PDM Workbench

PDM Workbench Release 13.0.0 for Aras Innovator

Release Notes

Version 2

Copyright

Contact

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

https://plm.t-systems-service.com/en/pdm-workbench

☎ +49 (0)40 30600 5544
✉ +49 (0)3915 80125688

Email: *cmi_support@t-systems.com*

Trademarks

Preface

About this Document

This document provides general release information for the PDM Workbench.

Related Documents

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

Manual Title	Version
PDM Workbench Installation & Administration Manual	13.0
PDM Workbench User Manual	13.0

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

Email: *cmi_support@t-systems.com*

Table of Contents

Table of Figures

CHAPTER 1

Platform Support

Supported Platforms

CATIA V5 Version V5-6R2016, V5-6R2017, V5-6R2018, V5-6R2019, and V5-6R2020

(other revisions on request):

CATIA V5 Client V5-6R2016	Windows 7 (64Bit)
CATIA V5 Client V5-6R2017	Windows 7 (64Bit), Windows 10 (64Bit)
CATIA V5 Client V5-6R2018	Windows 7 (64Bit), Windows 10 (64Bit)
CATIA V5 Client V5-6R2019	Windows 10 (64Bit)
CATIA V5 Client V5-6R2020	Windows 10 (64Bit)

Important notice:

CATIA V5-6R2014 SP2 has been retracted by Dassault Systèmes and is not supported. Please use SP3 instead.

Server Installation of Aras Innovator 12:

(Aras Innovator 11 and other service packs on request)

Aras Innovator Server 12	Windows Server 2012, Windows Server 2016
Aras Innovator Server 12 SP9	Windows Server 2012, Windows Server 2016

T-Systems licman20 license manager:

T-Systems licman20	please refer to the licman documentation

CHAPTER 2

Functional Changes

PDM to CAD Attribute Mapping only for CATIA Files claimed by the User

Changed PDM attribute values should only be written into CATIA files that are claimed by the user. This behavior should be optional.

Configuration

The behavior that read-only nodes should not automatically be modified is the new default behavior. The old behavior, writing differing PDM attribute values to read-only CATIA files, can be switched on by defining a global setting "AttributeMappingForReadOnlyNodes" with the value "true":

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

Extension of the "catiaEnvironment" Functionality

Before PWB 13.0.0, if the "Customer-Specific Environment" functionality was enabled, the user could only query for PDM Items fitting to the configured attributes.

Starting with PWB 13.0.0, it is possible to query for all PDM Items. PDM objects which do not fit to the configured environment variables are always loaded in ReadOnly mode.

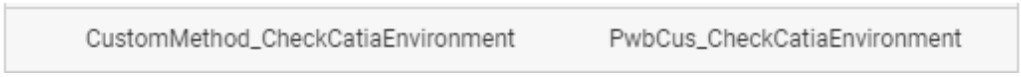
Configuration

To enable the new behavior, the optional setting "loadOthersReadOnly" has to be set to "true" (default: "false") in the PWB Schema file:

```
<catiaEnvironment loadOthersReadOnly="true">
    <envAttribute displayName="NLS_env_customer"
        pdm="pwb_customer"
        env="PWB_ENVIRONMENT_CUSTOMER" />
   <envAttribute displayName="NLS_env_project"
        pdm="pwb_project"
        env="PWB_ENVIRONMENT_PROJECT" />
</catiaEnvironment>
```

With "loadOthersReadOnly"="true" it is possible to use a custom server method to check if an item fits to the currently used catiaEnvironment.

Create a custom server method and set the value of the PWB Configuration setting "CustomMethod_CheckCatiaEnvironment" to your method name:



Picture 1: Sample "CustomMethod_CheckCatiaEnvironment" configuration

Sample Method:

```
// Sample CustomMethod_CheckCatiaEnvironment
//
// This method is called before an Item is send to CATIA.
// The method checks if the Item fits to the currently
// used CATIA environment.
// If not, the Item can neigther updated nor claimed in CATIA.


var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this);

Item ItemToCheck = this.getPropertyItem("ItemToCheck");
List<String> EnvironmentAttribueNames =
   PwbServerApiObj.StringToList(this.getProperty(
                                  "EnvironmentAttribueNames"));
List<String> EnvironmentAttribueValues =
   PwbServerApiObj.StringToList(this.getProperty(
                                  "EnvironmentAttribueValues"));

Item result = this.getInnovator().newItem();

if (ItemToCheck.getType()!= "CAD")
{
   // ignore CATIA environment settings
   result.setAttribute("EnvironmentOk", "true");
   return result;
}

if (EnvironmentAttribueNames != null &&
    EnvironmentAttribueValues != null &&
    EnvironmentAttribueValues.Count == EnvironmentAttribueNames.Count)
{
   for (int i = 0; i < EnvironmentAttribueNames.Count; i++)
   {
      string AttrValue = ItemToCheck.getProperty(
                         EnvironmentAttribueNames[i], "");
      if (AttrValue != EnvironmentAttribueValues[i] )
      {
         result.setAttribute("EnvironmentOk", "false");
         return result;
      }
   }
}

result.setAttribute("EnvironmentOk", "true");

return result;
```

Use Environment variable for Context Product

A designer must select a Context Product during selection of the CATIA start script. To avoid a second select it is possible to use an environment variable to set the Context Product.

Configuration

Set the environment variable:

```
PWB_CONTEXT_PRODUCT_ID=<Aras Id of the Context Product>
```

As the name of the Context Product is shown in the PWB Options dialog you can also set the environment:

`PWB_CONTEXT_PRODUCT=<Display name of the Context Product>`

Note:
The functionality is enabled by setting the PWB server configuration "CustomMethod_ContextProduct" to a custom method. The environment variables are only evaluated if this setting is not empty.

Let Aras Innovator delete orphaned files created at update

If you update a file in Aras Innovator, the original file gets orphaned. By default, the PDM Workbench deletes these orphaned files at the end of the update process.

Starting with Aras Innovator 12 it is possible to use the Aras Innovator variable item: `Force.Delete.Orphaned.Files=1`

to delete orphaned files. To make sure there is no conflict between Aras Innovator and the PDM Workbench about the deletion of the files you have to set the PWB configuration:

`DeleteOrphanFilesAtUpdate=false`

Performance improvement for Released Cache and Catalog

The "Released Cache" and "Catalog" functionality now uses a different kind of placeholder files in the PWB exchange directory. The placeholder files are deleted when closing CATIA V5.

Installation without PWB specific Environment Variables

By default, the PDM Workbench installer uses your standard CATIA V5 installation and environment to create a new CATIA V5 start script and a new CATIA V5 environment file which also holds the information about the PDM Workbench installation.

If you use a CATIA V5 start center that allows you to add an additional path for some 3rd party CATIA V5 modules, but you don't have the possibility to add extra PDM Workbench environment variables, you can install the PDM Workbench client without any additional environment variables.

1. Extract CATIA V5 specific package `PWBCATV5_R<XX>_<X>.<X>.<X>.zip` and copy the `win_b64` directory (`PWBCATV5_R<XX>_<X>.<X>.<X>\data\win_b64`) to your 3rd party CATIA V5 path.
2. Extract the Aras Innovator specific package `PWBCAD_<X>.<X>.<X>_Aras_<XXX>` and copy the content of `data\neutral\code\bin\` (`IOM.dll`, `PwbArasClientsCommon.exe`, `PwbArasFileClient.exe`, `PwbArasSoapClient.exe`) into `win_b64\code\bin\` of the previous copied directory.
3. From the extracted package `PWBCAD_<X>.<X>.<X>_Aras_<XXX>` copy the needed schema XML and DTD file to `win_b64\reffiles`.
    a. You have to copy the file `PWBSchema.dtd`.

b. If you want to use the CAD Structure data model, you have to copy the file `PWBSchema.xml`. If you want to use the BOM Part Structure data model, you have to copy `PWBSchema_Aras_PartStruc.xml` and rename it to `PWBSchema.xml`.

4. In the copied `PWBSchema.xml` file you have to adapt the following settings:

   a. PWB exchange directory
      `<xmap value=… />`
      To use a user specific xmap you can use system environment variables like follows:
      `<xmap value="${USERPROFILE}\xmap" />`

   b. Aras server URL
      `<soapTargetUrl value="…" />`

   c. Login data base(es)

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

If you use the “Open in CATIA” functionality, you also have to create the file `win_b64\code\bin\PwbListenerService.cfg`. (By default the file `PwbListenerService.cfg` is installed in the directory `config` of the PDM Workbench client installation. If the config file is located in the same directory like `PwbListenerService.exe` this configuration will be used):

```
LISTENER_PORT=8181
CATIA_DIR=C:\Program Files\Dassault Systemes\B30
CATIA_ENV_FILE=<full path>\catiaenv.txt
```

with:

LISTENER_PORT: Port number to connect from Aras Innovator (default: 8181)

CATIA_DIR: Installation directory of native CATIA V5

CATIA_ENV_FILE: Full path to a CATIA V5 environment file. This CATIA V5 environment must contain a PDM Workbench installation. It is possible to use a different CATIA V5 environment for the interactive CATIA V5 session.

The PwbListenerService must be installed manually. The service can be installed using Windows PowerShell:

```
PS C:\> New-Service -Name "PWBListenerService"
-DisplayName "PWBListener"
-BinaryPathName "<…>\win_b64\code\bin\PwbListenerService.exe"
-Description "Triggers PDM Workbench in CATIA V5 to load a
structure from Aras Innovator into CATIA V5"
-StartupType Automatic
```

Support user specific Exchange Directory when using “To Catia”

In previous releases (up to PWB 12) it was not possible to use user specific exchange directories when using the functionality “To Catia”. Now it is possible to set the exchange directory in the start script of CATIA V5 like:

```
SET PWB_XMAP=%USERPROFILE%\xmap
```

The “To Catia” functionality now uses the Windows temporary directory for some temporary files.

Add newly created and updated Part or CAD items to existing items

It is possible to link the Part and CAD items during the update process to an existing item. This can be done by using a custom method which is called at the end of the update process.

Examples:

1. The user wants to link a new top-level item to a selected Folder / Project item.
2. The user has several Change Items he has to work on. During the Update process the user selects the current Change Item he is working on from a list of his Change Items. This Change Item can be related to all updated Part or CAD items. This gives the possibility to understand later why a certain change was made.

The following Configuration shows a sample implementation to link a new top-level item to a selected Folder / Project item.

Configuration

There are two configuration points where a custom method can be called to configure this functionality.

One is called during login, and it returns the list of items (e.g. of a custom “Folder” type), to which CAD or Part items can be related to at the end of the update process.

The setting is “CustomMethod_PostProcUpdateInfo”, and its value has to be the name of a server-side C# method. Here is an example:

CustomMethod_PostProcUpdateInfo	PwbCus_PostProcUpdateInfo

Picture 2: Sample “CustomMethod_PostProcUpdateInfo” configuration

This method is supposed to return the names and values of a list of items which are stored in the PDM Workbench session. Entries from this list are then displayed in the update dialog. The user can select one of the items in the list. The ID of the selected item is then passed to the custom method at the end of the update process. In this method the newly created Part and CAD items can be related to the passed folder item.

The structure of the returned items is supposed to be like this:

‘Item type name’ + ‘|||’ + ‘name of item 1’ + ‘|’ + ‘ID of item 1’ + ‘||’ + ‘name of item 2’ + ‘|’ + ‘ID of item 2’ etc.

The pipe symbols (‘|’) are used for dividing the different entries in the string:

```
TypeName|||Item Name 1|ItemId1||Item Name 2|ItemId2||Item Name
3|ItemId3
```

This is an example of an actual returned string:

```
PwbFolder|||Folder Five|20DB73D8447748CFA067C07019B35ED4||Folder
Four|AAD93194CA8E464FB74166BFAA34994A||Folder
One|44978F9365C149AEA194F040502A85D1||Folder
Six|60EA79EE3E9D46E0A80EE72F99D43F26||Folder
Three|28AEB80AEDCC4C799335D3E3073D023B||Folder
Two|1E45E5D172F54A8CA5C6A00C85DDFA9A
```

This is an example implementation of the “CustomMethod_PostProcUpdateInfo” method:

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

```
CCO.Utilities.WriteDebug("_PwbOutput",
   "Called method -> PwbCus_PostProcUpdateInfo");

string TypeName = "PwbFolder";
string FolderInfoToReturn = TypeName;

FolderInfoToReturn += "|||";

var ResultList = PwbServerApiObj.PerformQuery(TypeName, "", "", "");

for (int i = 0; i < ResultList.Count; ++i)
{
   var ItemObj = ResultList[i];

   var Name = ItemObj.getProperty("name");
   var Id = ItemObj.getProperty("id");
   CCO.Utilities.WriteDebug(
      "_PwbOutput", "name:'" + Name + "', id:'" + Id + "'");

   FolderInfoToReturn += Name + "|";
   FolderInfoToReturn += Id;

   if (i < (ResultList.Count - 1))
   {
      FolderInfoToReturn += "||";
   }
}

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

OutputInfoDict.Add("PostProcUpdateInfo", FolderInfoToReturn);

Item OutputInfoItem =
   PwbServerApiObj.DialogAttrsDictionaryToItem(OutputInfoDict);

return OutputInfoItem;
```

The second configuration point is the method which relates the newly created part items to the selected folder item at the end of the update process. This method is called with information about the Part and CAD items which have been created and updated in the current update process.

The setting is “CustomMethod_PostProcUpdate”, and its value has to be the name of a server-side C# method. Here is an example:

Picture 3: Sample “CustomMethod_PostProcUpdate” configuration

The input information that is passed to the method looks like this:

<**UpdDlgSettings**>PwbFolder|28AEB80AEDCC4C799335D3E3073D023B</**UpdDlgSettings**>

<**RootNodeId**>B0AA1024532D42FB81916351006644CD</**RootNodeId**>

<**RootNodeFileName**>NewProduct3.CATProduct</**RootNodeFileName**>

<**RootPartId**>541AD2AB2F7249AFA46D91D7B75D7AAC</**RootPartId**>

<**RootNodePartNumber**>NewProduct3</**RootNodePartNumber**>

<**UpdatedFileIds**>EFE7E75D91804277917E9277D7725E85|44043BEF8305412A8CC7C06CD0524E9F|B0AA1024532D42FB81916351006644CD</**UpdatedFileIds**>

<**UpdatedFileNames**>NewPart4.CATPart|NewPart3.CATPart|NewProduct3.CATProduct</**UpdatedFileNames**>

<**UpdatedPartIds**>DD0C0CB5CDB74BE08463C21CA01D1D7F|B1AF4EBA36A447A8B820EEE811BB0890|541AD2AB2F7249AFA46D91D7B75D7AAC</**UpdatedPartIds**>

<**UpdatedPartNumbers**>NewPart3|NewPart4|NewProduct3</**UpdatedPartNumbers**>

<NewPartIds>DD0C0CB5CDB74BE08463C21CA01D1D7F|B1AF4EBA36A447A8B820EEE811BB0890|541AD2A
B2F7249AFA46D91D7B75D7AAC</NewPartIds>

<NewFileIds>44043BEF8305412A8CC7C06CD0524E9F|EFE7E75D91804277917E9277D7725E85|B0AA102
4532D42FB81916351006644CD</NewFileIds>

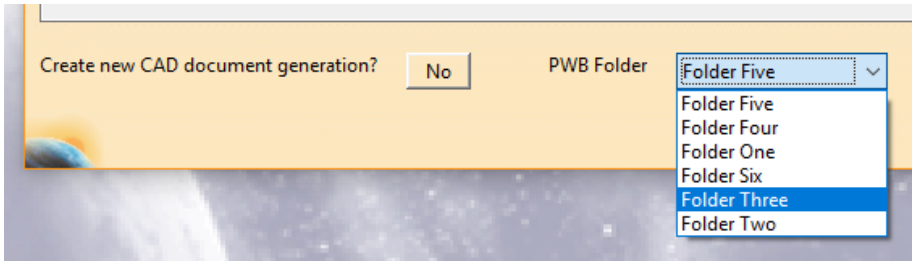
This is a sample implementation which relates the root part item of the updated structure to the selected folder item:

```
// Sample CustomMethod_PostProcUpdate
//
// This method is called at the end of a PWB update command after all
// regular PWB updates of that transaction were processed.
//
// It receives the CAD id of the top node of that PWB Update context
// and some additional xml information about updated children
//
// Should return this item or a new error item

Innovator InnovatorObj = this.getInnovator();

var PwbServerApiObj =
   new PwbServerAddin.PwbServerApi(this);

// Debug output of the received information.
string InputAttrXmlString = this.getProperty("InputAttrXmlString");
CCO.Utilities.WriteDebug(
   "_PwbOutput",
   "PwbCus_PostProcUpdate -> " +
   "InputAttrXmlString = '" + InputAttrXmlString + "'");

string AdditionalInfoXmlString =
   this.getProperty("AdditionalInfoXmlString");

CCO.Utilities.WriteDebug(
   "_PwbOutput", "PwbCus_PostProcUpdate -> " +
   "AdditionalInfoXmlString = '" + AdditionalInfoXmlString + "'");

// Parsing the structure.
string XmlString = "<Input>" + InputAttrXmlString + "</Input>";

var XmlDocObj = new XmlDocument();
XmlDocObj.InnerXml = XmlString;
XmlElement RootXmlElementObj = XmlDocObj.DocumentElement;

string UpdDlgSettingsString =
   RootXmlElementObj.SelectSingleNode("UpdDlgSettings").InnerText;
string RootNodeIdString =
   RootXmlElementObj.SelectSingleNode("RootNodeId").InnerText;
string RootNodeFileNameString =
   RootXmlElementObj.SelectSingleNode("RootNodeFileName").InnerText;
string RootPartIdString =
   RootXmlElementObj.SelectSingleNode("RootPartId").InnerText;
string RootNodePartNumberString =
   RootXmlElementObj.SelectSingleNode("RootNodePartNumber").InnerText;
string UpdatedFileIdsString =
   RootXmlElementObj.SelectSingleNode("UpdatedFileIds").InnerText;
string UpdatedFileNamesString =
   RootXmlElementObj.SelectSingleNode("UpdatedFileNames").InnerText;
string UpdatedPartIdsString =
   RootXmlElementObj.SelectSingleNode("UpdatedPartIds").InnerText;
string UpdatedPartNumbersString =
   RootXmlElementObj.SelectSingleNode("UpdatedPartNumbers").InnerText;
string NewPartIdsString =
   RootXmlElementObj.SelectSingleNode("NewPartIds").InnerText;
string NewFileIdsString =
   RootXmlElementObj.SelectSingleNode("NewFileIds").InnerText;
```

```
CCO.Utilities.WriteDebug(
   "_PwbOutput",
   "PwbCus_PostProcUpdate -> '" + UpdDlgSettingsString + "' / '" +
   RootNodeIdString + "' / '" + RootNodeFileNameString + "' / '" +
   RootPartIdString + "' / '" + RootNodePartNumberString + "' / '" +
   UpdatedFileIdsString + "' / '" + UpdatedFileNamesString + "' / '" +
   UpdatedPartIdsString + "' / '" + UpdatedPartNumbersString + "' / '" +
   NewPartIdsString + "' / '" + NewFileIdsString + "'");

if (UpdDlgSettingsString.Equals(""))
{
   return this;
}

string[] UpdDlgSettingsArray = UpdDlgSettingsString.Split('|');
if (UpdDlgSettingsArray.Length != 2)
{
   throw new Exception(
      "CustomMethod_PostProcUpdate -> UpdDlgSettingsArray.Length == " +
      UpdDlgSettingsArray.Length + " != 2 --> error");
}
string FolderType = UpdDlgSettingsArray[0];
string FolderId = UpdDlgSettingsArray[1];

Item FolderItemObj = InnovatorObj.getItemById(FolderType, FolderId);
PwbServerApiObj.CheckItem(
   FolderItemObj,
   "Item of type '" + FolderType +
   "' and ID '" + FolderId + "' not found");

string FolderName = FolderItemObj.getProperty("name");
CCO.Utilities.WriteDebug(
   "_PwbOutput",
   "PwbCus_PostProcUpdate -> FolderName:'" + FolderName +
   "' / '" + FolderItemObj.node.InnerXml + "'");

// Relate the folder item to new part items.
string[] NewPartIdsArray = NewPartIdsString.Split('|');
for (int i = 0; i < NewPartIdsArray.Length; ++i)
{
   string CurrentPartId = NewPartIdsArray[i];

   // Only relate the folder to the root part item.
   if (CurrentPartId != RootPartIdString)
   {
      continue;
   }

   Item PartItemObj = InnovatorObj.getItemById("Part", CurrentPartId);
   PwbServerApiObj.CheckItem(
      PartItemObj,
      "Part with ID '" + CurrentPartId + "' not found");

   string PartNumber = PartItemObj.getProperty("item_number");
   CCO.Utilities.WriteDebug(
      "_PwbOutput",
      "PwbCus_PostProcUpdate -> Part number:'" + PartNumber + "'");

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

   var NewRelationItem =
      PwbServerApiObj.CreateRelation(
         FolderItemObj, PartItemObj, "PwbFolderPart", Attributes);
}

return this;
```

Usage

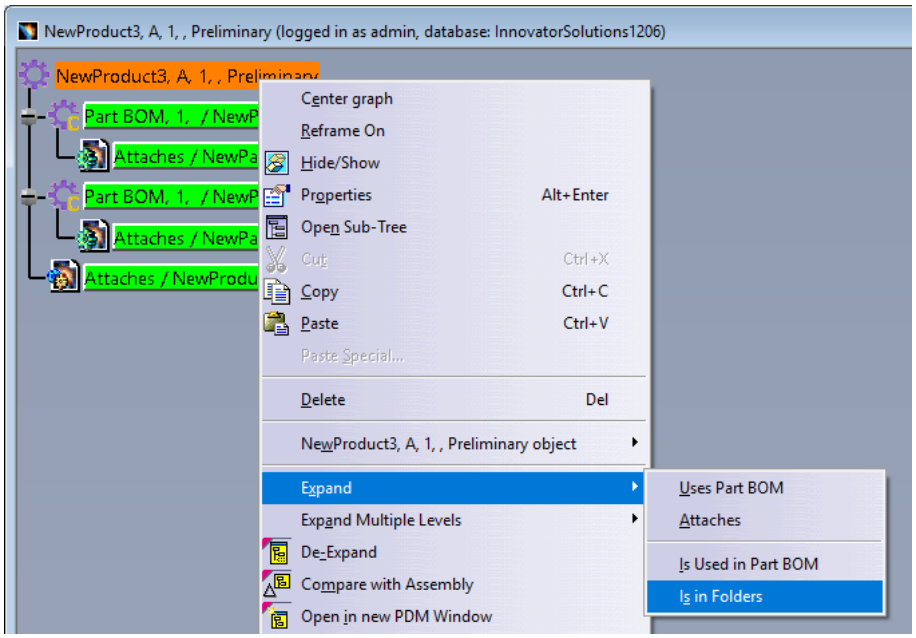
With the correct configuration the user can select an entry from a list of folder names in the update dialog. The list is the one returned by the custom method defined by the setting “CustomMethod_PostProcUpdate”:



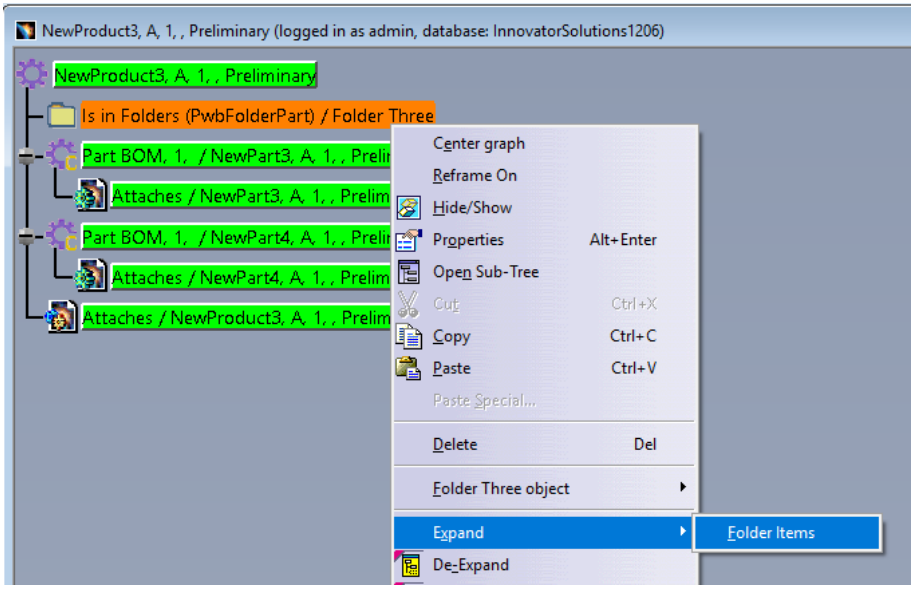
Picture 4: Folder list

The item that the user has selected will be the one that the newly created root Part item will be related to.

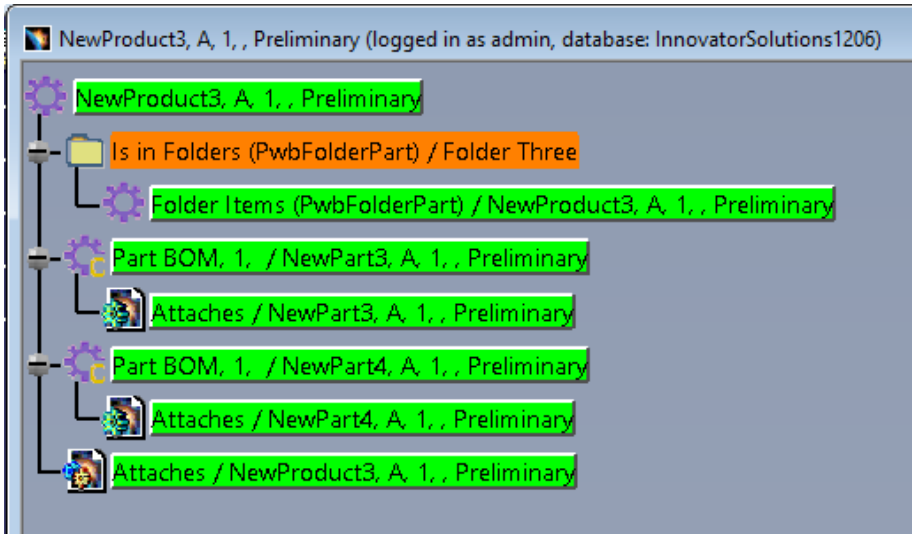
If the folder item and relationship is defined in the PWBSchema file the user can expand the relations in the PDM structure window and find out which folder, if any, a part item is related to, and which part items are related to a folder:



Picture 5: Expanding “Is in Folders” in the PDM structure window



Picture 6: Expanding "Folder Items" in the PDM structure window



Picture 7: Expanded folder items in the in the PDM structure window

Custom method to handle Catalog references

The standard PDM Workbench Catalog functionality stores the Aras Innovator Id or item_number (depending on the configuration) of the referenced items as a keyword inside the catalog file. This means this information is not available in Aras Innovator.

Thanks to this functionality it is possible to make this data available in Aras Innovator using a custom method which is called during update of a catalog.

Configuration

To enable this functionality, you must create a custom method to handle the referenced items of a catalog.

The PWB Configuration setting "CustomMethod_ProcessCatalogContent" must point to the method name of your custom method:

CustomMethod_ProcessCatalogContent	PwbCus_ProcessCatalogContent

Picture 8: Sample “CustomMethod_ProcessCatalogContent” configuration

The following method will write the referenced Id or item_number of the referenced items to a log file. If you work in CAD Structure mode, the list of referenced Parts is always empty.

```
/*
Sample method PwbCus_ProcessCatalogContent
This method allows to create / update links to referenced items in a
catalog
Set PWB Configuration CustomMethod_ProcessCatalogContent =
PwbCus_ProcessCatalogContent
*/

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

string CAD_Id             = this.getID();
string CAD_Type           = this.getType();
string CAD_Classification = this.getProperty("classification");

CCO.Utilities.WriteDebug("_PwbOutput",
   "PwbCus_ProcessCatalogContent -> CAD_Id:'" +
   CAD_Id + "', CAD_Class:'/" + CAD_Type + "/"+ CAD_Classification +
   "'");

int referencedPartCount = 0;

int.TryParse(this.getAttribute("referencedPartCount"),
   System.Globalization.NumberStyles.Integer,
   System.Globalization.CultureInfo.InvariantCulture,
   out referencedPartCount);

for (int i = 0; i < referencedPartCount; i++)
{
   String referencedPart =
      this.getAttribute("referencedPart_"+i.ToString());
   if (String.IsNullOrEmpty(referencedPart))
   {
      continue;
   }

   CCO.Utilities.WriteDebug("_PwbOutput",
      "PwbCus_ProcessCatalogContent -> referencedPart:' " +
      referencedPart );
}

int referencedCadCount = 0;

int.TryParse(this.getAttribute("referencedCadCount"),
   System.Globalization.NumberStyles.Integer,
   System.Globalization.CultureInfo.InvariantCulture,
   out referencedCadCount);

for (int i = 0; i < referencedCadCount; i++)
{
   String referencedCad =
      this.getAttribute("referencedCad_"+i.ToString());
   if (String.IsNullOrEmpty(referencedCad))
   {
      continue;
   }
   CCO.Utilities.WriteDebug("_PwbOutput",
      "PwbCus_ProcessCatalogContent -> referencedCad:' " +
      referencedCad );
}

Item result = InnovatorObj.newItem("result");
```

```
result.setAttribute("message", "Catalog Content handled ");

return result;
```

Fetch Part Number allow manual input

By default “Fetch Part Number” and “Fetch Nomenclature” present a list of possible new Part Numbers / Nomenclatures from a custom method. If the user needs a different number, the default functionality cannot be used.

Now it is possible to enter a new number manually if the custom method does not return a fitting number.

Configuration

The functionality has to be enabled in the settings section of the PWB Schema file:

```
<settings>
   <setting name="FetchPnAllowManualInput" value="true"/>
   <setting name="FetchNomenclatureAllowManualInput" value="false"/>
   <setting name="FetchPnUseNomenclatureForInstanceName" value="true"/>
   <setting name="FetchPnInputWidth" value="32"/>
</settings>
```

with

FetchPnAllowManualInput: enable manual input for “Fetch Part Number”.

FetchNomenclatureAllowManualInput: enable manual input for “Fetch Nomenclature / Rename”.

FetchPnUseNomenclatureForInstanceName: set default state of check box to create instance name from Part Number or from Nomenclature.

FetchPnInputWidth: dimension of the input fields.

Clear pre-filled Attributes from Create Dialog

Depending on the attribute mapping, many attributes in the create dialogs for CAD and Part are pre-filled with current CATIA V5 properties. In some cases, like copy structure, this causes additional effort and increases the possibility of wrong values in Aras Innovator.

This functionality offers the possibility to show a button to clear the content of the pre-filled attributes from the create dialog.

Configuration

Configure the register form to

a) create a clear button (info="ShowClearButton") and

b) keep attribute values when clear (pwbAttrInfo="DoNotClear"):

```
<form name="Register" info="ShowClearButton" >
   <formAttribute name="item_number" displayName="NLS_document_number"
                  widgetType="SingleLineEditor" mode="update"
                  visibleLength="15" allowedLength="256" required="true"
                  pwbAttrInfo="DoNotClear"/>
   <formAttribute name="name" widgetType="SingleLineEditor" mode="update"
                  visibleLength="15"
                  required="false" />
   <formAttribute name="description" widgetType="MultiLineEditor"
                  mode="update" visibleLength="15"
                  required="false" />
</form>
```

Support of Inactive List Values

Starting with Aras Innovator 12 SP8 there is a new feature "Inactive List Values". This feature allows the Aras Innovator administrator to inactivate List values going forward, while retaining the values as set on existing data. In Aras Innovator web client, starting in SP8, the List dropdown (and type-ahead) knows to omit any inactive values when setting new values, in a form or in a grid. When using a List dropdown (and type-ahead) for searching, the inactive values still appear.

Configuration

In the PWBSchema file, list values in a data source of type “ValueList” can have the XML attribute “pdmInfo”. If this attribute has the value “InactiveListValue” the list entry is removed from dialogs which update an item. It is still present in other dialogs, like the query dialog. The default value is “ActiveListValue”.

Example:

```
<dataSource name="Unit" type="ValueList">
   <value name="EA" displayName="NLS_EA" />
   <value name="IN" displayName="NLS_IN"/>
   <value name="FT" displayName="NLS_FT" pdmInfo="InactiveListValue" />
   <value name="MM" displayName="NLS_MM"/>
   <value name="CM" displayName="NLS_CM"/>
   <value name="M" displayName="NLS_M"/>
</dataSource>
```

Optional “Clear” Button in “Register” Dialogs of “CAD” Item Definitions

It is possible to add a “Clear” button to the “Register” dialogs of CATIA files. Clicking on this button clears all the values from the dialog attributes.

Configuration

```
<form name="Register" info="ShowClearButton">
   <formAttribute name="item_number" widgetType="SingleLineEditor" ... />
   <formAttribute name="name" widgetType="SingleLineEditor" ... />
   <formAttribute name="description" widgetType="MultiLineEditor" ... />
   <formAttribute name="is_template" widgetType="SingleCheckBox" ... />
   <formAttribute name="is_standard" widgetType="SingleCheckBox" ... />
</form>
```

CHAPTER 3

Environment and Configuration Variables

Server-side Configuration Variables

Configuration variable	Values

Variables PWBSchema file configuration

CHAPTER 4

Environment and Configuration Variables

Server-side Configuration Variables

Configuration variable	Values

Variables PWBSchema file configuration

CHAPTER 5

Data Model Changes

The following data model changes are introduced in this release of the PDM Workbench.

Added Classes

None.

Added Forms

None.

Changed Server Methods

CHAPTER 6

Bug Fixes

Version 13.0.0

.

CHAPTER 7

Functional overview older releases

Release	Windows Version	CATIA V5 Client Version	ARAS Innovator Version	Additional Functionalities	Part Structure	CAD Document Structure
3.5	Windows XP (32 Bit, 64 Bit) Windows 7 (64Bit)	V5R21 V5-6R2012 V5-6R2013	9.3	CATIA DesignTable functionality (configurable in Schema.xml)	X	X
				Bounding Box Management "Show Neigbour" functionality	X	X
				Automatic Part Creation	-	X
				Synchronize CAD structure to BOM	-	X
				“Current” and “Released” Expand Modes	-	X
3.6	Windows 7 (64Bit)	V5-6R2012 V5-6R2013 V5-6R2014	9.4 10.0	Support for the new CAD structure instance	-	X
				Standard Part Functionality	X	-
				Check for CAD document CATIA release at PDM update	X	X
				Extended attribute mapping functionality	X	X
				Local Workspace Information	X	X
				Optional Load of linked CATPart Files	X	X
				Newest Version Info Context Menu	X	X

Release	Windows Version	CATIA V5 Client Version	ARAS Innovator Version	Additional Functionalities	Part Structure	CAD Document Structure
3.7	Windows 7 (64Bit)	V5-6R2012 V5-6R2013 V5-6R2014	9.4 10.0	Non-BOM CATParts and CATProducts	X	X
				Configuration of BOM Part Structure	X	X
				Archives (Compress Files ZIP in ARAS)	X	X
				Standard Part functionality for CAD structure mode	X	X
				Check CAD Links	X	X
				Displaying part structure instances as separate nodes	X	X
				Possibility to call a server method for a PDM item	X	X
				Change in drawing attribute mapping configuration	X	X
3.8	Windows 7 (64Bit)	V5-6R2013 V5-6R2014 V5-6R2015	10.0 11.0	Saving PDM Session Information	-	X
				Part CAD Filter	X	X
				BOM Configuration Range Extension	X	X
				Revising as a different user ('ReviseAs')	X	X
				Improvements for custom context action functionality	X	X
				Show Neighborhood Improvement	X	X
3.9	Windows 7 (64Bit)	V5-6R2013 V5-6R2014 V5-6R2015	10.0 11.0	Select Type of additional Parts in Document mode	-	X
				Support Electrical / Tubing	X	X
				Reconnect at Update	X	X
				Configuring the size of the Query dialog	X	X

Release	Windows Version	CATIA V5 Client Version	ARAS Innovator Version	Additional Functionalities	Part Structure	CAD Document Structure
				“Lock All” / “Unlock All”	X	X
4.0	Windows 7 (64Bit)	V5-6R2014 V5-6R2015 V5-6R2016	11.0 11 SP5	Duplicate Structure	-	X
				Clean up / Housekeeping of PWB_XMAP Directory	X	X
				Support for CGRs as Native Files	X	X
				CATDrawing: Loading referenced Data as ‘Current’	X	X
				Simplify PDM tree view	X	X
				Unlock after Save	X	X
				Creation of Thumbnails by CATIA	X	X
				“External Owner” Attribute in Query Dialog	X	X
				Current Database displayed in Window Title Bars	X	X
5.0	Windows 7 (64Bit)	V5-6R2014 V5-6R2015 V5-6R2016	11.0 11 SP5	Additional Rep Types	X	-
				Allow deactivated CATProduct and CATPart instances	X	X
				Load Substructures in Context	X	X
				Automatically loading CATDrawings or linked CATParts from the Query Window	X	X
				Query dialog filter attribute values are kept when changing the type	X	X
				Unlink and Delete Newest Version	X	-
				Setting configuration information on structure relations	X	X

Release	Windows Version	CATIA V5 Client Version	ARAS Innovator Version	Additional Functionalities	Part Structure	CAD Document Structure
				Replacement of class ‘ArasUtil’ with class ‘PwbServerAddin.PwbServerApi’ in custom Server Methods	X	X
6.0	Windows 7 (64Bit)	V5-6R2015 V5-6R2016	11 SP9	Released Cache Mode	X	X
				Create Drawing CAD document: Automatically select loaded part in session if a single link exists	X	X
				Optional check for broken links at update	X	X
	Windows 7 (64Bit) Windows 10 (64Bit)	V5-6R2017		Create CAD in Parent	X	X
				Default value of Server Setting “ShowCreateDialogsDuringUpdate” has changed to “true”	X	X
7.0	Windows 7 (64Bit)	V5-6R2015 V5-6R2016	11 SP9	Support of CATIA Catalogs	X	X
				Additional Options for “Load with Links”	X	X
	Windows 7 (64Bit) Windows 10 (64Bit)	V5-6R2017		Displaying Attributes in PDM Nodes in several Lines	X	X
				Preselection of Nodes on the PDM Structure Window	X	X
8.0	Windows 7 (64Bit)	V5-6R2016	11 SP12	Comparing PDM Structure Trees	X	X
				Selecting Nodes in the PDM Structure Window	X	X
	Windows 7 (64Bit) Windows 10 (64Bit)	V5-6R2017 V5-6R2018		Alternative Icons for PDM Structure Nodes	X	X
				Inertia attributes mapping	X	X
				Rollback on File Upload Error	-	X

Release	Windows Version	CATIA V5 Client Version	ARAS Innovator Version	Additional Functionalities	Part Structure	CAD Document Structure
				"Select Date" Enhancement in the Query Dialog	X	X
9.0	Windows 7 (64Bit)	V5-6R2016	11 SP12	“Open in CATIA” from Aras Innovator Client	X	X
	Windows 7 (64Bit) Windows 10 (64Bit)	V5-6R2017 V5-6R2018		Enhancement of „Configurable Node Behavior / Skip Nodes”	X	X
10.0	Windows 7 (64Bit)	V5-6R2016	11 SP12	“Open in Aras” from CATIA V5	X	X
				Duplicate Structure	X	X
				Enhancement of „Duplicate Structure“: „originatedFrom” Property and Standard Part handling	X	X
	Windows 7 (64Bit) Windows 10 (64Bit)	V5-6R2017 V5-6R2018		Support generic Shape Representations	X	X
				Delete relations of non-loaded instances	X	X
				Support floating content in Catalog	X	X
	Windows 10 (64Bit)	V5-6R2019		Configurable Catalog Keywords	X	X
				Enhancement of „Reconnect“ Functionality: No longer connected with the “Auto Name” Functionality	X	X
				Use Server method for Quantity	X	X
11.0	Windows 7 (64Bit)	V5-6R2016	11 SP12 12	Validate Structure before Update	X	
	Windows 7 (64Bit)	V5-6R2017		Material	X	X

Release	Windows Version	CATIA V5 Client Version	ARAS Innovator Version	Additional Functionalities	Part Structure	CAD Document Structure
	Windows 10 (64Bit)	V5-6R2018				
	Windows 10 (64Bit)	V5-6R2019		Support Part BOM Classification	X	
				Manage Context Products	X	X
12.0	Windows 7 (64Bit)	V5-6R2016	12 12 SP7	CATProcess file support		
				Show Create Dialogs multi column	X	
	Windows 7 (64Bit) Windows 10 (64Bit)	V5-6R2017 V5-6R2018		Excluding child node types for creation		
				Template file support for 'create part' with templates depending on the part type		
	Windows 10 (64Bit)	V5-6R2019 V5-6R2020		PDM to CAD attribute mapping only for CATIA files claimed by the user		
				Extension of the "CATIA Environment" functionality	X	X
				Use Environment variable for Context Product	X	X
13.0	Windows 7 (64Bit)	V5-6R2016	12 12 SP9	PDM to CAD Attribute Mapping only for CATIA Files claimed by the User	X	X
				Extension of the "catiaEnvironment" Functionality	X	X
				Use Environment variable for Context Product	X	X
				Let Aras Innovator delete orphaned files created at update	X	X
	Windows 7 (64Bit) Windows	V5-6R2017		Performance improvement for Released Cache and Catalog	X	X
				Installation without PWB specific Environment Variables	X	X

Release	Windows Version	CATIA V5 Client Version	ARAS Innovator Version	Additional Functionalities	Part Structure	CAD Document Structure
	10 (64Bit)	V5-6R2018		Support user specific Exchange Directory when using "To Catia"	X	X
				Add newly created Part or CAD items to existing items	X	X
	Windows 10 (64Bit)	V5-6R2019 V5-6R2020		Custom method to handle Catalog references	X	X
				Fetch Part Number allow manual input	X	X
				Clear pre-filled Attributes from Create Dialog	X	X
				Support of Inactive List Values	X	X
				Optional "Clear" Button in "Register" Dialogs of "CAD" Item Definitions	X	X