PDM Workbench NX

PDM Workbench NX Release 19.0 for Aras Innovator

Installation & Administration Manual

Version 1

T Systems

Copyright

© 2005-2025 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*

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
		19.0	March 2025

This edition 19.0 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*

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
PDM Workbench NX Installation & Administration Manual	*19.0*
PDM Workbench NX User Manual	*19.0*

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

Table of Figures

CHAPTER 1

Overview

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

System Hardware and Software Requirements

Aras Innovator Server

Hardware

For the Aras Innovator 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 32

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.

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

Starting with Aras Innovator 32 the following .NET 8.0 module must be installed:

- .NET Desktop Runtime

NX

NX Client Version 1926, Version 1953, Version 2206, Version 2212, Version 2306, Version 2312, and Version 2406 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 “licman21” please refer to the *Licman 2.1 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

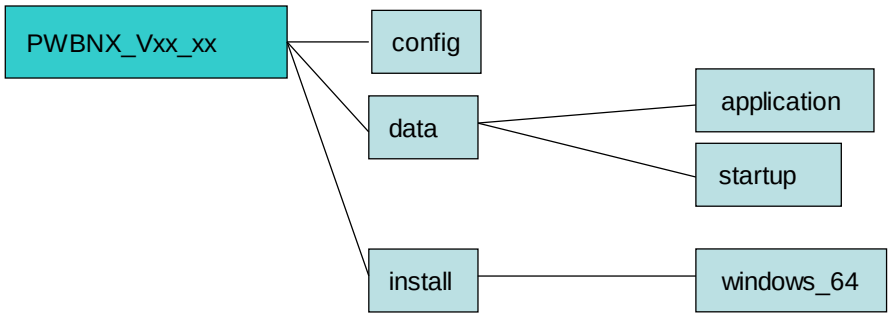
Use the Windows Explorer to locate the **D:\pwbnx\PWBNX_Vxx_xx.zip** file on the CD. Extract the content of the archive file to a temporary installation location.

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 PWBNX 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 **PWBNX_Vxx_xx\install\windows_64** of the installation package if you have installed the 64 Bit version of NX.

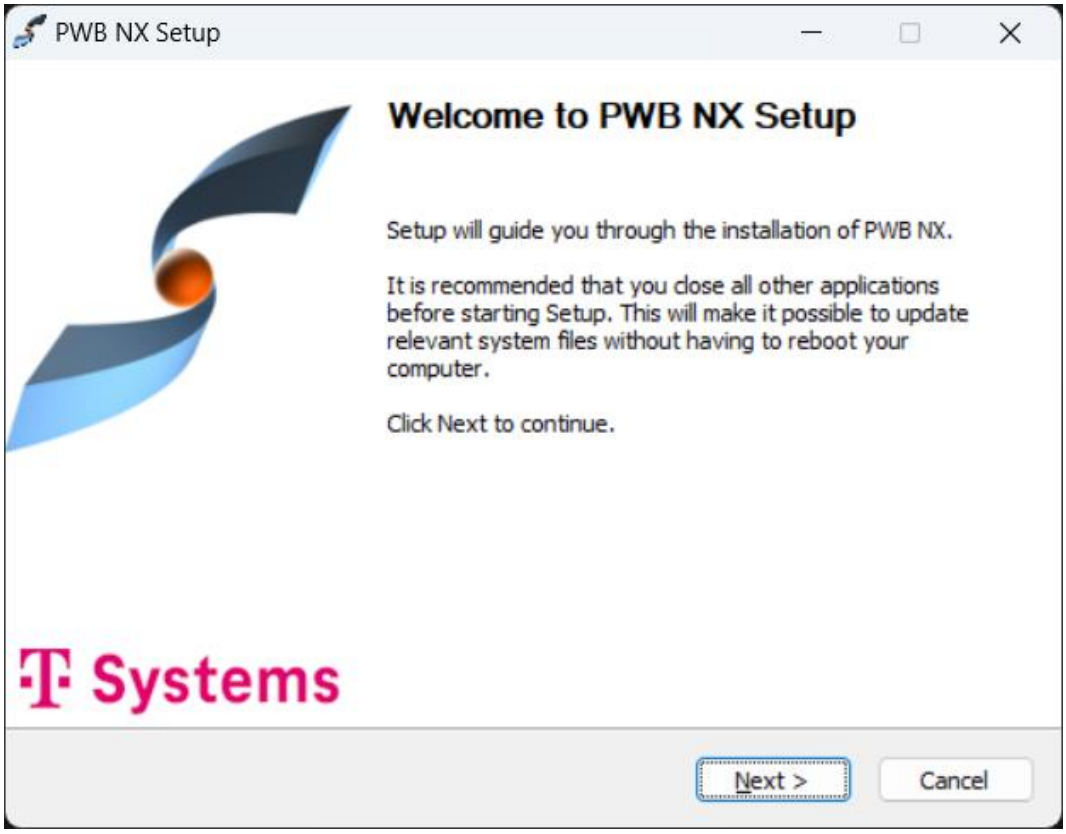
On **Windows Vista**/**Windows 7** the User Account Control (UAC) will be triggered and you will have to agree that the setup program may make changes to the computer. The installer is signed with a "**T-Systems International GmbH**" certificate to ensure its integrity and source.

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

The licman21 license manager has to be installed on the NX client host. For the installation of the license manager please refer to the *Licman 2.1 Installation Manual*.

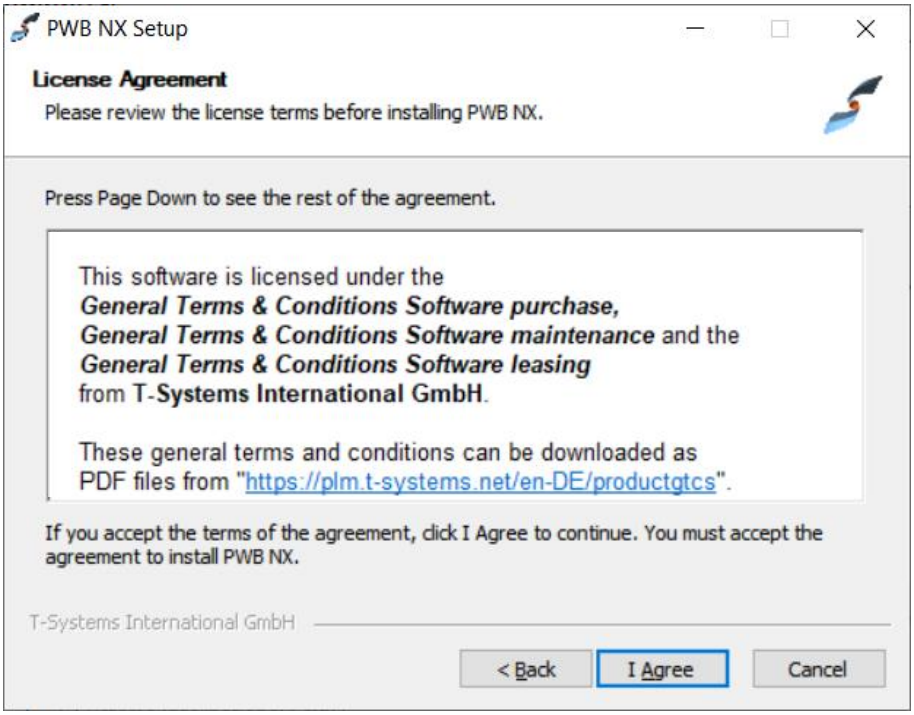
In the following the setup is shown step-by-step.

Installation process:



Picture 2: Welcome to the Installation

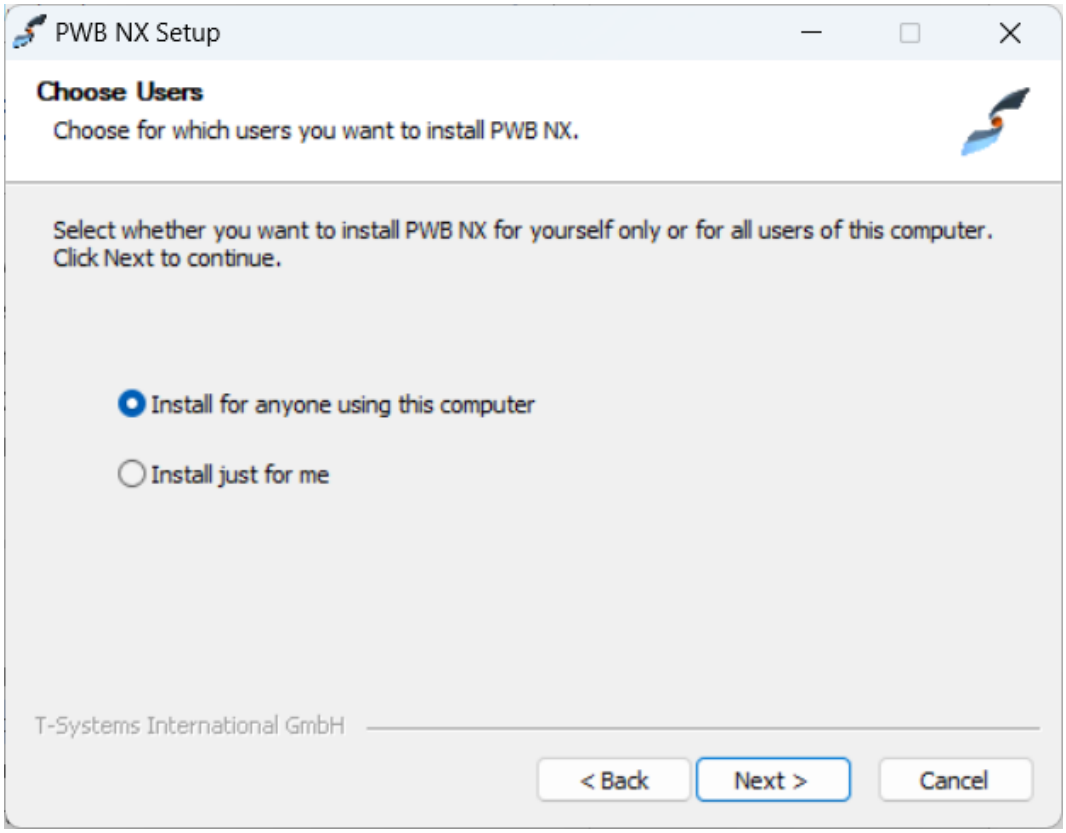
The installer software asks to approve the license terms (see *Picture 3: License Agreement*).



Picture 3: License Agreement

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

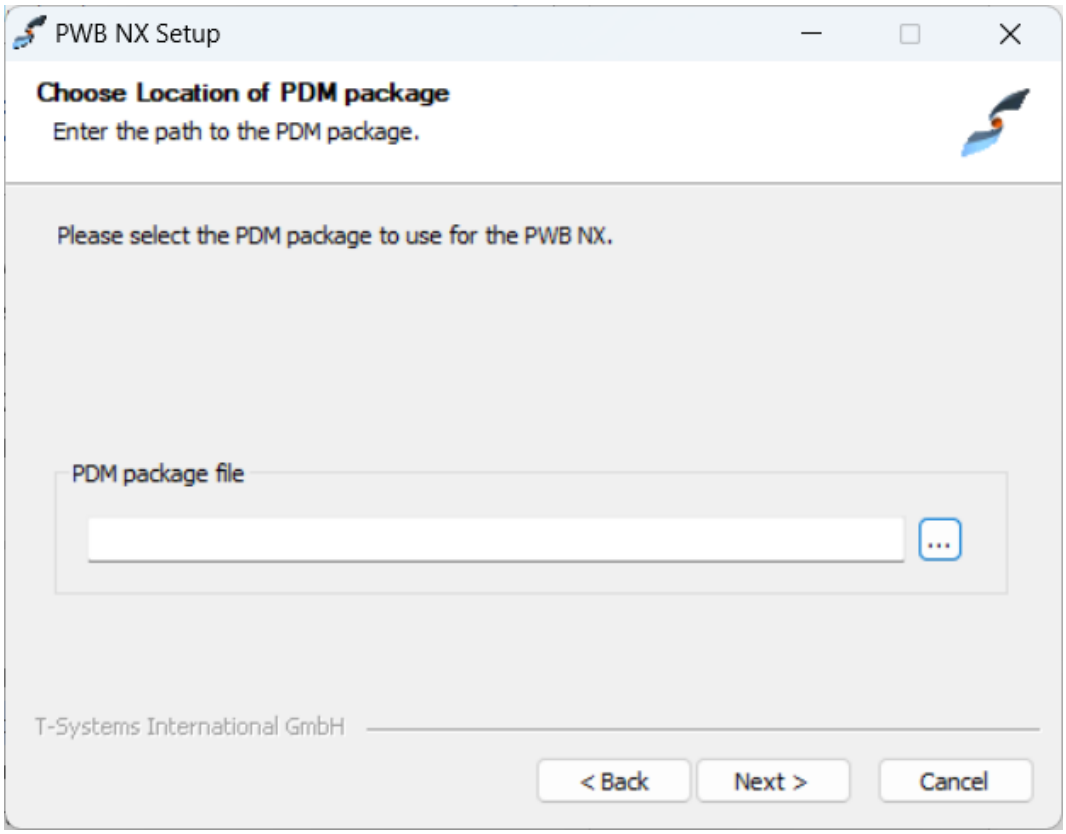
Next the installer will ask you to define the scope of the installation (see *Picture 4: Choose Users*). You can choose between an installation for anyone using the computer or just for the current user.



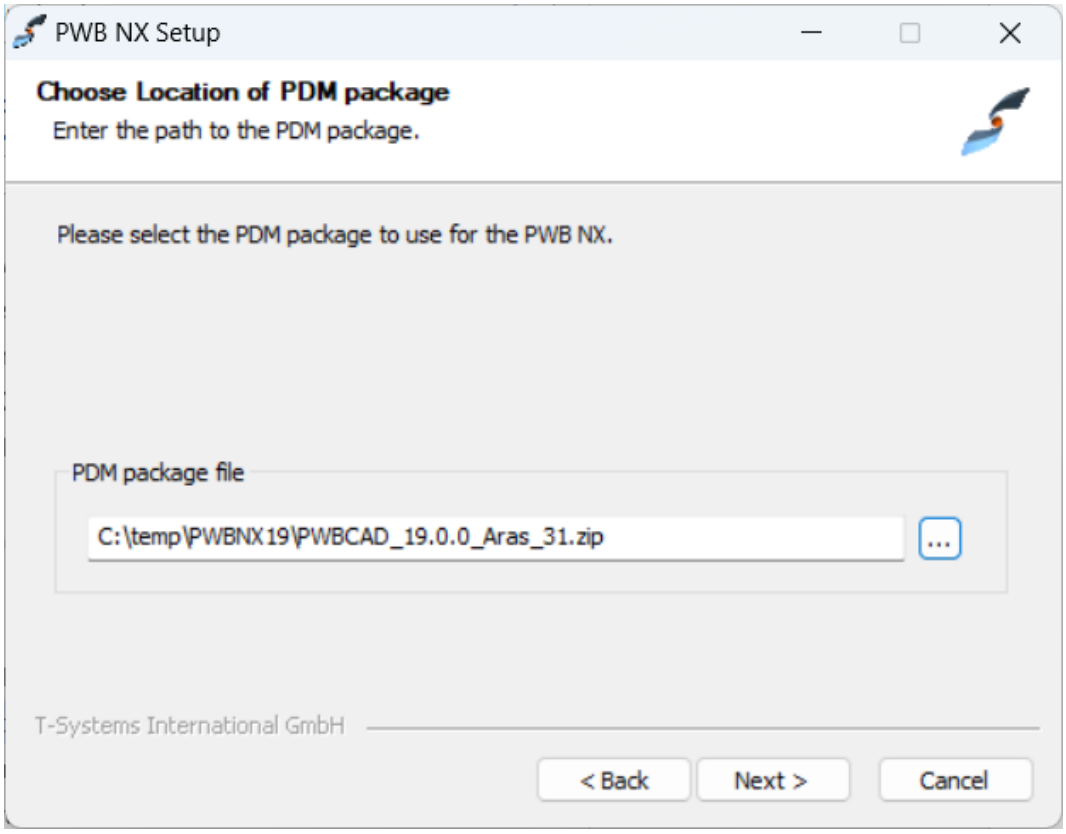
Picture 4: Choose Users

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)*).



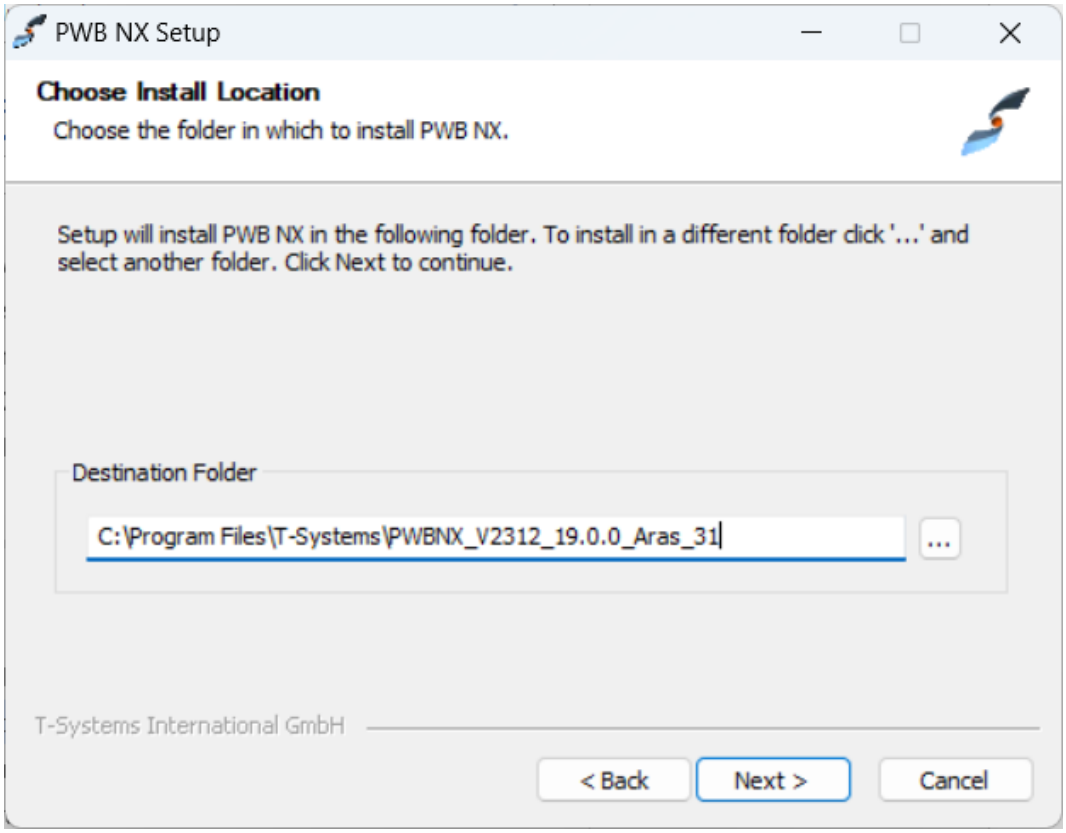
Picture 5: Choose Location of PDM package



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

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

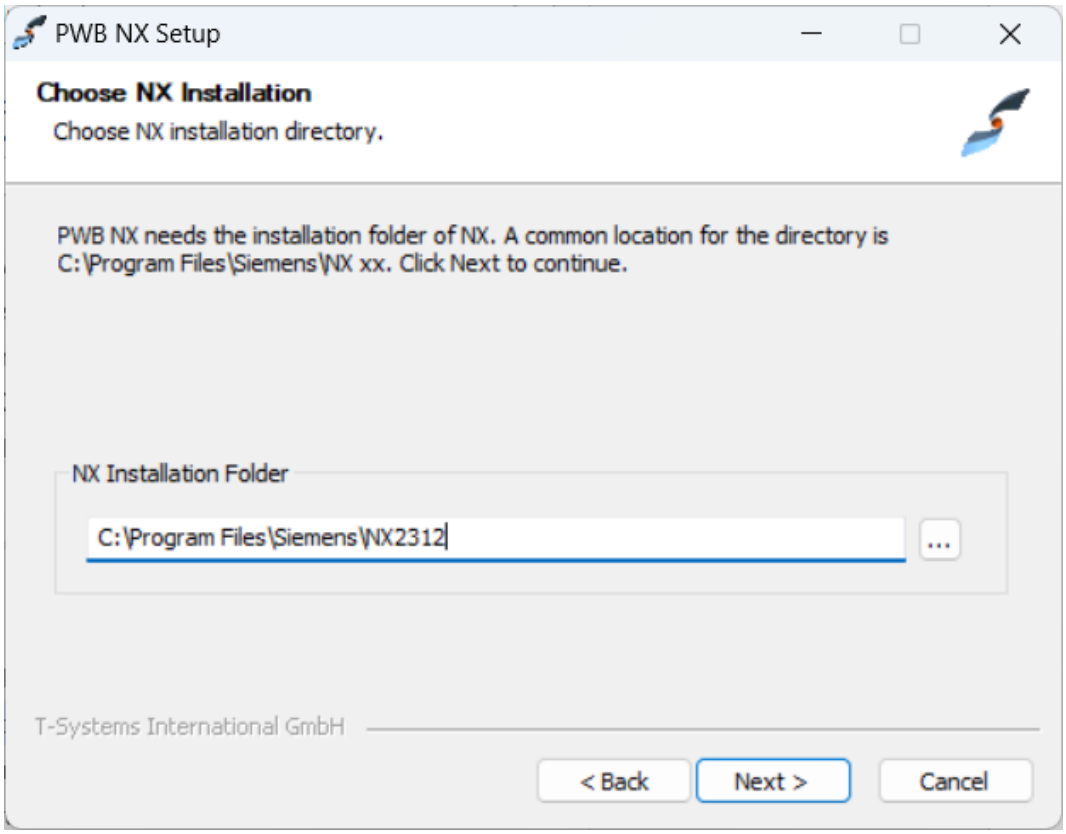
Next the installer will ask you for the target directory for the installation. You can use the given standard location or choose any other location (see *Picture 7: Choose Install Location*). The chosen folder must be empty or not existent.



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*).



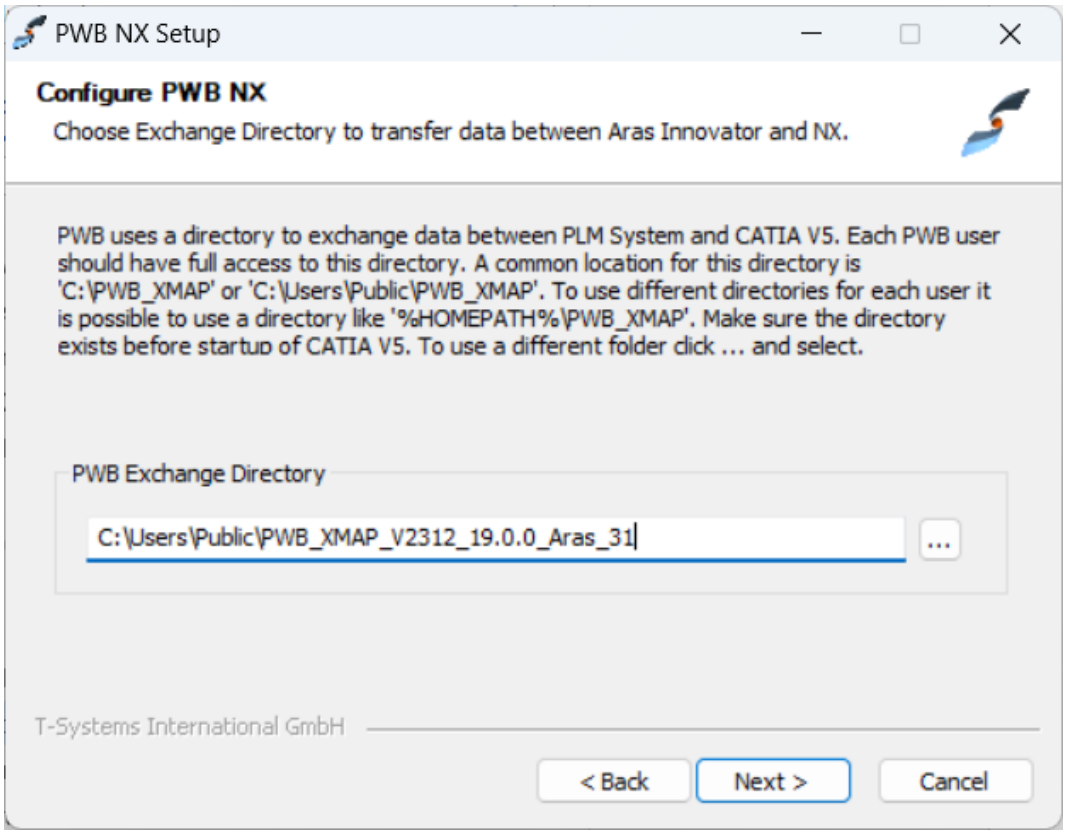
Picture 8: Choose NX Installation

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!

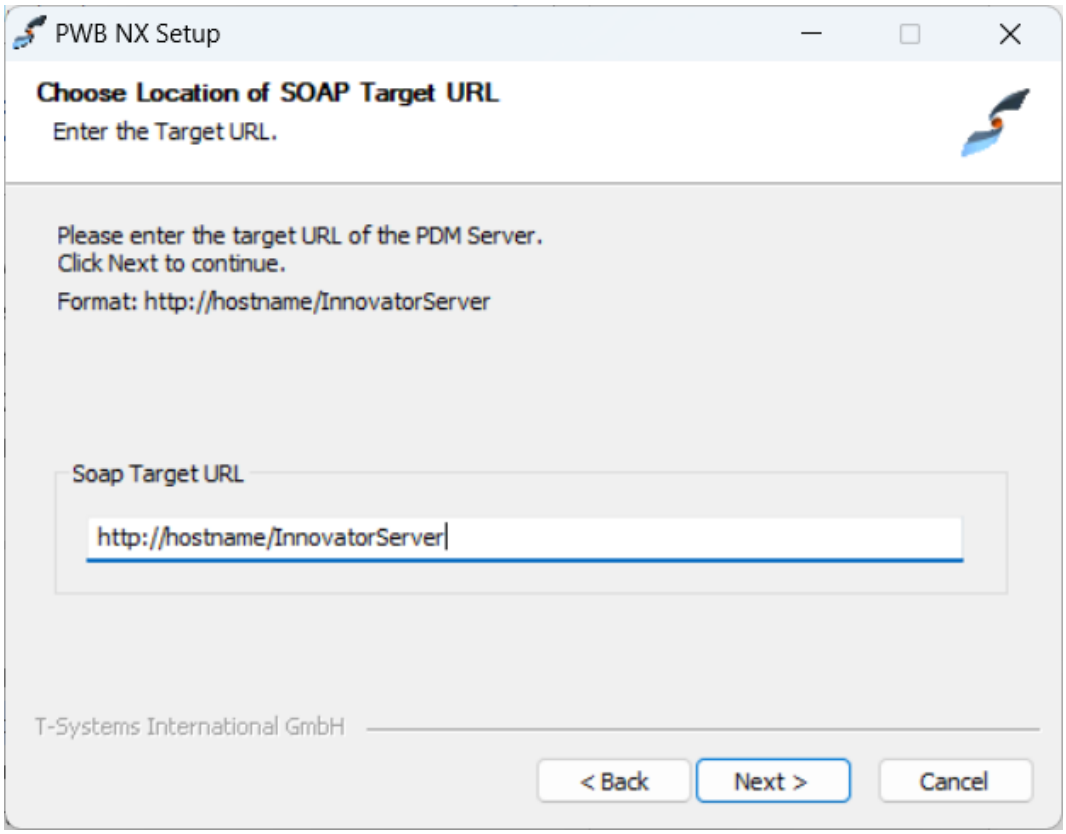


Picture 9: Choose Exchange Directory

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*).

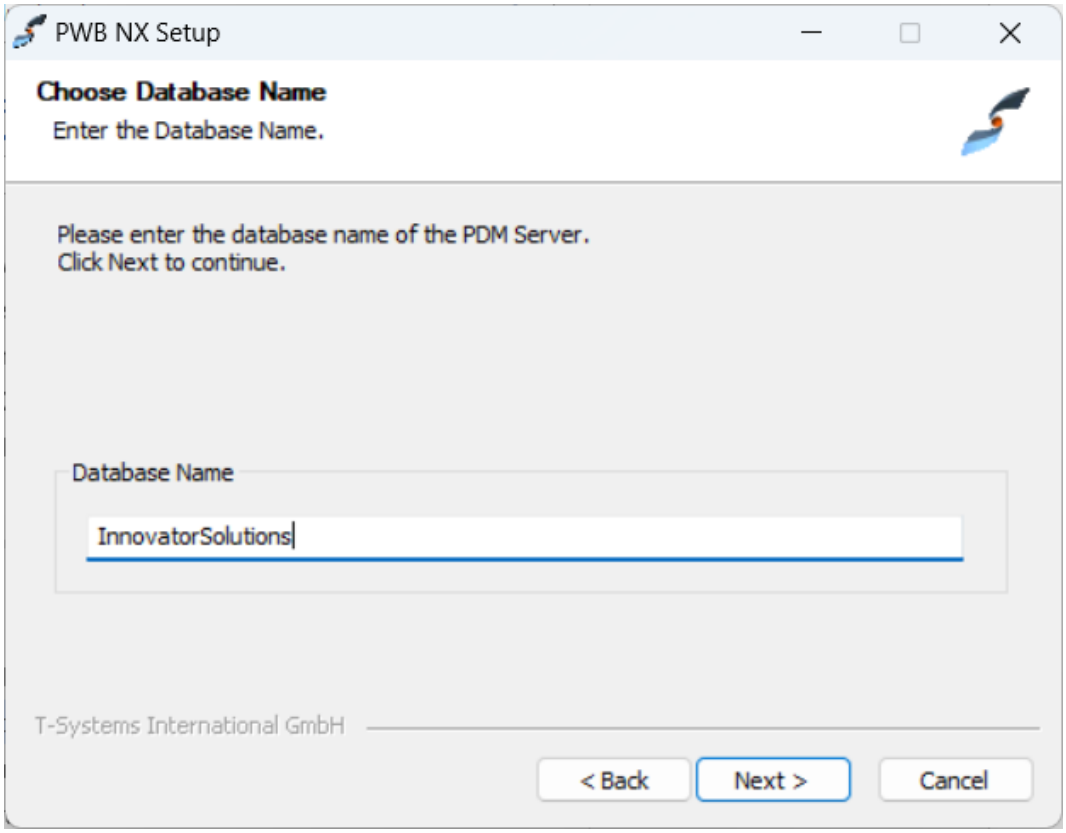
This URL defines the host on which the PDM Server is reachable.



Picture 10: Choose Location of SOAP Target URL

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

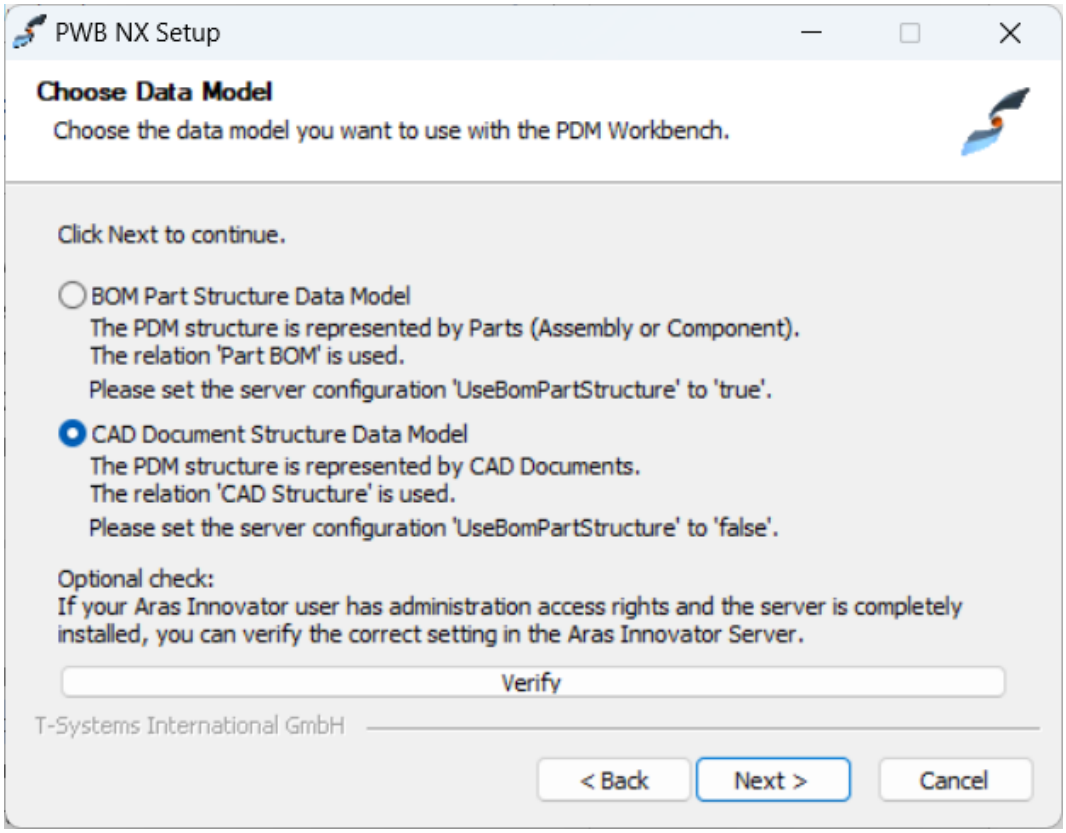
Now you have to add the Database Name (see *Picture 11: Choose Database name*).



Picture 11: Choose Database name

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

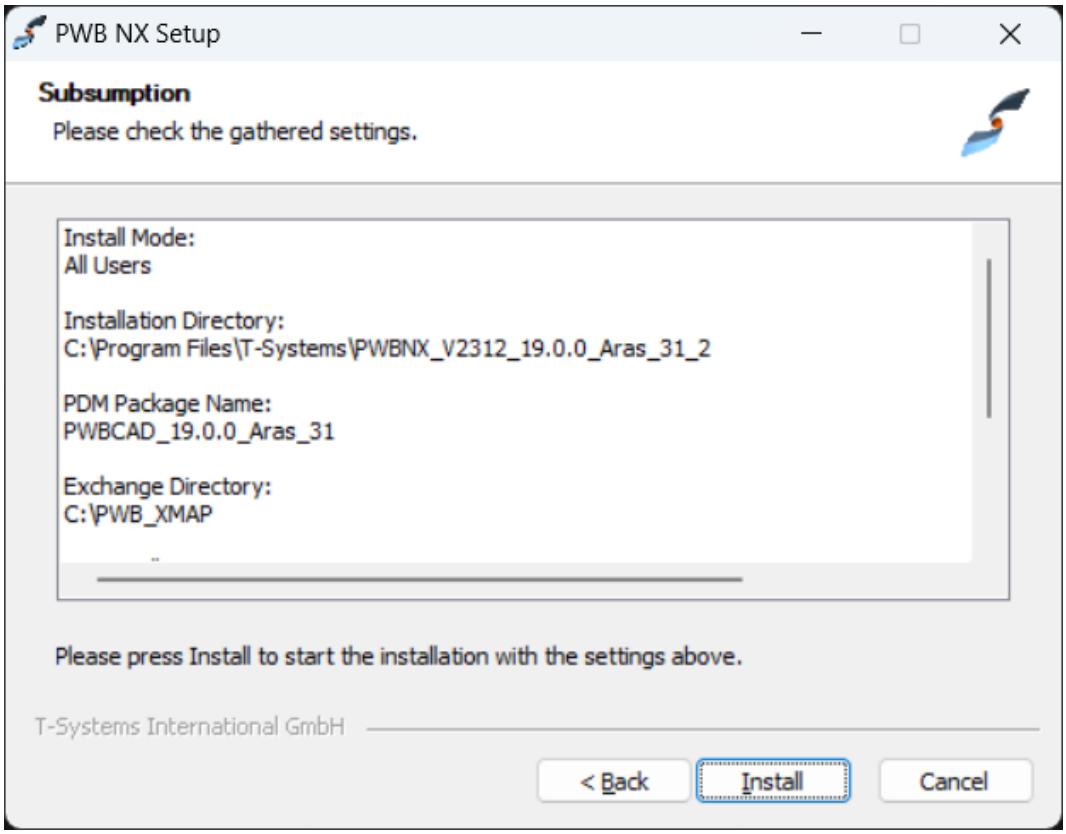
Finally, you have to choose the data model to be used by the PDM Workbench (see *Picture 12: Choose Data Model*).



Picture 12: Choose Data Model

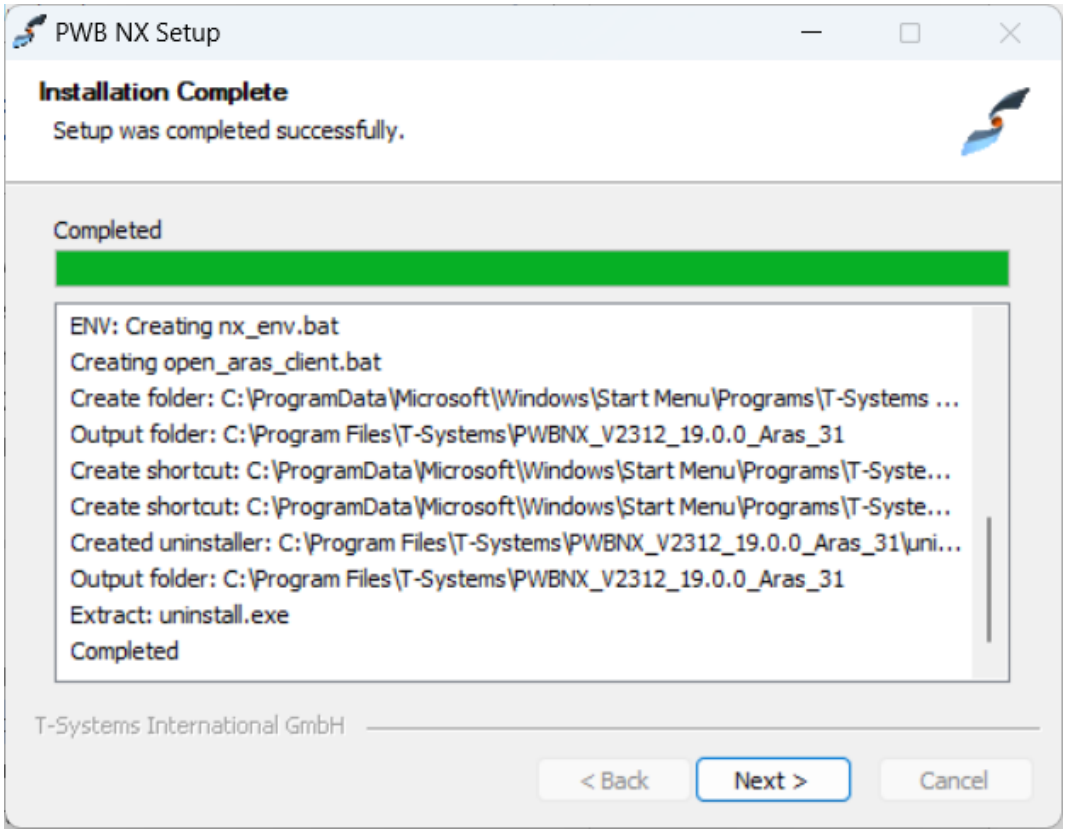
With the optional check you can directly log in into Aras Innovator with an administrative user and verify the setting of the data model (UseBomPartStructure) in the PWB Configuration object.

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

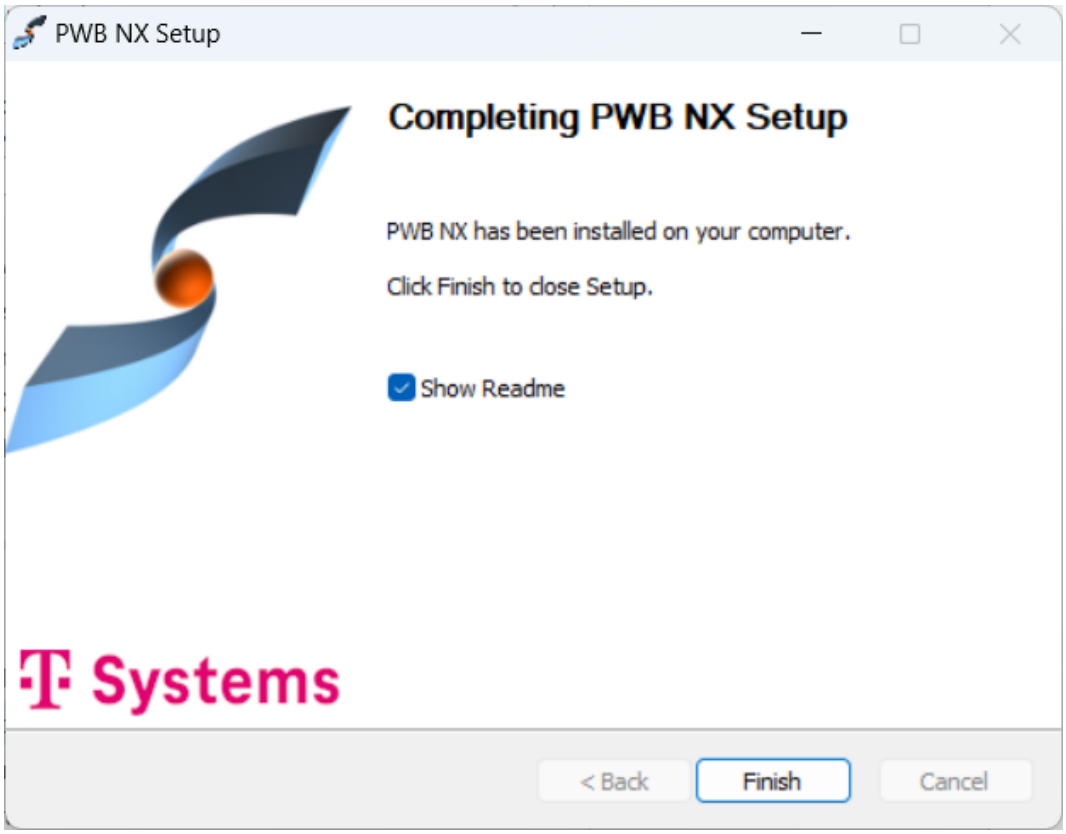


Picture 13: Subsumption

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



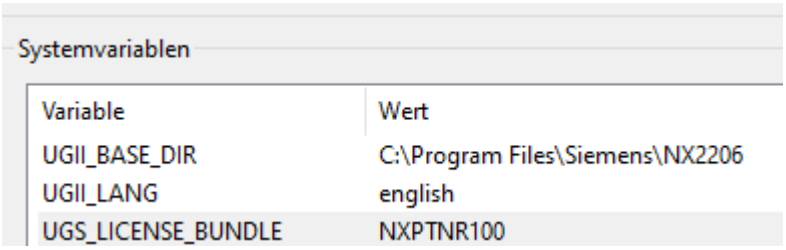
Picture 14: Installation progress



Picture 15: Installation finished

Required Windows Environment Variables for NX

Please check if the `UGS_LICENSE_BUNDLE` environment variable is set before start of the PDM Workbench NX session:



Picture 16: System variables – UGS_LICENSE_BUNDLE

If not, please add the valid NX license bundle.

Alternatively, the `UGS_LICENSE_BUNDLE` can also be set in the `nx_env.bat` file of the PDM Workbench NX client installation, e.g.: `SET UGS_LICENSE_BUNDLE=NXPTNR100`

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 un-installation please open a command line window as administrator.

Start the silent un-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

There are currently no required NX options.

License Manager Installation

The licman21 license manager has to be installed on the NX client host. For the installation of the license manager please refer to the *Licman 2.1 Installation Manual*.

(For software and manual download see: *https://plm.t-systems.net/en-DE/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 Schema file, which is placed in the config directory of the installation. The right server address must be defined within the "soapTargetUrl" XML element. The database must be defined within the "dataSource" XML element (<dataSource name="LoginDatabases" type="ValueList">) or no database is defined at all (then the possible databasese are received from the aras connection).
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 Aras Innovator thin client.

Testing the installation

For installation testing follow these steps:

1. Use: **Start→Programs→T-Systems PWB NX 1572 15.0.0 Innovator 1218→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 the 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, e.g. http://localhost/InnovatorServer
PWB_DATABASE_NAME	The name of the database, e.g. InnovatorSolutions
PWB_DEBUG	Set to “ON” to receive PWB debug output in the console.

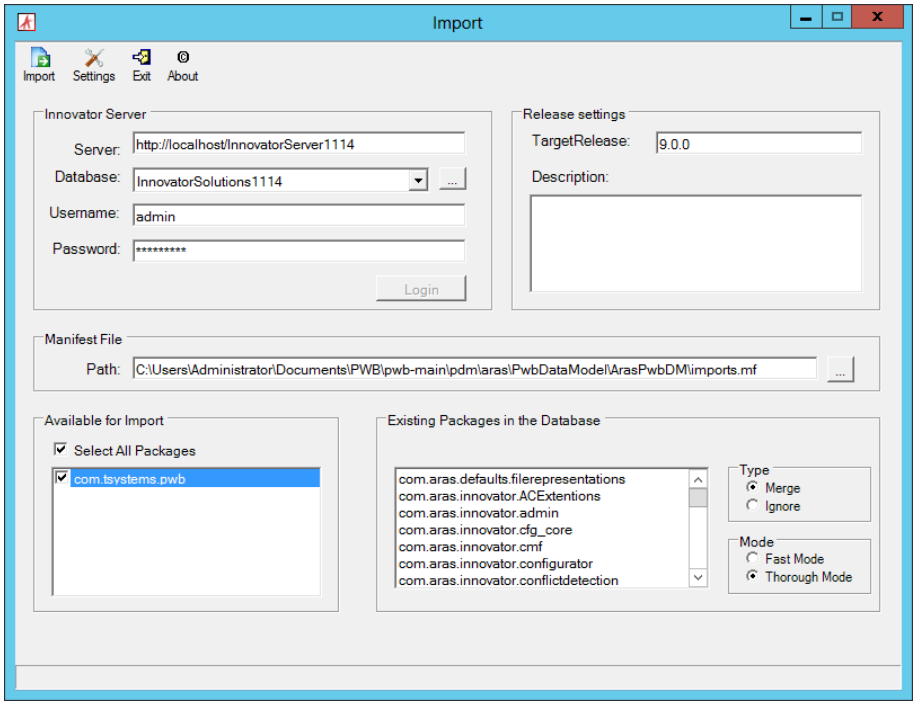
CHAPTER 3

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 at least five packages need to be imported to Aras Innovator with the Aras Innovator import utility[1]:



Picture 17: PDM Aras Innovator import utility

[1] The import utility has to be downloaded from the Aras homepage and to be installed. Link: 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 17: 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 17: 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="36">
<![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 $(pkgname)
{
...
```

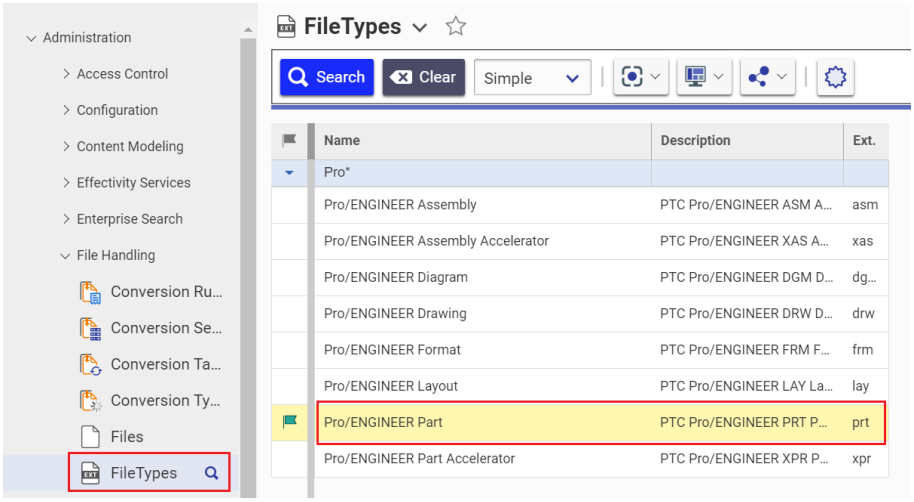
CHAPTER 5

Server Configuration

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

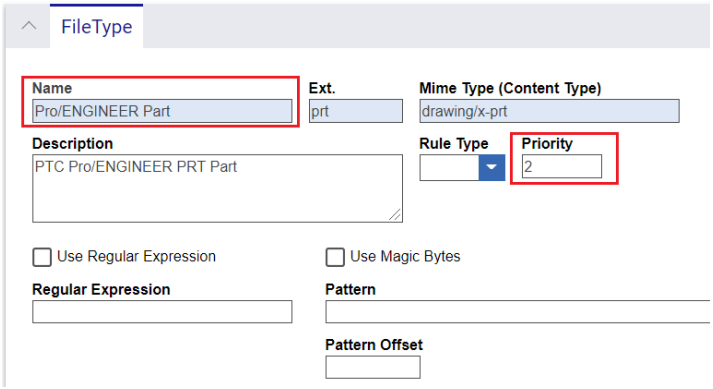
Priority of File Types

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:



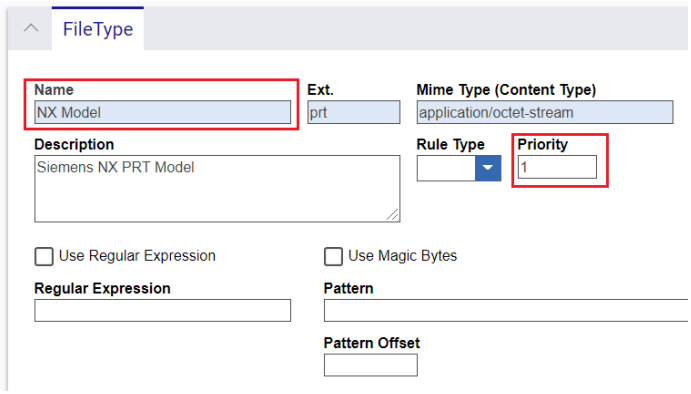
Picture 18: FileType “Pro/ENGINEER Part” with extension “prt”

➔



Picture 19: FileType “Pro/ENGINEER Part”

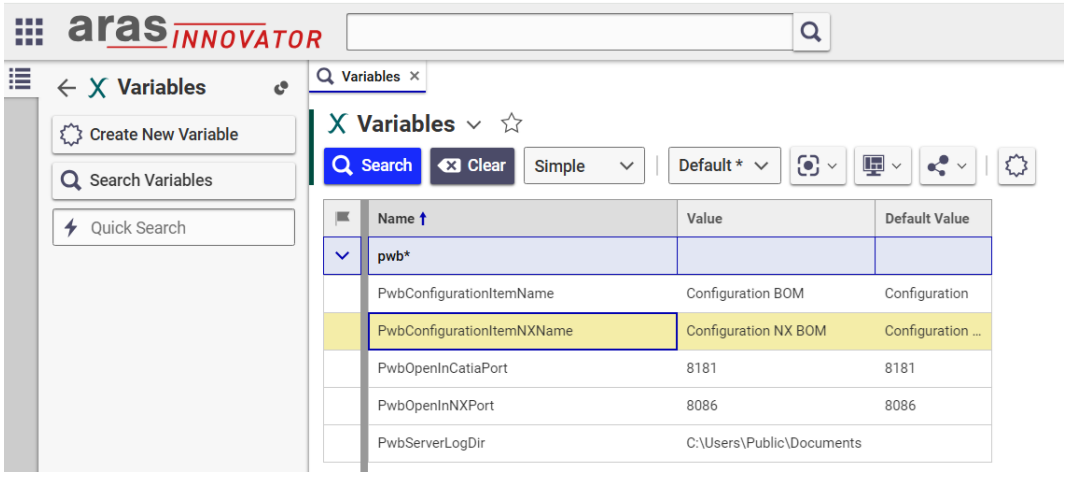
➔



Picture 20: FileType "NX Model"

Configuration Variables

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



Picture 21: Aras Innovator server configuration variables

- PwbConfigurationItemNXName

  The name of the PWB Configuration item which contains additional configuration information, like the attribute mapping configuration. Please see "Configuration Items" for more details.

  Within the schema file the corresponding server configuration (serverConfig) and the structure mode (UseBomPartStructure) may be set. Then the server configuration is checked during the PWB NX login.

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

<PWBSchemata>
    <PWBSchema system="Aras" customization="Aras"
               serverConfig="Configuration NX"
               UseBomPartStructure = "false"
               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.

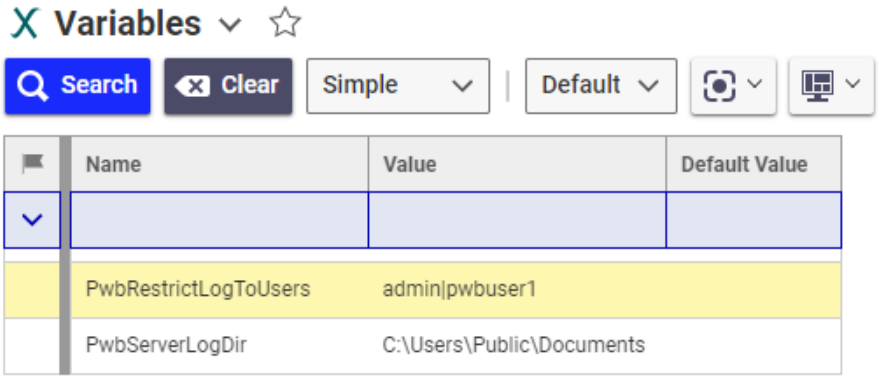
PWB Logging on Server

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

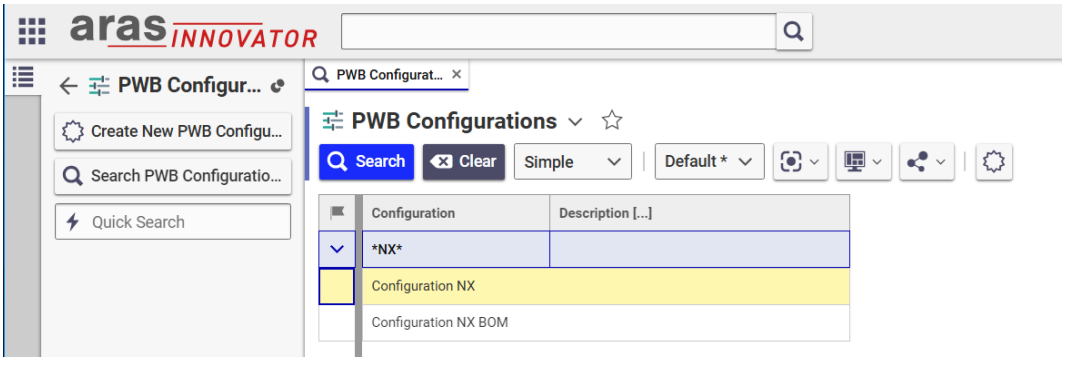
To enable general logging, set the Aras Innovator variable “PwbServerLogDir” to the log directory. To restrict the logging to a specific set of users, set the Aras Innovator variable “PwbRestrictLogToUsers” to a list of names (separated by ‘|’).



Picture 22: 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 PWB Configuration item (see *Picture 23: PWB Configuration item in Aras Innovator*) has to be used:



Picture 23: PWB Configuration item in Aras Innovator

CHAPTER 6

Configurations for specific Functionalities

This chapter describes the configuration of the PDM Workbench NX for specific functionalities. This configuration is done in the PDM Workbench NX Schema file.

Standard Configuration

Exchange Map

In the 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.

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 24: Sample UseBomPartStructure configuration

Default value: “false”

Optional. Possible values: “true”, or “false”.

BOM Part Structure Data Model

In the BOM Part Structure Data Model the PDM structure is represented by Parts. The relation “Part BOM” is used.

Each Part is described by a CAD Document which includes the NX file.

CAD Document Structure 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 25: 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 26: Sample MaxQueryResults configuration

Optional.

Update Configuration

“Create” Dialogs

The PWB Configuration item setting “ShowCreateDialogsDuringUpdate” has to be set to “true” in order to show the “Create” dialogs during the update process.

ShowCreateDialogsDuringUpdate	true

Picture 27: Sample ShowCreateDialogsDuringUpdate configuration

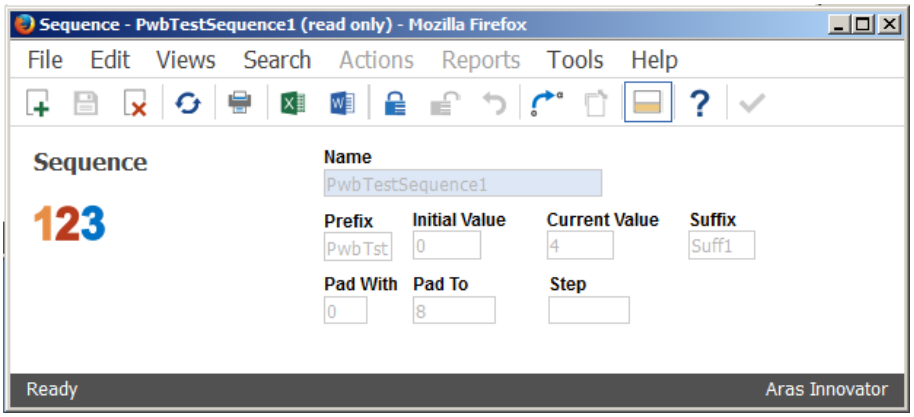
Default value: “true”

Optional. Possible values: “true”, or “false”.

Naming Configuration - Numbering

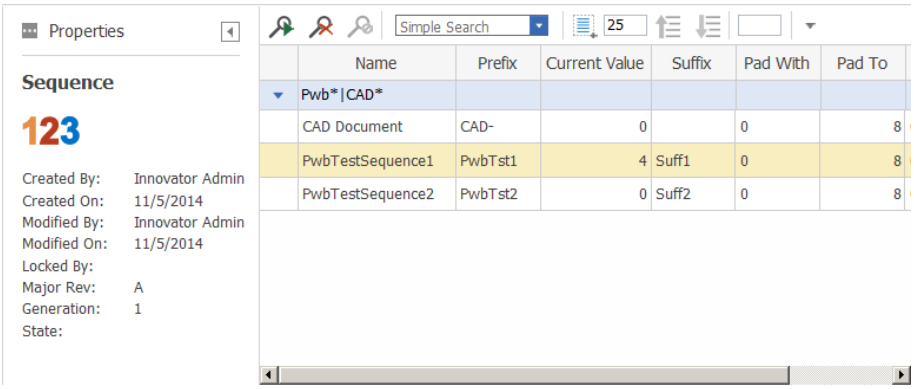
Autoname Support using Aras Innovator Sequence Items

First Sequence items which should be used for the “Autoname” functionality need to be created:



Picture 28: Sample Sequence item

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



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 PWB Configuration item 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 Schema file.

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 Schema file (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:

```
// CCO.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;
```

Standard Part Functionality

StandardPartAdmin

The PWB Configuration item setting “StandardPartAdmin” defines the “Standard Part Administrator” identity.

Only a standard part administrator can create or modify standard parts. A standard part administrator is defined as every identity which belongs to the identity which is set as the setting “StandardPartAdmin”. By default, it is the identity “Standard Part Administrator”, which is added with the PDM Workbench installation:

StandardPartAdmin	Standard Part Administrator

Picture 31: Sample StandardPartAdmin configuration

By default, the “Innovator Admin” identity is a standard part administrator.

Standard Part Functionality for BOM Part Structure Data Model

Standard parts might be defined by the standard part administrator.

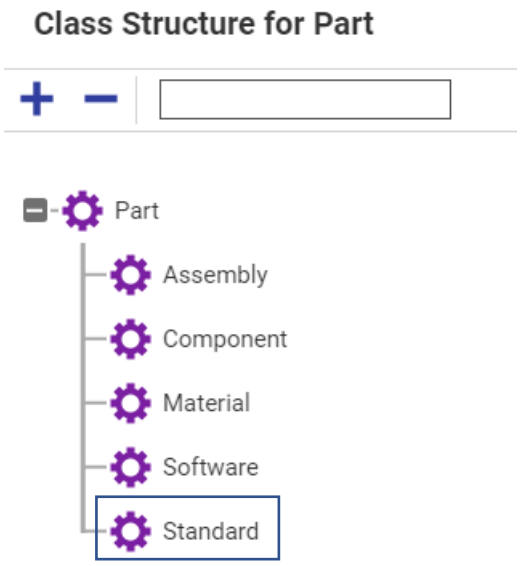
For standard parts the *is_standard* attribute of the CAD document item is set to *true* then and the related part item has the classification */Part/Standard*.

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 be 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 32: 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">

   ...

</object>
```

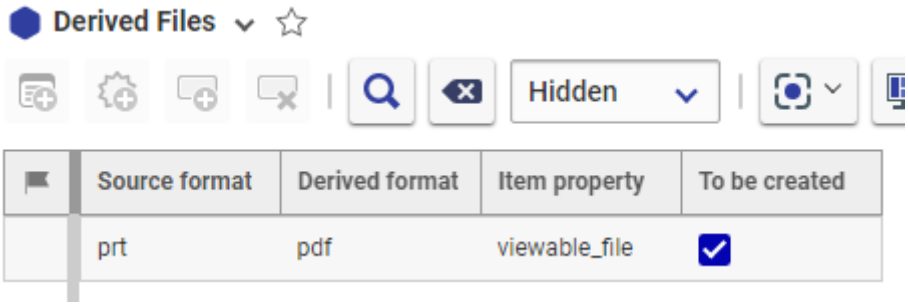
Derived Files Configuration

Derived viewable Files

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

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 33: derived files configuration

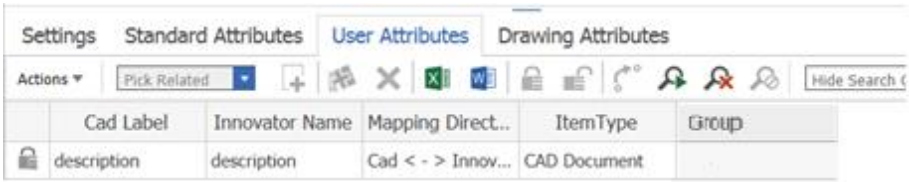
Attribute Mapping Configuration

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 Innovator. Other user defined attributes (applied to a component, reference set, …) cannot be mapped.

To define the attribute mapping, open the PWB 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 34: 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 Configuration item 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
t*						
test54	A	test54	Mechanical/Part	Preliminary	test54.prt	DE
test55	A	test55	Mechanical/Part	Preliminary	test55.prt	BR

Picture 35: 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 36: Define Group Attribute Mapping – Group

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

GroupAttributeNameForUserAttributes	country

Picture 37: 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 PDM Workbench, the values have to be configured.

Then for the CAD Document “Test54” the CAD attribute “sizeDE” is mapped to the Aras 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 Aras Innovator attribute “size”.

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

Versioning Configuration

Only one new Generation of a CAD Document per “Claim” Action

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 38: Sample ClaimedIsNewGenAttr configuration

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.

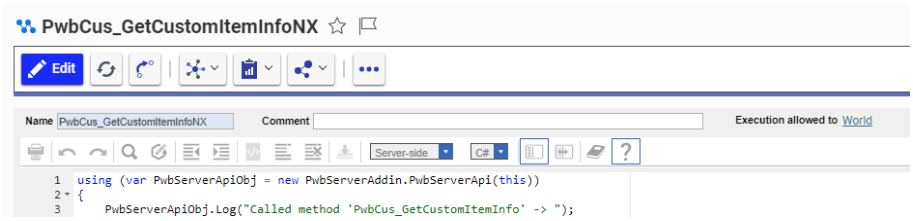
CAD Document generations which are read-only, for example because they are released or frozen can be claimed if they are current. In this case the new generation will be created by the claim process, and the first update will not create another new generation.

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 item setting “CustomMethod_GetCustomItemInfo”:

CustomMethod_GetCustomItemInfo	PwbCus_GetCustomItemInfoNX

Picture 39: Sample CustomMethod_GetCustomItemInfo configuration



Picture 40: 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());
```

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.

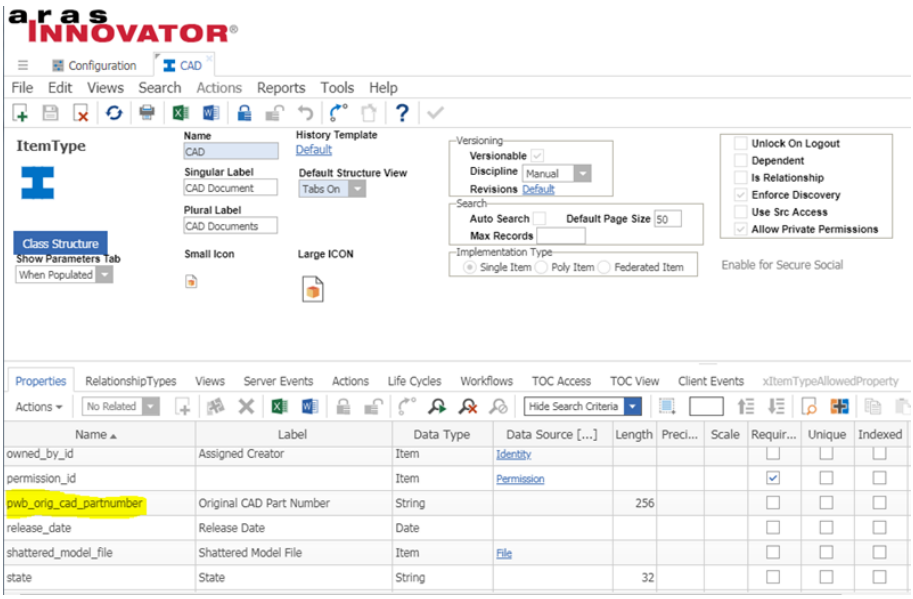
If you want to use a different name for the method, you have to set the name in the PWB Configuration setting “ReconnectAtUpdateMethod”:

ReconnectAtUpdateMethod	<Your Method name>

Picture 41: Sample ReconnectAtUpdateMethod configuration

If you want to use the original NX Part 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 NX Part Number:



Picture 42: 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 43: Sample OrigCadPartnumberAttr configuration

If this property is set, the original NX 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("model7");
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;
```

"Open in NX" from the Aras Innovator Client

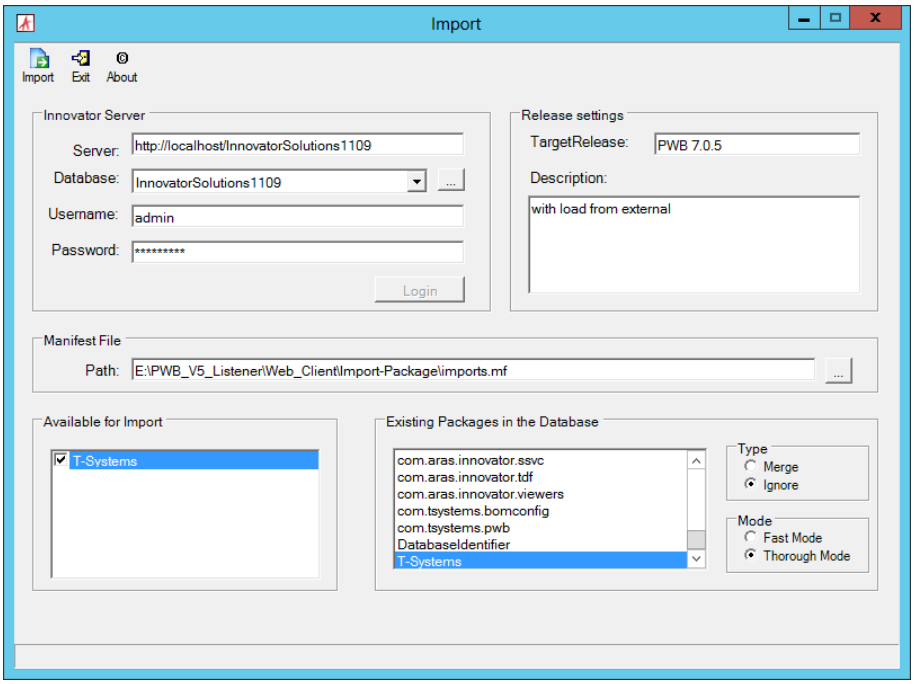
Single CAD or Part items, or structures, can be loaded in NX from the Aras Innovator web client.

The precondition for this functionality is that NX is started with PDM Workbench, and the user is logged in.

Configuration in Aras Innovator

Import the following package to the Aras Innovator environment with the Aras Innovator Package Import Utility:

“PwbDataModel_[XX.X]\Specific\OpenInCAD\OpenInNX\imports.mf” -
The "Open in NX" functionality.



Picture 44: Import utility

In Aras Innovator the variable “PwbOpenInNXPort” has to be set to a specific port, for example “8086”:

	Name	Value	Default Value
	PwbOpenInNX*		
	PwbOpenInNXPort	8086	8086

Picture 45: Innovator variable “PwbOpenInNXPort”

“Open in Aras” from NX Client

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

Configuration

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 Innovator code tree.

Modify the file (Aras 14 to Aras 23)
`\Innovator\Client\Modules\aras.innovator.core.MainWindow\deepLinking.ts` in your Aras Innovator code tree.

Modify the file (up to Aras 12) `\Innovator\Client\Modules\aras.innovator.core.MainWindow\deepLinking.js` in your Aras Innovator 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 46: File "deepLinking.ts or .js"

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

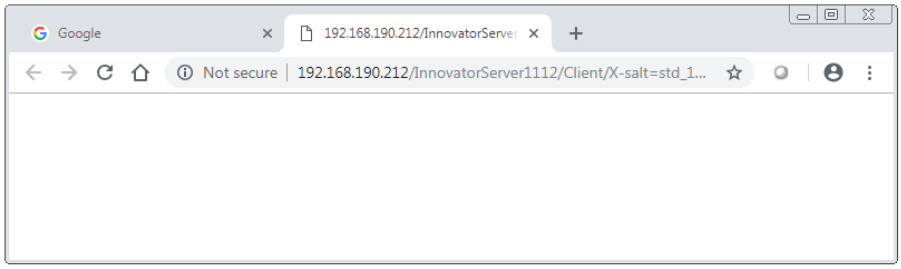
At all clients the browser cache of the default browser must be cleared to enforce the use of the changed script.

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

For later Aras Innovator version this behaviour may be changed.

For Aras Innovator 11 only

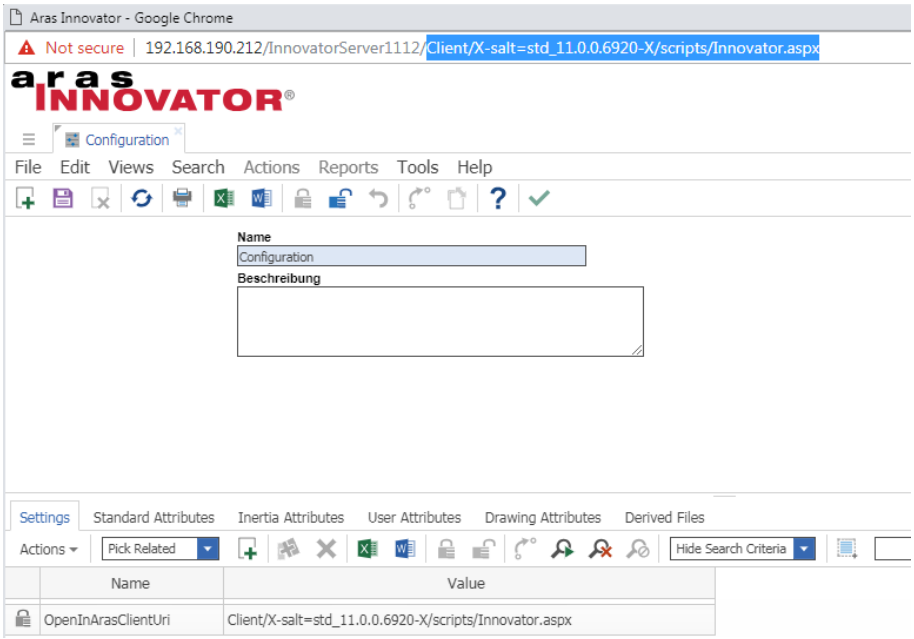
With Aras Innovator 11 the "Open in Aras" functionality opens a new empty tab in an additional default browser window.



Picture 47: Empty tab

To avoid this window, you must set the PWB Configuration setting "OpenInArasClientUri" to the client part of your Aras Innovator 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



Picture 48: Aras Innovator web client with Client URL

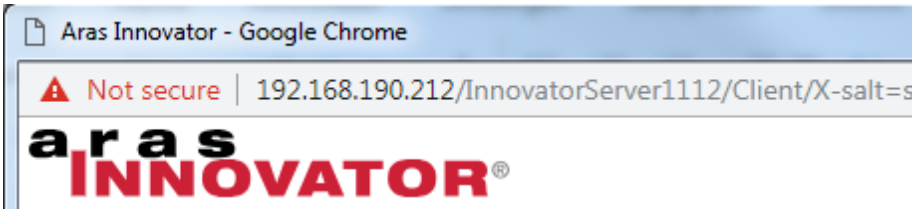
This setting must not exist in Aras Innovator 12.

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 49: Window title “Aras Innovator”

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

ArasWindowTitleSubString	Aras Innovator

Picture 50: Sample ArasWindowTitleSubString configuration

“Open in Aras” – Allow configurable Web Browser

By default, the “Open in Aras” functionality uses the default web browser. It is also possible to configure a different browser to open the selected object in Aras Innovator.

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

Configuration

To configure a web-browser the following environment variable must be set on the NX client.

```
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
```

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

Configuration

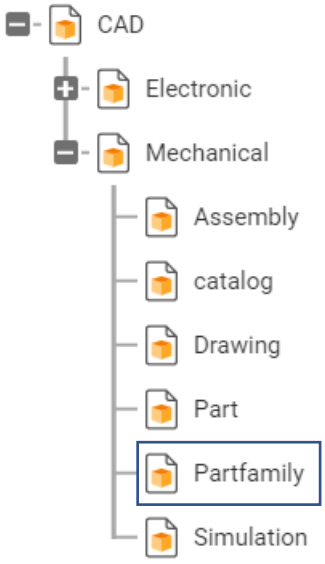
The PWB Configuration setting "UseCadIsMaster" has to be defined and set to "true" to enable the new behaviour.

Family Part Handling

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

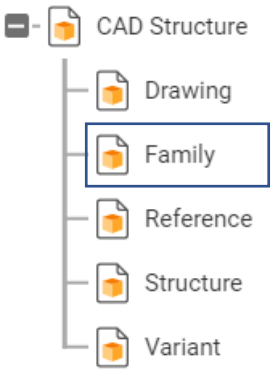
First the classifications for item type "CAD" and "CAD Structure" need to be adapted.

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



Picture 51: CAD item extension - /CAD/Mechanical/Partfamily

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



Picture 52: CAD Structure item extension - /CAD Structure/Family

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">

...

</object>
```

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 an 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.
- “serverConfig” Contains the name of the corresponding configuration on the server.
- “UseBomPartStructure” Contains the suitable structure mode defined on the server.
- “dynDlgServerMethod” Contains the main method on the server for the dynamic dialog definition.

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.

Configure in the Schema file:

```
...
<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 **Fehler! Textmarke nicht definiert.**
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[2] characters when creating a file.

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

- “isPartNumber” Optional.
- “autoName” Optional.
- “isDerived” Optional.

Example:

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

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

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

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

The *form* attribute definitions contain the following attributes:

- “name” Mandatory, must correspond to the PDM attribute's name.
- “displayName” Optional. 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**.

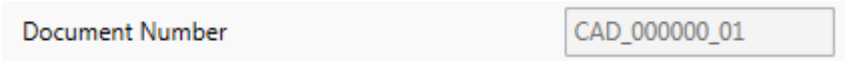
Description of the Widget Types

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.

You can find the detailed explanation of Data Sources in chapter **Data Sources**.

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

Picture 53: Single Line Editor Widget, update mode

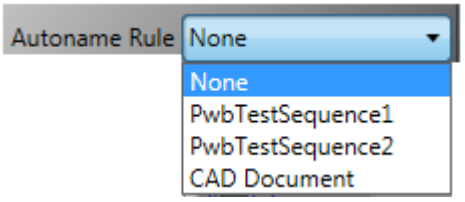


Picture 54: Single Line Editor Widget, output mode

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”.

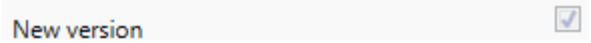
Picture 55: Multi Line Editor Widget, update mode

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 56: 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 57: Single Check Box Widget, select mode

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

Picture 58: Password Box Widget

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

Picture 59: 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>
```

Get databases from Server

During the installation you can configure the login database which is shown in the login dialog. It is possible to add more than one database value. It is also possible to remove all database values. Then all possible databases are received from the aras connection and are shown in the PDM login dialog.

```
<dataSource name="LoginDatabases" type="ValueList"
additionalValues="DefaultDatabase" >
      <value name="InnovatorSolutions" displayName=""/>
</dataSource>
```

Customizing PDM Workbench NX Menu

Adjust PDM Workbench default context actions

For every item context menus are available within the query dialog and the NX assembly navigator.

These context menues are customized within the schema file.

The context actions in the Schema file are predefined for the CAD mode and the BOM mode and there for the following item types: /CAD/Mechanical/Part, /CAD/Mechanical/Partfamily, /Part/Assembly, /Part/Standard.

Under each type there are the contextAction defined e.g. for the CAD object like the following

```
<object name="/CAD/Mechanical/Part" displayName="Model" icon="NXPart">

<contextAction name="Expand" displayName="Expand" usedIn="QueryDialog"  />
<contextAction name="Claim" displayName="Claim"
usedIn="QueryDialog|AssemblyDialog" disable="true" />

…
```

If a context action should not be visible for a specific object type, the context action can be commented out.

Each context action has the possible attributes:

displayName: The display name of the action in the context menu can be adjusted here.

usedIn: It can be adjusted where this context actions is visible, in the Query dialog or in the Assembly Navigator or both. If the context actions shouldn't be showed at all, remove or out comment the context action completely.

disable: this value defines if the context actions is visible when the actions is not possible in the context. If the value is false the context actions is still in the menu, but inactive (gray). It the value is true, the context action is not shown at all if the actions it not possible right now.

imgUrl: if you want to define another Image for the context actions in the Query dialog you can define the Image name here. It must be a valid Image which lays in the installation folder under /application. This is mainly used for the own custom context actions which don't have a image by default (see section below).

defaultAction: in the CAD Mode there is for the contextAction "Load Structure" also the attribute defaultAction which is used if a double click is done on the item. The default value is "LoadAsSaved", but is also possible to configure the default actions for the Load Structure with the values "LoadReleased" and "LoadCurrent".

Adjust and create own custom context actions

It is also possible to add own dynamically customized context actions being implemented on the server side. These actions may be added to the context menues of an item within the Query dialog or the Assembly Navigator.

To be able to use custom context actions first of all a main method for the dynamic dialog creation must be declared as dynDlgServerMethod attribute of the PWBSchema tag (see Attributes of the tag "PWBSchema":Attributes of the tag "PWBSchema":).

An example is the Promote action which is used from the PDM Workbench in CAD structure data model mode by default.

Custom context actions may be defined just as the normal context actions for every item type: /CAD/Mechanical/Part, /CAD/Mechanical/Partfamily, /Part/Assembly, /Part/Standard.

```
<customContextAction name="Pwb_Sample_DynContextPromote"
usedIn="QueryDialog|AssemblyDialog" confirm="false"
dialog="Pwb_Sample_ContextPromoteDlg" displayName="Promote to state"
button="PDM_WORKBENCH_DYN_DIALOG_PROMOTE" imgUrl="PWB_Ctx_Promote.bmp"/>
```

Here you have to set the following attributes:

name: The name of the customized server method being called after confirming the related dialog with OK.

dialog: The server method defining the dialog.

confirm: This parameter determines whether an additional user confirmation is required to start this context action.

button: For each action we need to define a button within the menu.men, which is lying in the startup directory of the installation folder. The button name needs to be referred here.

displayName: The display name of the action in the context menu can be adjusted here.

imgUrl: If you want to define another Image for the context actions in the Query dialog you can define the Image name here. It must be a valid image which lays in the installation folder under /application. This is mainly used for the own custom context actions which don't have a image by default (see section below).

For more detailed information about the implementation of own custom methods see PWB-Aras_ServerAddin_API_Documentation.docx.

Adjust standard NX actions

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 need 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_Tlb_Query.bmp
SENSITIVITY OFF
ACTIONS BL_query
```

with

```
BUTTON UG_FILE_OPEN
LABEL Query
TOOLBAR_LABEL Query
MESSAGE Queries the PDM-System.
BITMAP PWB_Tlb_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 are configured in the PDM 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 conditions 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.

Multiple 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 Schema file the setting “checkAuthoringToolVersion” has to be configured to prevent an update of NX files which are created with an older NX release..

If this setting is set to true `<checkAuthoringToolVersion value="true" />`, during an update there is a warning before files of an older NX release are overwritten.

The default value of this setting is “false”.

CHAPTER 8

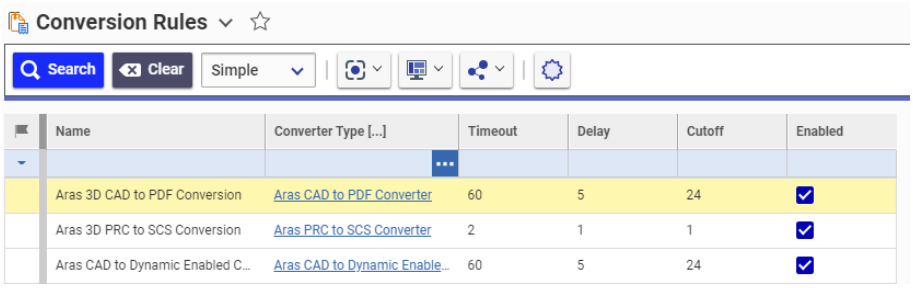
Troubleshooting

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,



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 NX fails when the Conversion Server is installed.

In some cases locking CAD documents in NX 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
```

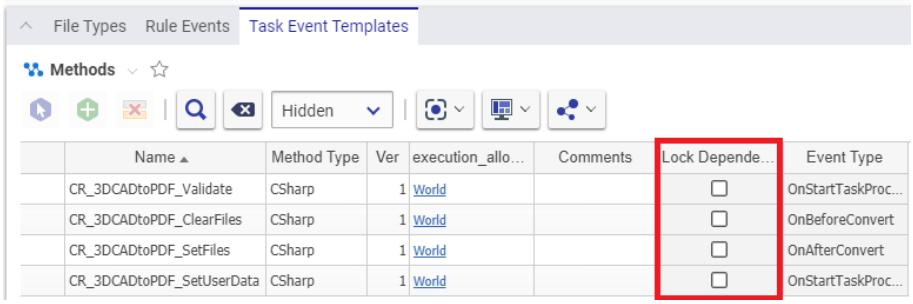
If that happens please create the server setting “EnableFileCloneCreation” and set it to the value “false”. That should fix the issue.

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 60: “Aras 3D CAD to PDF Conversion” Rule

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



Picture 61: “Uncheck Lock Dependencies Aras 3D CAD to PDF Conversion” Rule