("item_number"))
        {
            OrigCatiaPartNumber = CadDocInputDialogDict["item_number"];
        }
        else if (CatiaStdPropsDict != null)
        {
            OrigCatiaPartNumber = CatiaStdPropsDict["CadPartNumber"];
        }
    }
    else
    {
        if (CadDocInputDialogDict != null)
        {
            OrigCatiaPartNumber = CadDocInputDialogDict["item_number"];
        }
    }
}
else
{
    if (PdmClassification == "Assembly" || PdmClassification == "Component" )
    {
        if (PartInputDialogDict != null &&
            PartInputDialogDict.ContainsKey("item_number"))
        {
            OrigCatiaPartNumber = PartInputDialogDict["item_number"];
        }
        else if (CatiaStdPropsDict != null)
        {
            OrigCatiaPartNumber = CatiaStdPropsDict["CadPartNumber"];
        }
    }
    else
    {
        if (CadDocInputDialogDict != null)
        {
            OrigCatiaPartNumber = CadDocInputDialogDict["item_number"];
        }
    }
}
}

```

```

    }
}
if (!String.IsNullOrEmpty(OrigCatiaPartNumber) &&
    !PartnumbersToIgnoreHashSet.Contains(OrigCatiaPartNumber))
{
    Item QueryItem = InnovatorObj.newItem(PdmType, "get");
    QueryItem.setProperty(PwbServerApiObj.OrigCadPartnumberAttr(),
        OrigCatiaPartNumber);
    if (PdmClassification != null)
    {
        QueryItem.setProperty("classification", PdmClassification);
    }
    Item ExistingPdmItem = QueryItem.apply();

    if (ExistingPdmItem != null)
    {
        if (ExistingPdmItem.isError())
        {
            return null;
        }
        else if (ExistingPdmItem.isCollection())
        {
            return InnovatorObj.newError(
                "ReconnectAtUpdate Failed: Multiple Items of type '"+PdmType+
                "'/" +PdmClassification + "' with '"
                + PwbServerApiObj.OrigCadPartnumberAttr() + "' = '"
                + OrigCatiaPartNumber + "' found");
        }
    }
    return ExistingPdmItem;
}
return null;

```

## Create only one generation per claim

If the server setting 'ClaimedIsNewGenAttr' is set to *pwb\_claimed\_is\_new\_gen*, the creation of a new generation might be performed with respect to a standard process.

ClaimedIsNewGenAttr	pwb_claimed_is_new_gen
---------------------	------------------------

**Picture 39: PWB Configuration setting "ClaimedIsNewGenAttr"**

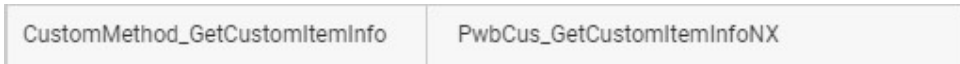
The default implementation creates only one new generation of a CAD document per claim.

This new generation will be created at the first update after the claim. Further updates will overwrite the newly created generation. If a new generation of the CAD document should be created explicitly then the user has to unclaim the CAD document and claim it again before performing the next update.

CAD document generations which are read-only, for example because they are released or frozen can be claimed if they are current. In this case the new generation will be created by the claim process, and the first update will not create another new generation.

If an already claimed CAD document becomes read-only later, then a new generation of the CAD document will be created at update, since the claimed generation cannot be overwritten.

If the default implementation needs to be changed, the default implementation can be overwritten by defining a C# server method whose name corresponds to the value of the server PWB configuration 'CustomMethod\_GetCustomItemInfo':



Picture 40: Setting “CustomMethod\_GetCustomItemInfo”



Picture 41: Custom method “PwbCus\_GetCustomItemInfoNX”

This method returns the information whether to create a new generation or not.

This is the default implementation which can be overwritten:

```
using (var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this))
{
    PwbServerApiObj.Log("Called method 'PwbCus_GetCustomItemInfo' -> ");

    string PdmActionStr = getProperty("PdmAction");
    PwbServerApiObj.Log(" -> PdmActionStr:'" + PdmActionStr + "'");

    string InputObjsStr = getProperty("CustomItemInfoInputObjs");
    PwbServerApiObj.Log(" -> InputObjsStr:'" + InputObjsStr + "'");

    IDictionary<string, IDictionary<string, string>> ConfigDict =
        new Dictionary<string, IDictionary<string, string>>();

    var ItemList = PwbServerApiObj.QueryItemsForCustomItemInfo(InputObjsStr);
    foreach (var ItemObj in ItemList)
    {
        PwbServerApiObj.Log(
            " -> '" + ItemObj.getType() + "' / '" + ItemObj.getID() + "'");

        IDictionary<string, string> CurrentItemConfigDict;
        GetItemConfigSettings(PwbServerApiObj, ItemObj, out
            CurrentItemConfigDict);

        ConfigDict.Add(ItemObj.getID(), CurrentItemConfigDict);
    }

    IDictionary<string, string> OutputInfoDict = new Dictionary<string, string>();
    OutputInfoDict.Add(
        "CustomItemInfoObjConfigSettings",
        PwbServerApiObj.StringDictDictToString(ConfigDict));

    return PwbServerApiObj.DialogAttrsDictionaryToItem(OutputInfoDict);
}

private void GetItemConfigSettings(
    PwbServerAddin.PwbServerApi PwbServerApiObj,
    Item ItemObj,
    out IDictionary<string, string> ItemConfigDict)
{
    ItemConfigDict = new Dictionary<string, string>();

    // By default, the same calls as in the hardcoded server method.
    int LockStatus = ItemObj.getLockStatus();

    bool IsEditAllowed = (LockStatus == 1);
}
```



```

bool IsClaimAllowed = (LockStatus == 0);
bool ClaimAsNewGeneration = false;

// Check for standard parts and for template files
if ((PwbServerApiObj.ItemIsStandardPart(ItemObj)) &&
    (!PwbServerApiObj.IsUserStandardPartAdmin()))
{
    IsEditAllowed = false;
    IsClaimAllowed = false;
}

if ((PwbServerApiObj.ItemIsTemplateFile(ItemObj)) &&
    (!PwbServerApiObj.IsUserTemplateFileAdmin()))
{
    IsEditAllowed = false;
    IsClaimAllowed = false;
}

// Released items
if ((ItemObj.getProperty("is_released") == "1") &&
    String.IsNullOrEmpty(ItemObj.getProperty("source_id"))) // not a relation
{
    IsEditAllowed = false;
    IsClaimAllowed = false;
    ClaimAsNewGeneration = false;
}

// Superseded items should also not be modified.
if (PwbServerApiObj.ItemIsSuperseded(ItemObj))
{
    IsEditAllowed = false;
    IsClaimAllowed = false;
}

// Reconnected items must not be modified
if (PwbServerApiObj.IsInReconnectContext())
{
    IsEditAllowed = false;
    IsClaimAllowed = false;
}

if (IsEditAllowed || IsClaimAllowed)
{
    PwbServerApiObj.CheckForEnvironmentAttributes(
        ItemObj,
        ref IsEditAllowed,
        ref IsClaimAllowed,
        ref ClaimAsNewGeneration);
}

// Debug info
ItemConfigDict.Add("ItemNumber", ItemObj.getProperty("item_number", ""));
ItemConfigDict.Add("Type", ItemObj.getType());
ItemConfigDict.Add("Id", ItemObj.getID());

ItemConfigDict.Add(
    "IsUpdateAllowed",
    (ItemObj.getProperty("pwb_update_allowed") == "1") ? "true" :
"false");

ItemConfigDict.Add(
    "ClaimedIsNewGen",
    (ItemObj.getProperty("pwb_claimed_is_new_gen") == "1") ? "true" :
"false");

bool IsReleased = (ItemObj.getProperty("is_released") == "1");
bool UpdateAllowed = !IsReleased;

if (!UpdateAllowed)
{
    ClaimAsNewGeneration = true;
}

bool NewGenHasAlreadyBeenCreated =
(ItemObj.getProperty("pwb_claimed_is_new_gen") == "1");

```

```

bool CreateNewGenAtUpdate = ((!UpdateAllowed) || (!NewGenHasAlreadyBeenCreated));

ItemConfigDict.Add("IsEditAllowed", IsEditAllowed.ToString().ToLower());
ItemConfigDict.Add("IsClaimAllowed", IsClaimAllowed.ToString().ToLower());
ItemConfigDict.Add("ClaimAsNewGeneration",
    ClaimAsNewGeneration.ToString().ToLower());
ItemConfigDict.Add("CreateNewGenAtUpdate",
    CreateNewGenAtUpdate.ToString().ToLower());

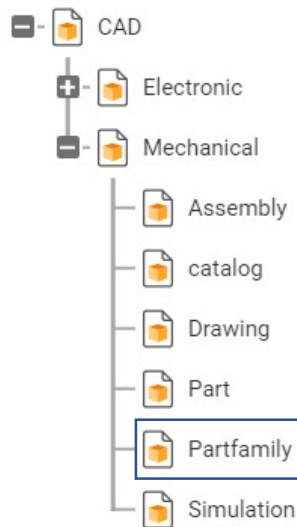
```

## Family Part Handling

To be able to work with family parts, the following data extensions have to be done on the server:

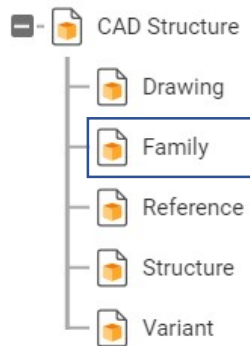
First the classifications for item type CAD and CAD Structure need to be adapted.

The classification "Mechanical / Partfamily" need to be added to the CAD item.



**Picture 42: CAD item extension**

Moreover, the classification "Family" is required for the CAD Structure item.



**Picture 43: CAD Structure item extension**

---

On the client side the schema file needs to be adapted. (see also: Chapter: Structure of the Schema File)

The /CAD/Mechanical/Partfamily item should be mentioned within the schema file.

```
<object name="/CAD/Mechanical/Partfamily" displayName="Family Template" icon="FamilyTemplate"> ...
```

---

## Standard Part Handling

Standard parts might be defined by the standard part administrator.

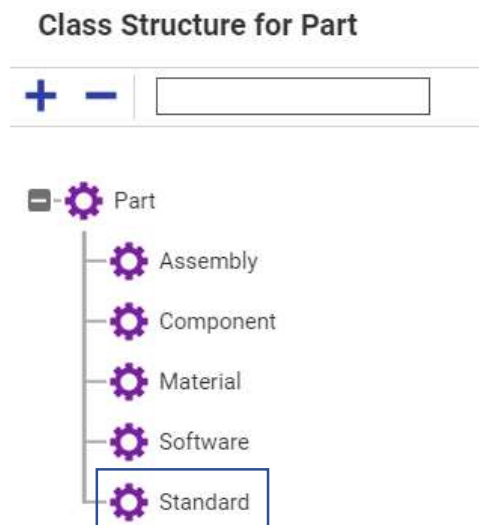
For standard parts the *is\_standard* attribute of the CAD document item is set to *true* then and the related part item has the classification /Part/Standard.

Key behaviour of a standard part is, that a reconnect to a given item in PDM is always tried during update from CAD to PDM.

To able to work with standard parts the following adjustments are required:

To provide key behaviour of standard parts, the reconnect mechanism should be configured. (See Chapter: Reconnect at Update).

Working with Part items, the Part item needs to be expanded by the "Standard" classification.



**Picture 44: Part item extension**

And on the client side the /Part/Standard item is requested in the schema file.

```
<object name="/Part/Standard" displayName="Standard Part" icon="StandardPart">...
```

---

# CHAPTER 7

## Client Schema File Configuration

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

---

### Structure of the Schema File

The main purpose of the PDM Workbench NX 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 NX and who needs to save the NX 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 NX.

```
<!-- 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 attribute *system*.
- “displayName” Contains the NLS (native language support) name of the PDM system or customization that is defined in the tag *PWBSchema*.
- “visibleLength” Contains the visible length of the display name to be shown in the dialogs of NX.
- “allowedLength” Contains the allowed length of the values inserted in the text editor widgets in characters.

### 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 display name for that object that the PDM Workbench NX users can see.

File “PWBSchema.xml”:

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

### Configuration settings

Now the configuration of the PDM Workbench NX can be defined in the Schema file.

The tags are described in detail in the previous chapter. In this list you can see if the configuration is optional or mandatory.

xmap	optional	see “Exchange map” on page 27
soapTargetUrl	optional	see “SOAP target URL” on page 27
sessionSettings	optional	see “Session settings” on page 27
lastModificationDateAttribute	mandatory	see “Last modification date attribute” on page 29
createparts	optional	See “Create Part mode” on page 28
UpdateNXDataStructureFromPDM	optional	See “Update NX Data Structure from PDM Option” on page 28
showReconnectAtUpdate	optional	See “Reconnect At Update” on page 28

### “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**.

***“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.
- “dataSource” Optional. The data source includes the possible values for this attribute.
- “isFileName” Optional. If it is set to “true” the value of the corresponding input file name is checked about illegal<sup>2</sup> 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" />
```

---

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

---

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. 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 the 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”.
- “dataSource” Optional. The value defines the link to a data source that is more special than the linked data source in the <attribute> tag.
- “displayOnly” Possible values are “true” or “false”. If “true”, then the display value of the value of the data source will be used.

Example:

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

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**

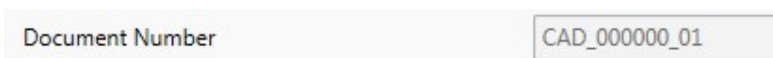
There are six different widget types available to build up dialogs with. All widget types except “SingleLineEditor”, “MultiLineEditor” and one mode of “NameValueList” can only be used on attributes that have certain kinds of Data Sources attached. Data Sources are a container of a limited set of values.

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

- |                  |  |
|------------------|--|
| SingleLineEditor | Supports “update” and “output” mode.<br>Can be used for attributes with no data source attached and also for attributes with data sources of type “SingleValue”. |
|------------------|--|



**Picture 45: Single Line Editor Widget, update mode**



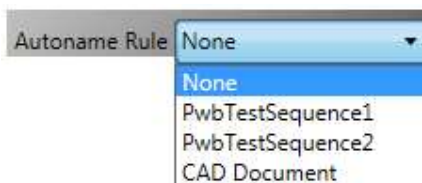
**Picture 46: Single Line Editor Widget, output mode**

**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”.



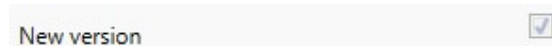
**Picture 47: Multi Line Editor Widget, update mode**

**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 48: 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 49: Single Check Box Widget, select mode**

**PasswordBox** Supports “update” mode.  
Can be used for attributes with no data source.



**Picture 50: Password Box Widget**

**URL** Supports “output” mode.  
Can be used for attributes with no data source.



**Picture 51: URL Widget**



---

## Login Form

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="PWBServerURL" displayName="Server"
      widgetType="URL" mode="output"
      visibleLength="15" required="true" />
    <pwbFormAttribute name="PWBLoginUser" displayName="Username"
      widgetType="SingleLineEditor"
      mode="update" visibleLength="15"
      required="true" entryAllowed="true" />
    <pwbFormAttribute name="PWBLoginPassword"
      displayName="Password"
      widgetType="PasswordBox" mode="update"
      visibleLength="15" required="false" />
    <formAttribute name="LoginDatabase" displayName="Database"
      widgetType="ComboBox" mode="update"
      visibleLength="15" required="true"
      entryAllowed="false" />
    <formAttribute name="pwbAutonameRule"
      displayName="Autoname Rule"
      widgetType="ComboBox" mode="update"
      visibleLength="15" required="false"
      entryAllowed="false" />
  </frame>
</form>
```

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

Mandatory.

---

## PDM Objects

XML tags *object* define PDM object classes that can be used in the PDM Workbench NX 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/Component" displayName="Part"
  icon="Part">
```

---

## Description of PDM Objects

The tag *description* 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" />
</description>
```

## Actions on PDM Objects

Some actions that can be performed with PDM objects are defined in the Schema file.

Menu 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" menu button, the type "Assembly" (display name) is included in the query dialog list, otherwise it is not.

## PDM Object Forms

The following forms can be defined for an object class:

"Query", "Properties"

Example:

```
<form name="Query">
  <formAttribute name="item_number" displayName="Document Number"
    widgetType="SingleLineEditor" mode="update"
    visibleLength="15" required="false"
    listViewRelevant="true" entryAllowed="true"
    dataSource="RecentlyUsedValueDataSource" />
  <formAttribute name="major_rev" displayName="Major Rev."
    widgetType="SingleLineEditor" mode="update"
    visibleLength="15" required="false"
    listViewRelevant="true" entryAllowed="true"
    dataSource="RecentlyUsedValueDataSource" />
  <formAttribute name="generation" displayName="Generation"
    widgetType="SingleLineEditor" mode="update"
    visibleLength="15" required="false"
    listViewRelevant="true" entryAllowed="true"
    dataSource="RecentlyUsedValueDataSource" />
  <formAttribute name="name" displayName="Name"
    widgetType="SingleLineEditor" mode="update"
    visibleLength="15" required="false"
    listViewRelevant="true" entryAllowed="true"
    dataSource="RecentlyUsedValueDataSource" />
```

---

```

<formAttribute name="state" displayName="State"
  widgetType="SingleLineEditor" mode="update"
  visibleLength="15" required="false"
  listViewRelevant="true" entryAllowed="true"
  dataSource="RecentlyUsedValueDataSource" />

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

<formAttribute name="created_on" displayName="Created on"
  widgetType="SingleLineEditor" mode="update"
  visibleLength="15" required="false"
  listViewRelevant="true" entryAllowed="true"
  dataSource="RecentlyUsedValueDataSource" />

<formAttribute name="modified_on" displayName="Modified on"
  widgetType="SingleLineEditor" mode="update"
  visibleLength="15" required="false"
  listViewRelevant="true" entryAllowed="true"
  dataSource="RecentlyUsedValueDataSource" />

<formAttribute name="created_by_id" displayName="Created by ID"
  widgetType="SingleLineEditor" mode="update"
  visibleLength="15" required="false"
  listViewRelevant="true" entryAllowed="true"
  dataSource="RecentlyUsedValueDataSource" />

<formAttribute name="modified_by_id"
  displayName="Modified by ID"
  widgetType="SingleLineEditor" mode="update"
  visibleLength="15" required="false"
  listViewRelevant="true" entryAllowed="true"
  dataSource="RecentlyUsedValueDataSource" />

<formAttribute name="locked_by_id" displayName="Locked by ID"
  widgetType="SingleLineEditor" mode="update"
  visibleLength="15" required="false"
  listViewRelevant="true" entryAllowed="true"
  dataSource="RecentlyUsedValueDataSource" />

</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 NX.

### 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:

---

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

Example:

```
<dataSource name="ItemTypes" type="ValueList">
  <value name="/CAD/Mechanical/Assembly" displayName="Assembly"/>
  <value name="/CAD/Mechanical/Part" displayName="Model" />
  <value name="/CAD/Mechanical/Drawing" displayName="Drawing" />
</dataSource>
```

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

Example:

```
<dataSource name="TrueOrFalse" type="BooleanValueList">
  <value name="1" displayName="NLS_true" booleanValue="true" />
  <value name="0" displayName="NLS_false" booleanValue="false" />
</dataSource>
```

### ***Complete example of using a data source tag:***

The attribute “new\_version” can be assigned exactly to “true” or “false”. Therefore we define a data source called “TrueOrFalse” and attach this container to the attribute description.

```
<attribute name="new_version" displayName="NLS_new_version"
  dataSource="TrueOrFalse" />

<dataSource name="TrueOrFalse" type="BooleanValueList">
  <value name="1" displayName="NLS_true" booleanValue="true" />
  <value name="0" displayName="NLS_false" booleanValue="false" />
</dataSource>
```

---

## **Customizing PDM Workbench NX Menu**

The PDM Workbench NX menu is integrated by default into the standard NX menu dialog.

But it is also possible to customize the NX menu to the individual needs.

It is possible to add PDM Workbench NX menu buttons to existing or new menus, toolbars or ribbon tabs. See the Menuscript User's Guide from the NX documentation.

The PDM Workbench NX also supports the replacement of the standard NX buttons by some PDM Workbench NX buttons.

The replacement is possible for the “Query”, “Update”, “Replace” and “Add Component” functions.

To do this, the “PWBSchema\_Aras\_NX.xml” and the “menu.men” file needs to be adapted.

Within the menu file the NX buttons are overwritten with the PDM Workbench NX action:

For example replace

```

BUTTON PDM_WORKBENCH_QUERY
LABEL Query
TOOLBAR_LABEL Query
MESSAGE Queries the PDM-System.
BITMAP PWB_T1b_Query.bmp
SENSITIVITY OFF
ACTIONS BL_query

```

with

```

BUTTON UG_FILE_OPEN
LABEL Query
TOOLBAR_LABEL Query
MESSAGE Queries the PDM-System.
BITMAP PWB_T1b_Query.bmp
SENSITIVITY OFF
ACTIONS BL_query

```

in the menu file.

In addition the ButtonName attribute of the "PWBSchema\_Aras\_NX.xml" configuration file needs to be adapted.

The displayName of the PDM Workbench NX function, here the 'Query' ButtonName, has to be replaced with the new button name (UG\_FILE\_OPEN).

```

<attribute name="ButtonName" displayName="Button" dataSource="ButtonNames"/>
<dataSource name="ButtonNames" type="ValueList">
  <value name="Query" displayName="UG_FILE_OPEN" />
  <value name="AddComponent" displayName="PDM_WORKBENCH_ADD_COMPONENT" />
  <value name="Replace" displayName="PDM_WORKBENCH_REPLACE" />
  <value name="Update" displayName="PDM_WORKBENCH_UPDATE" />
  <value name="CreateFile" displayName="PDM_WORKBENCH_CREATEFILE" />
</dataSource>

```

## Validation Rules for Update in Schema File

If no validation rules for an update is configured in the PWM Workbench NX Schema file, all files can be updated to Aras Innovator.

However you can define validation rules for an update. There are two different validation types:

- Error: no files can be uploaded until all error in the structure are resolved
- Warning: a warning is shown during update, but update can be proceeded nevertheless

The rules have to be defined the following way:

```

<validationRule type="Warning">
  <condition name="has3D" value="true"></condition>
  <condition name="isAssembly" value="true"></condition>
</validationRule>

```

The condition name can have the following values:

- has3D
- isAssembly
- isDrawing

The value can be true or false.

If all condition of a validation rule apply to a part file, an error or a warning is shown during update.

In the example above all files which are an assembly and contain geometry would show a warning during update, but update can be proceeded.

---

Multiples validation rules of both types (error and warning) can be configured.

---

### **Prevent update of NX files which were created with an older release**

In the Shema file configuration there is a setting called: « checkAuthoringToolVersion »  
if this setting is set to true `<checkAuthoringToolVersion value="true" />`, during a update there is a warning before files of a lower release are overwritten. The default value of this setting is false.

# CHAPTER 8

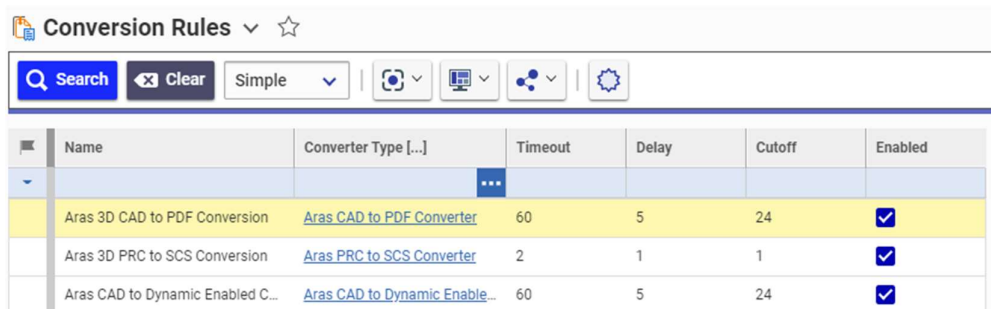
## Troubleshooting

This chapter lists error conditions and describes possible sources of such failures and how to solve them.

### Invalid CAD Instances with Aras 3D CAD to PDF Conversion

This issue has been observed in Aras Innovator SP14 and higher.

If the Aras 3D CAD to PDF Conversion is enabled in your Aras Innovator environment,



Name	Converter Type [...]	Timeout	Delay	Cutoff	Enabled
Aras 3D CAD to PDF Conversion	<a href="#">Aras CAD to PDF Converter</a>	60	5	24	<input checked="" type="checkbox"/>
Aras 3D PRC to SCS Conversion	<a href="#">Aras PRC to SCS Converter</a>	2	1	1	<input checked="" type="checkbox"/>
Aras CAD to Dynamic Enabled C...	<a href="#">Aras CAD to Dynamic Enable...</a>	60	5	24	<input checked="" type="checkbox"/>

please check the following method:

#### CR\_3DCADtoPDF\_SetFiles

At the end of the method, you can find the following code:

```
// When CAD assembly was converted for opening by 'Dynamic HOOPS Viewer'
appropriate 'CAD Instance'
// items should be created based on information from assembly tree XML
created by HOOPS converter
if (dynamicAssemblyEnabled)
{
    CCO.Utilities.WriteDebug(
        "_CR_3DCADtoPDF_SetFiles","dynamicAssemblyEnabled = true" );
    string rootCadId = depIds.ToArray()[0];
    string inParams =
        @"<root_cad_id>{0}</root_cad_id><master_xml_id>{1}</master_xml_id>";
    string res = string.Format(
        CultureInfo.InvariantCulture, inParams, rootCadId,
        files["shatteredModel"]);

    Item methodResult = inn.applyMethod(
        "CR_3DCADtoDyn_CreateCadInstances", res);
}
```

It is here deliberately written that 'CAD Instance' items will be created. This is done in the method CR\_3DCADtoDyn\_CreateCadInstances.

PDM Workbench already creates CAD Instances. Additional CAD instances will compromise the CAD Structure.

Change the line

```
if (dynamicAssemblyEnabled)
```

---

to

```
if (false)
```

In newer versions of Innovator the code looks like this:

```
// When CAD assembly was converted for opening by 'Dynamic HOOPS Viewer'  
appropriate 'CAD Instance'  
// items should be created based on information from assembly tree XML  
created by HOOPS converter  
if (dynamicAssemblyEnabled || streamingAssemblyEnabled)  
{  
    methodResult =  
        _dataAccessLayer.CreateCadInstances(  
            depIds, _files["shatteredModel"]);  
  
    if(methodResult.isError())  
    {  
        return methodResult;  
    }  
}
```

Please comment out the call of “\_dataAccessLayer.CreateCadInstances” and the corresponding error handling here.

---

## Locking CAD Documents in CATIA fails when the Conversion Server is installed.

In some cases locking CAD documents in CATIA fails with this server stack trace:

```
PdmObject.PerformAction (Lock)  
ArasUtil.AddOwner -> LockStatus:'0'  
ArasUtil.AddOwner -> caught exception:'An item with the same key has  
already been added.'  
    at System.Collections.Generic.Dictionary`2.Insert(TKey key, TValue  
value, Boolean add)  
    at  
System.Linq.Enumerable.ToDictionary[TSource,TKey,TElement](IEnumerable`1  
source, Func`2 keySelector, Func`2 elementSelector, IEqualityComparer`1  
comparer)  
    at Aras.Server.Core.FileContainerItemsOnAfterUpdate.Main(XmlDocument  
inDom, XmlDocument outDom, Object eventData) in  
E:\Builds\Innovator_RELS11-0-SP11\6812-RELS11-0-  
SP11\Innovator.git\CompilableCode\Core\InternalMethods\Files\FileContainer  
ItemsOnAfterUpdate.cs:line 67
```

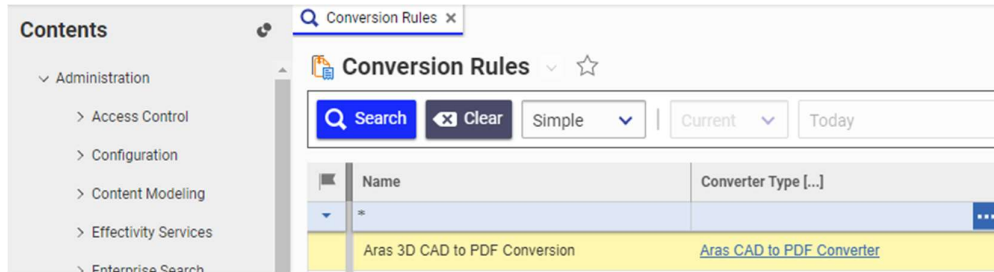
If that happens please create the server setting “EnableFileCloneCreation” and set it to the value “false”. That should fix the issue.

---

## Conversion Tasks fail for updated or new CAD files from PDM Workbench

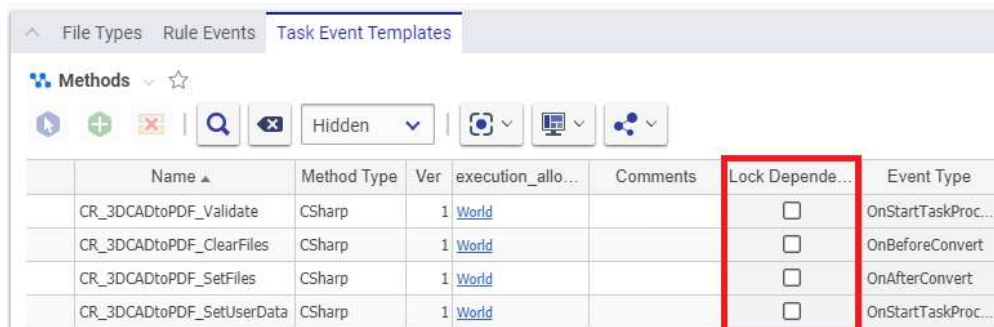
The Aras 3D CAD to PDF Conversion may fail after Update from PDM Workbench. In this case please check the “Aras 3D CAD to PDF Conversion” Rule:





**Picture 52: “Aras 3D CAD to PDF Conversion” Rule**

Edit the Rule and uncheck all checkboxes for “Lock Dependency” in the Task Event Templates Tab.



**Picture 53: “Uncheck Lock Dependencies Aras 3D CAD to PDF Conversion” Rule**